Inhaltsverzeichnis

Vorwort

Die höchste Pflicht des Intellektuellen unserer Zeit ist es,
die einfachsten Wahrheiten in den einfachsten Worten auszusprechen.

George Orwell

Eine Datenanalyseumgebung zur *komfortablen* und *flexiblen* Exploration multidimensionaler Daten zu entwerfen, ist das erklärte Ziel dieser Arbeit. Gleichzeitig — oder besser gesagt: gerade mit diesem Anspruch — soll eine Sprache zur *exakten* Beschreibung von interaktiven Analysesitzungen konzipiert werden. Die hierzu unverzichtbaren formalen Modelle und teilweise doch recht komplexen Definitionen, die große Abschnitte der zentralen Kapitel dieser Arbeit dominieren, mögen zunächst einmal scheinbar im Widerspruch zu obiger Forderung von Orwell stehen.

Und doch wollen wir — trotz bzw. als Komplementierung aller mathematischen Exaktheit — versuchen, in möglichst „einfachen“ Worten und stets mit direktem Bezug zur gewählten Anwendungsdomäne alle Entwurfsentscheidungen und Modellkonstrukte zu motivieren und zu erläutern. Aufgestellte Modelle ebenso wie eingeführte Begrifflichkeiten sollen sich direkt aus den praktischen Anforderungen des Datenanalysten sowie aus den jeweiligen Objekten der Datenanalyse herleiten, so daß mit dem formalen Unterbau gleichzeitig eine intuitiv nutzbare Analyseplattform entsteht. Der Leser bzw. der spätere Anwender von *VIOLA* mag beurteilen, inwiefern dies gelungen ist.

Eine Arbeit wie diese kann nicht ohne ein geeignetes Umfeld entstehen. Deshalb gebührt einer ganzen Reihe von Personen mein herzlicher Dank. Ich bedanke mich bei

- Herrn Prof. Dr. Hans-Jürgen Appelrath für die Freiheiten, die er mir als Doktorvater bei meiner Arbeit in seiner Abteilung bzw. im OFFIS gewährt hat, für ein stets offenes Ohr sowie für die richtige Mischung aus Forderung und Rückhalt,
- Herrn Prof. Dr. Hans-Joachim Lenz und Frau Prof. Dr. Stephanie Teufel dafür, daß sie sich zur Übernahme von Zweit- und Drittgutachten für diese Arbeit bereit erklärt und mir bereits im Vorfeld durch ihre Kommentare so einige Anregungen geliefert haben,
- Heidrun Ortleb, Frau PD Dr. Angelika van der Linde und Vera Kamp für ihre Unterstützung beim Einstieg in die Gebiete der Epidemiologie, Statistik bzw. multidimensionalen Datenanalyse,
- dem gesamten CARLOS–Team für eine angenehme Arbeitsatmosphäre,
- meinen Mitarbeitern Alexander Scharnofske und Martin Rohde für interessante fachliche Diskussionen,
- Ina Wellmann und Joachim Kieschke für viele hilfreiche Anregungen „aus Anwendersicht“,
- all meinen (ehemaligen) Diplomanden und Studienarbeitern, Teilnehmern meiner Projektgruppe und meiner Seminare sowie von mir betreuten studentischen Hilfskräften (insbesondere Ralf Hüsing, Jürgen Meister, Carsten Keltsch, Jörg Dannemann, Sascha Koch, Michael Brandt, Michael Suschke und Roland Ahlborn) für etliche gute Ideen, Denkanstöße und nicht zuletzt ihre Implementierungsarbeiten im Rahmen eines *VIOLA*–Prototypen,
- Ralf Hüsing und Martin Rohde für ihr akribisches Korrekturlesen,

- meinen Eltern dafür, daß sie mir meine bisherige akademische Laufbahn überhaupt erst ermöglicht haben,
- und — last, but not least — meiner Frau Michaela für tatkräftige Unterstützung beim finalen Korrektur–Marathon sowie für viel Geduld, Verständnis und unverzichtbare Aufmunterungen in den letzten „entbehrungsreichen“ Jahren.

Oldenburg, im Januar 2000

Frank Wietek

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Eine ureigene Aufgabe der Naturwissenschaften im speziellen, aber auch vieler anderer Wissenschaften im weiteren Sinne besteht darin, aus Daten Informationen zu extrahieren. In wissenschaftlichen und kommerziellen Anwendungen werden — mit steigenden Speicher-, Transfer- und Verarbeitungskapazitäten moderner Rechneranlagen zunehmend — große Datenmengen angesammelt, aus deren Auswertung wertvolle Erkenntnisse erhofft werden. Die Hilfsmittel zur Bewältigung dieser Aufgabe liefert die Datenanalyse.

Bei der Analyse von Datenbeständen wirken eine Vielzahl unterschiedlicher Fachdisziplinen zusammen — neben den jeweiligen Anwendungsdomänen vor allem aus dem Bereich der Statistik und der Informatik. Um jeweils möglichst effizient alle interessanten und bedeutsamen Informationen — nicht weniger, aber auch nicht mehr — aus einem gegebenen Datenbestand zu extrahieren, also auf möglichst *intelligente* Art Datenanalyse zu betreiben, bedarf es einer möglichst guten Kombination vorhandener Ansätze. Ein zentraler, übergreifender Aspekt ist vor diesem Hintergrund das Zusammenwirken von menschlichem Datenanalysten und rechnerbasiertem Analysesystem, das eine nahezu unentbehrliche Hilfe gerade bei der Analyse *großer* Datenbestände geworden ist. So charakterisiert Hand in [Han97b] einen zentralen Aspekt *intelligenter* Datenanalysen wie folgt:

> *Intelligent data analysis is [...] combining the complementary strategies of computers and humans.*

Mensch und Rechner sind also als sich ergänzende Analysepartner anzusehen. Beide sollten sich im Rahmen einer explorativen, also nach „verborgenen“ Strukturen suchenden Datenanalyse auf ihre jeweiligen Stärken konzentrieren können: der menschliche Datenanalyst auf seine Kreativität, die Fähigkeit zum Erkennen und zur Abstraktion genereller Eindrücke und Strukturen aus komplexen Datenmengen, zum Anwenden informeller Heuristiken sowie zum Aufstellen neuartiger Hypothesen auf der Basis von Zwischenergebnissen der Analyse; der Rechner dagegen auf die Verwaltung und Bereitstellung von Daten und Verfahren sowie die systematische und effiziente Durchführung komplexer Berechnungen.

Damit nun das interaktive Zusammenspiel zwischen Rechner und Anwender reibungslos funktionieren kann, bedarf es einer geeigneten Schnittstelle, die den jeweiligen Wissens- bzw. Planungsstand bzgl. Problemstellung, Lösungsstrategien, Analyseschritten, Ergebnissen und deren Interpretation zwischen den beiden Analysepartnern kommuniziert (vgl. auch [Ter93]). Insbesondere dem menschlichen Nutzer sind somit stets ein umfassender Überblick und flexible Eingriffsmöglichkeiten in den Ablauf einer Analysesitzung anzubieten, damit er sein Datenanalysepotential voll ausschöpfen und Analyseergebnisse — soweit dies die vorliegende Datenbasis zuläßt — korrekt interpretieren kann.

In diesem Sinne soll in dieser Arbeit eine rechnerbasierte Datenanalyseumgebung entworfen werden, die *intelligente* Datenanalysen ermöglicht, indem sie möglichst alle für die Analyse relevanten Informationen unmittelbar und umfassend zur Verfügung stellt. Es geht dabei *nicht* um den Einsatz von Verfahren der Künstlichen Intelligenz oder die Entwicklung eines statistischen Expertensystems, sondern „lediglich“ — und diesem Aspekt wird (zu) oft viel zu wenig Beachtung geschenkt — um eine *fundierte Basis* zur intuitiven und erfolgreichen

Datenanalyse durch Formalisierung und Nutzung differenzierter Metadaten, die die jeweilige Datenbasis näher beschreiben.[1]

In Abschnitt 1.1 wird der hier verfolgte Ansatz genauer in die bestehende Landschaft von Datenanalysesystemen eingeordnet, wobei zentrale Defizite aufgezeigt werden, die unter diesen bestehen. Wesentliche Anregungen zur Entstehung dieser Arbeit lieferten Arbeiten zum Aufbau des Epidemiologischen Krebsregisters Niedersachsen (EKN) [EKN00]. Somit soll als ein durchgängiges Anwendungsbeispiel, aber auch als zentrale Definition spezieller Anforderungen die deskriptive Epidemiologie, insbesondere die Krebsepidemiologie und -registrierung, dienen, also die Beschreibung der Verteilung von Krankheiten (Krebsfällen) und ihren Risikofaktoren in der Bevölkerung. Abschnitt 1.2 stellt in einem kleinen Exkurs einige Grundlagen der Epidemiologie vor. An der adäquaten Unterstützung dieses Anwendungsgebiets soll der hier vorgestellte Entwurf gemessen werden. Es gilt, die subjektive Zufriedenheit von Epidemiologen bei der rechnergestützten Datenanalyse durch intuitive Systembedienung, klar definierte Analyseergebnisse sowie Förderung der eigenen Kreativität und Hypothesenbildung während der Datenexploration zu erhöhen. Natürlich wird auch darauf geachtet, die entwickelten Konzepte weitgehend unabhängig von dieser Domäne bzw. übertragbar auf andere Anwendungsbereiche zu gestalten. Abschnitt 1.3 schließt diese Einleitung mit einem Überblick über die Arbeit ab.

1.1 Einordnung der Arbeit

Diese Arbeit beschäftigt sich speziell mit der Analyse multidimensionaler, aggregierter Daten als ein aktuelles, unter Schlagworten wie „OLAP“, „Data Warehousing“ oder „Decision Support“ diskutiertes Forschungsgebiet. Da sich alle Datenwerte in derartigen Datenbeständen jeweils auf Gruppen von Individuen beziehen, spricht man hier auch allgemein von *Makrodaten*. Diese stehen im Gegensatz zu den typischerweise relationalen *Mikrodaten*, die oftmals die Basisdaten bilden, aus denen Makrodaten durch Zusammenfassung von Einzelwerten gewonnen werden. Die Verarbeitung von Mikrodaten (sofern sie nicht direkt in Bezug zu Makrodaten stehen) wird in dieser Arbeit nicht näher betrachtet.

1.1.1 OLAP und Data Warehousing

Mit der Ansammlung immer größerer Datenbestände ist in letzter Zeit insbesondere in der betriebswirtschaftlichen Anwendung die systematische und komfortable Datenanalyse in das allgemeine Interesse gerückt. Auf der Basis von aus vielen operativen Datenbanken integrierten Data Warehouses dienen sogenannte *OLAP– (On–line Analytical Processing) Tools* zur Entscheidungsunterstützung [CD97]. Diese Werkzeuge modellieren den betrachteten Datenbestand i. a. als eine Menge multidimensionaler *Datenräume* (auch *Datenwürfel* oder *Data Cubes*), die durch eine Reihe *kategorieller* Attribute (*Dimensionen*), wie z. B. Produktgruppe, Verkaufsgebiet oder -zeitraum, aufgespannt werden und zu den jeweiligen Kombinationen von Kategorien auf verschieden granularen *Aggregierungsebenen* (wie Landkreis, Gemeinde oder Bundesland) die betreffenden *Maßzahlen* (*summarische Attribute*), wie Umsatz, Gewinn, Verkaufszahlen etc., enthalten.

Im Vordergrund bei der Nutzung von OLAP–Tools steht eine möglichst einfache Bedienbarkeit gerade durch „Nicht–Statistiker“ oder „Nicht–Programmierer“, vor allem durch die jeweiligen betrieblichen Entscheidungsträger, denen auf der Suche nach interessierenden Informationen eine flexible Navigation durch den Datenbestand zu ermöglichen ist.

Nun ist die Problematik der Analyse mehrdimensionaler Datenräume sicher nicht auf die Auswertung von Unternehmensdatenbanken beschränkt. So stellen etwa in der Epidemiologie Neuerkrankungszahlen oder Sterblichkeitsraten nach Alter, Geschlecht, Untersuchungsregion, betrachtetem Zeitraum und Art der Erkrankung typische auszuwertende multidimensionale Datenräume dar. Wie in der Betriebswirtschaft, dem originären Anwendungsgebiet von OLAP, ist auch hier Fachkräften aus verschiedenen Disziplinen, wie Sozialwissenschaften, Medizin, Dokumentation etc., eine komfortable, explorative Analyse des Datenmaterials zu ermöglichen.

[1] Derartige Beschreibungen bilden natürlich zugleich auch die Grundlage für jede wissensbasierte Datenverarbeitung.

Spezielle Datenanalysesysteme für die Epidemiologie sind rar; meist werden hier gängige Standard–Statistikpakete wie *SAS* oder *SPSS* verwendet, die sich in letzter Zeit zunehmend von batchorientierten Funktionsbibliotheken zu deutlich benutzungsfreundlicheren menübasierten Systemen mit (neben einer Vielzahl von Analysefunktionen) ansprechenden Möglichkeiten zur Datenvisualisierung gewandelt haben.

Die multidimensionale Struktur der in der Epidemiologie untersuchten (Makro–)Daten findet — in Zusammenhang mit den speziellen Charakteristika dieses Anwendungsgebiets, wie

- einem gegenüber OLAP–Anwendungen deutlich geringeren Datenumfang (jeweils höchstens bis zu wenigen Millionen Datensätzen), zugleich aber höheren Anforderungen an
- die hochinteraktive Analyse und Visualisierung sowie an
- die flexible und vergleichende Maßzahlwahl (vgl. auch Abschnitt 1.2) —

jedoch kaum Beachtung in diesen Systemen.

1.1.2 Schwächen bestehender Datenanalysesysteme

Die meisten der existierenden Datenanalysesysteme — sei es im allgemeinen oder speziell im OLAP–Bereich — konzentrieren sich vor allem auf die Bereitstellung umfangreicher Funktionsbibliotheken zu Datenmanagement, Statistik und Datenvisualisierung. Wenig Augenmerk legen sie jedoch auf die Verwaltung des dynamischen Analyseprozesses an sich, der oftmals aus langen, aufeinander aufbauenden oder sich in verschiedene Teilbereiche verzweigenden Sequenzen einzelner Analyseschritte besteht. Dem Benutzer fehlen der Überblick über diese „Historie“ der bereits durchgeführten Analyse sowie entsprechend auch Möglichkeiten zu deren einfacher Manipulation (vgl. auch z. B. [You96, SSW96]). Dies erschwert sowohl die Wahl und Anwendung geeigneter nachfolgender Analyseverfahren als auch die korrekte Interpretation der erhaltenen Ergebnisse.

Weiterhin ist auch die Integration aller Teilprozesse einer Datenanalyse oft nur unvollständig gelöst [Lee94]. So bieten OLAP–Tools zwar mächtige Operatoren zur Definition komplexer, um statistische Funktionen angereicherter Datenbankanfragen, behandeln aber kaum weitere Schritte nachfolgender *explorativer* Datenanalyse und -visualisierung, die auf Zwischenergebnissen aufbauend Untersuchungen iterativ verfeinert und ergänzt.[2] Gerade im Hinblick auf die multidimensionale Datenanalyse stellen gängige Statistikpakete kaum Operatoren zur intuitiven Navigation in komplexen Datenwürfeln zur Verfügung. Und die interaktive Datenvisualisierung schließlich findet nur zögerlich ihren Einzug aus Forschungsprototypen in die gängige Datenanalysesoftware, wobei es oft an einem einheitlichen Konzept ihrer Einbindung in die gesamte Analysefunktionalität mangelt.

Bedingt durch die genannten Defizite von Analysesystemen ergibt sich schließlich auch ein Bruch zwischen dem Entwurf einer Datenanalyse als Entwicklung bzw. Umsetzung einer Analysestrategie zur Bearbeitung eines Forschungsziels (oft auf Papier oder mit anderen externen Hilfsmitteln) einerseits und deren Implementierung und Durchführung durch den Rechner andererseits. Es fehlt eine intuitive, kognitiv angemessene Umsetzung von Strategien in eine vom Rechner zu verarbeitende Repräsentation unter Nutzung geeigneter Interaktionsmetaphern. Diese sollten eine direkte Manipulation von Analyseschritten ermöglichen, so daß sich der Benutzer voll auf die eigentliche Analyse konzentrieren kann und nicht durch Systembedienung bzw. Wechsel der Analyseumgebung abgelenkt wird [Lee94]. Auf der Basis eines jeweils klar definierten und dargestellten Kontextes der Analyse ist — gerade vor dem Hintergrund der hier zugrundegelegten Datenbestände von eher mittlerer, also insofern „handlicher“ Größe — eine aktive Einbeziehung des Benutzers in eine interaktive Datenexploration von zentraler Bedeutung [SBM92].

1.1.3 Visuelle Programmierung als Lösungsansatz

Die Visualisierung von Daten bildet die Grundlage für die Kommunikation zwischen Anwender und Analysesystem im Rahmen intelligenter Datenanalysen: Nur wenn der Datenanalyst sieht, *was* er analysiert, kann er geeignete neue Verfahren zur Analyse auswählen und so dem Analyseablauf eigene Impulse geben.

[2] Teilweise ist dies auch durch die Größe der in betriebswirtschaftlichen Anwendungen verarbeiteten Datenbestände bedingt.

In konsequenter Fortführung dieser Idee bietet die datenflußbasierte visuelle Programmierung eine ausgezeichnete Möglichkeit zur Lösung der im vorangegangenen Abschnitt angeführten Probleme: Nicht nur das Analyse*ergebnis* selbst, sondern der gesamte *Prozeß* der jeweiligen Analysesitzung wird visualisiert. Verschiedene Bausteine (Module), die jeweils die Ausführung einer Bearbeitungsfunktion repräsentieren (Datenzugriff und -management, Analyse- und Visualisierungsverfahren), werden in Form eines Ablaufdiagramms gemäß der Berechnung neuer (Zwischen–)Ergebnisse miteinander verbunden. Explorative Datenanalyse erfolgt innerhalb des Datenflußprogramms durch Parametrisierung oder Austausch bestehender Netzbausteine gegen ähnliche, ebenso passende Verfahren sowie Erweiterung des Graphen um auf den bisherigen Ergebnissen aufbauende Analyseschritte. Die Resultate vorgenommener Modifikationen werden gemäß des zugrundeliegenden Datenflußparadigmas zu den unveränderten Abschnitten der Analyse propagiert und in bestehenden Visualisierungen sichtbar gemacht.

Die visuelle Programmierung hilft auf diese Weise auch bei der Planung und schrittweisen Durchführung von Datenanalysen, deren genauer Ablauf im voraus oft noch gar nicht klar ist. Hierbei ist gerade die Repräsentation von Analyseverfahren als Bausteine, als „Black Boxes“, ein angemessenes Abstraktionsniveau, das eine intuitive Anwendung der Programmierumgebung erlaubt. Auch entspricht die datenflußbasierte Sichtweise dem natürlichen Analyseprozeß — Analysestrategien werden bildlich direkt umgesetzt [YS91]. Entwurf, Implementierung und Ausführung erfolgen integriert in *einem* System.

Hauptziel dieser Arbeit ist somit der Entwurf einer datenflußbasierten visuellen Programmierumgebung (*VIOLA — VIsual On–Line data Analysis environment*) für die multidimensionale Datenanalyse, die insbesondere typische epidemiologische Anwendungen unterstützt, aber auch in anderen, ähnlichen Anwendungsgebieten einsetzbar ist. Hierbei sollen durch eine umfassende metadatenbasierte Information des Benutzers über Gestalt und Ergebnisse der jeweils durchgeführten Analysesitzung *intelligente* Datenanalysen im oben definierten Sinne ermöglicht werden:

- Ein zugrundeliegendes Datenmodell *MADEIRA* (*Modelling Analyses of Data in Epidemiological InteRActive studies*) definiert jeweils die Semantik der betrachteten Daten und Ergebnisse.
- Die visuelle Umgebung *VIOLA* selbst repräsentiert den Ablauf der Analyse. Somit zeigt sie dem Anwender auf, wo er sich im Analyseprozeß befindet, welche Auswertungen er bisher vorgenommen hat, wie Zwischenergebnisse hergeleitet wurden und welche Verfahren er als nächstes anwenden kann. Weiterhin liefert sie die Grundlage für eine interaktive Manipulation der Daten und auch des Analyseablaufs.

VIOLA integriert sowohl Zugriff auf multidimensionale Datenbestände, Datenmanagement, also die Selektion interessierender Teilräume, und Navigation auf diesen Daten als auch Analyse und interaktive Visualisierung im Rahmen eines einheitlichen Konzepts zur datenflußbasierten Datenexploration. Auf der Basis eines unterliegenden, lediglich als Lieferant der Quelldaten fungierenden „Data Warehouses“ wird eine eigenständige Schicht zur flexiblen Manipulation multidimensionaler Datenräume definiert. Durch eine geeignete Modellierung der Datenflußrepräsentation einer Anfrage sowie einer Kapselung ihrer Verarbeitung kann das Gesamtsystem offen für die Anbindung unterschiedlicher Benutzungsschnittstellen für verschiedene Nutzergruppen und Anwendungszwecke bleiben. Neben der kanonischen Netzwerkdarstellung bieten sich etwa eine menübasierte Schnittstelle für einfache Standardauswertungen oder Skripte zur Automatisierung häufiger Abläufe an, wobei stets ein Wechsel auf die flexiblere Datenflußsicht möglich ist.

Schließlich bilden auch die Analysebausteine eines Datenflußprogramms eine geeignete Abstraktionsebene zur einfachen Systemerweiterung sowie zur Anbindung externer Datenbanken, Analyse- und Visualisierungsprogramme (vgl. [SSW96]). Hierdurch läßt sich das System generisch und damit auf verschiedene Anwendungsdomänen übertragbar gestalten.

Von bestehenden datenflußbasierten Systemen zur Datenanalyse hebt sich *VIOLA* ab durch die enge Verflechtung mit einem multidimensionalen Datenmodell (*MADEIRA*), die eine besonders gute und intuitive Unterstützung multidimensionaler Datenanalysen gestattet und die Basis für eine exakte Definition aller Analyseschritte bietet, sowie die Einbeziehung interaktiver Datenvisualisierung in ein homogenes Modell.

Wie bereits erwähnt, soll es nicht Ziel dieser Arbeit sein, ein Expertensystem zur Datenanalyse zu entwickeln. Außerdem sollen auch keine neuen Speicherungs- und Zugriffsstrukturen für multidimensionale Daten entwickelt oder spezielle Analyse- und Visualisierungsverfahren konzipiert und realisiert werden — lediglich deren flexible Anbindung ist zu ermöglichen. Auch die über mehrere Rechnersysteme eines lokalen Netzwerks (oder das Internet) verteilte Datenanalyse, also hier die Zuweisung von Knoten eines datenflußbasierten Analysegraphen an verteilte Ausführungsressourcen, wird, obgleich der vorgestellte Ansatz durchaus entsprechende Grundlagen bereitstellt, nicht näher betrachtet.

1.2 Das Anwendungsgebiet der Epidemiologie

Die Epidemiologie hat vor allem etwa seit Mitte dieses Jahrhunderts mit ihrer Konzentration auf nicht–infektiöse Krankheiten sowie durch die Entwicklung ausgefeilter Methoden einen Beitrag zur Lösung von medizinischen Problemen geliefert, die auf anderem Wege vorher nicht zugänglich schienen [Pfl73]. Inzwischen ist sie als die Basisdisziplin der Gesundheitswissenschaften anzusehen und bildet gewissermaßen das Bindeglied zwischen Ursachenforschung und öffentlichem Gesundheitswesen [KS95].

Im folgenden wird ein kurzer Einblick in grundlegende Begriffe und Vorgehensweisen der Epidemiologie gegeben, um eine Begriffswelt für ein konkretes Anwendungsszenario zu definieren, das im weiteren Verlauf zur Illustration der entwickelten Konzepte herangezogen werden soll. Weiterhin werden spezielle Anforderungen an diese Arbeit verdeutlicht, die sich im Hinblick auf einen praktischen Einsatz des entwickelten Systems ergeben. Eine gute Einführung in das Themengebiet der Epidemiologie findet sich in [KS95] oder — speziell für die deskriptive (Krebs–)Epidemiologie — in [EBR94].

1.2.1 Definition der Epidemiologie

Die klassische Definition der Epidemiologie stammt von MacMahon und Pugh [MP70]:

> *Epidemiology is the study of the distribution and determinants of disease frequency in men.*

Ihr charakteristischer Ansatz zur Lösung medizinischer Probleme ist also die Loslösung von der Betrachtung und Behandlung von Individuen in der Medizin hin zum Vergleich von Krankheitshäufigkeiten in unterschiedlichen Bevölkerungsgruppen. Hierbei verzichtet sie im allgemeinen — im Gegensatz zu anderen medizinischen Teildisziplinen — auf ein experimentelles Vorgehen und beschränkt sich auf die Beobachtung ausgewählter Gruppen von Individuen.

Eine moderne Definition, die das Zusammenwirken unterschiedlicher Fachgebiete (neben Medizin und Biologie etwa Sozialwissenschaften, Statistik, Biometrie oder Geographie) in der Epidemiologie betont, geben Hellmeier, Brand und Laaser in [HBL93]:

> *Epidemiologie ist die Bearbeitung von Fragen aus dem Bereich der Medizin, der Gesundheitssystemforschung und der Gesundheitswissenschaften mit Methoden der empirischen Sozialforschung und der Statistik.*

Die Epidemiologie entwickelt und definiert Größen zur Beschreibung gesundheitlicher Sachverhalte sowie methodische Ansätze zur Untersuchung gesundheitlicher Fragestellungen. Die Sozialwissenschaften liefern ihr Werkzeuge zur Durchführung von Studien; die Statistik hilft bei der Präsentation des Datenmaterials, beim Umgang mit Unsicherheiten und zufälligen Variationen, bei der Kontrolle von Störgrößen, der Feststellung von Assoziationen zwischen verschiedenen Faktoren sowie der Beurteilung der Aussagekraft ihrer Ergebnisse.

Man unterscheidet grob zwei Arbeitsgebiete der Epidemiologie: die deskriptive und die analytische Epidemiologie, die voneinander hinsichtlich ihrer Rolle beim Aufstellen und Überprüfen von Hypothesen zum Entstehen von Krankheiten abgegrenzt werden können.

Die *deskriptive Epidemiologie* beschreibt Zustand und Entwicklung der Verteilung und Häufigkeit von Krankheiten in einer *Studienpopulation*, die typischerweise (unter anderem) durch ein Beobachtungsgebiet

definiert ist. Sie stellt Eigenschaften der untersuchten Bevölkerungsgruppen dar und macht Angaben zum Vorliegen und zum Ausmaß von mit dem Auftreten der Krankheit möglicherweise in Zusammenhang stehenden Einflußfaktoren, sogenannten *Expositionen*. Indem die deskriptive Epidemiologie einen Überblick über die Bedeutung von Krankheiten verschafft, das Krankheitsgeschehen im Hinblick auf evtl. noch unbekannte *Risikofaktoren* überwacht und auf Veränderungen aufmerksam macht, bildet sie die Basis für Abschätzungen der Anforderungen an das Gesundheitssystem sowie für die Generierung neuer Hypothesen zur Krankheitsentstehung.

Studien der deskriptiven Epidemiologie (auch *Korrelationsstudien* genannt), wie etwa ökologische oder Querschnittstudien, verfolgen einen explorativen Ansatz. In einer *ökologischen Studie* stehen nur aggregierte Daten bzgl. Krankheitshäufigkeiten und Expositionen in verschiedenen Gruppen (z. B. administrativen Einheiten wie Landkreisen, Gemeinden etc.) zur Verfügung. Somit sind keine Aussagen über Individuen möglich; lediglich auf aggregierter (ökologischer) Basis werden Zusammenhänge gesucht, untersucht und dargestellt. *Querschnittstudien* erheben dagegen zu einem festen Zeitpunkt für alle Individuen der Studienpopulation sowohl Gesundheit als auch das Vorliegen bestimmter Einflußfaktoren. Auch hier sind keine ursächlichen Abhängigkeiten ableitbar, da vor allem keine Aussage über die zeitliche Abfolge von Exposition und Erkrankung vorliegt und zusätzliche, evtl. das Ergebnis störende Einflußfaktoren nicht berücksichtigt werden.

Die *analytische Epidemiologie* versucht, aufgrund gezielter Hypothesen die *Ätiologie* von Krankheiten, also den Einfluß bestimmter Einflußfaktoren (Expositionen) auf deren Entstehung und Verlauf, zu erforschen. Hierzu vergleicht sie — in vielen Fällen zwei — ausgesuchte Bevölkerungsgruppen bzgl. der jeweiligen Krankheitshäufigkeiten einerseits sowie ihrer Wohn- und Arbeitsumgebung, ihrer Lebensweise, besonderen Belastungen und (z. B. genetischen oder infektionsbedingten) Dispositionen andererseits. Somit dient sie der Identifikation von Lebensumständen, die das Auftreten bestimmter Krankheiten begünstigen bzw. mit diesem verknüpft sind; Risikofaktoren werden identifiziert und quantifiziert. Die betrachtete Studienpopulation ist jeweils repräsentativ für diejenigen Gruppen der Gesamtbevölkerung auszuwählen, über die bestimmte Aussagen getroffen werden sollen. Beispiele für Studientypen in der analytischen Epidemiologie bilden *Fall–Kontroll–Studien*, die meist retrospektiv vorliegende Expositionen von Erkrankten (*Fällen*) und Gesunden (*Kontrollen*) miteinander vergleichen, oder *Kohortenstudien*, die demgegenüber von zwei Personengruppen (Kohorten) ausgehen, von denen eine bzgl. eines bestimmten Einflußfaktors exponiert ist und die andere nicht.

1.2.2 Datenanalyse in der deskriptiven Epidemiologie

Im Kontext dieser Arbeit ist speziell die Sichtweise ökologischer Studien der deskriptiven Epidemiologie von Interesse, die vollständig von der Betrachtung des Individuums abstrahiert. Zu verschiedenen *Populationen* werden eine Reihe von Maßzahlen errechnet, die vor allem *Inzidenz* (die Menge an Neuerkrankungen), *Prävalenz* (den Krankenbestand) und *Mortalität* (das Ausmaß von Todesfällen) sowie daraus abgeleitete Eigenschaften der betrachteten Bevölkerungsgruppen beschreiben. Ein typischer Anwendungsfall für derartige Betrachtungen sind *Krankheits-*, speziell *Krebsregister*, in denen zu einer jeweils betrachteten Population Daten über Erkrankungsfälle gesammelt, miteinander abgeglichen und ausgewertet werden.

Im wesentlichen sind folgende Arten von Kenngrößen (Maßzahlen) und Analysetypen zu unterscheiden:

- *Raten*, das sind auf die jeweilige Bevölkerungsgröße und einen bestimmten Beobachtungszeitraum bezogene *Fallzahlen*, und *Risiken*, das sind zu einem erwarteten Vergleichswert in Beziehung gesetzte Raten, bilden die Grundlage der meisten Betrachtungen. Eine wichtige Rolle spielen hierbei verschiedene Varianten der *Standardisierung*, durch die Maßzahlen zu Populationen, die im Hinblick auf bekannte Einflußfaktoren unterschiedliche Charakteristika aufweisen, miteinander vergleichbar gemacht werden. So werden z. B. in *direkt standardisierten Raten* durch geeignete Gewichtungen von *Teilpopulationen* auf Basis einer als Bezugsgröße verwendeten *Standardpopulation* Unterschiede hinsichtlich Altersstruktur oder Verteilung der Geschlechter „herausgerechnet“ (vgl. [Rot86]).

- Raumbezogene *Clusteranalysen* versuchen außergewöhnliche Verteilungen von Krankheitshäufigkeiten über Teilregionen eines Studiengebiets sowie Bereiche mit besonders hohen oder niedrigen Erkrankungs-

raten zu identifizieren. Neben verschiedenen Varianten der graphischen Darstellung der räumlichen Verteilung interessierender Maßzahlen in *thematischen Karten* [BMG89] existieren hierzu eine Vielzahl von *Clusterindizes*, die bestimmte Eigenschaften der räumlichen Maßzahlverteilung durch eine Kenngröße beschreiben und statistisch im Hinblick auf Signifikanz beurteilen [AB96, BG96]. Die raumbezogene Analyse erfreut sich aufgrund vielfältiger Möglichkeiten zur Korrelation der Analyseergebnisse mit raumbezogenen Hintergrunddaten großer Beliebtheit, ist jedoch in ihrer Aussagekraft nicht unumstritten [Rot90].

- *Trendanalysen* betrachten die zeitliche Entwicklung verschiedener Maßzahlen, allem voran die stetige Zu- oder Abnahme oder signifikant herausragende Zeitabschnitte (siehe etwa [CED$^+$93]).

- *Überlebenszeitanalysen* klassifizieren Fälle nach der Länge der Zeitperiode zwischen Erstdiagnose und Tod. In die Analyse der prozentualen Überlebensraten einer Ausgangspopulation fließen auch Vergleichsdaten aus allgemeinen Sterbetafeln ein [BSV$^+$95, PH91].

- Während die bisher dargestellten Größen und Verfahren im wesentlichen auf Fall- und Bevölkerungszahlen als Ausgangsdaten basieren, werden in Korrelationsstudien oftmals auch krankheitsunabhängige, häufig raum- und zeitabhängige oder auf andere Attribute des Datenbestandes bezogene Hintergrunddaten mit den Erkrankungsdaten z. B. in *Regressionsanalysen* in Beziehung gesetzt. Diese versuchen, Abhängigkeiten zwischen zwei oder mehr Variablen durch einfache Funktionen zu beschreiben. Aus den Ergebnissen derartiger Betrachtungen können Hypothesen zur näheren Untersuchung in speziellen analytischen Studien gewonnen werden.

- Gerade im Kontext der Registrierung von Erkrankungen spielen Indikatoren zur *Qualitätssicherung* im Register eine nicht unerhebliche Rolle. Ergänzend zur Heranziehung externer Vergleichsdaten, die eine Beurteilung der Vollständigkeit des Datenbestandes ermöglichen, werden hierbei auch die Verteilung der Ausprägungen spezieller medizinischer Attribute sowie das jeweilige Spektrum der Beteiligung unterschiedlicher Meldergruppen betrachtet [PCF$^+$94]. Dies ermöglicht zusätzlich Aussagen über die Validität der vorliegenden Angaben.

- Schließlich werden zu vielen der genannten Maßzahlen *Konfidenzintervalle* und *Signifikanzniveaus* berechnet. Etwas vereinfacht dargestellt (für eine exakte Definition siehe etwa [HEK91]) beschreiben Konfidenzintervalle auf der Basis eines unterliegenden statistischen Modells und einer gegebenen Irrtumswahrscheinlichkeit den Bereich rein „zufälliger“ Schwankungen um den jeweils beobachteten Wert. Liegt also ein zugehöriger erwarteter Wert innerhalb dieses Intervalls, kann nicht von einer signifikanten Abweichung gesprochen werden. Analog gibt ein Signifikanzniveau die Irrtumswahrscheinlichkeit eines entsprechenden statistischen Tests an, mit der eine beobachtete Differenz zum Erwartungswert als signifikant bezeichnet werden kann.

Im wesentlichen orthogonal hierzu sind zumindest drei Analysekontexte zu unterscheiden, in denen die genannten Methoden genutzt werden:

- In *Ad–hoc–Anfragen* an den Datenbestand werden zum einen abgegrenzte, einzelne Fragestellungen, z. B. resultierend aus Anfragen aus dem Gesundheitswesen, der Politik oder der allgemeinen Öffentlichkeit, untersucht, zum anderen aber auch im typischen Sinne explorativer Datenanalysen bisher verborgene Zusammenhänge und Strukturen gesucht. Hierzu werden zum Datenbestand ermittelte Maßzahlen bzw. Ergebnisse von Analyseverfahren auf unterschiedliche Weise in Tabellen, Diagrammen, Graphiken und thematischen Karten visualisiert und einander gegenübergestellt.

- Eine wesentliche Aufgabe der Epidemiologie, speziell von Krankheitsregistern, besteht darin, frühzeitig auf signifikante Änderungen und Auffälligkeiten im Gesundheitszustand der Bevölkerung hinzuweisen. Hierzu wird im Rahmen eines (*Inzidenz–* oder auch *Mortalitäts–*) *Monitoring* [SHDA90] kontinuierlich auf Basis festgelegter Auswertungsfolgen der jeweilige Datenbestand nach signifikanten Unregelmäßigkeiten durchsucht bzw. mit jedem neu aufgetretenen Fall überprüft, ob bestimmte

Häufungskriterien erfüllt sind (vgl. etwa [Rog97, CMCI82]). Dieses Vorgehen muß die Problematik des *multiplen Testens*[3] [HV95] berücksichtigen und einhergehen mit einer sorgfältigen und verantwortungsbewußten Verifikation der Auswertungsergebnisse. Eine vollständige Automatisierung — etwa unter Nutzung von Konzepten *aktiver Datenbanksysteme* [ABJK94] — wird i. a. aufgrund mangelnder Flexibilität sowie unklarer Aussagekraft abgelehnt.

- Im Rahmen einer umfassenden *Gesundheitsberichterstattung*, die die Basis für die Kommunikation der Ergebnisse epidemiologischer Forschung bildet [SW89], werden Gruppen und Sequenzen von Auswertungen routinemäßig für verschiedene Parametrisierungen (etwa für eine Reihe von Erkrankungsgruppen) durchgeführt. Ein typisches Beispiel bilden hier etwa Krebsatlanten sowie Jahresberichte aus der Krebsregistrierung (z. B. [Saa96, BW97, Ham95, PHR+94, HW97b]). Auch wenn ein allgemeines einheitliches Schema zugrunde liegt, ist doch durch spezielle Analysen oder Variationen auf Besonderheiten einzelner Szenarien einzugehen.[4]

Tabelle 1.1 stellt diese Kontexte den oben angeführten Maßen und Verfahren gegenüber. Während in explorativen Ad–hoc–Analysen prinzipiell alle Arten von Auswertungen durchgeführt werden, stehen neben der direkten Überwachung von Raten und Risiken im Monitoring Cluster- und Trendanalysen bzw. bei der Berichterstellung Trend- und Überlebenszeitanalysen[5] sowie Korrelationsstudien im Vordergrund.

	Ad–hoc–Anfragen	Monitoring	Gesundheitsberichterstattung
Raten, Risiken etc.	•	•	•
Clusteranalysen	•	•	○
Trendanalysen	•	•	•
Überlebenszeitanalysen	•	○	•
Korrelationsstudien	•	○	•
Qualitätssicherung	•	○	○

Tabelle 1.1: Nutzung epidemiologischer Maßzahlen und Verfahren in verschiedenen Analysekontexten (• – voll zutreffend, ○ – eingeschränkt zutreffend)

1.2.3 Anforderungen an diese Arbeit

Es folgen einige zentrale Anforderungen, die sich — über die allgemeinen Überlegungen aus Abschnitt 1.1 hinaus bzw. diese konkretisierend — aus den typischen Arbeitsvorgängen der Datenanalyse in der deskriptiven Epidemiologie an das in dieser Arbeit konzipierte System *VIOLA* ergeben:

- Die Basisdaten bilden — zumeist relativ einfache — Maßzahlen zu Teilen von Studienpopulationen, die nach einer Vielzahl von Kriterien klassifiziert sind. Wichtige Merkmale sind Wohnort, Erkrankungszeitpunkt, Alter, Geschlecht und *Diagnose* (Art der Erkrankung, oft codiert nach der *International Classification of Diseases (ICD)*). Im Rahmen explorativer Analysen müssen an diese Attribute geknüpfte Hintergrund- und Vergleichsdaten, z. B. raum- oder zeitbezogene Schadstoffbelastungen, einfach und flexibel mit den analysierten Daten korreliert werden können.

[3] Die Irrtumswahrscheinlichkeit (das Signifikanzniveau) eines statistischen Tests ist i. a. nur für eine einfache Durchführung wohldefiniert und steigt bei multipler Durchführung je nach Unabhängigkeit der Testfälle an; entsprechend sind hier Korrekturen vorzunehmen.

[4] In jedem Fall ist natürlich eine — letztendlich nicht automatisierbare — textuelle Interpretation der Ergebnisse unabdingbar.

[5] Die umfassende Darstellung von Ergebnissen raumbezogener Analysen — insbesondere die Nutzung thematischer Karten — wird hier kontrovers diskutiert. Ein Vorbehalt besteht im Hinblick auf die leicht überzuinterpretierende Signalwirkung derartiger Darstellungen.

- Wie es der interdisziplinäre Charakter der Epidemiologie nahelegt, sollen verschiedene Benutzergruppen mit unterschiedlichen Kenntnisständen im Hinblick auf Datenanalyse und Epidemiologie unterstützt werden. Insbesondere hat der typische Nutzer keine vertieften Programmierkenntnisse, verfügt zwar zumindest über Grundkenntnisse der Epidemiologie, ist aber nicht notwendigerweise Experte der statistischen Datenanalyse. Wichtig sind somit intuitive Bedienbarkeit, flexible Hilfestellung und unterschiedliche Benutzungsschnittstellen für verschiedene Benutzergruppen.

- Für komplexe Sequenzen von Auswertungen, wie sie für Berichterstellung und Inzidenzmonitoring typisch sind, sind unterstützende Techniken vorzusehen, die eine Strukturierung der Abläufe ermöglichen und einen Überblick über diese gewähren. Im Rahmen des Monitoring ist eine geeignete Teilautomatisierung zu erreichen, die die vorgenommenen Auswertungsfolgen für den Benutzer transparent und flexibel modifizierbar macht. Die explorative (zeitnahe, aber retrospektive) Analyse unter Benutzerkontrolle steht jedoch stets im Vordergrund. Es soll kein System erstellt werden, das in ständigem Betrieb automatisch für jeden neu in die Datenbasis aufgenommenen Fall auf auffällige Häufungen prüft (wie es z. B. das typische Szenario der Anwendung kumulativer Summenstatistiken [Rog97] oder oben erwähnter aktiver Datenbanksysteme ist).

- Die Größe typischerweise auszuwertender Datenbasen[6] ergibt sich im wesentlichen durch Falldatenbestände von bis zu einigen Millionen Fällen, die nach bis zu etwa 50 verschiedenen Merkmalen klassifiziert sind, sowie den zugehörigen Bevölkerungsdaten mit unter zehn Merkmalen. In einzelnen Auswertungen sind meist jedoch nach Einschränkung des Datenbestandes nur jeweils wenige (oft zwei oder drei, i. a. nicht mehr als fünf) klassifizierende Merkmale gleichzeitig von Interesse. Zur Auswertung herangezogene Hintergrundinformationen haben eine noch geringere Dimensionalität. Die betrachteten Attribute haben einige wenige (Geschlecht) bis einige tausend (Art der Erkrankung, Wohnort etc.) Ausprägungen, die typischerweise auf bis zu 10 bis 20 Aggregierungsebenen betrachtet werden. Insgesamt kann erfahrungsgemäß davon ausgegangen werden, daß ein Ausschnitt der Basisdaten, der für eine bestimmte Untersuchung (z. B. die Betrachtung alters- und geschlechtsstandardisierter Raten in ihrer räumlichen Verteilung) herangezogen wird, in einem Datenwürfel mit höchstens wenigen Millionen, meistens aber deutlich weniger Zellen bereitgestellt werden kann. Gerade die „großen“ Dimensionen (also solche mit vielen möglichen Ausprägungen) werden entweder nur zusammen mit „kleinen“ Dimensionen betrachtet oder auf relativ groben Aggregierungsebenen miteinander verknüpft.

Wir werden im Laufe der Arbeit, speziell natürlich bei der Diskussion des konkreten Anwendungsbeispiels in Abschnitt 6.3, aber auch schon als Basis für einzelne Entwurfsentscheidungen, mehrfach auf diese Anforderungen zurückkommen sowie einige Teilaspekte noch näher beleuchten und differenzieren.

1.3 Überblick über die Arbeit

Kapitel 2 stellt zunächst einige grundlegende Begriffe und Konzepte aus den Bereichen Datenanalyse, visuelle Programmierung, multidimensionale Datenmodellierung sowie zur Nutzung von Metadaten im Rahmen einer umfassenden Daten- und Datenanalysebeschreibung vor. Anschließend geht Kapitel 3 auf bestehende Klassen von Datenanalysesoftware sowie einige Beispielsysteme ein, um daraus zentrale Anforderungen an *VIOLA* abzuleiten. In Kapitel 4 wird *MADEIRA* als grundlegendes Datenmodell für die Analyseumgebung *VIOLA* entworfen, deren Konzeption dann Gegenstand von Kapitel 5 ist. Hierbei werden auch Aspekte der Datenvisualisierung, der datenflußbasierten Anfrageverarbeitung sowie der Gestaltung der Benutzungsoberfläche eines adäquaten graphischen Editors angesprochen. Schließlich werden in Kapitel 6 einige Implementierungsdetails, etwa die Idee einer offenen Client–Server–Architektur, spezifische Benutzungsschnittstellen und Ansätze zur Optimierung der Verarbeitung datenflußbasierter Analysen, sowie — als konkretes Anwendungsbeispiel —

[6] Als Richtlinie sollen die oben angeführten Erkrankungsregister bzw. entsprechende Registerberichte dienen. Andere epidemiologische Untersuchungen betrachten eher kleinere Datenbestände.

die Anwendung von *VIOLA* im Kontext des Niedersächsischen Krebsregisters betrachtet. Eine Zusammenfassung und ein Ausblick auf noch ausstehende, auf *VIOLA* aufbauende Arbeiten im Bereich der intelligenten Datenanalyse, auf die Nutzung der entwickelten Konzepte in der Krebsepidemiologie und anderen Anwendungsdomänen sowie auf mögliche Systemerweiterungen runden die Arbeit in Kapitel 7 ab.

Kapitel 2

Grundlagen

Ziel dieser Arbeit ist die Entwicklung eines Datenanalysesystems zur Verarbeitung multidimensionaler Daten mit Mitteln der datenflußbasierten visuellen Programmierung. Entsprechend bilden die *Datenanalyse*, die *multidimensionale Datenmodellierung* und die *visuelle Programmierung* drei zentrale Eckpfeiler, deren geeignete Integration und Verknüpfung im Zentrum der Betrachtungen steht. Als Mittel der Integration (und somit vierte Säule) wird eine detaillierte formale Modellierung und Beschreibung aller relevanten Strukturen und Operationen in Form von *Metadaten* dienen. Im folgenden werden jeweils grundlegende Begriffe, Anforderungen und Konzepte der genannten Domänen vorgestellt, um eine breite Basis für die Grundideen der in den Kapiteln 4 und 5 konzipierten Modelle zu legen. Es wird insbesondere herausgearbeitet,

- was den Prozeß *intelligenter* Datenanalyse ausmacht und aus welchen einzelnen Teilschritten er im wesentlichen besteht,
- wie visuelle Programmierumgebungen charakterisiert werden können und warum der Ansatz der Datenflußprogrammierung für die Anwendung in der Datenanalyse besonders sinnvoll ist,
- welche Gestalt multidimensionale Daten haben, welche Operationen auf ihnen definiert sind und wie beides — als Grundlage einer Analyseumgebung — auf syntaktischer und semantischer Ebene formal modelliert werden kann sowie
- welche besondere Rolle Metadaten im Rahmen von Datenanalysen einnehmen.

Die beiden letztgenannten Aspekte werden in Kapitel 4 als Grundlage von Entwurf und Einordnung des Datenmodells *MADEIRA* dienen, während die beiden erstgenannten Themenkreise vor allem zur Konzeption und Bewertung der Analyseumgebung *VIOLA* in Kapitel 5 herangezogen werden sollen.

Im Kern geht es in dieser Arbeit um die *bessere* Unterstützung von Datenanalysen, d. h. um die komfortable Datenexploration in einer integrierten, rechnerbasierten Analyseumgebung. Aus diesem Grund soll hier als erstes in Abschnitt 2.1 auf deren Eigenheiten und spezielle Anforderungen eingegangen werden, bevor Abschnitt 2.2 die visuelle Programmierung als ein Werkzeug zu ihrer Durchführung vorstellt. Anschließend werden dann in Abschnitt 2.3 die Domäne multidimensionaler Daten als ein spezielles Anwendungsgebiet der Datenanalyse präsentiert und in Abschnitt 2.4 Arten und Bedeutung von Metadaten in Datenverwaltung und -analyse behandelt.

2.1 Datenanalyse

Rohe Daten allein sind wertlos; erst durch die (korrekte) Interpretation von Daten werden aus ihnen Informationen, die dem Verständnis für die Charakteristika der jeweils betrachteten Datenquelle dienen können. Aufgabe und Ziel der *Datenanalyse* besteht nun gerade darin, diese Ableitung von Informationen bzw. — durch deren weitergehende Verarbeitung — von Wissen aus Daten durch geeignete Verfahren und Vorgehensweisen zu un-

terstützen bzw. erst zu ermöglichen [FPSS96a]. Eigenschaften eines Datenbestandes lassen sich allgemein in Form von Abhängigkeiten oder Mustern beschreiben. So formuliert Hand in [Han97b]:

Finding patterns is what data analysis is all about.

Wenn auch ihr allgemeines Ziel mit diesen Worten recht einfach zu spezifizieren ist, so ist die Aktivität der Datenanalyse selbst doch hochkomplex: In einem oft iterativen, stark verzweigten und zyklischen *suchenden* Prozeß werden in vielen kleinen Schritten jeweils unterschiedliche Verfahren zur Ableitung neuer Daten ausgewählt und auf den Datenbestand oder bereits vorliegende Zwischenergebnisse angewendet, um die zugrundeliegenden Basisdaten besser zu verstehen [YL95, LP88].

Als eigenständige wissenschaftliche Disziplin ist die Datenanalyse noch relativ jung. Sie entstammt vor allem der Statistik, erhält aber zunehmend auch aus Bereichen der Informatik, wie dem Maschinellen Lernen, der Mustererkennung, der Künstlichen Intelligenz (KI) oder aus der Informationsvisualisierung neue Impulse [Han97b]. Mitunter werden die Bezeichnungen „Statistik“ und „Datenanalyse“ auch synonym verwendet (vgl. etwa [Hub94]). In den meisten wissenschaftlichen Veröffentlichungen sowie auch im allgemeinen Sprachgebrauch steht jedoch die Statistik für eine wissenschaftliche Fachdisziplin, die eine Reihe von Analysetechniken definiert und bereitstellt, während die Datenanalyse deren praktische Anwendung in einem komplexen Datenverarbeitungsprozeß betrachtet [Def89]. In diesem Sinne sollen die Begriffe auch in dieser Arbeit verstanden werden.

Vor allem aufgrund von Anforderungen betriebswirtschaftlicher und sozioökonomischer Anwendungen ist die Datenanalyse in den letzten Jahren zunehmend in das öffentliche, aber auch das wissenschaftliche Interesse gerückt. Als zentrale Ursachen hierfür sind drei eng miteinander verknüpfte Aspekte zu nennen (vgl. [Han97b] oder [DF95]):

1. Die Verfügbarkeit immer größerer Speicherkapazitäten zu immer geringeren Kosten führt zu einer explosionsartig steigenden Flut an gespeicherten *Daten*, die zudem durch den stetigen Ausbau der Vernetzung von Informationssystemen in lokalen Netzen oder über das rasant wachsende Internet für immer mehr Nutzer zugreifbar sind und nach einer systematischen und effizienten Auswertung verlangen.
2. Die breite Verfügbarkeit immer leistungsfähigerer Hard- und Software zur Datenanalyse geht — gerade auch im Hinblick auf die interaktive und visuell unterstützte Datenexploration — einher mit der Verfolgung neuer, komplexer *Fragestellungen*.
3. Zur Bearbeitung dieser neuen Probleme werden neue Verfahren und *Modelle* entwickelt und eingesetzt, die in großem Umfang auch auf Konzepte der Informatik, etwa der Künstlichen Intelligenz, zurückgreifen und eine effiziente Informationsextraktion ermöglichen.

Im folgenden werden zunächst in Abschnitt 2.1.1 einige der Statistik sowie der Datenanalyse untergeordnete bzw. nahestehende Forschungsbereiche und Anwendungsdomänen vorgestellt sowie entsprechende Begriffe definiert. Im Anschluß betrachtet Abschnitt 2.1.2 die Untergliederungen des Datenanalyseprozesses in einzelne Komponenten, die als Interaktionen zwischen menschlichem Analysten und rechnergestütztem Datenanalysesystem modelliert werden sollen. Als herausragendes Medium zur Interaktion im Rahmen des Prozesses der Datenexploration wird die Visualisierung, insbesondere die interaktive Graphik, in Abschnitt 2.1.3 näher vorgestellt. Schließlich werden in Abschnitt 2.1.4 Anforderungen an die *intelligente* Datenanalyse motiviert, worunter hier im wesentlichen die reibungslose Kommunikation und effektive Kooperation zwischen Analyseumgebung und Anwender verstanden werden soll.

2.1.1 Begriffsdefinitionen und Abgrenzung benachbarter Domänen

Die *Statistik* ist zweifellos die „Mutter“ der Datenanalyse. Sie befaßt sich mit der Analyse von Beobachtungen, die unter dem Einfluß des Zufalls oder in derartig komplexen Situationen, daß die Angabe kausaler Abhängigkeiten praktisch nicht möglich ist, entstanden sind [LW92, HEK91]. Sie dient der Auswertung von

durch Experiment oder Erhebung gewonnenen Daten und bildet somit ein Werkzeug aller empirisch arbeitenden Wissenschaften. Allgemein werden zwei grundlegende Zweige der Statistik unterschieden: Aufgabe der *beschreibenden* (oder *deskriptiven*) *Statistik* ist die Aufbereitung des Datenmaterials in Form von graphischen Darstellungen und Tabellen sowie durch Berechnung empirischer Kenngrößen, wie Mittelwert, Median oder Varianz von Meßreihen. Die *schließende* (auch *konfirmatorische* oder *induktive*) *Statistik* dagegen behandelt — ausgehend von Modellen der Wahrscheinlichkeitsrechnung und basierend auf Maßzahlen der deskriptiven Statistik[1] — die Analyse und Beurteilung der Daten sowie die Ermittlung der jeweils zugrundeliegenden, durch Wahrscheinlichkeitsverteilungen beschreibbaren Zufallsgesetze mittels Schätz- und Testverfahren. Sie zieht Schlüsse und bewertet im Vorfeld aufgestellte Hypothesen, indem sie quantitative Aussagen über Unsicherheiten trifft und Fehlerwahrscheinlichkeiten bestimmt.

Explorative Datenanalyse

Die *explorative Datenanalyse (EDA)* erweitert die Palette deskriptiver Verfahren — teilweise auch unter Einbeziehung von Modellen der schließenden Statistik — um Methoden und Vorgehensweisen zur interaktiven Extraktion relevanter Informationen aus oft großen Datenmengen. Hierbei werden die kognitiven Fähigkeiten des Menschen in vielerlei Hinsicht stärker einbezogen, als es bei anderen Ansätzen zur Datenanalyse in der Regel der Fall ist. Populär geworden ist die EDA als Zweig der Statistik vor allem durch Arbeiten von Tukey in den 70er Jahren [Tuk77]. Er versteht unter explorativer Datenanalyse

> *[. . .] looking at data to see what it seems to say [. . .].*

Die konfirmatorische Datenanalyse erfordert die von den Daten unabhängige Formulierung von Hypothesen sowie eine die Anforderungen des aufgestellten Modells erfüllende Datenerhebung, um bereits vorhandene Vermutungen zu bestätigen oder zu verwerfen. Demgegenüber stehen bei der EDA die Daten selbst im Mittelpunkt der Analyse. Ihr Ziel ist es, durch effektive Betrachtung oftmals großer Datenmengen unter verschiedenen Blickwinkeln „Neues zu entdecken“, d. h. neue Fragestellungen und Hypothesen abzuleiten sowie neue Erkenntnisse zu gewinnen. Während durch die Einschränkung auf einzelne Hypothesen bzw. Aspekte der Daten evtl. wichtige Informationen übersehen werden können, findet die explorative Datenanalyse auch unerwartete Zusammenhänge, Abhängigkeiten, Besonderheiten und globale Strukturen, die charakteristische Eigenschaften der betrachteten Objektmengen widerspiegeln und häufig durch einzelne Maßzahlen nicht beschrieben werden können (vgl. auch [Hin87, Boc92]).

Die EDA stellt gegenüber der schließenden Statistik weniger strenge Anforderungen an die untersuchten Daten, die zudem oft, wie z. B. spezielle Verteilungsannahmen oder die Unabhängigkeit von Variablen, gar nicht überprüfbar wären. Somit dient sie gerade zur Untersuchung von Daten, die nicht im Hinblick auf vorgegebene Hypothesen und die Anwendbarkeitsbedingungen spezieller Tests erhoben wurden. Darstellungen der Daten und Analyseergebnisse sollen beim Analysten Substanzwissen um Zusammenhänge abrufen und das weitere Vorgehen dynamisch beeinflussen. Anwendung von Analyseverfahren und Ergebnisinterpretation wechseln einander ab, wobei auch im Weg zu einem bestimmten Resultat wichtige Informationen enthalten sind.

Zum einen definiert die EDA eine Vielzahl neuer, häufig relativ einfacher Verfahren zur Datenbeschreibung und -analyse, die jeweils unterschiedliche Eigenschaften eines untersuchten Datensatzes hervorheben (für einen grundlegenden Überblick siehe etwa [Boc92, Tuk77]). Hierbei sind vor allem zu nennen

- Verfahren zur *graphischen Datenanalyse* (zur *Datenvisualisierung*), also Darstellungsmethoden, die Daten leichter und effizienter erfaßbar machen und eine ausgezeichnete Möglichkeit bieten, den menschlichen Anwender interaktiv in den Analyseprozeß einzubeziehen (vgl. [dTSS86] und Abschnitt 2.1.3), sowie

[1] Zum einen werden zu untersuchende Hypothesen aus Resultaten der Datendeskription abgelegt, zum anderen dienen deskriptive Kenngrößen der Hypothesenprüfung.

- *robuste Verfahren* (vgl. [HEK91]), die eine weitgehende Unabhängigkeit von Modellvoraussetzungen gewähren.

Zum anderen und vor allem aber beschreibt die explorative Datenanalyse eine neuartige Art des Umgangs mit den Daten [Bor92, AC98]. Dieser vollzieht sich in einem interaktiven Prozeß auf der Suche nach Strukturen und passenden Modellen. Abbildung 2.1 skizziert dieses Konzept der EDA.[2] Insbesondere die Reihenfolge von Datenerhebung bzw. -betrachtung und Festlegung einer Fragestellung, das Aufbauen einer Erwartungshaltung sowie die Betonung des iterativen Vorgangs unterscheiden dieses Modell von dem der konfirmatorischen Datenanalyse. Durch geeignete Datenpräsentationen wird dem Analysten ermöglicht, verborgene Strukturen in den Daten zu entdecken. Auf diese Weise gewonnene Ideen führen zu neuen Fragestellungen und weiteren Untersuchungen der Daten, nun unter neuen Blickwinkeln — ein schrittweise verfeinernder Zyklus entsteht, wobei generierte Ergebnisse ohne die Kenntnis und Beschreibung des Weges ihrer Herleitung kaum richtig zu interpretieren sind [Hub94]. Selten gibt es *ein* bestes Verfahren, oft liefert erst die Kombination der Ergebnisse *mehrerer* Methoden einen guten Einblick in die Struktur der Daten.

Abbildung 2.1: Der interaktive Zyklus explorativer Datenanalysen (nach [Wol92])

Die Ziele explorativer Datenanalysen bestehen nach [Goo83] in

- der Präsentation von Daten in einer Art und Weise, die den menschlichen Fähigkeiten zur (insbesondere visuellen) Informationsaufnahme und -verarbeitung gerecht wird,
- der Erkennung von *interessanten* Mustern, die den betrachteten Daten innewohnende Charakteristika widerspiegeln (vgl. auch die Definition des *Knowledge Discovery in Databases* weiter unten in diesem Abschnitt),
- der Formulierung von Hypothesen zur Erklärung dieser Muster sowie
- der Verbesserung der Erklärungsfähigkeit bestehender Hypothesen.

Wie Tukey in [Tuk77] formuliert, macht hierbei die *Strategie der EDA* (vgl. [AC98]) aus

> *[. . .] to simplify description and to describe one layer deeper [. . .] to make data more easily and effectively handleable by minds [. . .].*

Die EDA will jedoch keineswegs andere, ältere Bereiche der Statistik verdrängen; für die Bearbeitung konkreter Fragestellungen ist i. a. eine Zusammenarbeit von explorativer und konfirmatorischer Analyse nötig und sinnvoll [Liu96]. Im Verlauf der Datenexploration aufgestellte Vermutungen sind mit Hilfe der schließenden Statistik in weiteren Folgestudien zu überprüfen. Tukey vergleicht die EDA mit der Arbeit eines Detektivs, der (auch zufällige oder durch Störgrößen hervorgerufene) Zusammenhänge findet und (oft auch falsche oder irreführende) Hypothesen aufstellt, und die konfirmatorische Analyse mit der Justiz, die dann die Sicherheit der Vermutungen beurteilt. Hieraus folgert er:

> *Exploratory data analysis can never be the whole story, but nothing else can serve as the foundation stone — as the first step.*

Wissensbasierte Analysesysteme

Vor allem die explorative Datenanalyse greift zur Wahl geeigneter Verfahren auf Heuristiken zurück. Hierbei fließen neben theoretischem Wissen auch viele intuitive Entscheidungen ein [HI82]. Es gibt nur selten *die*,

[2] Ähnliche Analysezyklen bilden auch die Grundlage wissenschaftlicher Datenvisualisierung, vgl. [Dye90, UFK+89].

anhand allgemeiner Regeln zu bestimmende, korrekte Methode zur Lösung eines Problems. Die konkrete Verfahrenswahl hängt ab von Anwendungsgebiet, Gestalt der Daten, feinen Nuancen der Fragestellung sowie nicht zuletzt subjektiver Einschätzung des Analysten. Mit der Anwendung immer komplexerer Modelle zur Untersuchung von Kombinationen von Einflußfaktoren in großen Datenmengen und zum Nachweis immer kleinerer Unterschiede werden zunehmend viele verschiedene Verfahren mit oft sehr speziellen Anwendbarkeitsbedingungen entwickelt. Gerade auch angesichts der Vielzahl von in Datenanalysesystemen angebotenen Methoden braucht deren Nutzer, der — aufgrund ihrer weiten Verbreitung — i. a. kein Statistik–Experte, sondern Spezialist eines Anwendungsgebiets ist, einen Überblick über die bestehenden Möglichkeiten sowie Unterstützung bei der korrekten und sinnvollen Anwendung von Auswertungsverfahren.

Ziel muß es sein, durch Integration von entsprechendem Wissen Datenanalysesoftware intelligenter und damit nützlicher zu machen [Def89]. Die Aufgabe eines wissensbasierten Beratungssystems (eines statistischen Expertensystems) muß darin bestehen, den Benutzer — unter Berücksichtigung der jeweiligen Datencharakteristika, von zusätzlichen Metadaten, Beschreibungen von Verfahren und deren Anwendbarkeit sowie von allgemeinem Wissen über die betrachtete Anwendungsdomäne — bei seiner Entscheidung für einzusetzende Techniken zu unterstützen und ihn dazu zu ermuntern, verschiedene geeignete Verfahren zur genaueren und umfassenden Analyse der Daten zu verwenden [Cha81].

Wissensbasierte Analysesysteme begleiten sowohl Methodenwahl als auch -anwendung, Ergebnispräsentation und -interpretation [CLW+90]. Sie stellen die jeweils unterliegenden Annahmen bereit, schützen vor Verfahrensmißbrauch, führen den Anwender und liefern ihm Interpretationshilfen [Han86]. Ein statistisches Expertensystem übernimmt die Rolle eines beratenden Statistikers, kann ihn jedoch in der Regel nicht vollständig ersetzen. Im Sinne eines Assistenten kann es einerseits autonome Entscheidungen zur Problemlösung treffen, also Analysewege selbständig logisch fortführen oder verfeinern, Analysevarianten ausprobieren und Vorschläge erfolgversprechender Wege machen [Ama97]. Andererseits kann der Systembenutzer alle Entscheidungen des Systems korrigieren und stets die Führungsrolle übernehmen [AC98].

Die wissensbasierte Umsetzung der explorativen Datenanalyse entspricht einer verfeinernden *Planung* von Analyseabläufen [AC96b] oder anders gesehen der *Suche* in einem Suchraum geeigneter Verfahren und Modelle bzw. möglicher Abhängigkeitsstrukturen [LP88]. Gale faßt die Aufgabe der KI in der Statistik in der Bereitstellung *statistischer Strategien* zusammen [Gal86] (vgl. auch [Cha81]). In [GP82] definiert er diese als Antwort auf die Fragen

- Was suche ich?
- Wann suche ich es?
- Wie suche ich es?
- Warum suche ich es?
- Was tue ich damit, wenn ich es gefunden habe?

In [Han86] werden Analysestrategien in ähnlicher Weise als formale Beschreibungen der Selektionen, Aktionen und Entscheidungen beim Gebrauch statistischer Methoden im Laufe einer Analyse eingeführt.

Diese Betrachtungen zur wissensbasierten Analyseunterstützung sollen im Rahmen dieser Arbeit als Grundlage für die Konzeption eines Systems zur *intelligenten* Datenanalyse (vgl. Abschnitt 2.1.4) ausreichend sein. Eine detailliertere Einführung in diese Thematik findet sich z. B. in [Haa94], erste Arbeiten auf diesem Gebiet sowie zentrale Ideen sind sehr differenziert in [Gal86] dargelegt, und beispielsweise in [Wit85] werden Entwurf, Implementierung und Anwendung eines speziellen statistischen Expertensystems sehr ausführlich beschrieben.

Entwicklung der Datenanalyse in den 90er Jahren

Lange Zeit distanzierten sich viele Statistiker von den Ideen der explorativen Datenanalyse; die „Jagd nach signifikanten Aussagen“ war verpönt, da sie dem strengen Modell des Hypothesen–Testens und der mathematisch–formalen Belegbarkeit von Analyseergebnissen zuwiderläuft [Unw94]. Zudem gab es nur wenige Tools, die den Explorationsprozeß direkt unterstützten. Übersehen wurde bei dieser Argumentation jedoch

oft, daß die praktische Anwendung statistischer Verfahren, gerade auch durch „Statistik–Laien“, in den meisten Fällen in einen explorativen Analyseprozeß eingebettet ist. Den Unterschied macht lediglich der Blickwinkel auf diesen Prozeß bzw. die Art und Weise der Nutzung und Interpretation seiner (Zwischen–)Ergebnisse aus.

Sieht man die Datenanalyse als den gesamten Prozeß von der Planung einer Studie bis zur Betrachtung und Verwendung ihrer Ergebnisse, so finden sich Ansätze und Verfahren der EDA in fast allen Stufen dieses Ablaufs [Unw94]:

- bei der Prüfung der Datenqualität,
- bei der Erkennung von Ausreißern bzw. auffälligen Einzelwerten,
- natürlich im Rahmen der Suche nach Mustern und Strukturen,
- als Beleg und Konkretisierung von Ergebnissen konfirmatorischer Analysen sowie
- zur Ergebnisillustration und -kommunikation.

Vor diesem Hintergrund werden im folgenden die Begriffe der Datenanalyse und der *explorativen* Datenanalyse synonym verwendet (wie ähnlich auch in [Hub86] motiviert). Unter dem Begriff der Statistik wird demgegenüber das Angebot der einzelnen Methoden, Modelle und Maßzahlen insbesondere aus der schließenden, aber auch der beschreibenden Statistik subsumiert (vgl. etwa [GMPS96]).

Im Sinne dieser Begriffsbildungen begannen spätestens mit Beginn der 90er Jahre die klaren Grenzen zwischen deskriptiver, explorativer und konfirmatorischer Analyse zu verschwimmen. Demgegenüber trat zum einen die Nutzung von rechnerbasierten Analysesystemen, insbesondere die Verwendung teilautomatisierter Verfahren als Ergänzung interaktiver Methoden in den Vordergrund — vgl. etwa [MT89], wo die große Bedeutung des Computers für die Zukunft der (insbesondere graphischen und interaktiven) Datenanalyse herausgestellt wird. Zum anderen kamen — wie schon eingangs dieses Kapitels dargestellt — in den letzten Jahren entscheidende Impulse aus betriebswirtschaftlichen Anwendungen der Datenanalyse. So werden in diesem Kontext die Bereiche

- *Data Warehousing* und *OLAP* (mit dem Schwerpunkt interaktiver, durch den Anwender gesteuerter Analysen) sowie
- *Knowledge Discovery in Databases (KDD)* und *Data Mining* (die sich eher auf die Automatisierung von Teilprozessen der Datenexploration konzentrieren)

unterschieden. Diese beiden Domänen werden im folgenden näher betrachtet.

On–line Analytical Processing und Data Warehousing

Der Begriff des *On–line Analytical Processing (OLAP)* wurde von Codd 1993 in [CCS93] in einem Thesenpapier über funktionale und technische Anforderungen an Datenanalysesysteme für betriebliche Entscheidungsträger eingeführt und hat sich seitdem schnell etabliert. OLAP–Tools dienen der interaktiven Ad–hoc–Auswertung von aus vielen Einzelsystemen zusammengeführten sehr großen betrieblichen Datenbeständen, sogenannten Data Warehouses. Die betrachteten Daten bilden multidimensionale Felder, werden typischerweise als Datenwürfel modelliert und können durch Zusammenfassung von Maßzahlen über Würfeldimensionen auf verschiedenen Aggregierungsebenen ausgewertet werden. So werden die Codd–Thesen zur Evaluation von OLAP–Produkten in [PC99] zum Begriff der „Fast Analysis of Shared Multidimensional Information“ zusammengefaßt (vgl. auch [Cla98] für einige entsprechende Grundideen).

Neben der Konzentration auf effiziente Verfahren zur Verwaltung und Abfrage multidimensionaler Daten in adäquaten Datenbanksystemen[3] steht auch im Rahmen von OLAP der explorative Analysecharakter im Vordergrund. Weiterhin spielen wiederum flexibel generierbare graphische Darstellungen eine zentrale Rolle bei der schnellen und effektiven Analyse vor allem durch „Statistik–Laien“. Thomsen beschreibt in [Tho97] OLAP folgendermaßen:

[3] Hierbei wird oft auch technischen Aspekten der Systemimplementierung, die über die eigentliche Datenauswertung hinausgehen, wie Mehrbenutzerfähigkeit, Client–Server–Architekturen, Transaktionsmanagement, Recovery, Persistenz etc., große Bedeutung beigemessen.

> *OLAP is the process of creating and managing multidimensional enterprise data for analysis and viewing by the user who seeks an understanding of what the data is really saying.*

Diese Definition verdeutlicht die nahe Verwandtschaft zur Idee der explorativen Datenanalyse. Der *OVUM*–Report zum Thema „OLAP“ [WKC96] betont darüber hinaus noch stärker den interaktiven Charakter von OLAP:

> *OLAP is the interactive, multidimensional analysis of business information on an enterprise scale.*

Auch wenn typische OLAP–Analysen in der Iteration ähnlicher Anfragen bzw. der Navigation durch verschiedene Dimensionen und Aggregierungsebenen der betrachteten Datenbestände bestehen, so findet sich in der Regel in OLAP–Tools doch keine explizite Repräsentation bzw. Unterstützung des Prozeßcharakters von Analysesitzungen. Dies geht teilweise so weit, daß sich jede neue Anfrage (wenn auch durch Caching–Mechanismen effizient realisiert) prinzipiell wiederum auf die Ausgangsdaten bezieht, auf denen Selektionen, Aggregationen, Maßzahlberechnungen, weitere Analysen und schließlich Visualisierungen in einem für den Benutzer nicht weiter aufzuschlüsselnden einzigen gemeinsamen Schritt vorgenommen werden.

Eine praxisnahe Einführung in die OLAP–Thematik wird etwa in [Tho97, WKC96, PC99] gegeben. In Abschnitt 2.3 wird noch genauer auf die datenbanktechnische Sichtweise auf OLAP–Systeme eingegangen werden. Was die eigentliche Datenanalyse anbelangt, definiert OLAP gegenüber der EDA keine grundlegend neuen Konzepte.

Knowledge Discovery in Databases und Data Mining

Die bessere Rechnerunterstützung und (Teil–)Automatisierung von Analyseschritten bei der Auswertung großer Datenmengen ist das zentrale Ziel des *Knowledge Discovery in Databases (KDD)* [FU+96]. Deutlich stärker als im Rahmen von OLAP wird hier der Prozeßcharakter der Datenanalyse herausgestellt, wobei gegenüber Diskussionen der EDA auch die eigentliche Auswertung vor- und nachbereitende Schritte, wie Datensammlung, -management, -selektion und -transformation sowie die Nutzung von Analyseergebnissen miteinbezogen werden. In diesem Sinne findet sich unter dem KDD–Begriff die umfassendste und allgemeinste Charakterisierung des Datenanalyseprozesses. Zwar steht hier oft die Automatisierung der Hypothesengenerierung und -validierung im Vordergrund, dies bezieht sich jedoch nur auf einzelne Abschnitte des Analysezyklus. Der gesamte Prozeß wird weiterhin interaktiv durch den menschlichen Analysten geleitet, iteriert und explorativ modifiziert. Außerdem wird auch der Skalierbarkeit des KDD große Bedeutung beigemessen. Speziell große Datenbestände werden verarbeitet, woraus sich zum einen der Wunsch nach Automatisierung und zum anderen der häufig anzutreffende Bezug zu Konzepten des Datenbankmanagements [IM96, Man96] erklärt. Auch Visualisierungsverfahren sowie die Integration von Techniken der KI, des Maschinellen Lernens und ähnlicher Domänen spielen eine zentrale Rolle (siehe etwa [KK95]).

Eine vielzitierte Definition des *Knowledge Discovery in Databases* findet sich in [FPSS96a]:

> *Knowledge discovery in databases is the non–trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*

Die Nicht–Trivialität der Analyse impliziert hierbei sowohl die explorative, iterative Suche über mehrere Einzelschritte als auch die Automatisierung von Teilen dieses Ablaufs. Gefundene Muster sollen charakteristisch für die betrachtete Datenquelle sein und auch für andere als den jeweils betrachteten Teildatensatz gelten; unerwartete Zusammenhänge, die für das Anwendungsgebiet eine gewisse Relevanz und Nutzbarkeit aufweisen, sollen gefunden und in verständlicher, einfacher Form präsentiert werden können.

Abbildung 2.2 zeigt die Zusammensetzung des KDD–Prozesses aus mehreren Teilprozessen. Hierunter sind zu nennen (vgl. auch [Wro98, BA96]):

1. das „Verstehen“ der Anwendung, die Formulierung einer Fragestellung und Zielsetzung,
2. die Datensammlung und/oder -auswahl, evtl. Testdatengenerierung bzw. -selektion,

3. die Datenbereinigung und -integration,
4. die Datentransformation in ein für die Analyse geeignetes Format, oft unter Reduktion der Datenmenge oder ihrer Attribute (dies geschieht häufig im Rahmen einer explorativen Voranalyse unter Nutzung erster Datenvisualisierungen, vgl. [WK97]),
5. der eigentliche Analyseschritt, das sogenannte *Data Mining*,
6. die Visualisierung, Bereinigung und Interpretation der Ergebnisse sowie
7. deren Nutzung und die Umsetzung gewonnener Erkenntnisse.

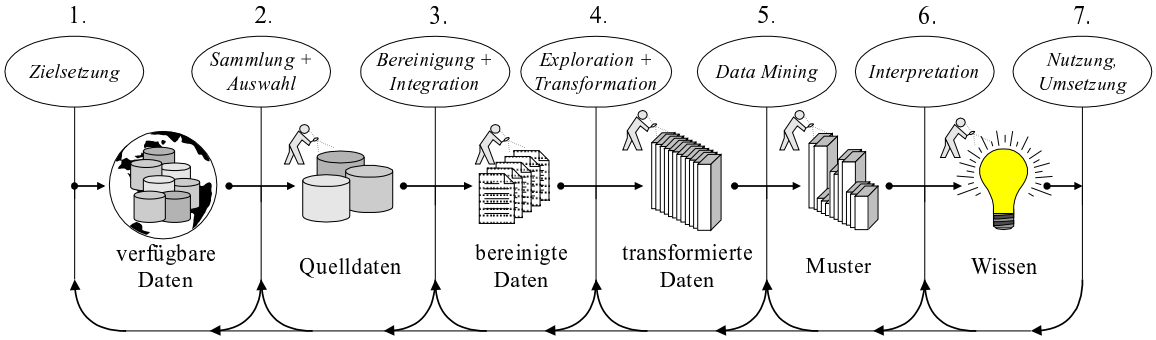

Abbildung 2.2: Überblick über den Knowledge Discovery Prozeß (nach [FPSS96b])

Alle Schritte dieses Ablaufs können — gesteuert durch Entscheidungen des Datenanalysten — iteriert und verfeinert werden, stets ist ein Rücksprung zu früheren Phasen möglich, wenn etwa neue Daten herangezogen oder neue Aspekte betrachtet werden sollen.

Einführungen und Überblicksarbeiten zum *Knowledge Discovery in Databases* finden sich u. a. in [FU+96, FPSSU96, WK97, Man96, Wro98, DF95].

Als *Data Mining* wird der eigentliche Analyseschritt im Rahmen des KDD–Prozesses bezeichnet.[4] Das Data Mining sucht und generiert aus den jeweiligen Daten neuartige Hypothesen nach vorgegebenen Mustern und bewertet diese autonom und/oder in Interaktion mit dem Experten. Eine wiederholte, verfeinerte Formulierung und Validierung präzisiert mehr und mehr die Hypothese, die schließlich zu einem Modell der gefundenen Strukturen führt. Hier wird die Abgrenzung zur schließenden Statistik besonders deutlich: Während dort gegebene Hypothesen bestätigt oder verworfen werden, gilt es hier, neue Hypothesen aufzustellen und zu bewerten.

OVUM definiert Data Mining in [WK97] wie folgt:

> *Data mining is the automated analysis of large or complex data sets in order to discover significant patterns or trends that would otherwise go unrecognised.*

Auch der Data Mining–Prozeß selbst läßt sich wiederum in drei Phasen unterteilen, die in der Regel mehrfach durchlaufen werden:

1. die Auswahl der Data Mining–Methode bzw. der Art der zu findenden Muster,
2. die Selektion der zu verwendenden Data Mining–Technik und deren Parameter sowie schließlich
3. die Durchführung der Datenanalyse.

Verschiedene Aufgaben des Data Mining sind durch die Art der jeweils betrachteten Zusammenhänge und Strukturen zu charakterisieren. In der Literatur (s. o.) werden vor allem folgende Methodengruppen genannt:

[4] Im allgemeinen Sprachgebrauch werden „Data Mining“ und „KDD“ oft auch synonym verwendet.

- Verfahren, die Datenobjekte in Klassen einteilen — hierzu gehören
  - Klassifikationsregeln zur Unterteilung einer Objektdomäne,
  - charakteristische Regeln zur zusammenfassenden Beschreibung von Teilmengen von Objekten und
  - Gruppierungen „ähnlicher“ Objekte in Cluster,
- die Beschreibung von Zusammenhängen zwischen Objektattributen, vor allem
  - Abhängigkeitsmodelle zur Darstellung von allgemeinen Abhängigkeiten zwischen Variablen,
  - Verbindungsmuster, die etwa in Form von Assoziationsregeln Abhängigkeiten zwischen Attributausprägungen in Objektgruppen spezifizieren und
  - sequentielle Analysen bei expliziten Objektfolgen,
- Vorhersagen und Regressionen sowie
- die Beschreibung von Ausnahmen, Abweichungen und Veränderungen.

Je nach Verfahrensklasse kommen verschiedene Techniken des Data Mining zum Einsatz, die jedoch oft nicht nur für eine einzige Fragestellung anwendbar sind. Häufig genannt werden z. B. Entscheidungsbäume und Regeln, (nichtlineare) Regressionen, Klassifikationsmethoden, Clusteranalyseverfahren, neuronale Netze, fallbasiertes Schließen, genetische Algorithmen, Bayes'sche Verfahren, probabilistische Netze oder Fuzzy–Logik.

Generell lassen sich die Methoden des KDD und Data Mining zwischen OLAP–Ansätzen (die selbständig kein neues Wissen entdecken) und Verfahren des Maschinellen Lernens (die vollautomatisch Konzepte erlernen und ihre Ergebnisse durch Erfahrungen verbessern, vgl. [Mit97]) einordnen [Wro98]. Gegenüber dem Maschinellen Lernen, das prinzipiell sehr ähnliche Verfahren einsetzt, betont das KDD nach [Man96] auch viel stärker

- die Skalierbarkeit seiner Anwendung,
- das Streben nach einfachen Beschreibungen bzw. Modellen und
- die Nutzung möglichst einfacher, verständlicher Algorithmen,
- die zentrale Rolle der Daten selbst im Gegensatz zur Bedeutung bestimmter übergeordneter Konzepte
- sowie die interaktive Anwendung der gewonnenen Erkenntnisse.

Aktuelle Entwicklungen, Zusammenfassung und Ausblick

In Abb. 2.3 erfolgt nochmals eine ungefähre Einordnung der hier angesprochenen Ansätze zur Analyse von Daten anhand der vier Kriterien

1. Betonung der prozeßbasierten Sichtweise gegenüber der Bereitstellung einzelner Verfahren,
2. Grad der Rechnerunterstützung bzw. der Automatisierung von Abläufen,
3. Skalierbarkeit für große Datenmengen sowie Beschäftigung mit Fragen des Datenmanagements bzw. der Datenbankanbindung und
4. Interaktivität der Analyse.

Genauer gesagt wird dargestellt, mit welcher Bedeutung die jeweiligen Termini zur Bezeichnung von Teilbereichen/Herangehensweisen der Datenanalyse in der Regel verwendet werden — die Übergänge sind stark fließend, und letztendlich subsumiert der Begriff der „Datenanalyse“ alle von ihnen. Zusätzlich werden im Falle von EDA und OLAP aktuell deutliche Entwicklungen bzgl. der beiden erstgenannten Aspekte aufgezeigt.

Wie bereits oben erwähnt, kann das KDD als umfassendste Sicht auf den Prozeß der Datenanalyse angesehen werden. Der interaktive Charakter von Analysen wird — wenn auch theoretisch postuliert — in vielen Arbeiten auf diesem Gebiet jedoch noch vernachlässigt. Zudem ist strittig, ob große Datenmengen besser automatisiert oder interaktiv auszuwerten sind. Während die meisten der oben angeführten Arbeiten hierin die

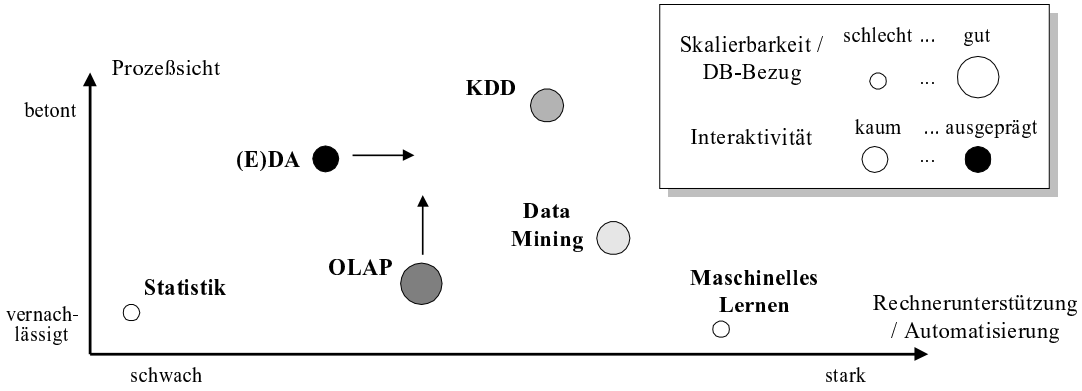

Abbildung 2.3: Einordnung von grundlegenden Ansätzen zur Auswertung von Daten

Stärke des Data Mining sehen, stellen Verfechter der EDA (vgl. z. B. [AC96a]) die Möglichkeiten der Datenexploration, auf Basis visueller Darstellungen einen ersten Eindruck zu erhalten und die menschlichen kognitiven Fähigkeiten in die Analyse einzubeziehen, in den Vordergrund. Weiterhin bestehen oft noch Unklarheiten über die Semantik bzw. Gemeinsamkeiten der jeweiligen Begriffsbildungen. So wird etwa in [AC98] KDD als Pendant zur aus der Statistik hervorgegangenen EDA für den Bereich der KI charakterisiert oder in [FPSS96a] KDD als eigentliche Datenanalyse, EDA dagegen allenfalls als ein Ansatz zur Realisierung von Teilschritten des Analyseprozesses angesehen.

Insgesamt sind jedoch die Übereinstimmungen zwischen KDD und Data Mining auf der einen und OLAP und EDA auf der anderen Seite unverkennbar [Lee94]. Neuere Arbeiten, von denen im folgenden einige kurz genannt werden sollen, streben die noch engere Kombination und Integration der genannten Ansätze an, um noch bessere Möglichkeiten zur flexiblen und vielseitigen Datenexploration zu bieten.

In [BA96] wird sehr ausführlich der Bedarf einer am menschlichen Anwender orientierten (*human–centered*), interaktiven Umsetzung des KDD–Prozesses motiviert und anhand des Begriffs der *Datenarchäologie* (vgl. [BST+93]) diskutiert. Lediglich rechnerbasierte *Werkzeuge* werden genutzt, und nur *Teilprozesse* werden automatisiert. Ähnlich wird in [DDI97] die Vision des *On–line Mining* vorgestellt: In einem interaktiven, iterativen Prozeß wechseln Datenmanagement, statistische Analysen, Mining–Techniken und Visualisierungen einander ab, wobei wissensbasierte Methoden Verfahrenswahl und Ergebnisfilterung unterstützen.

Die noch stärkere Einbeziehung des Menschen mit Hilfe von Datenvisualisierungen im Rahmen einer visuellen Datenexploration wird u. a. in [KKS94, LG96] sowie unter dem Stichwort des *Visual Data Mining* in [GHK+96] gefordert. Visualisierungen sollen zum Beobachten, Bewerten und Leiten des Analyseprozesses dienen, insbesondere im Hinblick auf Zielformulierungen, Methodenwahl und Ergebnisfilterung.

Schließlich sind noch Arbeiten zu nennen, die die Zusammenführung von Data Mining und OLAP (vor allem der Navigation durch Aggregierungsebenen, hier „Cubing“ genannt) propagieren. Han stellt in [Han97a] unter dem Schlagwort des *OLAP–Mining* verschiedene Möglichkeiten hierzu vor:

- Cubing zur Selektion von interessanten Datenräumen zum Mining,
- Cubing zur interaktiven Verfeinerung von Mining–Ergebnissen oder
- in das Mining integriertes Cubing zur Mustersuche auf verschiedenen Aggregierungsebenen.

Weiterhin wird die Anwendung von Verfahren des Data Mining im System *DBMiner* um ein interaktives Backtracking sowie den benutzergesteuerten Vergleich und die Kombination von Mining–Verfahren ergänzt. Allgemein verspricht sich Han von der Einbindung von OLAP–Konzepten in das Data Mining eine solidere Fundierung des Datenbankmanagements, eine Erhöhung der Interaktivität sowie eine erleichterte Selektion von Mining–Verfahren. Umgekehrt werden in [SAM98] Mining–Verfahren in ein OLAP–Tool integriert. Diese

führen den Benutzer bei der Navigation in „interessante“, also auffällige Bereiche des jeweiligen Datenraums. In [Par97] werden beide Blickwinkel in einer einheitlichen Sicht auf OLAP und Data Mining als gleichberechtigte und sich ergänzende Analyseansätze vereint.

2.1.2 Interaktionen als Komponenten des Datenanalyseprozesses

Bisher wurden Teile des Datenanalyseprozesses lediglich auf relativ grober Abstraktionsebene betrachtet. Im folgenden wollen wir uns genauer anschauen, welche Arten von Einzelhandlungen als Analyseschritte auftreten können. Diese Sichtweise wird später (in Kapitel 5) auch eine wesentliche Rolle bei der Modellierung von Datenanalysen als aus Einzelbausteinen zusammengesetzte Datenflußprogramme spielen. Zudem läßt sich ein Datenanalysesystem natürlich nur adäquat konzipieren, wenn die zu unterstützenden Abläufe im Detail bekannt sind.

In der Arbeitspsychologie unterscheidet Leontjew [Leo82] (vgl. auch [Uli92])

- Tätigkeiten „anhand der sie initiierenden Motive“,
- Handlungen „als bewußten Zielen untergeordnete Prozesse“ und
- Operationen, „die unmittelbar von den Bedingungen zur Erlangung des konkreten Ziels abhängen“.

Abbildung 2.4 veranschaulicht die entsprechenden, oftmals, aber nicht immer hierarchischen Zusammenhänge und zeigt zudem die Parallelen zur Datenanalyse auf: Eine bestimmte Aufgabenstellung führt zu einer Datenanalyse(sitzung), einzelne Schritte der Datenanalyse sollen als *Interaktionen*[5] (mit dem Analysesystem bzw. mit den betrachteten Daten) bezeichnet werden, die wiederum weiter in technische Details der Systembedienung bzw. der Datenverarbeitung zerlegt werden können. In ähnlicher Weise werden auch in [Lee94] Interaktionen der Datenanalyse modelliert.

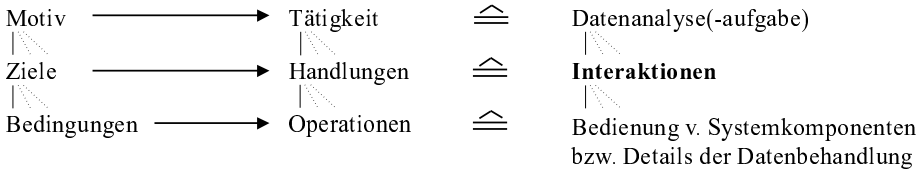

Abbildung 2.4: Die hierarchische Komposition von Tätigkeiten nach Leontjew

Aus kognitionspsychologischer Sichtweise (vgl. [And89]) bilden Interaktionen gerade die Abstraktionsebene, auf der der Datenanalyst seine Untersuchungen strukturiert und plant. Entsprechend kann die Interaktion auch als Metapher zur objektorientierten Modellierung und rechnerbasierten Implementierung der elementaren „Bausteine“ von Datenanalysen dienen (vgl. [Hub94] sowie die Diskussion der visuellen bzw. Datenflußprogrammierung in Abschnitt 2.2.2 und 2.2.4). Die konkrete Umsetzung von Interaktionen — auf der Basis technischer Details der Bedienung eines Analysesystems oder der rechnerunabhängigen Datenbehandlung — braucht an dieser Stelle nicht genauer zu interessieren.[6]

Interaktionen im Analyseprozeß lassen sich grob den Bereichen

- *Navigation* im bisherigen Analyseverlauf,
- *Datenwahl*, also Datenselektion und -management,

[5] Dieser Begriff ist nicht zu verwechseln mit Interaktionen, wie sie in der Software–Ergonomie, der Mensch–Maschine–Kommunikation oder im Multimedia–Bereich verstanden werden [P+94, Obe94, Shn87, BS98, Bol94]. Dort entspricht eine Interaktion im engeren Sinne eher der kleinsten Einheit bei der Systembedienung in Abb. 2.4, wenn auch dieser Begriff oftmals nicht so exakt und restriktiv definiert ist. Generell ist aber auch in diesen Bereichen die dreistufige Modellierung der Systembedienung üblich (siehe etwa [P+94] mit der Unterscheidung von Ziel, Aufgabe und Aktion).

[6] Zum Beispiel in [LG96] wird eine derartige Umsetzung diskutiert und vorgenommen.

- Anwendung *statistischer Verfahren* oder anderer konkreter Analysetechniken und
- *Datenvisualisierung*

zuordnen [Lee94]. Hierbei fallen Datenwahl, Statistikanwendung und Visualisierung nach [Obe94] klar in die Klasse der *Anwendungsinteraktionen*, die den eigentlichen Gegenstand der Analyse betrachten, während der Navigation eine vielschichtige Aufgabe zwischen Anwendungs- und *Steuerungsinteraktionen* zukommt: Sie dient zum einen der Koordination und Verwaltung von Analyseschritten, liefert aber auch selbst wesentliche Erkenntnisse im Hinblick auf untersuchte Zusammenhänge.

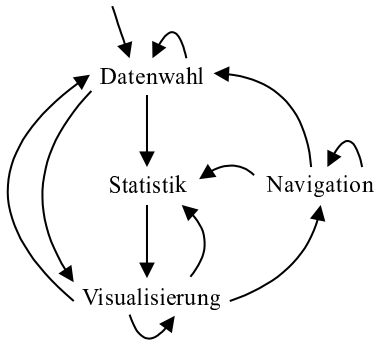

Abbildung 2.5: Ein einfaches Interaktionsmodell

Unter einem *Interaktionsmodell* soll eine Spezifikation möglicher Abfolgen oder Kombinationen von Interaktionen verstanden werden, die im Rahmen einer Analyse zur zusammenhängenden Bearbeitung einer Fragestellung möglich und sinnvoll sind[7] bzw. durch eine konkrete Datenanalyseumgebung unterstützt werden.

Ein exemplarisches, recht grobes und flexibles Interaktionsmodell zeigt Abb. 2.5 in Form eines gerichteten Graphen. Viele verschiedene Abfolgen von Interaktionen sind hier möglich, wobei eine Analyse jedoch stets mit der Selektion von Daten (der „Datenwahl“) beginnt. Weiterhin erfolgt nach einer Verfahrensanwendung („Statistik“) zunächst die Visualisierung der Ergebnisse; Navigationsschritte folgen auf Visualisierungsinteraktionen und führen zu einer Modifikation bzw. Erweiterung der Datenwahl oder der Anwendung eines weiteren statistischen Verfahrens. In Kapitel 3 werden wir diese Darstellungsweise nutzen, um verschiedene Klassen von Analysesystemen grob voneinander abzugrenzen.

Springmeyer untersucht in [SBM92] etwas detaillierter die Menge der für den Analyseprozeß und dessen Umsetzung in rechnergestützten Analyseumgebungen relevanten Interaktionen. Sie unterscheidet grob nach Auswertung (Interaktion mit Darstellungen und Anwendung mathematischer Verfahren) sowie Integration und Aufarbeitung gewonnener Erkenntnisse durch das „Festhalten/Formulieren von Ideen“. In beiden Bereichen spielt das „Manövrieren“, also zum einen die Navigation durch den Analyseprozeß bzw. die Benutzungsoberfläche des Analysesystems sowie zum anderen die Datenselektion und das Datenmanagement, eine wesentliche Rolle. Alle vier Interaktionsklassen werden jeweils noch feiner untergliedert, siehe Abb. 2.6.

Zu beachten ist, daß sich auch hier Interaktionen nicht direkt auf die Bedienung eines Analysesystems beziehen müssen, sondern auch davon unabhängige Tätigkeiten eines Anwenders spezifizieren können. So beschreibt etwa das Klassifizieren im Rahmen der Interaktion mit Darstellungen eine Aktion des Analysten beim Betrachten einer Graphik, ohne ein konkretes Analyseverfahren anzuwenden.

Der Aspekt der Navigation im Analyseprozeß wird in [Ama97] noch genauer untersucht. Hier werden Vorwärts- und Rückwärtsschritte, der Sprung an eine beliebige Position in der Analyse, die Übersicht über die gesamte Historie einer Ergebnisherleitung sowie — als Managementoperationen auf dem Analyseprozeß — das Kopieren, Verschieben, Wiederholen oder Löschen von Analysezweigen genannt.

Näher mit Datenselektion und -management — allgemein im Kontext des Datenbankzugriffs, nicht nur beschränkt auf Datenanalysen — beschäftigt sich Larson [Lar86]: Unter dem Obergriff des „Browsing“ in Datenbeständen wird zwischen dem Strukturieren (Anordnen) von Datenobjekten, dem Filtern, dem Schwenken über benachbarte Instanzen sowie dem Zooming unterschieden. Bei [CRS85] werden unter den Begriffen „Scanning“, „Browsing“, „Searching“, „Exploring“ und „Wandering“ verschiedene Abfolgen von Navigationsschritten bei der Suche in Datenbeständen charakterisiert und in Form von Graphen veranschaulicht. Unterschieden wird etwa danach, ob die Navigation zielgerichtet ist, Zyklen aufweist, ob sie auf „benachbarte“ (Daten–)Bereiche beschränkt oder sehr ausufernd ist.

[7] Dies läßt sich etwa aus der Betrachtung von Interaktionsprofilen (Protokollen) typischer Analysesitzungen ableiten.

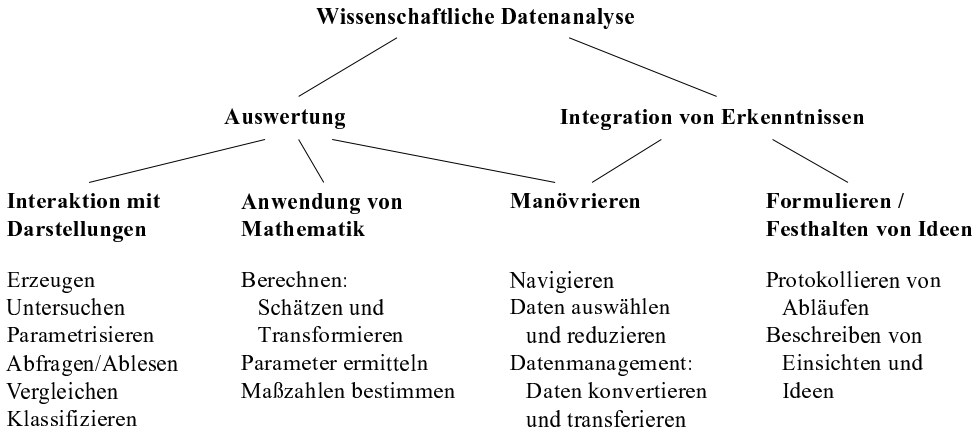

Abbildung 2.6: Interaktionsklassen der Datenanalyse (nach [SBM92])

Eine Brücke zwischen der Sicht auf die Phasen des Explorationsprozesses (wie im vorigen Abschnitt 2.1.1 dargestellt) und der Klassifikation von Analyseinteraktionen wird von Huber in [Hub86] geschlagen. Die dort vorgestellten, iterativ durchführbaren Analyseaktivitäten Inspektion, Modifikation/Vergleich, Modellierung und Interpretation entsprechen im wesentlichen auch der Gruppierung von Springmeyer. Für Modifikation und Vergleich wird jedoch von Huber nicht unterschieden, ob sie im Sinne von Datenmanagement und Navigation oder im Rahmen der direkten Manipulation einer graphischen Darstellung zum Einsatz kommen.

Die Visualisierung schließlich wird von Shneiderman in [Shn96] mit ähnlichen Ergebnissen wie bei Springmeyer untersucht. Abgesehen vom Erzeugen und Klassifizieren entsprechen die „visuellen Aufgaben“ Überblick, Zoom und Filter, Abfrage von Details und In–Beziehung–Setzen/Vergleichen gerade den Interaktionen von Springmeyer. Zusätzlich nennt Shneiderman im Kontext der Visualisierung noch (aus der zentralen Interaktionsklasse des Manövrierens bei Springmeyer) den Einblick in die Historie sowie die Ablage von Visualisierungsergebnissen (dem Datenmanagement zuzuordnen). Einen Überblick über weitere Arbeiten zur Klassifikation von Interaktionen mit Visualisierungen gibt auch [LG96].

2.1.3 Datenvisualisierung

Die herausragende Bedeutung der Visualisierung in der wissenschaftlichen Forschung wurde auf einer Sitzung der *Division of Advanced Scientific Computing* der *National Science Foundation* in Washington 1986 mit folgenden Worten formuliert [MDB87]:

> *Visualization in Scientific Computing [...] offers a way to see the unseen.*

In ähnlichem Sinne äußert sich Tukey in [Tuk77]:

> *The greatest value of a picture is when it forces us to notice what we never expected to see.*

Kein anderer Ansatz vermag so effektiv und effizient gleichzeitig sowohl globale Muster und Strukturen als auch feine Details, sowohl quantitative als auch qualitative Informationen aus großen Datenmengen zu extrahieren wie die visuelle Datenanalyse [Cle94]. Gegenüber tabellarischen Darstellungen etwa zeigen Graphiken, unterstützt durch Freiräume zur zweckbestimmten Gestaltung, viel mehr unterschiedliche Aspekte eines Datensatzes, die zudem deutlich einprägsamer vermittelt werden können [NBOH96]. Darüber hinaus bietet die Datenvisualisierung exzellente Möglichkeiten zur interaktiven Anfragemodifikation mit direktem visuellen Feedback [KKS94]. Insgesamt kann sie als das wichtigste Werkzeug der explorativen Datenanalyse angesehen werden [dTSS86, Cle93, Hin87], denn nur mit der adäquaten Darstellung von Analyseergebnissen wird der

Analyst in die Lage versetzt, neue Hypothesen und Strategien zur Untersuchung des betrachteten Datensatzes zu entwickeln und zu verfolgen.

Visuelle Darstellungen kommen im Rahmen von Durchführung und Kommunikation von Datenanalysen auf drei zentralen Aufgabengebieten zum Einsatz [NBOH96, Cle94], und zwar bei

- der visuellen Dateninspektion zur Hypothesengenerierung,
- der Prüfung von Modellvorstellungen und
- der Ergebnispräsentation.

Erkenntnisse der Kognitionspsychologie belegen, daß gerade zur Aufbereitung und Vermittlung schwieriger Zusammenhänge Bilder als natürlicher und gegenüber textuellen Darstellungen leichter verständlich angesehen werden können [And89]. So kann das menschliche Gehirn in zwei Hälften unterteilt werden, von denen die linke (eher) für das logische und analytische Denken, die rechte dagegen (mehr) für die gefühlsbezogene und ästhetische Kognition auf der Basis audiovisueller Informationen zuständig ist. Während die linke Gehirnhälfte Informationen vorwiegend sequentiell verarbeitet, geschieht dies in der rechten zumeist parallel. Somit können Bilder weitaus schneller als Texte aufgenommen werden und hinterlassen zugleich einen deutlich tieferen Eindruck.[8] Details zur Perzeption visueller Informationen werden etwa in [HW97a] sehr detailliert dargestellt.

Das menschliche visuelle System ist aufgrund langer evolutionärer Prozesse spezialisiert auf das effektive und effiziente Erkennen von Strukturen, das Filtern relevanter Zusammenhänge und die parallele Aufnahme sehr vieler Informationen [Goo83]. Spence bezeichnet in diesem Zusammenhang das Erkennen, Klassifizieren und Erinnern visueller Muster als die am weitesten entwickelten menschlichen Fähigkeiten zur Informationsverarbeitung [SL90], woraus nochmals die Bedeutungen von Visualisierungen für die explorative Datenanalyse deutlich wird.

Terminologie

Den folgenden Ausführungen wird ein einfaches Modell einer *Graphik* zugrunde gelegt (vgl. Abb. 2.7):

- Eine Graphik dient der Visualisierung von einem oder mehreren homogenen Datenbeständen, die jeweils aus einer Menge von *Datenobjekten* bestehen.
- Ein Datenbestand beschreibt jeweils eine Menge von Variablen (Eigenschaften) von Realwelt–Objekten. Jedes Datenobjekt aus dem Bestand weist für jede dieser Variablen ein *Attribut* mit einer entsprechenden Ausprägung auf. (Ein Datenbestand hat also die Gestalt einer Relation auf den Wertebereichen der betrachteten Variablen bzw. Attribute.)
- Die Visualisierung des Datenbestandes in einer Graphik erfolgt, indem den Datenobjekten — unter Berücksichtigung einer ausgewählten Menge der vorliegenden Variablen — *Darstellungssymbole* zugeordnet und in die Graphik eingetragen werden.
- Jedes Darstellungssymbol besitzt eine Menge *graphischer Attribute*, etwa Position, Höhe, Breite, Farbe etc.
- Die Ausprägungen der graphischen Attribute eines Symbols ergeben sich aus den Ausprägungen der Attribute der jeweils zugeordneten Datenobjekte.
- Einem Symbol können ein oder — nach Zusammenfassung (Aggregierung) von Datenobjekten mit gleichen Attributausprägungen, ggf. auch nach vergröbernden Klassifikationen auf bestimmten Variablen — mehrere Datenobjekte zugeordnet sein.
- In letzterem Fall können auch Maßzahlen, die durch die Anwendung von Aggregierungsfunktionen auf den Objektgruppen berechnet wurden (insbesondere die Anzahl) zur Definition der graphischen Attribute herangezogen werden.[9]

[8] Hier sei an die bekannte Arbeit von Miller aus dem Jahr 1956 über „die magische Zahl 7“ als Begrenzung menschlicher Fähigkeiten zur sequentiellen Informationsaufnahme erinnert [Mil56].

[9] Natürlich kann man die erhaltenen Gruppen, deren gemeinsame Attributausprägungen und über diesen definierte Maßzahlen wiederum

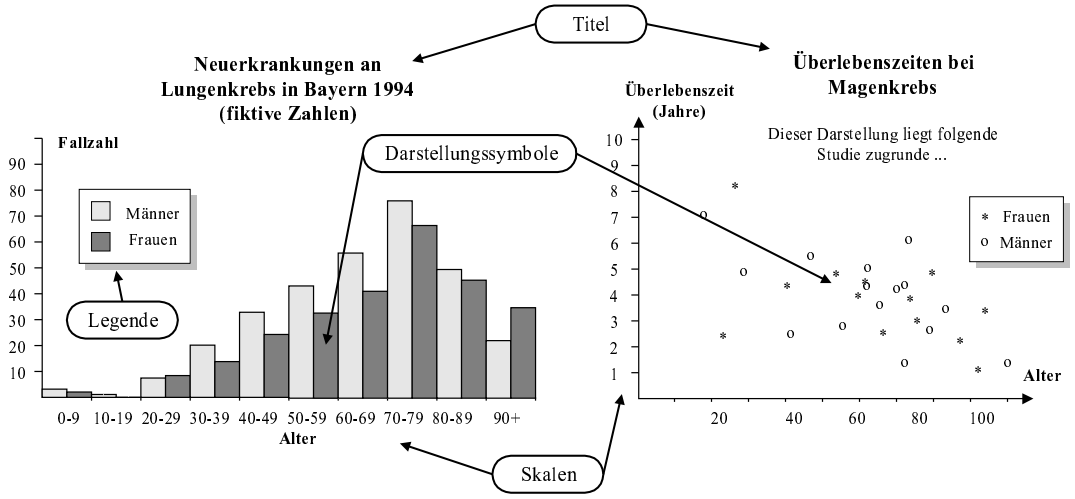

Abbildung 2.7: Ein Balkendiagramm und ein Scatterplot

An Bestandteilen einer Graphik wird grob zwischen dem *Titel* (ggf. inklusive Erläuterungen), dem *Darstellungsrahmen* (bestehend aus Skala und Legende) sowie dem eigentlichen *Inhalt* der Graphik, der Menge der (evtl. beschrifteten) Darstellungssymbole, unterschieden. Als *Darstellungskomponenten* werden sowohl Darstellungssymbole als auch Bestandteile des Rahmens bezeichnet, und unter dem Begriff der *Visualisierungsparameter* werden allgemein alle Merkmale verstanden, die das Erscheinungsbild einer Graphik näher spezifizieren. Dies sind neben den unmittelbar zur Repräsentation der Attribute von Datenobjekten verwendeten graphischen Attribute auch weitere Merkmale beliebiger Darstellungskomponenten.

Abbildung 2.7 zeigt mit einem Balkendiagramm und einem Scatterplot zwei Beispiele typischer Graphiken. Datenobjekte bilden jeweils Erkrankungsfälle (über die hier interessierenden Variablen Alter, Geschlecht und Überlebenszeit). Darstellungssymbole sind im Balkendiagramm Boxen (mit den graphischen Attributen x–Koordinate und Höhe), im Scatterplot dagegen einzelne Zeichen (mit x– und y–Koordinate sowie ASCII–Code). Das Balkendiagramm faßt Gruppen von Datenobjekten nach Aggregierung in Altersgruppen zusammen und visualisiert die Anzahl der Gruppenelemente als Balkenhöhe (sowie Geschlecht und Alter als x–Koordinate des Balkens). Im Scatterplot dagegen werden die einzelnen Datenobjekte direkt visualisiert.

Dieses Graphikmodell wird in Abschnitt 5.2.3 im Rahmen von *VIOLA* noch formalisiert und verfeinert werden. Hier sollen die eingeführten Begriffe zunächst einmal zum weiteren Verständnis ausreichend sein.

Gestaltung *guter* Graphiken

Die Aufgabe von Graphiken in der Datenanalyse besteht nach Tufte in der effizienten Kommunikation komplexer Ideen [Tuf83]. Graphiken sollten

- in erster Linie die Daten zeigen (also alles Überflüssige, sogenannten *Chartjunk*, vermeiden und die *Data–Ink–Ratio*, den Anteil Datenwerte repräsentierender Graphikelemente, maximieren),
- den Betrachter über den Inhalt — nicht die Methode — nachdenken lassen,
- die Aussage der Daten nicht verfälschen (vgl. Abb. 2.8 als Beispiel für eine Graphik mit einem hohen *Lügenfaktor*, also einem Mißverhältnis zwischen Relationen in den Daten einerseits und Variationen in graphischen Attributen andererseits: das Verhältnis der Längen der markierten Linien beträgt etwa

als Datenobjekte eines neuen Datenbestandes auffassen. Eine flexiblere Betrachtungsweise kann jedoch mitunter hilfreich sein, siehe etwa weiter unten die Diskussion des Linking von Graphiken. In Abschnitt 2.3.2 werden wir noch genauer auf diese Unterscheidung von *Mikro-* und *Makrodaten* eingehen.

8.8:1, das der entsprechenden Datenwerte dagegen nur etwa 3:2 — Tufte berechnet aus den jeweiligen prozentualen Differenzen einen Lügenfaktor von $783\%/53\% \approx 14.8$),

- viele Zahlen auf kleinem Raum darstellen,
- Zusammenhänge in großen Datenmengen zeigen,
- zum Vergleich von Daten ermutigen,
- die Gestalt der Daten auf verschiedenen Ebenen (von Übersicht bis Feinstruktur) enthüllen,
- mit verbalen und statistischen Beschreibungen integriert sein,
- einem klaren Zweck, nämlich Exploration, Präsentation, Beschreibung oder Verzierung, dienen.

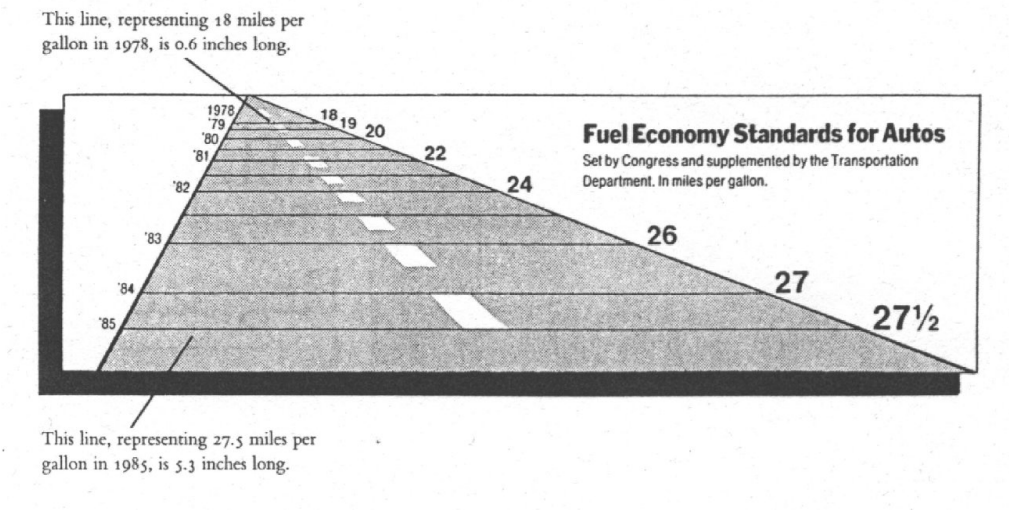

Abbildung 2.8: Verfälschte Relationen in einer Graphik mit hohem Lügenfaktor (aus [Tuf83])

Cleveland faßt diese Aspekte in [Cle94] unter den Begriffen der

- Übersichtlichkeit (die Daten sollten hervorstechen, auf Überflüssiges ist zu verzichten) und
- Verständlichkeit (insbesondere der informativen Beschriftung)

von Graphiken zusammen. Insgesamt definiert Tufte exzellente Graphiken als

> *[. . .] that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.*

Die Verwendung von Graphiken zur objektiven Darstellung von Datensammlungen im obigen Sinne erfordert eine Vielzahl differenzierter Überlegungen zur visuellen Wahrnehmung von Informationen. Der gleiche Sachverhalt kann in unterschiedlichen Präsentationsformen völlig unterschiedlich aufgenommen werden. Dies kann zu Verzerrungen und Verfälschungen führen (vgl. Abb. 2.8), z. B. rein zufällige Werteverteilungen zu auffälligen Abhängigkeiten oder Clustern werden lassen, aber auch in der explorativen Analyse dazu genutzt werden, verschiedene Aspekte der Daten zu betrachten, zu vergleichen und „versteckte“ Strukturen zu extrahieren.

Die „Kunst“ der Graphikerstellung besteht darin, die interessierenden Daten so auszuwählen, aufzubereiten und visuell umzusetzen (zu „verschlüsseln“), daß sie vom Betrachter (im vom Ersteller beabsichtigten Sinne) korrekt wahrgenommen und wieder „entschlüsselt“ werden können. In der Regel ist es hierzu nötig, viele verschiedene Graphiken zu einem Datensatz zu erstellen, mit Varianten zu experimentieren und diesen iterativ zu verbessern [Cle94].

Die Art der Codierung von Datenwerten durch graphische Attribute (etwa Position, Länge, Größe, Farbe, Schraffur etc.) beeinflußt wesentlich Schnelligkeit und Genauigkeit der Informationsaufnahme (vgl. eine empirische Untersuchung hierzu in [CM84] sowie ein noch differenzierteres Ranking von Attributen in [Mac86]). Eine nicht unwichtige Rolle spielt hierbei die jeweilige Aufgabe der Graphik [Cle94, Ber74], grob zu unterscheiden nach[10]

- Identifikation auffälliger Einzelwerte oder auch Ablesen, Suchen oder Interpolieren konkreter Ausprägungen,
- Gruppierung oder Vergleich (Unterscheidung, Ordnung oder Verhältnisschätzung) von Datenwertegruppen sowie
- Erkennen globaler Strukturen.

Als sehr umfassende und lesenswerte Arbeiten zur Erstellung „guter“ Graphiken sind verschiedene Bücher von Tufte, Cleveland und Bertin zu nennen. [Tuf83] und [Cle94] konzentrieren sich hierbei eher auf grundlegende Konstruktionsprinzipien und Gestaltungsmittel sowie auf Regeln zu ihrem sinnvollen Einsatz; [Tuf91] und [Cle93] orientieren sich mehr an konkreten Graphiken und Visualisierungsbeispielen sowie deren Parametrisierungsmöglichkeiten. Bertin legt in [Ber74] mit einer umfassenden Taxonomie graphischer Komponenten und — parallel dazu — von Aspekten der visuellen Wahrnehmung die Grundlage einer Grammatik für Graphiken.

Die meisten der genannten Arbeiten klassifizieren Graphiken vor allem nach der Anzahl darstellbarer Merkmale sowie deren Typ (quantitativ, qualitativ, etc.) und Kardinalität. Eine weitere grundlegende Unterscheidung trifft Bertin nach der Nutzung der beiden — neben der Gestaltung von Darstellungssymbolen — durch die Zeichenfläche gegebenen Darstellungsdimensionen. Hiernach ergeben sich die Klassen

- *Diagramm* (Darstellung der Beziehungen zwischen mehreren Merkmalen),
- *Netz* (Darstellung der Beziehungen zwischen Ausprägungen eines Merkmals),
- *Karte* (ein Spezialfall von Netzen mit der Raumdimension als betrachtetem Merkmal) und
- *Symbole* (Verzicht auf die Darstellung von Beziehungen außerhalb der einzelnen Darstellungselemente),

die natürlich auch kombinierbar sind. Einen ausführlichen, aber weniger an einzelnen Gestaltungsmerkmalen ausgerichteten Überblick über das Angebot statistischer Graphiken gibt auch [NBOH96].

Schließlich seien hier mit [GRKM94, ISF94] noch einige Ansätze zur automatischen, wissensbasierten Graphikerstellung erwähnt, die sich an oben angeführten Grundsätzen orientieren. Ihnen liegt jeweils eine formale Beschreibung der darzustellenden Datensätze in Form von Metadaten (vgl. [RM90] sowie Abschnitt 2.4) sowie eine Charakterisierung der zugrundeliegenden Datenanalyseaufgabe zugrunde, woraus etwa über ein Regelsystem geeignete Visualisierungsformen gefunden werden können. Etwas anders geht MacKinlay in [Mac86] vor: Hier werden zunächst viele verschiedene Graphiken, die die interessierenden Informationen aus einem Datensatz ausdrücken können, generiert und anschließend anhand der Effizienz der Darstellung, die über die jeweils eingesetzten graphischen Attribute definierbar ist, bewertet. Dieses Verfahren wird vor allem dann interessant, wenn komplexe Graphiken zusammengesetzt werden, die gleichzeitig verschiedene Merkmale zu unterschiedlichen Aufgabenstellungen visualisieren sollen.

Interaktive Graphiken

Eine effiziente und effektive Analyse insbesondere großer Datenmengen verlangt die (inter-)aktive Einbeziehung des Benutzers in die Datenexploration [SBM92, UFK+89]. Vor allem werden spezielle Konzepte zur Interaktion mit den Analyseergebnissen benötigt. Graphiken und Tabellen dürfen nicht (nur) ein Endprodukt der Analyse sein, sondern sollten ihrerseits in dynamischer Form als Ausgangsbasis für vergleichende bzw. tiefergehende Untersuchungen dienen [RCK+97]. Die *interaktive Graphik* zeichnet sich hierbei als Spezialfall

[10] Man beachte, wie diese Differenzierung teilweise der Klassifikation von Aufgaben des Data Mining in Abschnitt 2.1.1 ähnelt.

der *dynamischen Graphik* dadurch aus, daß der Benutzer selbst in die Änderung von Visualisierungsparametern eingreifen kann, während diese bei dynamischen Graphiken (etwa in rotierenden Punktwolken) automatisch erfolgt und lediglich gestartet und wieder abgebrochen werden kann [The96b].

Grundlegend für interaktive Visualisierungen ist also die direkte Manipulation von Graphiken und die unmittelbare Visualisierung sich daraus ergebener Veränderungen (vgl. [Shn83]).[11] Ganz im Sinne der EDA wird also der Mensch durch das direkte „Anfassen" von Graphiken noch stärker und direkter zum steuernden Element der Exploration. Die Interaktion eröffnet auf effiziente Weise noch mehr unterschiedliche Blickwinkel auf die Daten und gestattet einen direkteren und differenzierteren Datenvergleich. Das „Risiko", wichtige Aspekte eines Datenbestandes zu übersehen oder durch schlechte Darstellungen zu verfälschen bzw. falsch zu interpretieren, sinkt deutlich.

Mit der Verfügbarkeit immer leistungsfähigerer Rechnersysteme steigen die Möglichkeiten zur Implementierung interaktiver Datenvisualisierungen. Unwin bezeichnet dementsprechend den Einsatz interaktiver Graphiken sowie die Nutzung von Analysesystemen, die diese unterstützen, als „moderne explorative Datenanalyse" [Unw94].

An Grundelementen und -methoden interaktiver Visualisierung sind nach [BCW88, The96b, NBOH96] zu nennen:

- die interaktive *Selektion* von Darstellungssymbolen bzw. Datenobjekten (vor allem als Grundlage anderer Techniken) — dies kann über einen Pointer, eine in Größe und Position dynamisch modifizierbare Box (auch *Brushing* genannt), seltener auch andere geometrische Selektionswerkzeuge, wie Linien (*Slicer*), Kreise oder frei definierbare Regionen (ein sogenanntes *Lasso*), erfolgen,
- das *Highlighting*, also das Hervorheben von Datenelementen in einer Graphik aufgrund der Spezifikation von Wertebereichen über Selektionsparameter oder bei der *Identifikation* einzelner Datenobjekte,
- das direkte *Abfragen* näherer Informationen zu beliebigen Graphikkomponenten (sowohl zu Bestandteilen des Graphikrahmens also auch in Form eines *Labelling*, also des Beschriftens einzelner Darstellungssymbole bzw. Datenobjekte),
- das *Löschen* von Symbolen bzw. der zugehörigen Datenelemente aus einer Graphik,
- die Anzeige von *Warnungen* bzw. Hinweisen auf möglicherweise falsch zu interpretierende Aspekte einer Graphik,
- die dynamische *Skalierung* und *Parametrisierung* einer Graphik (hierunter fällt auch die benutzergesteuerte *Rotation* oder die Einschränkung der betrachteten Wertebereiche),
- das *Overlaying* von Graphiken, also das Übereinanderlegen zweier Darstellungen im gleichen oder einem vergleichbaren Koordinatensystem, sowie
- das *Linking* zweier Graphiken, d. h. allgemein der Austausch von Informationen und Parametrisierungen zwischen Graphiken.

Die Basis für die meisten dieser Verfahren bildet die formale Modellierung der Abbildung von Datenobjekten auf Darstellungssymbole sowie der Attribute eines Datenobjekts auf graphische Darstellungsattribute. Jede Komponente einer Graphik ist die visuelle Wiedergabe einer bestimmten Information, entweder — als Darstellungssymbol — zu einem Datenobjekt bzw. zu den Ausprägungen einiger seiner Attribute oder — in Skalen oder der Legende — zu einer Variablen auf den Daten. Über die Manipulation dieser Graphikkomponenten kann nun direkt mit Datenobjekten interagiert bzw. eine Parametrisierung der Darstellung vorgenommen werden. Ein entsprechendes Softwaremodell diskutiert [HO91] im Detail. Als allgemeine Einführung in die Thematik interaktiver Graphiken können [The96b, CM88, NBOH96] dienen. Dort finden sich auch jeweils einige Anwendungsbeispiele. Im folgenden wird noch kurz auf zwei Varianten interaktiver Visualisierung, die im Rahmen dieser Arbeit von besonderem Interesse sein werden, genauer eingegangen, nämlich Dynamic Queries und das Linking von Graphiken.

[11] Eine ähnliche Idee liegt auch der visuellen Programmierung zugrunde, siehe Abschnitt 2.2.2.

Ein einfaches, effektives Beispiel der interaktiven Parametrisierung von Graphiken bilden sogenannte *Dynamic Queries* [Shn94]. Die Restriktion des zu betrachtenden Teildatenbestandes erfolgt hier über direkt mit der visuellen Darstellung verbundene Schieberegler oder ähnliche Komponenten der Benutzungsoberfläche. Durch diese kann der in der Graphik betrachtete Datenausschnitt für bestimmte Attributdimensionen in Realzeit modifiziert werden. Dieses Wandern durch den betrachteten Datenraum ermöglicht zum einen die flexible Suche nach interessierenden Teilbereichen und gibt darüber hinaus der i. a. ein- oder zweidimensionalen Graphik noch eine dritte, durch die Benutzerinteraktion eingeführte Dimension. Abbildung 2.9 zeigt ein Beispiel für Dynamic Queries auf einer Landkarte, die Erkrankungsraten visualisiert. Interaktiv kann der interessierende Zeitraum sowie — über verschiedene bevölkerungsbezogene Parameter — die Menge der darzustellenden Gebiete selektiert werden.

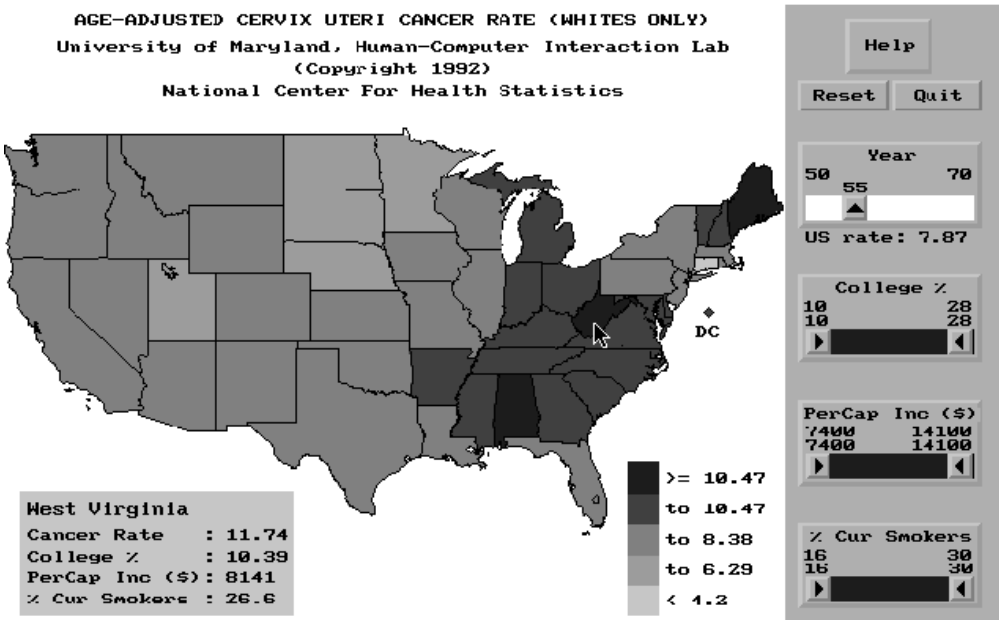

Abbildung 2.9: Einsatz von Dynamic Queries in der raumbezogenen Krebsepidemiologie (aus [PJ94])

Als ein weiterführender Ansatz zur interaktiven Manipulation von Graphiken koppelt das *Linking* zweier oder mehrerer Graphiken nicht Datenmanagement (Selektion) und Visualisierung, sondern mehrere, i. a. gleichberechtigte Visualisierungen dynamisch miteinander (vgl. etwa [CM88, The96b]). Die Grundidee besteht hierbei darin, diejenigen Teile der Darstellungen, die sich jeweils auf die gleichen Datenobjekte beziehen, zueinander in Verbindung zu setzen und über diese Verbindung Informationen auszutauschen. Gerade bei der Untersuchung komplexer, also großer, hochdimensionaler Datenbestände, die sowohl diskrete als auch stetige Variablen enthalten, erleichtert das Linking die flexible Exploration. Darüber hinaus kann auch aus der Interaktion selbst, also dem unmittelbaren Verändern und In–Beziehung–Setzen von Graphiken, neues Wissen gewonnen werden.

Eine typische Variante des Linking stellt das *gelinkte Highlighting* dar: In einer Graphik selektierte Teilbereiche eines Datenraums bzw. Teilmengen der enthaltenen Datenobjekte werden auch in einer weiteren Darstellung der gleichen Daten hervorgehoben.[12] Ein schönes Beispiel hierfür findet sich in Abb. 2.10. Dort werden Beobachtungen tropischer Wirbelstürme mit Hilfe des Analysesystems *Regard* [Unw94] untersucht. Im linken unteren Histogramm sind Stürme mit hoher Zuggeschwindigkeit selektiert. Entsprechende Beobachtungsorte und y–Koordinaten werden in Landkarte und zweitem Histogramm hervorgehoben.

[12] Hier macht sich die einheitliche Sicht auf Darstellungssymbole zu Datenobjekten und solchen zu Mengen von Datenobjekten, wie sie im oben vorgestellten Graphikmodell eingeführt wurde, bezahlt. Das Hervorheben von Teilmengen der Datenobjekte erfolgt durch das

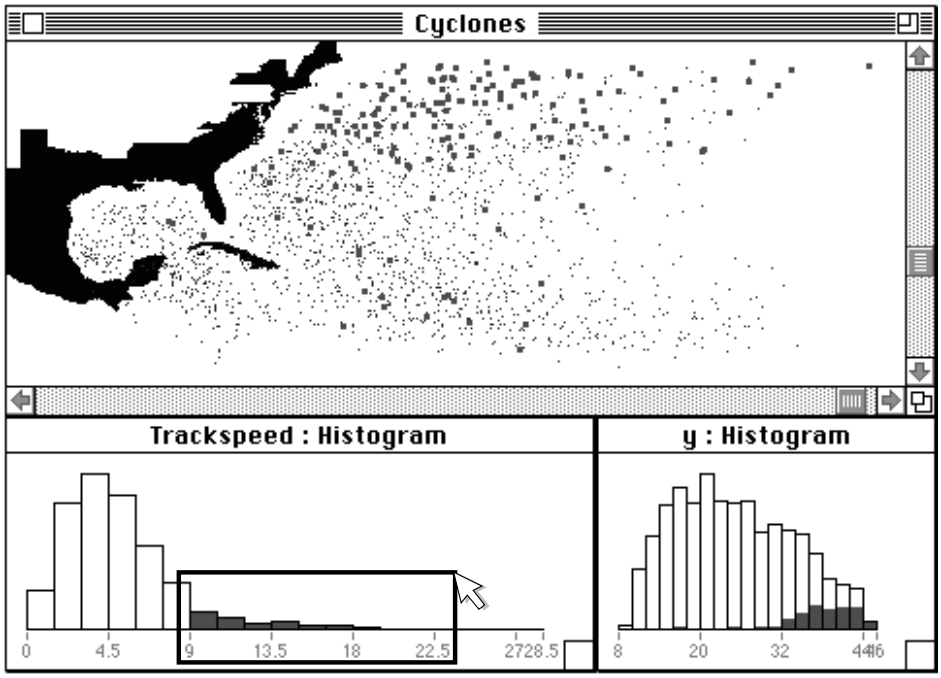

Abbildung 2.10: Linking von Graphiken (aus [The95])

In den meisten Fällen ist die Verbindung zwischen gelinkten Graphiken — dies können auch durchaus mehr als zwei sein — bi- bzw. multidirektional, d. h. Selektionen in einer beliebigen Graphik werden zugleich auf alle anderen übertragen.

Allgemein unterscheidet man zwischen *empirischem* und *algebraischem Linking* [NBOH96]. Beim empirischen Linking werden entweder in allen beteiligten Graphiken stets die identischen *Datenobjekte* (nur ggf. in Gestalt anderer Darstellungssymbole) hervorgehoben (wie in Abb. 2.10) oder aber stets die gleichen *Variablen* (Attribute) des betrachteten Datenraums (wiederum natürlich in unterschiedlicher Form) dargestellt. Letztere Variante ist etwa im Analysesystem *ViSta* [YFM93] realisiert — dort können z. B. durch Selektion von Zellen aus Scatterplot–Matrizen[13] Paare von Variablen spezifiziert werden, die in anderen Diagrammen genauer zu betrachten sind. Demgegenüber definiert das algebraische Linking Abbildungsfunktionen, die aus der Selektion einer Graphik die Darstellung in einer anderen berechnen. Somit ist hier der Bezug auf denselben Datenraum nicht zwangsläufig erforderlich.

In [The96b] werden orthogonal zu dieser Differenzierung *kalte*, *warme* und *heiße* Darstellungssymbole bzw. Graphiken unterschieden. Während erstgenannte kein Linking erlauben, erfolgt das Linking im Falle warmer Objekte erst nach expliziter Anforderung und für heiße Objekte stets sofort.

Ein allgemeinerer Linkingbegriff wird von Wilhelm in [Wil97] im Sinne eines beliebigen Informationsaustauschs zwischen Datenobjekten und/oder Darstellungskomponenten (Skalen, Legende etc.) eingeführt. Neben obigen Varianten diskutiert er u. a.

- das Linking von Skalen (Minimum, Maximum, Einteilung) und damit der Datenselektion (ähnlich den Dynamic Queries),
- die Übernahme von benutzerdefinierten Ordnungen für nominale Variablen,
- die gleichzeitige Abfrage näherer Informationen für entsprechende Datenobjekte in verschiedenen Gra-

Highlighting eines Teiles des Darstellungssymbols.

[13] Zu allen Paar–Kombinationen einer Menge von Variablen werden die jeweiligen Scatterplots, also Punktwolken, in einer Matrix angeordnet.

phiken sowie

- das Linking zwischen Daten aus verschiedenen Quellen mit zumindest teilweise übereinstimmenden Attributen.

Einige dieser Varianten finden sich auch in dem Visualisierungssystem *DEVise* [LRB+97]. Weiterhin wird von Wilhelm das Problem der Überlagerung verschiedener, nacheinander ausgeführter Selektionen angesprochen. In Form von *Selektionssequenzen* (vgl. [The96b]) können unterschiedliche Arten der Selektion (gemäß der Aufstellung auf Seite 28) auf verschiedenen Graphiken über beliebige Mengenoperatoren verknüpft werden. Derartige Sequenzen werden in Form von SQL–Anfragen modelliert und können auch nachträglich modifiziert oder auf anderen Datensätzen wiederverwendet werden. Allgemein eröffnen sich somit über das Linking weitere Möglichkeiten zum komfortablen Datenmanagement, das ohne ein explizites Wechseln zwischen Visualisierungs- und Datenmanipulationsoperationen auskommt.

2.1.4 Ziele intelligenter Datenanalysen

Abschließend kommen wir noch einmal auf den Begriff der „intelligenten Datenanalyse“ zu sprechen, der die grundlegende Motivation dieser Arbeit definiert. Im Vorwort von [BH99] wird unter diesem Begriff zwischen zwei Aspekten differenziert, nämlich

1. der Entwicklung und Anwendung *intelligenter rechnerbasierter Werkzeuge*, das sind (nach Berthold und Hand) Systeme

   [...] which probe more deeply into structure than first generation methods [...],

   die also zu sehr differenzierten Betrachtungen der jeweiligen Daten in der Lage sind, und
2. der *intelligenten Anwendung* von Datenanalysesystemen, dem wohlüberlegten Einsatz von Analyseverfahren und der fundierten Interpretation ihrer Ergebnisse.

Wie bereits in Kapitel 1 dargestellt, steht in dieser Arbeit die zweite Sichtweise, hierbei vor allem die enge Kooperation zwischen menschlichem Datenanalysten und rechnerbasiertem Analysesystem, also die adäquate Synchronisation ihrer jeweiligen Wissens- und Planungsstände durch Austausch möglichst aller relevanten Informationen im Vordergrund (vgl. [Han97b, Liu96]). Diese Grundidee entspricht auch der aktuellen Entwicklung des integrierten, sich abwechselnden und ergänzenden Einsatzes von Verfahren aus OLAP und Data Mining, EDA und schließender Statistik, interaktiver Visualisierung und automatisierter Mustersuche (vgl. Abschnitt 2.1.1, speziell [Han97a, Wro98]).

Im Rahmen intelligenter Datenanalyse sollen in geeigneter Weise Fähigkeiten von Mensch und Computer kombiniert, d. h. die einzelnen Teilaufgaben (etwa Erkennung von Strukturen, Hypothesengenerierung und effiziente Berechnungen) im Rahmen einer Analyse geeignet auf diese beiden Aufgabenträger aufgeteilt werden. Datenanalyseumgebungen sollten in Gestalt eines Assistenten ein explorativ es Vorgehen unterstützen, komfortablen Zugriff auf alle relevanten Verfahren bieten sowie gerade dem Nicht–Experten die nötige Hilfestellung zu deren korrekter Anwendung gewähren bzw. dem Menschen jeweils schon bei der Wahl des nächsten Analyseschrittes helfen. Grundlage hierfür bildet die klare Vorgabe von Analysezielen[14] sowie die Formalisierung von Strategien. Wichtige Hilfsmittel zur Lösung realer Problem bilden praxisnahe Modelle und Vorgehensweisen, effiziente Datenpräsentationen — insbesondere adäquate Verfahren zur interaktiven Datenvisualisierung — sowie die Datensätze beschreibende Metadaten, die eine sinnvolle Selektion von Analyseverfahren ermöglichen [Han97b, BH99].

Diese grundlegenden Forderungen und Konzepte sind die Basis dafür, „unintelligente Datenanalysen“ zu vermeiden, deren Gefahren sich aus der zunehmenden Verfügbarkeit von Analysesystemen auch für „Statistik–Laien“ sowie der Mächtigkeit, aber auch Komplexität neuartiger Analysemethoden ergeben. (Vergleiche hierzu

[14] Dies steht nicht im Widerspruch zu den Grundideen explorativer Datenanalyse. Auch hier sind stets mehr oder weniger allgemeine Fragestellungen sowie bestimmte Arten interessierender Muster vorgegeben, die natürlich noch im Laufe der Analyse verfeinert oder ergänzt werden können. Wenn sich der Grad einer Zielorientierung auch nur schwer festlegen und messen läßt, so führt eine *völlig* orientierungslose Suche doch nur sehr selten zu einem sinnvollen Ergebnis.

auch [FL90], wo die Bedeutung der graphischen Datenanalyse als den „Gefahren“ der Computer–Nutzung entgegensteuerndes Element herausgestellt wird.)

Letztendlich spielt immer noch der Mensch die Schlüsselrolle bei der Extraktion von Wissen aus Daten [SSW96] — zum einen, weil er es ist, der die interessierenden Fragestellungen formuliert, im Laufe einer Analyse evtl. auch noch modifiziert und die Relevanz von Ergebnissen aufgrund schwer zu formalisierenden Hintergrundwissens einstufen kann, und zum anderen aufgrund seiner überragenden und (auf absehbare Zeit immer noch) dem Rechner überlegenen kognitiven Fähigkeiten. Und nur wenn bei ihm ein gewisses Grundverständnis für die in diesem Abschnitt dargelegten Grundideen der (explorativen) Datenanalyse besteht, kann er die Möglichkeiten eines rechnerbasierten Analysewerkzeugs — sei es noch so „intelligent“ — richtig nutzen und die Bedeutung von explorativ generierten Hypothesen und so gefundenen Strukturen richtig einschätzen: nicht als finale Erkenntnisse, als nahezu beliebig erzeugte Ergebnisse, sondern stets nur als *Bausteine* zum Verständnis des zugrundeliegenden Datenraums.

2.2 Visuelle Programmierung

Mit der Entwicklung immer leistungsfähigerer Hard- und Software sucht man etwa seit Ende der 70er Jahre verstärkt nach Möglichkeiten, Programme verständlicher sowie Programmierung „natürlicher“ und damit für größere Anwendergruppen zugänglich zu machen [TG86]. Ausgehend von der Erkenntnis, daß dem Menschen komplexe Zusammenhänge und Abläufe viel besser und effizienter über anschauliche visuelle Darstellungen als über abstrakte Beschreibungen vermittelt werden können, soll die Konstruktion von Programmen idealerweise durch „Zusammenstecken und Ausprobieren“ erfolgen. Hierbei kommen gegenüber der klassischen, textuellen Programmierung vorwiegend graphische Notationen und interaktiv manipulierbare Softwarekomponenten zum Einsatz. Als Bezeichnung für derartige Ansätze hat sich der Begriff „visuelle Programmierung“ etabliert [Sch98].

Ähnlich wie die Datenvisualisierung eine zentrale Rolle für die (explorative) Datenanalyse spielt (vgl. Abschnitt 2.1), so zielt auch die visuelle Programmierung auf eine adäquatere, „bessere“ Informationsvermittlung zwischen Mensch und Rechnersystem. Die Visualisierung von *Daten* wird nun im Rahmen der Erstellung von Software fortgeführt zur Visualisierung von Programm*strukturen* und *-abläufen*.

Bevor in den Abschnitten 2.2.2 bis 2.2.5 näher auf einzelne Aspekte visueller Programmierung (insbesondere verschiedene visuelle Programmierparadigmen sowie Argumente für und gegen ihre Nutzung im allgemeinen und speziell in der Datenanalyse) eingegangen wird, wird zunächst in Abschnitt 2.2.1 konkretisiert, was unter dem allgemeinen Begriff „visuelle Programmierung“ verstanden werden kann und soll.

2.2.1 Begriffsbildung

Generell läßt sich — der Begriffswelt der Kognitionspsychologie entsprechend, die zwischen der Verarbeitung visueller und verbaler Information durch den Menschen unterscheidet [And89] — die *visuelle* von der *verbalen (textuellen) Programmierung* durch die Art der Verwendung *visuell informativer* Elemente abgrenzen. Hiermit sind nach Schiffer [Sch98] solche Komponenten bzw. Attribute gemeint, die (ohne vorherige „Übersetzung“) *ausschließlich* visuell wahrgenommen werden können, z. B.

- graphische Komponenten (Diagramme, Farben etc.),
- geometrische Attribute (Form, Größe etc.),
- topologische Eigenschaften (Verbindungen, Überlagerungen, Berührungen etc.) oder
- typographische Merkmale (Seitengestaltung, Schriftart etc.).

Begründet in der entsprechenden graphischen Anordnung von Programmierkonstrukten wird oft auch von *zwei-* oder *mehrdimensionaler*[15] *Programmierung* im Gegensatz zur traditionellen *linearen* oder *eindimensionalen*

[15] Tatsächlich gibt es Arbeiten, die versuchen, drei oder sogar noch mehr Dimensionen zur visuellen Programmierung zu nutzen, z. B. [Naj94].

Programmierung gesprochen [Mye86].

Der Begriff der visuellen Programmierung ist sehr weit gefaßt. So lassen sich hierunter im weitesten Sinne alle Aspekte graphischer Benutzungsschnittstellen — als ein visuelles Angebot aktivierbarer Befehle — betrachten [CKLI94]. Gerade vor diesem Hintergrund verwässern unzählige Publikationen aus den unterschiedlichsten Forschungsbereichen die klare Definition dessen, was den Kern visueller Programmierung ausmacht.

In der Fachliteratur werden visuelle Programmierung sowie visuelle Programmiersprachen in einer Vielzahl von Variationen über die Verwendung „bedeutungsvoller“ graphischer oder visueller Elemente im Programmierprozeß definiert (siehe u. a. [Shu86, Mye86, TG86, Cha87, CKLI94, BGL95, Men96]), wobei jedoch die Abgrenzung zu

- der Verwendung graphischer Softwareentwicklungsumgebungen mit komfortablen Editoren bei der Programmierung,
- der Programmvisualisierung, also der reinen *Darstellung* von Programmstruktur oder -ablauf (und nicht der *Konstruktion* von Programmen),
- der Programmierung graphischer Benutzungsoberflächen oder
- der Nutzung informeller Notationen, die nicht vom Rechner verarbeitet werden können und sollen,

ebenso wie die genaue Definition des Begriffs „bedeutungsvoll“ und der Umfang der Berücksichtigung textueller Elemente im Programmierprozeß sehr unterschiedlich und oftmals auch unklar formuliert sind.

Entsprechend werden derartige Aspekte auch in gängigen Klassifikationen visueller Programmierung einbezogen. So unterscheidet etwa der Ansatz von Shu aus [Shu86] (vgl. Abb. 2.11) auf oberster Ebene[16] nach

- visuellen Umgebungen
  - zur Visualisierung von Daten oder Datenstrukturen,
  - zur Programmvisualisierung, also der Illustration der Struktur eines Programms oder der Programmanimation zur Analyse von Laufzeitkomplexität, Ressourcenbedarf und algorithmischen Aspekten, oder
  - zur Unterstützung von Softwareentwurf und Anforderungsspezifikation im Rahmen des Software–Engineering–Prozesses
- und visuellen Sprachen
  - zur Bearbeitung visueller Daten (mit anderen Worten: zur Bildverarbeitung),
  - zur Unterstützung visueller Interaktion (bei Shu speziell zum Entwurf von Piktogrammen, in ähnlich aufgebauten Klassifikationen auch oft zur Gestaltung von Benutzungsoberflächen) oder
  - zur Programmierung mit visuellen Ausdrücken.

Während hier die ersten beiden Klassen „visueller“ Sprachen beliebige (auch textuelle) Programmiersprachen umfassen, die allein visuelle Informationen verarbeiten und selbst i. a. kaum das Attribut „visuell“ verdienen, ist die letztgenannte Klasse diejenige, die die eigentliche *visuelle* Programmierung erlaubt. Auch in den genannten visuellen Umgebungen bezieht sich der Begriff „visuell“ allein auf die Form der dargestellten Informationen, wobei natürlich oft auch die Verwendung graphischer Benutzungsoberflächen eine große Rolle spielt.

In ähnlicher Weise definiert Chang in [Cha87] eine Klassifikation visueller (Programmier–)Sprachen danach, ob die Sprachkonstrukte und/oder die verarbeiteten Objekte inhärent visuell sind, woraus sich die vier Klassen

1. Sprachen zur Unterstützung visueller Interaktion[17],
2. visuelle Programmiersprachen,
3. Sprachen zur Verarbeitung visueller Informationen und

[16] Eine weitere Verfeinerung dieses Schemas findet sich in [Shu88]. Hierauf kommen wir auch in Abschnitt 2.2.3 nochmals zurück.

[17] Wieso im Rahmen dieser Klassifikation nochmals eine Abgrenzung zu beliebigen verbalen Programmiersprachen erfolgt, die nicht–visuelle Informationen verarbeiten, wird nicht genauer motiviert.

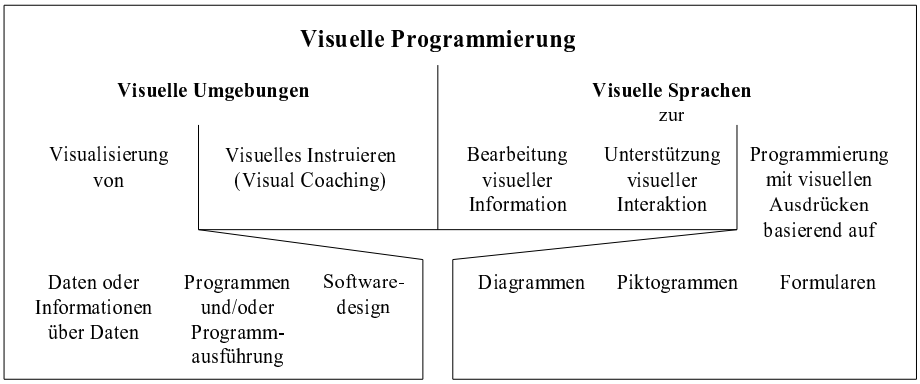

Abbildung 2.11: Klassifikation visueller Programmierung nach Shu

4. Piktogrammsprachen zur Verarbeitung visueller Informationen[18]

ergeben. Chang stellt fest, daß in den betrachteten Sprachen generalisierte *Piktogramme (Icons)* sowohl für Datenobjekte als auch für Prozesse (Sprachkonstrukte) definiert werden, die jeweils einen visuellen und einen logischen Aspekt (eine Bedeutung) aufweisen. Diese beiden Aspekte sind in den verschiedenen Sprachklassen für Daten und Prozesse unterschiedlich ausgeprägt bzw. werden durch spezifische Abbildungen auseinander bestimmt. Eine *visuelle Sprache* definiert Chang auf dieser Basis allgemein als eine Menge ikonischer Sätze über einem gegebenen Iconsystem.

Auch in [TG86] werden visuelle Sprachen über die Integration von graphischen Darstellungen in eine Programmiersprache zur symbolischen Repräsentation von Daten, Operationen, Kontrollstrukturen und Abläufen sowie die Nutzung visueller Komponenten zur Gestaltung eines graphischen Editors als geeignete Benutzungsschnittstelle beschrieben. Die Hauptziele der Entwicklung visueller Programmiersprachen werden auf dieser Grundlage in [MT90] unter den Gesichtspunkten der

- Syntax (der übersichtlichen Darstellung der Beziehungen zwischen graphischen Elementen),
- Semantik (einer möglichst klaren Bedeutung einer Komponente) und
- Pragmatik (der Nutzung geeigneter bekannter Metaphern aus dem jeweiligen Anwendungsgebiet)

zusammengefaßt.

Den in dieser Arbeit folgenden Betrachtungen soll die Definition *visueller Programmiersprachen* zugrunde gelegt werden, die Schiffer in [Sch98] als Zusammenfassung in der Literatur angeführter Begriffsbildungen gibt:

> VISUELLE PROGRAMMIERSPRACHEN *sind Programmiersprachen mit visuell informativen Elementen, die syntaktisch oder semantisch bedeutungstragend sind.*

In diesem Sinne ist z. B. ein Pfeil zwischen zwei Knoten eines Datenflußdiagramms oft von syntaktischer, seine eventuelle Einfärbung dagegen meist von semantischer Bedeutung.

Da in vielen Fällen nicht klar zwischen einer visuellen Programmiersprache und der jeweils zu ihrer Nutzung bereitgestellten Umgebung zu unterscheiden ist — so sind z. B. in dem visuellen Programmiersystem *Prograph* [CP88] Fenster bedeutungstragende Sprachelemente —, wird hier keine gesonderte Definition *visueller Programmierumgebungen* oder *-systeme* gegeben. Im Gegensatz zu obigen Klassifikationen liegt einer visuellen Programmierumgebung also stets eine visuelle und nie eine verbale Programmiersprache zugrun-

[18] Hiermit wird lediglich eine Untermenge visueller Programmiersprachen nach Shu hervorgehoben.

de.[19] In diesem Sinne ist auch *visuelle Programmierung* einfach als Programmierung unter Verwendung einer visuellen Programmiersprache zu verstehen — auch dies eine gegenüber den Klassifikationen von Shu und Chang deutlich eingeschränkte Betrachtungsweise. Die Berechnungsvollständigkeit (Turing–Mächtigkeit) der betrachteten Sprache soll hier keine besondere Rolle spielen (vgl. etwa [Bur95]).

Weiterhin werden auch *visuelle Sprachen*, die nicht zur Programmierung, also zur vollständigen Beschreibung der Eigenschaften von durch ein Rechnersystem verarbeiteter Software, sondern allein zur Kommunikation von Informationen entwickelt sind, nicht näher betrachtet.

Unter *Programmvisualisierung* soll im weiteren sowohl die *Programmanimation* als auch die visuelle Darstellung von Programmdaten oder -strukturen verstanden werden.

In der allgemeinen System- und Anwendungsentwicklung hat sich die visuelle Programmierung sicherlich noch nicht durchgesetzt [CKLI94]. Hierzu spielen für eine effiziente und effektive Programmentwicklung i. a. doch weit mehr Kriterien ein wichtige Rolle als nur die „bessere“ Wahrnehmung visueller Darstellungen gegenüber abstrakten Beschreibungen (wobei auch dieser Aspekt sicherlich noch differenzierter zu sehen ist). In Abschnitt 2.2.2 werden einige Vor- und Nachteile, Möglichkeiten und Probleme visueller Programmierung näher beleuchtet.

Gemäß obiger Definition visueller Programmierung wird weiterhin deutlich, daß es — je nach Ausmaß des Einsatzes visueller Elemente — auch zwischen verbaler und visueller Programmierung eine Vielzahl von Abstufungen des Grades der Visualisierung geben kann: begonnen bei der Nutzung von Fettschrift oder Farben bis zur Programmierung völlig ohne Text. Abschnitt 2.2.3 stellt einige Klassifikationen visueller Programmierung vor, die sich vor allem auf die graphischen Mittel zur Repräsentation von Programmkomponenten und die jeweils unterliegenden visuellen Programmierparadigmen konzentrieren.

Die Sinnhaftigkeit und Nützlichkeit der vielfältigen möglichen Konzepte hängt sicherlich jeweils auch am zu unterstützenden Anwendungsgebiet. So haben, wenn auch — wie oben dargestellt — der Erfolg auf breiter Front bisher ausgeblieben ist, einzelne Anwendungsdomänen gerade in den letzten Jahren auch kommerziell erfolgreiche visuelle Programmiersysteme hervorgebracht, die gut auf die jeweiligen Erfordernisse abgestimmt sind. Zu nennen sind hier etwa

- die Entwicklung von (oft prototypischen) Systemen mit komplexen Benutzungsoberflächen und relativ einfacher Anwendungslogik, z. B. mit *VisualAge* von *IBM*,[20]
- die Bereitstellung (meist deklarativer) visueller Anfragesprachen für Datenbanksysteme (vgl. [CCLB97, DRR+96, Lar86] für einen Überblick),
- die Erstellung multimedialer Präsentationen oder interaktiver multimedialer Anwendungen durch Autorenwerkzeuge [Bol95, BS98] oder
- Geschäftsprozeßmodellierung und Workflow–Management, z. B. auf Basis ereignisgesteuerter Prozeßketten [VB96, Rum99].

Auch in der Datenvisualisierung und -analyse werden zunehmend visuelle, auf dem Datenflußparadigma basierende Programmierumgebungen eingesetzt. In Abschnitt 2.2.4 wird dieses Paradigma, das auch der Analyseumgebung *VIOLA* zugrunde liegen wird, genauer vorgestellt und seine Eignung zur Unterstützung der Datenanalyse diskutiert. Einige Beispielsysteme finden sich dann in Abschnitt 3.4.

Abschnitt 2.2.5 geht schließlich noch auf einige aktuelle Forschungsthemen im Bereich der visuellen Programmierung ein. Für eine umfassende Einführung in die visuelle Programmierung sei auf [Sch98] verwiesen.[21] Einen Einblick in aktuelle Forschungsaktivitäten bieten die Tagungsbände des „*IEEE Workshop*

[19] Natürlich kann diese visuelle Sprache vor oder im Rahmen der Ausführung eines visuellen Programms auch zunächst in eine höhere verbale Sprache übersetzt werden, anstatt den Code direkt zu interpretieren oder zu compilieren.

[20] Systeme wie *Visual C++* von *Microsoft*, durch die über entsprechende Editoren allein die Benutzungsoberflächen visuell programmiert werden, die jedoch weiterhin eine verbale Codierung des eigentlichen Anwendungsprogramms erfordern, sind hier aufgrund fehlender Berechnungsvollständigkeit der klar abgegrenzten visuellen Teilsprache nicht als typische Beispiele anzusehen.

[21] An diesem Werk orientieren sich auch weite Teile dieses Abschnitts.

(Symposium) on Visual Languages“, der erstmals 1984 abgehalten wurde, sowie das „*Journal on Visual Languages and Computing*“, das 1990 erstmal erschien, sowie die großteils immer noch aktuellen Sammelbände [Gli90a, Gli90b, BM95].

2.2.2 Vor- und Nachteile visueller Programmierung

Der Einsatz visueller Programmierung wird allgemein vor allem aus Erkenntnissen der Kognitionspsychologie motiviert, nach der visuelle Darstellungen vom Menschen besonders wirkungsvoll verarbeitet werden können [And89] (vgl. die Diskussion der Datenvisualisierung in Abschnitt 2.1.3). Diese Überlegungen fußen auf zwei grundlegenden Sachverhalten:

1. Analytische Aufgaben und visuelle Eindrücke sprechen in unterschiedlicher Weise die Fähigkeiten des menschlichen Gehirns zur Informationsverarbeitung an — die linke Gehirnhälfte unterstützt sequentielle Verarbeitungsprozesse, die rechte parallele.[22] Es erscheint sinnvoll, bei der analytischen Problemlösung unter Nutzung visueller Informationen die Fähigkeiten beider Gehirnhälften zu kombinieren und somit höhere Verarbeitungskapazitäten auszuschöpfen sowie durch Parallelisierung effizienter zu sein.

2. Der Mensch löst die Welt um sich herum in Grundeinheiten und Beziehungsgeflechte zwischen diesen auf. Es wird stets versucht, neue Eindrücke durch Einordnung in diese semantischen (propositionalen) Netze auf bereits bekannte Konzepte abzubilden — je besser dies gelingt, umso nachhaltiger wird die neue Information aufgenommen. Diese Einordnung von Informationen ist, so gut es geht, durch visuelle Strukturierungsmittel zu unterstützen, wobei die Verwendung intuitiver Metaphern äußerst hilfreich sein kann.

Darüber hinaus gehen Green und Petre in [GP96] näher auf die „Psychologie der Programmierung“ ein: Jeder Programmierer baut sich eine mentale Repräsentation eines betrachteten Programms auf — diesen Prozeß gilt es durch Verdeutlichung von Programmstruktur und Zusammenhängen zwischen Komponenten sowie eine möglichst einfache und direkte Abbildung zwischen Problemstellung und Programm in einer visuellen Programmierumgebung zu fördern. Auch die direkte Interaktion mit den Programmkomponenten bei der Entwicklung und im Rahmen von Testläufen kommen dem Programmverständnis zugute. Die Entwicklung eines Programms erfolgt i. a. nicht linear, sondern durch Einfügen neuer Teile in ein bestehendes Gerüst oder Teilprogramm. Diese Vorgehensweise kann ebenfalls durch direkte Manipulationsmöglichkeiten in einer Umgebung, die den Wechsel zwischen verschiedenen Abstraktionsebenen gestattet, unterstützt werden. Schließlich wird auch darauf hingewiesen, daß alle Notationen — egal ob textuell oder visuell — jeweils bestimmte Aspekte der darzubietenden Informationen betonen und andere zugleich vernachlässigen. Das Ziel muß es sein, jeweils auch die verborgene Sicht (zumindest potentiell) sichtbar zu machen.

Auf den Aspekt der direkten Manipulation von Objekten einer Benutzungsschnittstelle kommt Shneiderman in [Shn83] genauer zu sprechen: Direkte Manipulation macht den Computer „transparent“, indem sie dem Benutzer erlaubt, sich auf seine eigentliche, mit *Hilfe* des Rechners bearbeitete Aufgabe zu konzentrieren. Sie vermittelt ein Gefühl der Systembeherrschung, Kompetenz und Leichtigkeit sowie Spaß am Erlernen und Benutzen eines Rechnersystems. Neben den Aspekten der schnellen Erlernbarkeit und subjektiven Zufriedenheit nennt Shneiderman als zentrale Ziele einer software–ergonomisch guten Benutzungsschnittstelle auch die Effizienz der Aufgabendurchführung, geringe Fehlerraten und gute Erinnerung der Systembedienung [Shn87]. All dies sind natürlich gleichermaßen auch Ansatzpunkte zum Einsatz visueller Programmierung.

Eine Vielzahl von Aussagen und in der Literatur aufgestellten Thesen zu Vorzügen und speziellen Zielen visueller Programmierung sind in [Sch98, Bla96, Men96, TG86] zusammengestellt. Einen Überblick über eine ganze Reihe empirischer Studien zur Über- (oder Unter-)legenheit graphischer gegenüber textuellen Notationen gibt [Whi97]. Eine verbreitete pauschale These lautet:

[22] Sicherlich gibt es von Mensch zu Mensch trotzdem signifikante Unterschiede in der Art und Geschwindigkeit der Verarbeitung bestimmter Informationen sowie in der Herangehensweise an bestimmte Probleme.

Visuelle Elemente vereinfachen die Programmierung und führen zu „besseren“ Programmen.

Diese Annahme ist in dieser allgemeinen Form sicher zu naiv und leicht zu widerlegen — andernfalls wären visuelle Programmiersprachen in der allgemeinen Programmierung heute sicher viel weiter verbreitet. Dagegen sprechen auch technische Eigenschaften von Software und ihrer Beschreibungsmittel, die sich der bildlichen Repräsentation prinzipiell entziehen.

Da eine pauschale Auflistung von Vor- und Nachteilen visueller Programmierung zumindest schwierig zu differenzieren, wenn nicht gar aufgrund der Breite möglicher Einsatzgebiete irreführend wäre, wird nachfolgend statt dessen in Anlehnung an [Sch98] anhand von fünf „klassischen“ Argumenten für den Einsatz visueller Programmierung diskutiert, unter welchen Umständen graphische Darstellungen textuellen Notationen überlegen sind:

1. Argument: Visuelle Programme erleichtern die Kommunikation. Gerade in der Kommunikation zwischen Anwender bzw. Auftraggeber und Programmierer sowie im Übergang von der Entwurfs- in die Implementierungsphase kann die visuelle Programmierung eine *gemeinsame* Sprache definieren. Über ausdrucksstarke Bilder lassen sich Informationen besser, kompakter und eleganter vermitteln.

Die Normierung der Informationsdarstellung ist hierbei jedoch schwierig. Abstrakte Sachverhalte sind zudem oftmals kaum visuell umsetzbar. Weiterhin können visuelle Darstellungen nicht so präzise wie textuelle Notationen (etwa klare Bezeichner) sein und können leicht von den zu vermittelnden Kernaussagen ablenken (vgl. [Tuf83] zur Gestaltung informativer Graphiken).

Insgesamt erscheint eine Kombination von Graphik und Text auf der Basis klarer, jeweils im Einzelfall festzulegender bzw. zu nutzender Konventionen sinnvoll, wobei insbesondere Dateninhalte sowie grobe Abläufe und Strukturen gut visuell vermittelt werden können.

2. Argument: Visuelle Programme sind leicht verständlich. Bilder gelten, gerade für Ungeübte und Nicht–Programmierer, als ein natürliches Kommunikationsmittel. Nach [Tuf91] können durch ihren Einsatz — verglichen mit textuellen (tabellarischen) Darstellungen — komplexe Daten und Zusammenhänge oftmals deutlich leichter interpretiert werden.

Vor allem Liniendiagramme werden jedoch schnell unübersichtlich, semantische Details sind nur schwer zu codieren und korrekt zu interpretieren. Wiederum sind die Stärken visueller Programmierung somit eher im Bereich von überblicksartigen Entwurfs- und Architekturskizzen sowie in Kombination mit einer Programmvisualisierung anzusiedeln.

3. Argument: Visuelle Programmierung ist leicht erlernbar. Visuelle Programmierumgebungen bieten einen fast schon spielerischen Umgang mit der Programmierung und können aufgrund direkter Manipulation und meist einfacher Syntax leichter motivieren und größere Anwendergruppen ansprechen.

Es besteht jedoch die Gefahr der Frustration, wenn die visuelle Sprache nicht mächtig genug ist. Weiterhin kann der interaktive Umgang gerade für größere Programme zu umständlich werden. Schließlich wird auch der Prozeß der abstrakten Problemlösung nicht gefördert — gerade komplexe Probleme führen schnell zu schlecht strukturierten Lösungen.

4. Argument: Visuelle Programmierung überwindet Sprachbarrieren. Bilder und Piktogramme können zwar international verständlich gestaltet werden, erreichen aber nicht den gleichen Standardisierungsgrad und die gleiche Aussagekraft wie die verbale Sprache.

5. Argument: Visuelle Programmierung heißt „optimale“ Kognition. Gemäß der zu Beginn dieses Abschnitts dargestellten kognitionswissenschaftlichen Erkenntnisse können visuelle Programme effizienter wahrgenommen und besser erinnert werden. Dies steht jedoch bei der Programmierung weniger im Vordergrund als ein guter Überblick über das jeweilige Programm. Zu dessen Gewährung (dem direkten Zugriff auf alle Programmteile sowie dem Wechsel zwischen Detail- und Gesamtsicht) bietet visuelle

Programmierung zwar gute Möglichkeiten, visuelle Programme sind jedoch nicht an sich gut strukturiert (vgl. Abb. 2.12).

Die geringe Darstellungsdichte erfordert außerdem oft eine künstliche Programmaufteilung mit unklarer Leserichtung bzw. aufwendiger Navigation (das oft zitierte *Screen–Space–Problem*). Letztlich erschweren auch gerade Probleme der prägnanten visuellen Codierung wesentlicher Details die schnelle Erfassung und dauerhafte Erinnerung von Programmstruktur und -semantik.

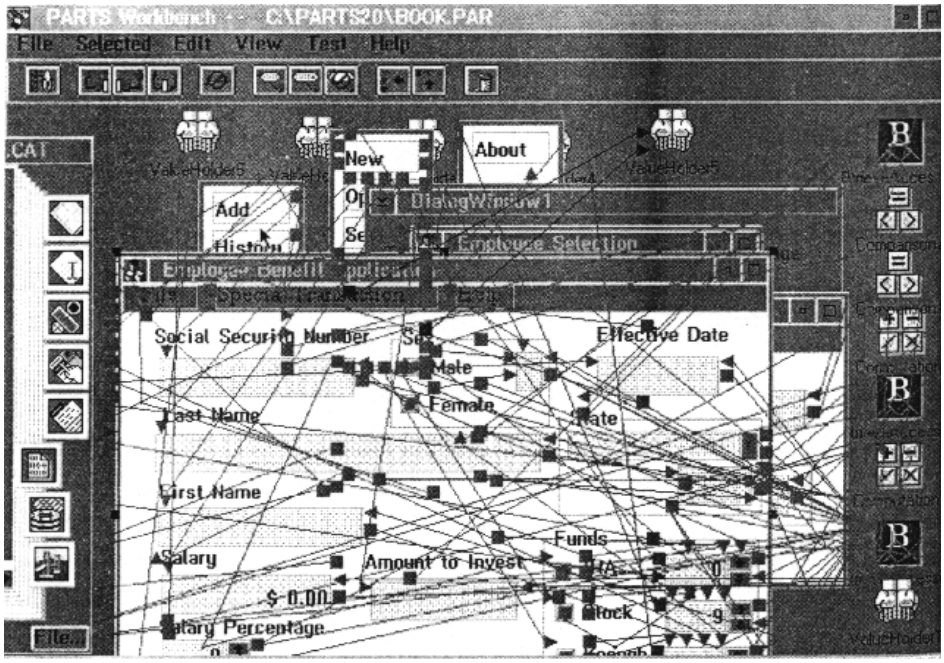

Abbildung 2.12: Ein „einfaches“ Programm in der objektorientierten visuellen Programmiersprache *PARTS* (aus [Sch98])

Somit wird klar: Ein Bild sagt nicht immer mehr als 1000 Worte — Bilder können durch dekorative Elemente auch „zu viel sagen“, auch ein Wort kann aufgrund seines höheren Abstraktionsgrades ausdruckskräftiger sein und „mehr als 1000 Bilder sagen“, und schließlich kann ein Bild unterschiedlichen Betrachtern aufgrund fehlender Exaktheit (und die steht ja gerade im Zentrum der Programmierung) auch oft „zu Unterschiedliches sagen“ [Sch98]. Gerade bei der Entwicklung großer Programme wird der überproportionale Komplexitätszuwachs durch visuelle Programmierung deutlich. Bestimmte Aspekte von Software können nicht oder nur schlecht bildlich repräsentiert werden (z. B. Rekursionen). Das Screen–Space–Problem, die Gefahr unübersichtlicher Darstellungen sowie die häufig ineffiziente Verarbeitung sind als weitere potentielle Nachteile visueller Programmierung zu nennen.

Das Hauptziel visueller Programmierung kann i. a. nicht die bessere Verständlichkeit sein [KHA97]. Die visuelle ist der verbalen Programmierung weder generell überlegen noch intuitiver und nutzt auch die Kapazitäten des menschlichen Gehirns nicht prinzipiell besser aus [Bla96]. Sie bietet jedoch auf höherem Abstraktionsniveau sehr gute Möglichkeiten zur Strukturierung und — auf der Basis expliziter Repräsentation von Teilkomponenten und deren direkter Manipulation — zum Verständnis von Problemen und Problemlösungen. Weitere Potentiale liegen in der Beschränkung auf wenige syntaktische Konzepte und im Rückgriff auf über die formalen Sprachbestandteile hinausgehende *sekundäre Notationen*, wie z. B. Layout und Kommentierung [BBB+95, GP96, Whi97]. Hierbei ist auch über eine geeignete Kombination von Text und Graphik nachzudenken.

Vor- und Nachteile visueller Programmierung sind stark nutzer- und problemabhängig. Neue Erkenntnisse zur Sinnhaftigkeit des Einsatzes visueller Programmierung sind vor allem aus weiteren empirischen Studien zu erwarten [Whi97].

2.2.3 Klassifikation und Evaluation visueller Programmiersprachen

In diesem Abschnitt werden verschiedene häufig referenzierte Klassifikationen visueller Programmierung sowie Kriterienkataloge, die neben ihrer Einordnung auch zugleich zur Evaluation visueller Programmiersprachen dienen können, vorgestellt. Einige von ihnen werden in Kapitel 5 auch zur Bewertung von *VIOLA* herangezogen werden.

Die bereits in Abschnitt 2.2.1 angesprochenen Klassifikationen von Shu und Chang dienen im wesentlichen dazu, visuelle Programmiersprachen „im engeren Sinne“ von benachbarten Themenfeldern abzugrenzen. Darüber hinaus unterscheidet Shu noch näher nach dem Erscheinungsbild der jeweiligen visuellen Darstellung [Shu86] (siehe Abb. 2.11). Hiernach basieren visuelle Programmiersprachen auf

- Diagrammen (der imperativen Programmierung entsprechend, etwa Steuerfluß- oder Zustandsüberführungsdiagramme),
- Piktogrammen (hiermit wird eine sehr breite Palette abgedeckt) oder
- Formularen.

Diese Klassifikation vernachlässigt das der Programmiersprache jeweils zugrundeliegende konzeptionelle Modell; unklar ist etwa die Einordnung von Datenflußprogrammen, die Piktogramme innerhalb eines (Datenfluß–)Diagramms nutzen. Beispielorientierte visuelle Programmierung wird wiederum — ohne Berücksichtigung der etwaigen Verwendung von Diagrammen, Piktogrammen oder Formularen — separat unter visuellen Umgebungen zum visuellen Instruieren klassifiziert.

Zusätzlich definiert Shu ein grobes dreidimensionales Koordinatensystem zur Evaluation visueller Sprachen im Hinblick auf Sprachniveau (Ausdrucksmächtigkeit), Anwendungsbereich (allgemein oder speziell) und Visualisierungsgrad (Anzahl visueller Konstrukte, auch im Verhältnis zu textuellen Konstrukten) (siehe Abb. 2.13). Eine klare Metrik zur Einordnung konkreter Sprachen liegt jedoch hierfür nicht vor.

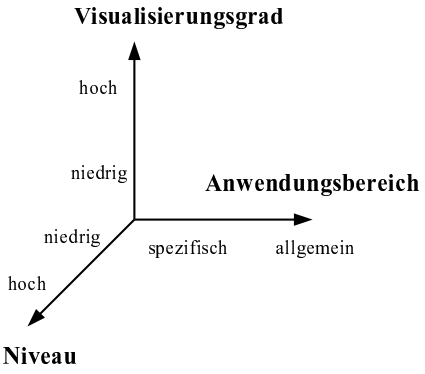

Abbildung 2.13: Ein Koordinatensystem zur Bewertung visueller Programmiersprachen nach Shu

Ein ähnliches Bewertungssystem wird von Menzies in [Men96] vorgeschlagen, wo nach Art der visuellen Ausdrücke (Diagramme, Formulare, etc.), Zweck (Visualisierung, visuelle Programmierung, Animation, etc.) und Programmierparadigma unterschieden wird. Letzteren Aspekt werden wir weiter unten noch vertiefen.

Myers unterteilt in [Mye86, Mye90] (visuelle) Programmiersysteme — etwas eigenwillig — danach in acht Klassen, ob sie beispielorientiert (oder nicht), visuell (oder nicht)[23] sowie compilierend oder interpretierend sind. Unter dieser Hauptklassifikation erfolgt eine Differenzierung nach Spezifikationstechnik bzw. Programmnotation — und zwar nach textuellen Sprachen, Steuerflußdiagrammen und Varianten davon, Petrinetzen, Datenflußdiagrammen, gerichteten Graphen und Varianten davon, Matrixsprachen, Puzzlediagrammen, Formularen, Piktogrammsätzen, Tabellenkalkulationen und beispielhaften Notationen (Programmierung durch dynamische Sprachmittel, etwa Drag–and–Drop). Außerdem gibt es noch die nicht näher spezifizierte Rubrik „keine Sprache“.

Anders als bei Shu und Myers spielt das jeweilige Programmierparadigma bei Burnett und Baker [BB94] eine zentrale Rolle. Das Ziel ihrer Klassifikation (siehe Tab. 2.1) ist vor allem die systematische Erfassung

VPL:	**VISUELLE PROGRAMMIERSPRACHEN**
VPL-I.	**Umgebungen und Werkzeuge für visuelle Programmiersprachen**
VPL-II.	**Sprachklassifikationen**
A.	Paradigmen
1.	Sprachen für Parallelprogrammierung
2.	Constraintbasierte Sprachen
3.	Datenflußsprachen
4.	Formular- und tabellenbasierte Sprachen
5.	Funktionale Sprachen
6.	Imperative Sprachen
7.	Logische Sprachen
8.	Multiparadigmen–Sprachen
9.	Objektorientierte Sprachen
10.	Beispielorientierte Sprachen
11.	Regelbasierte Sprachen
B.	Visuelle Repräsentationen
1.	Diagrammsprachen
2.	Piktogrammsprachen
3.	Sprachen basierend auf statischen Bildsequenzen
VPL-III.	**Spracheigenschaften**
A.	Abstraktion
1.	Datenabstraktion
2.	Prozedurale Abstraktion
B.	Steuerfluß
C.	Datentypen und -strukturen
D.	Dokumentation
E.	Ereignisbehandlung
F.	Ausnahmebehandlung
VPL-IV.	**Fragestellungen zur Sprachimplementierung**
A.	Ansätze für Berechnungsmodelle (bedarfsgesteuert, datengesteuert etc.)
B.	Effizienz
C.	Parsen
D.	Interpreter und Compiler
VPL-V.	**Sprachzweck**
A.	Universalsprachen
B.	Datenbanksprachen
C.	Bildverarbeitungssprachen
D.	Visualisierungssprachen
E.	Sprachen zur Generierung von Benutzungsschnittstellen
VPL-VI.	**Theorie visueller Programmiersprachen**
A.	Formale Definition visueller Programmiersprachen
B.	Piktogrammtheorie
C.	Fragestellungen zum Sprachentwurf
1.	Kognition und Entwurf von Benutzungsschnittstellen
2.	Effiziente Nutzung von Bildschirmplatz
3.	Lebendigkeit
4.	Anwendungsbereich
5.	Typprüfung und Typtheorie
6.	Fragestellungen zur visuellen Darstellung (z. B. statische Repräsentation, Animation)

Tabelle 2.1: Das Klassifikationsschema für visuelle Programmiersprachen nach Burnett und Baker

von Literatur zur visuellen Programmierung in Anlehnung an das (und als unabhängige Ergänzung des) *ACM Computing Review Classification System* [ACM92].

Das Schlagwortsystem kann jedoch gleichermaßen zur Sprachklassifikation genutzt werden. Die Vorteile dieses Katalogs liegen vor allem in seiner Erweiterbarkeit und der flexiblen Kombinierbarkeit von Kriterien. Burnett und Baker schlagen eine Einordnung ihrer Klassifikation visueller Programmiersprachen auf einer übergeordneten Ebene der Bereiche „visuelle Datenbanksysteme“, „Sprachen zur Bildverarbeitung“, „visuelle Umgebungen für textuelle Sprachen“, „virtuelle Realität“ und „Visualisierung“ vor.

An dieser Stelle wollen wir lediglich die Klassifikation der Programmierparadigmen genauer erläutern; auf andere Aspekte wird teilweise noch in Abschnitt 2.2.5 eingegangen. Wie in [AB89] oder [Sch98] dargestellt, lassen sich Programmierumgebungen über die in Tab. 2.1 genannten Konzepte weiter gruppieren nach

- Systemen, die auf verbalen Sprachen basieren (steuerfluß-, funktions-, datenfluß-, objekt- und constraintorientierte Ansätze — letztere inklusive logischer Sprachen),

[23] Dieses überflüssig erscheinende Kriterium dient allein zur Identifikation beispielorientierter, nicht–visueller Systeme.

- Systemen, die auf visuellen Sprachen basieren, die also kein Gegenstück in der verbalen Programmierung haben und somit inhärent visuell sind (regel-, beispiel- und formularorientierte Ansätze), und
- Systemen für spezielle Bereiche (parallelitäts- und multiparadigmenorientierte Sprachen).

Im einzelnen können die genannten Sprachklassen der ersten beiden Gruppen[24] folgendermaßen charakterisiert werden:

Steuerflußorientierte Systeme entstammen dem imperativen Programmierparadigma. Die Abfolge von Befehlen wird spezifiziert über

- Anweisungssequenzen (u. a. in *Steuerflußdiagrammen* wie z. B. in *Pict* [GT84], durch *Nassi–Shneiderman–Diagramme* [NS73], *Puzzlediagramme* oder, wie in *VIPR* [CDZ94], durch geschachtelte Kreise),
- *Komponentennetze* aus *Softwarekomponenten* [dM94], die Ereignisse repräsentierende Token auf Kanälen zwischeneinander austauschen, oder
- *Transitionsnetze*, die *Zustandsdiagramme* von Automaten [HU90] oder *Petrinetze* [Rei91] beschreiben.

Datenflußbasierte Systeme beruhen auf dem Datenflußmodell: Verfügbarkeit und Anforderung von Daten bestimmen die Ausführungsreihenfolge, nicht ein übergeordneter Befehlszähler. Dieses Paradigma, das viele existierende visuelle Programmiersysteme verfolgen, wird in Abschnitt 2.2.4 näher beschrieben.

Funktionsorientierte Systeme korrespondieren mit der funktionalen Programmierung. Sie erweitern das Datenflußkonzept um Funktionen höherer Ordnung, polymorphe Funktionen, implizite Typsysteme und ähnliches, sind diesem ansonsten sehr nahe, interpretieren allenfalls die entsprechende Sichtweise *noch* strenger (siehe etwa [Pos94]). Ihre geringe Verbreitung legt nahe, daß das große Bedürfnis nach mathematischer Formalisierung besser durch verbale Programmiersprachen befriedigt werden kann.

Objektorientierte Systeme erweitern das Steuerflußparadigma im üblichen Sinne objektorientierter Modellierung und Programmierung: Programmbausteine sind Objekte, die Attribute besitzen und über Nachrichten gemäß visuell modellierter Verbindungen mit anderen Objekten kommunizieren. Ereignisverbindungen (Benutzerinteraktionen, Zustandsänderungen) verknüpfen Ereignisse mit Nachrichten, Resultatverbindungen verknüpfen Ergebnisse von Nachrichten mit Nachrichten, Argumentverbindungen beziehen Nachrichtenargumente auf ein Ergebnis, und Attributverbindungen gleichen Attribute ab. Beispiele bilden *VisualAge* von *IBM*, das einen graphischen Benutzungsschnittstellen–Editor mit einer objektorientierten visuellen Programmiersprache integriert, das System *Prograph* [CP88], das Datenfluß und Objektorientierung kombiniert, oder eine Erweiterung von *VIPR* [CDZ95].

Zu den bekannten Vorteilen objektorientierter Programmierung (Erweiterbarkeit, Wiederverwendbarkeit, Wartbarkeit etc.) kommt hier noch die direkte Manipulierbarkeit von Objekten. Aufgrund guter Möglichkeiten zur engen Verbindung von Problem, Lösungsentwurf und -realisierung[25] ist der objektorientierte Ansatz neben dem Datenflußmodell als das erfolgreichste visuelle Programmierparadigma anzusehen (vgl. [BGL95] sowie [RBM+92] für einen Überblick). Als Nachteile sind die Gefahr der Unübersichtlichkeit sowie Schwierigkeiten bei der Modellierung von Methoden, Vererbung, Objekterzeugung und Polymorphismus zu nennen.

Die vorangegangenen vier Paradigmen lassen sich — je nach Granularität der jeweiligen Bausteine — auch unter dem Blickwinkel der *komponentenbasierten Softwareentwicklung* [Gri98] betrachten. Diese wurde in

[24] *Multiparadigmenorientierte Sprachen* kombinieren einfach verschiedene andere Ansätze; *parallelitätsorientierte Systeme* sind zu divergent, um die ihnen zugrundeliegenden Konzepte hier kurz umreißen zu können.

[25] Auch die aktuelle Entwicklung im Software–Entwurf geht deutlich in die Richtung objektorientierter Modellierungssprachen, wie z. B. mit der *Unified Modelling Language (UML)*. Bei Verwendung entsprechender Entwurfsumgebungen, die Code–Generatoren integrieren, können derartige Sprachen auch als Kombination visueller und verbaler (objektorientierter) Programmierung angesehen werden.

den letzten Jahren vor allem als „Weiterentwicklung“ und unter Verwendung objektorientierter Programmierung stark vorangetrieben. Klar gekapselte, autonome und eigenständig nutzbare, aber vor allem auch (meist verteilt) in Kombination mit anderen Komponenten wiederverwendbare und adaptierbare *Softwarekomponenten* kommunizieren mit der Außenwelt über standardisierte Schnittstellen einer gegebenen Rahmenarchitektur, die hierzu Basisdienste bereitstellt. Beispiele für derartige Architekturen, sogenannte Middleware [Ber96], sind etwa *ActiveX* von *Microsoft* oder die *Common Object Request Broker Architecture* (*CORBA*) der *Object Management Group* (*OMG*).

Constraintorientierte Systeme modellieren Werte bzw. Variablen als Objekte und Constraints als Beziehungen zwischen diesen. Dieser Ansatz entspricht der deklarativen logischen Programmierung; dort werden Beziehungen als Relationen repräsentiert. Existierende Systeme (etwa *ThingLab* [Bor81] oder Ansätze von Menzies [Men96]) haben vor allem mit der ineffizienten Ausführung komplexerer Programme zu kämpfen, die außerdem auch schnell unübersichtlich werden.

Regelorientierte Systeme realisieren Programme als Folgen von Bildtransformationen, wie etwa in *ChemTrains* [BL93]. Sie sind sehr einfach zu nutzen, haben aber einen recht begrenzten Anwendungsbereich, wenn die unterliegenden Regelsysteme wiederum nicht zu komplex und unübersichtlich werden sollen. Ihre formale Grundlage (siehe etwa die Spezifikation über Graphgrammatiken in *PROGRES* [SWZ95]) macht sie trotzdem zu einem evtl. zukunftsweisenden Ansatz aus dem Bereich der inhärent visuellen Sprachen.

Beispielorientierte Systeme generieren Programme aus vom Benutzer vorgeführten Abläufen. Die Programmierung erfolgt nach [Mye90] entweder *mit* Beispielen (nur Übersetzung und Wiederholung der Benutzeraktionen, quasi durch einen Makrorekorder) oder *durch* Beispiele (Verallgemeinerung und Übertragung der Aktionen). Gerade letzterer Ansatz ist sicher sehr interessant, seine Umsetzung aber sehr problematisch und noch weit entfernt von wirklich nützlichen Anwendungen. Ein System zur Programmierung mit Beispielen ist *HI–VISUAL*, das von einer datenflußbasierten [HTI90] zu einer beispielorientierten Umgebung [MKHI95] weiterentwickelt wurde.

Formularorientierte Systeme schließlich beruhen auf dem Konzept der Tabellenkalkulation. Sie unterstützen in ihrem Grundmodell weder Datenabstraktion, algorithmische Konstrukte noch sequentielle Abläufe. Darüber hinausgehende Entwicklungen wie etwa *Forms/3* [BA94] machen eher einen unpraktischen Eindruck. Sinnvolle Anwendungen dieses Paradigmas finden sich im Datenbankbereich, etwa mit *QBE* (*Query by Example*) [Zlo80].

Stärker der Evaluation visueller Programmiersysteme widmen sich Arbeiten von Kiper und Green. In [KHA97] wird versucht, das recht grobe Evaluationsschema von Shu zu verfeinern und Metriken zur Systembewertung zu definieren. Visualisierungsgrad und Funktionalität (Turing–Mächtigkeit bzw. Adäquatheit für einen Anwendungsbereich) werden aus Shu's Dimensionen übernommen, das Sprachniveau wird — ähnlich wie bei Menzies — verallgemeinert zur Art und Anzahl der unterstützten Programmierparadigmen. Hinzu kommen die Dimensionen „Verständlichkeit“ (für unterschiedliche Nutzergruppen) und „Skalierbarkeit“ (Nutzbarkeit für „große“ Aufgabenstellungen). Für alle fünf Dimensionen werden recht einfache Metriken bzw. Spezialisierungen zu ihrer Bewertung angegeben; insgesamt bleibt das Schema jedoch recht ungenau.

Green und Petre wenden in [GP96] ein „System kognitiver Dimensionen“ zur Beurteilung von Programmierumgebungen auf visuelle Programmiersysteme an — deren Stärken und Schwächen werden im Hinblick auf kognitive Aspekte der „besseren“ Programmierung im Sinne von Abschnitt 2.2.2 bewertet:

- In welchem Maße dienen prozedurale sowie Daten– und Kontrollstrukturen–Abstraktionen der Strukturierung und Verständlichkeit?
- Wie klar und direkt ist die Abbildung zwischen Problemdomäne und Programm gelungen?
- Gewährt eine konsistente Sprachdefinition auf der Basis weniger grundlegender Konzepte gute Erlernbarkeit und intuitive Nutzung?

- Ist die Notation kompakt und prägnant und ermöglicht so eine einfache Programmierung sowie eine Reduzierung des Screen–Space–Problems?
- Werden durch das Sprachdesign, speziell durch syntaktische Feinheiten, Fehler in der Anwendung begünstigt?
- Gibt es „mental schwierige Operationen“, die gerade in kombinierter Anwendung nur schwer verständlich und nutzbar sind?
- Sind alle Abhängigkeiten ohne versteckte Nebeneffekte offen und klar dargestellt?
- Müssen Programmierer im Hinblick auf Layout, zu nutzende Konstrukte, Verbindungen etc. Entscheidungen fällen, bevor alle hierzu nötigen Informationen vorhanden sind?
- Ist zur fortschreitenden Evaluation die Ausführung von unvollständigen Programmen, also ein inkrementelles Programmieren, möglich?
- Ist die Rolle jeder Programmkomponente im Rahmen des Ganzen klar erkennbar?
- Werden sekundäre Notationen (neben der formalen Syntax und Semantik) für zusätzliche Kommentare und Erläuterungen genutzt (etwa Layout, Kommentare, Namen etc.)?
- Wie einfach sind „kleine“ Änderungen durchzuführen?
- Ist stets der gesamte Code sichtbar, oder sind zumindest stets zwei beliebige Programmteile gleichzeitig darstellbar?

Hierbei bestehen zwischen allen betrachteten Dimensionen jeweils unterschiedlich starke, oft konkurrierende Abhängigkeiten; jede Sprache realisiert notwendigerweise einen Trade–off zwischen den Kriterien.

2.2.4 Datenflußbasierte visuelle Programmierung und ihre Nutzung in der Datenanalyse

Die *datenflußbasierte visuelle Programmierung* beruht auf dem Konzept der *Datenflußprogrammierung*, das insbesondere als Grundlage entsprechender Rechnerarchitekturen (sogenannter *Datenflußarchitekturen*) dient (für einen Überblick siehe [Sha92b]). Datenflußprogramme beschreiben die datengesteuerte Abfolge von Instruktionen in Diagramm- oder textueller Form (verschiedene Datenflußsprachen werden etwa in [HYM+92] vorgestellt).

Ein weiteres Anwendungsfeld von *Datenflußdiagrammen* bildet das Software–Engineering [PS94]. Softwareentwürfe werden hier durch Netze von Begrenzern (Systemschnittstellen), Lagern (Datenspeichern) und Prozessen als Knoten sowie Flüssen als Kanten beschrieben.

Allgemein ist ein Datenflußprogramm ein gerichteter Graph mit *Datenquellen*, *Datensenken* und beliebigen *Operatoren* (*Transformatoren*) als Knoten (auch *Module*, *Prozessoren*, *(Funktions–)Bausteine*, oder *Komponenten* genannt) und *Kanälen* als Kanten. Auf den Kanten fließen Daten, die konzeptionell über abstrakte *Token* transportiert werden, jeweils von einem *Ausgang* eines Knotens zu einem *Eingang* eines anderen. Ein- und Ausgänge werden auch als *Ports* oder *Anschlußpunkte* bezeichnet.

Abbildung 2.14 zeigt ein einfaches Beispiel anhand einer epidemiologischen Untersuchung. Aus verschiedenen Datenquellen (links) werden Fall- und Bevölkerungsdaten zu einer Studien- und einer Vergleichspopulation gelesen. Die Daten der Studienpopulation werden durch einen Restriktionsbaustein jeweils auf einen bestimmten Zeitraum eingeschränkt, zu dem anschließend durch Aggregierung ein Gesamtwert ermittelt wird; weiterhin sollen nur Krebserkrankungen interessieren. Nachfolgend werden verschiedene Maßzahlen (Raten) durch gleichnamige Operationen berechnet, die dann (rechts) in Tabelle und thematischer Karte als Datensenken visualisiert werden.

Im Gegensatz zum Steuerflußparadigma gibt es in der Datenflußprogrammierung keinen abstrakten Befehlszähler, sondern die Daten selbst sind *aktive* Einheiten, deren jeweiliges Vorliegen eine implizite Befehlsreihenfolge bestimmt, wobei auch parallele Abläufe möglich sind. Eine Operation kann jeweils ausgeführt werden (schalten), wenn alle Eingangsdaten zur Verfügung stehen. Hier unterscheidet man zwischen

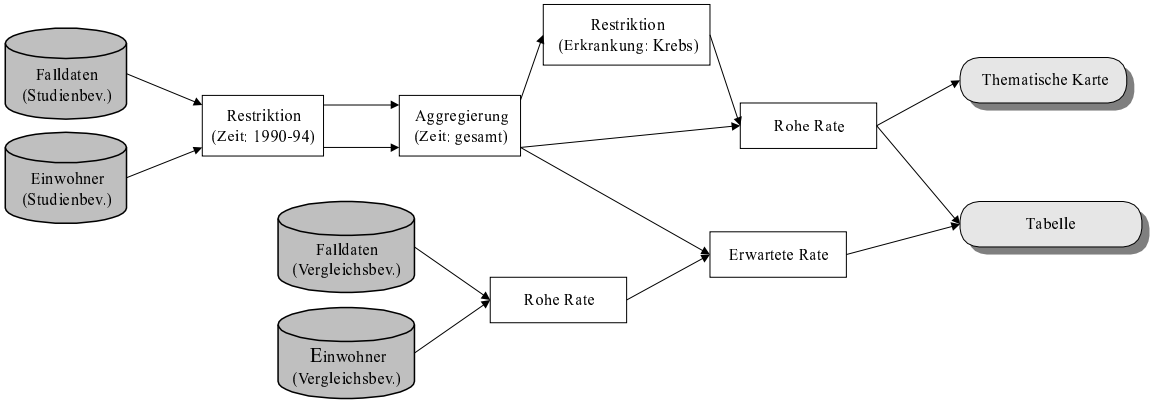

Abbildung 2.14: Ein einfaches Datenflußprogramm zur epidemiologischen Datenanalyse

- *datengetriebener (data–driven) Verarbeitung*, bei der alle in diesem Sinne möglichen Berechnungen beginnend bei den Datenquellen auch „sofort“ durchgeführt werden, und
- *anforderungsgetriebener (demand–driven) Verarbeitung*, die von zu berechnenden Datensenken ausgehend nur für diese nötige Berechnungsschritte anstößt. (In Abb. 2.14 bräuchte etwa zum Zeichnen einer thematischen Karte nur der obere Teil des Graphen durchlaufen werden, insbesondere wäre kein Zugriff auf die Daten zur Vergleichsbevölkerung nötig.)

Die Ausführung von Operationen erfolgt im Grundmodell des Datenflußparadigmas nebeneffektfrei[26], Verarbeitungsschritte haben also keine nicht–lokalen, zeitabhängigen Auswirkungen. Somit gibt es auch keine Zustandsvariablen.

Tanimoto unterscheidet unterschiedliche Ebenen der Lebendigkeit visueller Programmierumgebungen, je nachdem, ob die Ausführung eines Programms jeweils explizit angestoßen werden muß, stets nach jeder Veränderung von Programm oder Eingabeparametern bzw. -daten erfolgt oder ob (quasi in Form einer Animation) ein kontinuierlicher Datenfluß vorliegt [Tan90].

Die datenflußbasierte Programmierung ist unter den visuellen Programmierparadigmen besonders beliebt und verbreitet, weil

- Aspekte der physikalischen Welt (z. B. elektronische Schaltungen, allgemein alle natürlichen Abläufe ohne übergeordnete Kontrollinstanz) leicht und direkt abbildbar sind,
- sie aufgrund ihrer einfachen Grundkonzepte einen geringen Lernaufwand erfordert,
- die Top–Down–Zerlegung in Subprozeduren problemlos unterstützt wird,
- eine parallele Umsetzung ohne spezielle Anweisungen erfolgen kann und
- eine illustrative Visualisierung der Programmausführung zum besseren Programmverständnis und -test unmittelbar möglich ist [Sch98].

Zur effizienten Bearbeitung praktischer Problemstellungen wird das ursprüngliche Datenflußmodell in der Regel um jeweils einige der folgenden Konzepte erweitert:

- Interaktionsmöglichkeiten und Parametrisierungen von Operationen,
- Kontrollstrukturen wie *Schleifen* sowie *Verzweigungsoperatoren* (*Verteiler*, *Distributoren*, *Switches*) und *Routingoperatoren* (*Selektoren*, *Auswahloperatoren*, *Merger*), die aus mehreren Aus- bzw. Eingängen einen (oder mehrere) zur Datenweiterleitung selektieren,

[26] Dies ist etwa bei der Behandlung strukturierter Datenwerte zu beachten: Werden, um eine effiziente Verarbeitung zu gewähren, in einer Implementierung des Modells nur Verweise auf Datenobjekte zwischen den Operatoren propagiert, ist sicherzustellen, daß durch „verteiltes“ Lesen und Schreiben keine nicht–modellierten Abhängigkeiten entstehen.

- Möglichkeiten zur expliziten, datenunabhängigen Festlegung der Ausführungsreihenfolge oder
- Zustandsvariablen zur Vermeidung von Mehrfachberechnungen sowie zur Unterstützung von Schleifen.[27]

Durch die Integration dieser Konzepte büßt das Datenflußmodell jedoch wieder einen Teil seiner konzeptionellen Einfachheit bzw. der Übersichtlichkeit von Datenflußprogrammen ein, so daß stets zwischen Sprachmächtigkeit und -komplexität bzw. Handhabbarkeit abzuwägen ist [ACSW96, Hil93].

Ein umfassender Überblick über datenflußbasierte visuelle Programmiersysteme wird in [Hil93] gegeben. Auf einige Umgebungen, die speziell dem Bereich der Datenanalyse und -visualisierung entstammen, wird in Kapitel 3.4 noch näher eingegangen, wobei auch spezifische Aspekte der Umsetzung des Datenflußparadigmas betrachtet werden.

Eignung visueller Programmierung für die Datenanalyse

Die Eignung des Datenflußmodells zur Unterstützung der Datenanalyse wird in [YS91] sehr differenziert motiviert: So wie allgemein neuartige Entwicklungen und Konzepte zunächst in vagen Strukturen, in groben „Bildern“ entstehen, so hilft die Datenflußmodellierung dem Datenanalysten bei der Planung, Strukturierung und schrittweisen Durchführung gerade explorativer Analysen, deren Ablauf in der Regel von vornherein noch gar nicht ganz klar ist. In einer entsprechenden Programmierumgebung erfolgt eine direkte Umsetzung von Analysestrategien [AC96a, YL95] über einen Analyseentwurf in Analyseabläufe und deren Ausführung. In der Sprache *DFQL* [DRR+96] werden derartige Analysen als erweiterte Datenbankanfragen interpretiert — alle Schritte von Datenbankzugriff über nachfolgende analytische Berechnungen bis zur Ergebnisvisualisierung werden durch ein einheitliches Paradigma unterstützt. Hierbei kommt zweifellos die Datenflußsicht der inhärenten Natur eines (prozeduralen und interaktiv bearbeiteten und modifizierten) Analyseprozesses deutlich näher als etwa eine deklarative Anfragespezifikation. In ähnlicher Weise folgt eine derartige Analyseumgebung auch der Forderung von Huber nach einer Analysesprache, die Datenbankmanagement, statistische Analyse und Visualisierung integriert und dabei sowohl den Ablauf einer Analyse verwaltet und interaktiv zugänglich und modifizierbar macht als auch einfach erweiterbar ist [Hub94].

Weiterhin entsprechen die Anforderungen der Datenanalyse unmittelbar dem Ansatz, die Verarbeitungskapazitäten der rechten Gehirnhälfte durch visuelle, strukturierende Darstellungen stärker in die Programmierung einzubeziehen. Zum einen lassen sich Datenanalysen, wie in Abschnitt 2.1.2 dargestellt, ganz natürlich in Folgen bzw. Netze einzelner Analyseschritte und Interaktionen zerlegen [Lee94, SBM92]. Somit bildet deren Repräsentation als Operatorbausteine in Form von „Black Boxes“ ein angemessenes Abstraktionsniveau, das eine intuitive Anwendung der Programmierumgebung erlaubt. Zum anderen macht auch das Anordnen von Daten unter Identifikation der ihnen innewohnenden Strukturen den Kern explorativer Datenanalysen aus — ein Aspekt, der entsprechend durch die interaktive Manipulation von Datenobjekten und eine Strukturierung der jeweils zu ihrer Berechnung genutzten Analyseverfahren in einer visuellen Analyseumgebung zu unterstützen ist.

Mit Hilfe einer datenflußbasierten Analyseumgebung läßt sich also der Explorationsprozeß an sich verwalten. Der Nutzer erhält einen Überblick über vorgenommene Auswertungen, berechnete (Zwischen–)Ergebnisse und vor allem den genauen Weg ihrer Herleitung, ohne den sie kaum korrekt interpretierbar und nutzbar sind [Hub94]. Somit bildet die Umgebung die Grundlage für Benutzerinformation, Navigation und Manipulation von Daten und Verfahrensabläufen [AC96a]. Wie man an zahlreichen erfolgreichen wissensbasierten Systemen zur Datenanalyse sieht (vgl. [Ama97]), hilft die explizite Repräsentation des Datenanalyseprozesses — also die klare Darstellung von Beziehungen zwischen Daten, Modellen, Verfahren und Ergebnissen — beim besseren Verständnis des Analyseablaufs sowie der erhaltenen Ergebnisse. Auch können interaktiv interessierende Teilbereiche, Pfade oder Kontexte näher betrachtet oder hervorgehoben werden. In [BG89] wird weiterhin die Bedeutung umfassender Interaktionsmöglichkeiten mit dem jeweiligen Analysesystem herausgestellt. Vier

[27] Oftmals gilt für derartige Variablen das Einmalzuweisungsprinzip, d. h. ihnen wird je Ausführung eines Datenflußprogramms bzw. je Schleifendurchlauf nur einmal ein Wert zugewiesen. Somit ist quasi weiterhin Nebeneffektfreiheit gewährt.

Wege der Interaktion werden herausgearbeitet: Modifikation des Analyseprozesses an sich, Modifikation von Analyseparametern, direkte Interaktion mit Analyseergebnissen sowie direkte Betrachtung, Selektion und Manipulation der zugrundeliegenden Daten. Für die Umsetzung all dieser Aspekte bietet das Datenflußparadigma eine solide Grundlage.

Wir wollen hier noch einmal kurz die in Abschnitt 2.2.2 diskutierten Thesen zum Einsatz visueller Programmierung aufgreifen. Es wird deutlich, daß gerade die Datenanalyse ein sinnvolles Anwendungsgebiet ist:

Erleichterte Kommunikation: In der Datenanalyse werden Daten und Abläufe auf der Basis einer intuitiven gemeinsamen Metapher (des Datenanalyseprozesses) kommuniziert. Entwurfsaspekte sowie grobe, überblicksartige Strukturen, nicht Programmdetails stehen im Vordergrund.

Verständlichkeit: Die visuelle Programmierung stellt als Fortführung der Datenvisualisierung ein natürliches Kommunikationsmittel dar. (Wie in [CKLI94] herausgestellt wird, entspricht auch die Datenvisualisierung der Verwendung einer visuellen Sprache.)

Erlernbarkeit: Gerade in der *explorativen* Datenanalyse kommen der spielerische Charakter der Programmierung sowie die Möglichkeiten zur Motivationserhöhung durch direkte Interaktion voll zum Tragen. Außerdem werden erstellte Programme bei weitem nicht so komplex wie in anderen Domänen.

Internationalität: Dieser Aspekt spielt hier kaum eine Rolle.

Optimale Kognition: Im Vordergrund steht die Gewährung eines umfassenden Überblicks über ein Datenanalyseprogramm, nicht die Erinnerung von Details.

Hils nennt in [Hil93] verschiedene Charakteristika von Anwendungsdomänen, in denen speziell datenflußbasierte Programmiersysteme am erfolgreichsten sind:

- Das Anwendungsgebiet ist begrenzt, es existieren einprägsame visuelle Metaphern.
- Speziell Nicht–Programmierer oder Programmieranfänger sind angesprochen; nicht die Sprachmächtigkeit, sondern der geringe Lernaufwand steht im Vordergrund.
- Manipulation und Transformation von Daten bilden den Schwerpunkt der Anwendung.
- Systembenutzer sind mit visuellen Repräsentationen von Daten vertraut.

Auch diese Kriterien stimmen weitgehend mit Charakteristika der Datenanalyse überein. Aus der Reihe der oben aufgeführten allgemeinen Vorzüge datenflußbasierter Programmierung lassen sich außerdem die einfache Top–Down–Zerlegung von Programmen (zur Strukturierung von Analysen) und die illustrative Programmanimation (die den Analyseprozeß direkt wiedergibt) sehr gut zur Analyseunterstützung nutzen.

Eine grundlegende Motivation des Einsatzes visueller Programmierung zur Verbesserung des Verständnisses softwaretechnischer Sachverhalte lieferte Knuth bereits Anfang der 70er Jahre [Knu73]:

> *An algorithm must be seen to be believed, and the best way to learn what an algorithm is all about is to try it.*

Gerade in der Datenanalyse gilt diese Aussage wie in kaum einer anderen Domäne. Denn hier ist bereits die Entwicklung eines Algorithmus, also der Aufbau einer Analysesitzung, Teil seiner Anwendung. Durch die enge Verzahnung von Verfahrenswahl und Ergebnisberechnung bzw. -interpretation rückt auch das detaillierte Verständnis für die Struktur des jeweiligen Analyseprozesses in den Vordergrund. Die Kombination flexibler Exploration einerseits mit dem Bedarf an genauer Dokumentation des jeweiligen Analyseablaufs zur Überprüfung von Korrektheit sowie zur Reproduzierbarkeit einer Analyse andererseits macht das besondere Wesen der Datenanalyse aus [Hub86]. Da ist der Schritt zur datenflußbasierten Visualisierung und interaktiven Manipulation derselben nicht mehr weit.

2.2.5 Aktuelle Forschungsthemen

Visuelle Programmierung befreit sich nur langsam von der Beschränkung auf kleine oder prototypische (Beispiel–)Anwendungen — bei der Entwicklung *umfangreicher* und *komplexer* Softwaresysteme zur praktischen Lösung *realer* Probleme wurden visuelle Programmiersprachen bisher kaum eingesetzt. In diesem Kontext wird in [GQ94] die Übertragung des *Programming–in–the–Small* auf das *Programming–in–the–Large* unter Verwendung komplexer Softwarekomponenten als Programmbausteine diskutiert. Burnett stellt in [BBB+95] die *Skalierbarkeit* visueller Programmierung in das Zentrum zukünftiger Forschungsaufgaben und -gebiete:

- effizientere Bildschirmplatznutzung für noch bessere Interaktionsmöglichkeiten und einen noch besseren Überblick über das entwickelte Programm,
- Integration von Dokumentation, Kommentierung und weiteren sekundären Notationen (vgl. auch [Whi97, GP96]),
- geeignete prozedurale und Daten–Abstraktionen,
- genauere Spezifikation von Sprachsyntax und -semantik (unter Nutzung statischer Programmrepräsentationen)
- automatische Typprüfungen für eine einfachere und korrektere Programmierung,
- Datenbankanbindungen zur Gewährung der Datenpersistenz,
- effizientere Übersetzung, Ausführung und Ergebnisanzeige,
- Entwicklung vollständiger Entwicklungsumgebungen (für Entwurf, Test, Debugging, Versionskontrolle, Portierung etc.).

In vielen dieser Bereiche werden zudem die neuartigen Möglichkeiten, die die visuelle gegenüber der verbalen Programmierung bietet, bisher kaum ausgeschöpft. Ähnliche Forderungen stellt Myers in [Mye90]; ein besonderes Augenmerk wird hier auf die Verbesserung von Übersichtlichkeit und Strukturierung visueller Programme gelegt (worin eigentlich ein Stärke der visuellen Programmierung liegen sollte). Wie auch in [EM95] wird weiterhin der Bedarf zur Integration visueller und verbaler Programmierung aufgezeigt — eine sicherlich anspruchsvolle Zielsetzung.

Viele Forschungsarbeiten befassen sich mit der Lösung des Screen–Space–Problems. Es werden neben prozeduralen Abstraktionen Varianten des Zooming [GQ94, KMK95] oder Fisheying [CS96] sowie entsprechende Editoren vorgestellt, die die Sichtbarkeit von Programmelementen und deren Auffinden kontrollieren [SKA94]. Die visuelle Duplizierung von Programmbausteinen [KMK95] dient in ähnlicher Weise der Erhöhung der Übersichtlichkeit.

Visuelle Kontrollstrukturen sind oftmals nur schwer verständlich oder zeigen nicht alle relevanten Zusammenhänge [Han94, GP96]. Hier sind sicherlich noch umfangreiche Forschungen erforderlich; Varianten der Iteration werden etwa in [AB90, AD97] diskutiert.

Der Bedarf einer Formalisierung visueller Sprachsyntax und -semantik wird unter anderem in [Wit98, CKLI94, Mye90] motiviert. Oftmals unterstützen die jeweiligen Programmierumgebungen über syntaxgesteuerte Editoren bereits die Erzeugung korrekter Programme[28], somit ist die formale Sprachdefinition nicht von gleicher Bedeutung wie im Fall verbaler Sprachen, die in der Regel über beliebige Texteditoren nutzbar sein sollen. Sie kann dennoch sehr wertvoll zur automatischen, kontextsensitiven Unterstützung der Abstraktion und Kapselung, des Programmlayouts, der einfacheren (strukturierten) Implementierung der Programmverarbeitung sowie zur Ermöglichung der Ausführung halbfertiger Programme sein. Außerdem können so auch Umgebungen entwickelt werden, die unterschiedliche visuelle Sprachen verarbeiten können, siehe etwa [CTODL95] zur Definition visueller Compilergeneratoren. Ansätze zur formalen Sprachspezifikation bilden u. a. das Parsen mit Graphgrammatiken [RS97], die Definition formaler Semantik über räumliches Schließen [Haa95], attributierte Multimengen–Grammatiken [GR90], Relationen–Grammatiken [CGN+91] oder positionale Grammatiken [CTODL95], die Konstrukte zur Beschreibung der räumlichen Beziehungen zwischen den jeweiligen Terminalen und Nicht–Terminalen enthalten.

[28] Die wenigsten visuellen Programme werden mit „Malprogrammen“ spezifiziert.

Aus den gleichen Überlegungen heraus sind auch Typsysteme zur automatischen Kontrolle visueller Beziehungen bzw. zur Anwendbarkeitsüberprüfung von Programmkomponenten äußerst hilfreich [Bur93, WA94, Pos94]. Im System *Weaves* [GR91] werden hierbei Softwarekomponenten in einer framebasierten Notation beschrieben, so daß die automatische Suche nach jeweils geeigneten Bausteinen, transparente Typkonvertierungen und sogar die automatische Erstellung von Teilnetzen in einem datenflußbasierten System (hier „Weaving“ genannt) möglich werden.

Insgesamt muß es das Ziel visueller Programmierung sein, sich — wie etwa in der beispielorientierten Programmierung — noch stärker von der Beschränkung auf visuelle Präsentationen herkömmlicher Paradigmen zu lösen und neue Arten der Interaktion (Stichwort „Multimedia“) einzubeziehen, um das Potential visueller Darstellungen noch besser auszuschöpfen und neue Anwendungsgebiete zu erschließen [CKLI94, HI94]. Konkrete Konzepte sind jedoch noch sehr dünn gesät.

2.3 Modellierung und Verwaltung multidimensionaler Daten

Wie bereits in Kapitel 1 dargestellt, soll die in dieser Arbeit konzipierte Datenanalyseumgebung speziell die Exploration und Auswertung multidimensionaler Datensätze unterstützen, wie sie insbesondere auch in der deskriptiven Epidemiologie (vgl. Abschnitt 1.2) verarbeitet werden.

Unter *multidimensionalen Datensätzen* sind aggregierte, also jeweils über Mengen von Einzelangaben zusammengefaßte Daten aus Datenbeständen (in der Epidemiologie: Teilpopulationen), die durch verschiedene Attribute *klassifiziert*[29] sind, zu verstehen. Eine typische Sichtweise ist die eines mehrdimensionalen Feldes in Form eines „Datenwürfels“[30]: Die Dimensionen des Datenwürfels entsprechen den Kriterien der Klassifikation; in den Zellen stehen die betrachteten Maßzahlen. Ein Beispiel wären etwa die Lungenkrebsfallzahlen in Niedersachsen im Jahre 1993, klassifiziert nach den drei Dimensionen Altersgruppe, Geschlecht und Wohnort auf z. B. Gemeindeebene.

Eine derartige Konzentration der Betrachtung auf mehrdimensionale Datenfelder erscheint für die statistische Auswertung von Datensammlungen, die durch Experimente oder Erhebungen gewonnenen wurden, durchaus angemessen: Erst eine Klassifikation der Daten gemäß verschiedener Kriterien ermöglicht die Definition und Suche interessierender Strukturen und Zusammenhänge; der einzelne Beobachtungswert oder Fall ist nur von geringem Interesse. So basieren z. B. auch moderne Datenanalysesprachen wie *S–Plus* [VR94] oftmals auf einer Basisdatenstruktur „Matrix“, die aggregierte Daten beschreibt und deren Verarbeitung durch eine große Palette von Operatoren unterstützt wird. Die Behandlung von (Einzelfällen entsprechenden) Record–Strukturen spielt eine sicher nicht zu vernachlässigende, aber doch untergeordnete Rolle. [You96] nennt neben diesen beiden grundlegenden Datenstrukturen noch quadratische Matrizen als speziell zu behandelnden Datentyp. Dieser ist jedoch leicht als zweidimensionaler Datenwürfel mit gleichartigen Dimensionen zu modellieren.

Dennoch darf eine Modellierung dieses Diskursbereichs die den aggregierten Daten zugrundeliegenden Einzelfalldaten nicht völlig außer acht lassen:

- So wie die Aggregierung der Basisdaten am Beginn der Datenanalyse steht, sollte abschließend auch ihre Extraktion aus Würfelzellen möglich sein, um eine weitergehende Datensammlung oder an die multidimensionale Analyse anschließende Detailbetrachtungen vorzubereiten.

- Die Wahl der zur Zusammenfassung der Daten genutzten Aggregierungsfunktionen spielt eine bedeutende Rolle für die korrekte Interpretation und Weiterverarbeitung der Daten im Analyseprozeß. So kann die Semantik ermittelter Maßzahlwerte nur auf Basis der jeweils zugrundeliegenden Mikrodaten exakt

[29] Unter *Klassifikation* wird hier und im folgenden allgemein die — oft, aber nicht notwendigerweise disjunkte und vollständige — Zusammenfassung von Elementen einer Menge zu Teilmengen verstanden. Disjunkte und vollständige Klassifikationen werden als *Partitionierung* bezeichnet.

[30] Eigentlich handelt es sich hierbei i. a. um einen Datenquader, da die einzelnen Dimensionen in der Regel weder semantisch noch von der Kardinalität her gleich sind. Im Einklang mit dem gängigen Sprachgebrauch soll hier aber beim Begriff des Datenwürfels geblieben werden.

definiert werden (beginnend etwa bei der Betrachtung von Inzidenz- oder Mortalitätszahlen über den Unterschied zwischen durchschnittlichen Jahresbevölkerungen und Personenjahren bis hin zu komplexeren statistischen Größen, die alle letztendlich aus Ausprägungen der Attribute von Einzelfällen errechnet werden).

- Schließlich können auch im Laufe einer Analyse ermittelte Maßzahlen, Indizes, Signifikanzniveaus statistischer Tests oder Parameter komplexerer statistischer Modelle, die sich auf den gesamten jeweils untersuchten Datenbestand beziehen, sowohl als nulldimensionale Datenwürfel als auch (insbesondere in Form von Recordstrukturen) wiederum als neue Mikrodaten betrachtet werden.

Im folgenden werden nun einige Grundbegriffe und aktuelle Forschungsgebiete multidimensionaler Datenmodellierung und -analyse näher vorgestellt. Hierbei werden Datenbanksysteme zur Verwaltung multidimensionaler Datenbestände allgemein als *multidimensionale Datenbanksysteme (DBS)* bezeichnet. Analog wird auch von *multidimensionalen Datenbanken* bzw. *Datenbankmanagementsystemen (DBMS)* gesprochen.

2.3.1 Anwendungsgebiete und Charakteristika

Klassische Anwendungsgebiete multidimensionaler Datenmodellierung sind die Verarbeitung sozioökonomischer Daten sowie die Datenverwaltung für entscheidungsunterstützende betriebliche Informationsysteme. Vornehmlich auf den ersten Bereich konzentrierten sich seit Beginn der achtziger Jahre Arbeiten auf dem Gebiet *statistischer Datenbanken* und *Datenbanksysteme*. Als grundlegende Problemstellung wurde hierbei die geeignete Integration von Datenbeständen aus Experimenten und Erhebungen in einem DBMS zum Zwecke der statistischen Analyse formuliert [Sho82]. Existierende DBMS boten keine adäquate Modellierung und keine ausreichende Unterstützung des Datenzugriffs sowie der statistischen Analyse der mehrdimensionalen Datenbestände. Statistische Analysepakete andererseits wiesen große Schwächen auf dem Gebiet des Datenmanagements auf. Ziel sollte somit eine bessere Integration von Datenverwaltung und Analyse sein. Schwerpunktmäßig behandelt wurden vor allem Fragen der geeigneten konzeptionellen Datenmodellierung. Einen Überblick über diese Thematik geben [Mic91b] und [Ruf97] sowie die Tagungsbände der Konferenzreihe „*Statistical and Scientific Database Management*“.

Seit Beginn der neunziger Jahre findet die Verarbeitung multidimensionaler Daten zunehmend Beachtung im Bereich von *Decision Support Systemen* in betriebswirtschaftlichen Anwendungen [MMS96]. Mit der Integration verteilter operativer Datenbestände großer Unternehmen in sogenannten *Data Warehouses* oder (kleineren) *Data Marts* [Kim96, Inm92, CD97] ging die Forderung nach Werkzeugen zur einfachen Analyse dieser Daten einher. Insbesondere betrieblichen Entscheidungsträgern sollte eine einfache Navigation in den (hier z. B. durch die Dimensionen Produkt, Zeitraum und Verkaufsgebiet aufgespannten) mehrdimensionalen Datenräumen ermöglicht werden. Hierauf zielt das Gebiet des *On–line Analytical Processing (OLAP)* (vgl. Abschnitt 2.1.1). Ein Schwerpunkt der Forschungsarbeiten in diesem Bereich liegt — neben der Konzeption geeigneter Benutzungsschnittstellen zur Datenanalyse — auf der Entwicklung von Konzepten zur Erhöhung der Effizienz des Datenzugriffs [WB97, Wid95, Leh99].

Eigenheiten von OLAP–Systemen bzw. allgemein multidimensionaler Datenbanksysteme lassen sich in einem Vergleich mit für klassische *OLTP*–Anwendungen (*On–line Transactional Processing*) genutzten relationalen Datenbanksystemen anhand einer Reihe von Merkmalen skizzieren (vgl. auch die Kriterien in Tab. 2.2):

- Statistische Daten beziehen sich stets auf wenige Attribute von Gruppen von Objekten bzw. Ereignissen — nicht auf Einzelfälle und deren gesamte Attributmenge.

- Der einmal erfaßte Datenbestand ist — abgesehen von Fehlerkorrekturen und der Fortschreibung (Ergänzung) von Daten über die Zeit — statisch. Die Datenanalyse steht im Vordergrund, so daß die transaktionale Konsistenzerhaltung vernachlässigt werden kann.

- Operationen auf den Daten sind vor allem dynamische Zusammenfassungen und Analysen. Hierbei werden i. a. viele abgeleitete Datensätze erzeugt, die wiederum verwaltet werden müssen.

- Weil oftmals in einer Datenbank Datenbestände aus vielen Quellen aggregiert und über die Zeit fortgeschrieben werden müssen und da im Rahmen einer Datenanalyse von verschiedenen Nutzern viele neue, abgeleitete Datensätze erzeugt werden, spielt die Datenbeschreibung durch Metadaten eine besondere Rolle. Hierunter fällt insbesondere auch die Betrachtung verschiedener Arten von Nullwerten.

	OLTP	**OLAP**
Anwendungsbereiche	Administration und Kontrolle	Analyse, Decision Support
Auswertungsmethode	schlüsselbasiert, einzelsatzweise	verdichtend, verlaufsorientiert
Datenverwaltungsziele	transaktionale Konsistenzerhaltung	zeitbasierte Versionierung
Transaktionsart	kurze Schreib-/Lesetransaktionen	lange Lesetransaktionen

Tabelle 2.2: Abgrenzung von OLTP und OLAP (nach [Ruf97])

Eine differenzierte Gegenüberstellung der beiden (recht ähnlichen, jedoch lange Zeit weitgehend voneinander unbeeinflußten) Forschungsgebiete „Statistische Datenbanken“ und „Data Warehousing“ (die Einrichtung, Unterhaltung und Nutzung von Data Warehouses) bzw. „OLAP“ liefert [Sho97]. Ruf faßt die behandelten Themen unter dem Begriff der „Verwaltung und Auswertung empirisch erhobener Massendatenbestände“ zusammen [Ruf97].

Als ein weiteres Forschungsgebiet, das — aus einer ganz anderen Richtung kommend — entscheidende Beiträge zur Verarbeitung multidimensionaler Daten geliefert hat, sei hier schließlich noch die Bildverarbeitung genannt [Bau94]. Auch hier sind geeignete Operatoren zur effizienten Verwaltung von zwei-, drei- oder höher dimensionalen diskreten Datenmengen gefragt. Während Betrachtungen zu Speicherstrukturen und Zugriffsmechanismen relativ direkt auf die beiden oben genannten Anwendungsgebiete übertragbar sind, stellen sich jedoch etwas andere Anforderungen an die zu unterstützenden Operationen auf den Datenbeständen. Im Vordergrund stehen pixel-, also zellenbezogene Operationen und weniger Aggregierungen über verschiedene anwendungsspezifische Attribute. Auch wenn die Parallelen zu diesem Forschungsgebiet sicher interessant sind, trifft diese Sichtweise doch nicht die Kernproblematik dieser Arbeit und wird im folgenden somit nicht näher betrachtet.

2.3.2 Einige Grundbegriffe

Im Bereich der multidimensionalen Datenmodellierung hat sich bisher noch keine einheitliche Begriffswelt etabliert. Oftmals werden verschiedene Bezeichnungen synonym verwendet; andererseits können gleiche Begriffe auch mit unterschiedlicher Semantik auftreten. Im folgenden wird ein Überblick über Varianten von Bezeichnungen für grundlegende Sachverhalte und Strukturen gegeben. Eine eindeutige Begriffsdefinition im Kontext dieser Arbeit erfolgt dann in Kapitel 4.

Zur grundlegenden Abgrenzung multidimensionaler, aggregierter Daten von typischerweise relational repräsentierten Einzelfalldaten dienen die Bezeichnungen Mikro- und Makrodaten [Han92].

> MIKRODATEN *sind Basisdaten über Einheiten einer Gesamtpopulation, also Individuen, einzelne Objekte oder Ereignisse, aus Erhebungen oder Experimenten [Len94b].*

Während *Mikrodaten* also z. B. in Datensätzen zu einzelnen Tumorerkrankungen oder Verkaufstransaktionen bestehen, beschreiben *Makrodaten* (auch *summarische*, *Summen-* oder *aggregierte Daten* genannt) Eigenschaften von Gruppen von Mikrodaten (hier etwa Fallzahlen von Teilpopulationen oder Verkaufszahlen in Geschäftsbereichen).

> MAKRODATEN *sind aus Mikrodaten abgeleitete, gruppierte oder aggregierte Daten, die nach verschiedenen Attributen klassifiziert sind [Len94b].*

Auch wenn Makrodaten typischerweise als Datenwürfel modelliert werden, ist auch hier eine relationale Darstellung mit den klassifizierenden Attributen als Primärschlüssel der beschriebenen Eigenschaften durchaus natürlich und auch in Implementierungen des multidimensionalen Datenmodells (vgl. die OLAP–orientierten Datenmodelle am Ende von Abschnitt 2.3.3) üblich. Im Sinne dieser zu Mikrodaten ähnlichen Darstellungsweise verwendet Lehner in [Leh98] den Begriff „Mikrodaten“ allgemein für die in einer Datenbank vorliegenden Ausgangsdaten der Analyse, egal ob diese im obigen Sinne Mikro- oder Makrodaten repräsentieren.[31] Sofern der Bezug auf die genutzte Datenbasis überhaupt relevant ist, soll in dieser Arbeit statt dessen der Deutlichkeit halber von Basisdaten gesprochen werden.

> *Als* BASISDATEN *werden die in einer Datenbank materialisierten Ausgangsdaten einer Datenanalyse bezeichnet.*

Als eine „Zwischenstufe“ zwischen Mikro- und Makrodaten führt [FFH96] *Mesodaten* ein. Diese stellen quasi Makrodaten dar, die anstelle der aggregierten Maßzahlen über Eigenschaften der jeweils repräsentierten Gruppen Mengen von Identifikatoren der betroffenen Mikrodatensätze umfassen.

Die Struktur von Makrodaten

Entsprechend einer Relation im relationalen Datenmodell bildet ein *Datenwürfel* die grundlegende Datenstruktur zur Repräsentation einer Menge von Makrodaten. Synonyme Begriffe sind auch *Data Cube* sowie — insbesondere im Zusammenhang mit der Ableitung neuer Datensätze im Rahmen einer Analyse — *statistisches Objekt* und *Summary Set*.[32]

Die zur Klassifikation von Mikrodaten genutzten Daten werden *Parameterdaten*, die durch Makrodaten beschriebenen Eigenschaften *gemessene Daten* (auch *Maßzahlen*, *Maße* oder *Fakten*) genannt, wobei diese beiden Mengen von Daten nicht zwangsläufig disjunkt sein müssen. So kann etwa ein Attribut *Alter* einerseits zur Klassifikation eines Personenbestandes dienen, aber andererseits auch Gegenstand von Auswertungen sein. *Kategorielle (qualifizierende, klassifizierende) Attribute* beschreiben Parameterdaten, *summarische (quantifizierende) Attribute* die gemessenen Daten [Sho82]. In einem Datenwürfel definieren die kategoriellen Attribute somit ein multidimensionales Gitter zur Adressierung der *Zellen*, in denen die Werte der betrachteten summarischen Attribute stehen. Zellen können — dargestellt durch *neutrale Elemente* oder *Nullwerte*[33] — auch „leer“ sein, wenn zu betreffenden Kombinationen von Kategorien keine Daten vorliegen, die durch diese beschriebene Teilmenge der betrachteten Grundgesamtheit von Mikrodaten leer ist oder aber gar nicht existieren kann, also aufgrund von anwendungsbezogenen Randbedingungen immer leer ist (z. B. Frauen mit Prostatakrebs).[34] In letzterem Fall spricht man auch von *strukturellen Nullwerten*.

Oft wird von einem einzelnen summarischen Attribut pro Datenwürfel ausgegangen. Entsprechend werden Makrodaten mit mehreren summarischen Attributen zu einem Satz kategorieller Attribute mitunter auch als *komplexe Datenwürfel* bezeichnet.

Kategorielle Attribute heißen häufig auch *Dimensionen* (eines Datenwürfels). Unter einer Dimension wird meist — neben der Festlegung der Ausprägungen (oder *Kategorien*) des Attributs — auch die Spezifikation einer *Kategorien-* oder *Klassifikationshierarchie* verstanden, die Kategorien zu Gruppen bestimmter Granularität in Form von *Aggregierungsebenen* zusammenfaßt (z. B. Gemeinden zu Landkreisen oder Regierungsbezirken).

Die genannten Bezeichnungen werden sowohl im Rahmen der Beschreibung konkreter Makrodatensätze als auch zur Modellierung von Attributdomänen, die unabhängig von bestimmten Makrodatenbeständen sind, verwendet. In vielen Modellen multidimensionaler Daten wird zwischen diesen beiden Aspekten aber gar nicht

[31] Wie [Sho97] feststellt, handelt es sich hierbei im Forschungsgebiet statistischer Datenbanken oftmals um Mikrodaten, während Data Warehouses üblicherweise bereits Makrodaten materialisieren.

[32] Hier und im folgenden seien jeweils auch die englischsprachigen Bezeichnungen genannt, sofern sie auch im Deutschen üblicherweise verwendet werden oder keine angemessene Übersetzung existiert.

[33] Der Begriff des Nullwertes ist hier somit etwas weiter gefaßt als etwa in relationalen Datenbanksystemen üblich.

[34] Die Unterscheidung zwischen Fällen, in denen keine Daten vorliegen, und solchen, in denen keine Angaben vorliegen *können*, ist jedoch oftmals fließend, da entsprechende anwendungsbezogene Randbedingungen nicht immer klar zu definieren oder auch zeitlich veränderlich sind.

unterschieden. Abbildung 2.15 faßt die Modellierung von Makrodaten und deren Ableitung aus Mikrodaten noch einmal anschaulich zusammen.

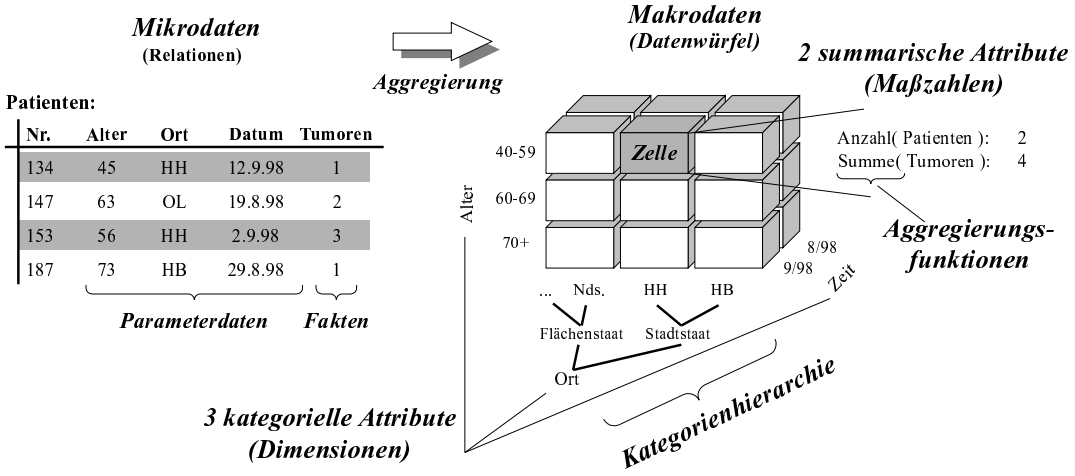

Abbildung 2.15: Mikro- und Makrodaten

Operationen auf aggregierten Daten

Die Basisoperation zur Verarbeitung multidimensionaler Daten ist die *Aggregierung*. Bei der Berechnung von Makro- aus Mikrodaten werden mittels einer vom jeweils betrachteten summarischen Attribut abhängenden *Aggregierungs- (Summierungs-)funktion* die Faktendaten der einzelnen Mikrodaten für jede der durch die kategoriellen Attribute definierten Gruppen (entsprechend den Zellen des Datenwürfels) zu einer Maßzahl zusammengefaßt (vgl. Abb. 2.15). Oftmals handelt es sich hierbei um recht einfache Funktionen, wie Anzahl (z. B. von Erkrankungsfällen), Summe (in Abb. 2.15 etwa von Tumoren betrachteter Personen), Minimum, Maximum, Durchschnitt etc. Werden bereits bestehende Makrodaten gemäß einer Zusammenfassung von Kategorien einer oder mehrerer Dimensionen vergröbert (z. B. Fallzahlen nach Monat und Gemeinde zu Fallzahlen nach Jahr und Landkreis), spricht man ebenfalls von Aggregierung. Je nach Typ des summarischen Attributs muß hierzu auf die zugrundeliegenden Mikrodaten zurückgegriffen werden, oder die Berechnung kann direkt durch Anwendung einer spezifischen Aggregierungsfunktion auf die Makrodaten erfolgen.[35] Diese Aggregierungsfunktion kann — muß aber nicht — die gleiche sein wie die, die bei der Verarbeitung der Mikrodaten genutzt wurde (im Beispiel Fallzählung etwa die Summe anstelle der Anzahl). Bisweilen wird der Begriff der Aggregierungsfunktion auch synonym zur Maßzahl oder zum summarischen Attribut verwendet.

Auch die Ableitung von Makrodaten aus anderen Makrodaten unter Anwendung beliebiger zusammenfassender, „vergröbernder“ Funktionen (also unabhängig vom in den Ausgangsdaten vorliegenden summarischen Attribut bzw. dessen Berechnung aus Mikrodaten) sowie die „Eliminierung“ von Dimensionen bei der Berechnung komplexer Maßzahlen fallen unter „Aggregierung“. Abbildung 2.16 gibt einige Beispiele zur Aggregierung; die Standardisierung bildet hierbei eine mit den Teilpopulationsgrößen einer Standardpopulation gewichtete Summe der alters- und geschlechtsspezifischen Erkrankungsraten (also der Fallzahlen bezogen auf die Größe der jeweiligen Teilpopulation).

Synonyme zum Begriff der Aggregierung sind *Verdichtung*, *Konsolidierung*, *Roll–up* oder *Summierung*. Wird eine Dimension eines Datenwürfels durch vollständige Aggregierung (Bildung einer „Randsumme“) über

[35] Im üblichen Sprachgebrauch ist der Begriff der Aggregierung — wie auch die Bezeichnung vieler anderer hier vorgestellter Operationen — nicht auf das tatsächliche Durchführen von Berechnungen beschränkt, sondern umschreibt auch das Navigieren in Datenbeständen auf verschiedenen Aggregierungsebenen über eine entsprechende Benutzungsoberfläche.

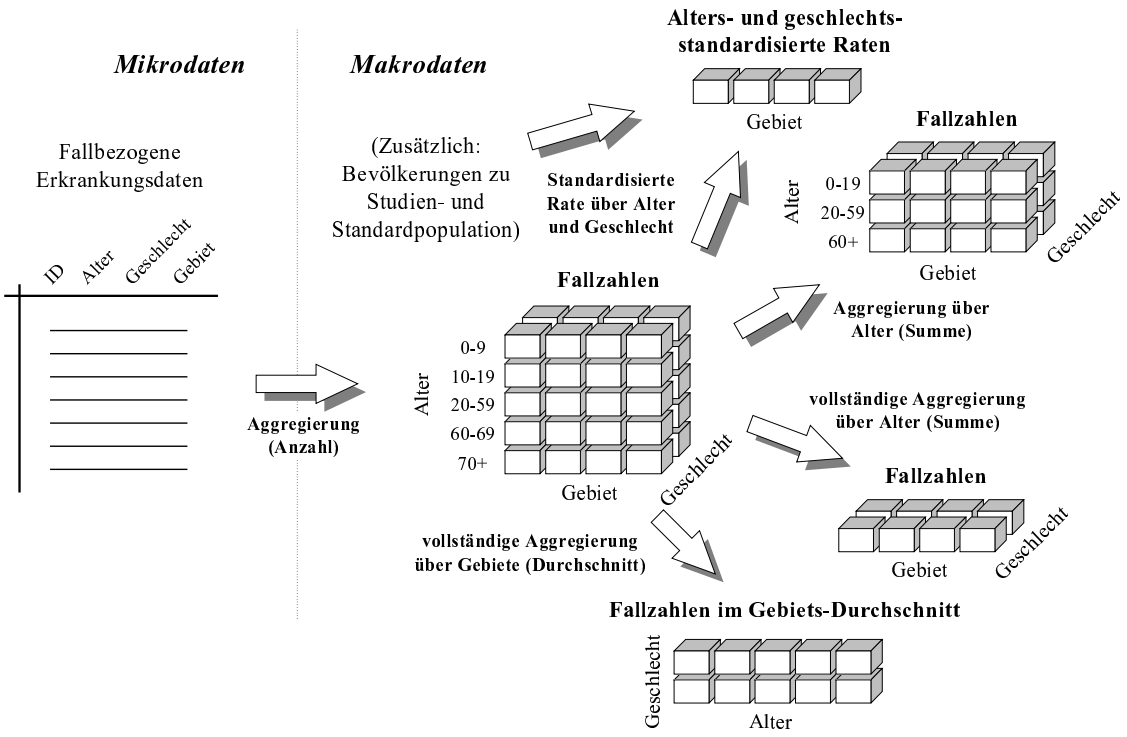

Abbildung 2.16: Varianten der Aggregierung

alle seine Kategorien quasi eliminiert, spricht man auch von *Projektion*.[36] Die Begriffe *Gruppierung* und *Klassifikation* stehen gleichermaßen auch für die (datenunabhängige) Zusammenfassung von Kategorien im Rahmen einer Kategorienhierarchie. Vor allem in Tabellenkalkulationsprogrammen gebräuchlich ist der Begriff des *Pivotisieren* als das Auswählen darzustellender Dimensionen durch vollständige Aggregierung über alle anderen (quasi ein Drehen des Datenwürfels und Betrachten von Randsummen) [CD97].

Die inverse Operation zur Makrodaten–Aggregierung ist die *Disaggregierung* (auch *Drill–down*). Zu ihrer *exakten* Durchführung müssen natürlich die betrachteten Mikrodaten oder feinere Makrodaten vorliegen. Andernfalls können auch Schätzverfahren zum Einsatz kommen. Wird bei der Disaggregierung ein neues kategorielles Attribut hinzugefügt, spricht man von *Erweiterung* des Datenwürfels.

Wie im relationalen Datenmodell ist natürlich auch eine *Restriktion (Selektion, Einschränkung, Slicing)* auf den Dimensionen von Datenwürfeln möglich. Als *Dicing* wird üblicherweise das gleichzeitige Einschränken über mehrere Dimensionen bezeichnet. Chaudhuri versteht unter „Slicing and Dicing“ speziell die Verringerung der Dimensionalität eines Datenwürfels durch Einschränkung kategorieller Attribute auf jeweils einen einzelnen Wert [CD97]. Shoshani bemerkt in [Sho97], daß mitunter auch „Slicing“ synonym zur Aggregierung verwendet wird.

Neben obigen Basiselementen runden noch eine Reihe weiterer Operationen, die teilweise aus dem relationalen Datenmodell übernommen werden können, teilweise auch anwendungsspezifisch realisiert werden müssen, ein multidimensionales Datenmodell ab:

- Das *Drill–through* steht für die Abfrage einem Datenwürfel zugrundeliegender Basisdaten.
- Unter *Drill–across* oder *Join* wird allgemein das Verknüpfen verschiedener Datenwürfel bzw. Maßzahlen über gemeinsame Dimensionen verstanden.

[36] Hier wie auch bei vielen anderen Operationen auf multidimensionalen Daten findet man deutliche Parallelen zum relationalen Datenmodell.

- Operationen auf den Domänen der summarischen Attribute induzieren in natürlicher Weise über eine zellenweise Bearbeitung *zellenbezogene Operationen* auf Datenwürfeln (z. B. die Invertierung eines oder die Addition zweier Datenwürfel).
- Auch auf Makrodaten können (weitgehend analog zum relationalen Modell) die gebräuchlichen *Mengenoperationen*, wie *Vereinigung*, *Durchschnitt* oder *Differenz*, definiert werden. Da diese Operationen auf einer Betrachtung von Datenwürfeln als *Mengen* von Zellen basieren, ist die Ergebnis*menge* ggf. durch Nullwerte zu einem vollständigen kartesischen Produkt über Kategorien der jeweiligen Dimensionen aufzufüllen.

Die Bereitstellung bzw. Art der Implementierung dieser Operationen ist sicherlich von DBMS zu DBMS sehr uneinheitlich und berührt in Teilen auch bereits auf diesen aufsetzende Anwendungen.

2.3.3 Varianten der multidimensionalen Datenmodellierung

Als Basis für Entwurfsentscheidungen bei der Gestaltung des Datenmodells *MADEIRA* in Kapitel 4, das durch die in dieser Arbeit entwickelte Analyseumgebung zu unterstützen ist, werden im folgenden grundlegende Modellierungsansätze sowie Spezialitäten einiger bestehender multidimensionaler Datenmodelle vorgestellt. Die Gliederung dieses Abschnitts orientiert sich vor allem an der veränderten Schwerpunktsetzung multidimensionaler Datenmodellierung im Laufe der achtziger und neunziger Jahre. Überblicksdarstellungen, insbesondere zu den älteren Modellen, geben etwa [Ruf97, Mic91a]. In Kapitel 4 werden dann einige Aspekte dieser Modelle zur Definition von Anforderungen an *MADEIRA* sowie zur Abgrenzung des in dieser Arbeit verfolgten Ansatzes herangezogen.

Graphbasierte Datenmodelle

Die ersten Ansätze zur Modellierung multidimensionaler Daten konzentrierten sich auf die Beschreibung der Struktur *statistischer Tabellen* bzw. entsprechender Datenräume in Form azyklischer gerichteter Graphen (siehe insbesondere [Raf91]), die zugleich auch die Grundlage für eine Benutzungsschnittstelle zur visuellen Datendefinition und Anfrage liefern (vgl. z. B. [RF92]). Summarische und kategorielle Attribute sowie die auf diesen definierten Kategorienhierarchien sind in derartigen Tabellen typischerweise auf Spalten- und Zeilenköpfe verteilt (siehe Tab. 2.3).

Lungenkrebs-Fallzahlen und -Raten 1990				**Niedersachsen**				**Hessen** ...
				Reg.–Bez. Weser–Ems	**Reg.–Bez. Hannover**	...	**Gesamt**	...
Geschlecht	**w**	**Alter**	**0 – 19**	3 / 7.4	2 / 5.9		11 / 6.1	
			20 – 59	67 / 89.4	83 / 98.2	...	271 / 99.1	...
			60+	...	...		...	
	m	**Alter**	**0 – 19**				...	
			20 – 59	...	...	...	...	
			60+					

Tabelle 2.3: Beispiel einer statistischen Tabelle

Die im 1981 vorgestellten *SUBJECT*–Modell [CS81] verwendeten Graphen repräsentieren Zusammenfassungen orthogonaler Klassifikationskriterien durch Kreuzproduktknoten (X) sowie Gruppierungen von Attributen oder deren Unterteilungen bzw. Ausprägungen durch Clusterknoten (C) (siehe Abb. 2.17 — der linke Teil des Baumes stellt die Zeilenköpfe, der rechte die Spaltenköpfe dar). Somit werden nur streng hierarchische Kategorienhierarchien unterstützt; zusammengesetzte summarische Attribute können über X–Knoten modelliert

werden. Auch Gruppen von Datenwürfeln sind mittels eines Netzes von C–Knoten darstellbar, wobei durch Ausnutzung gemeinsamer Teilgraphen eine kompaktere Notation möglich ist.

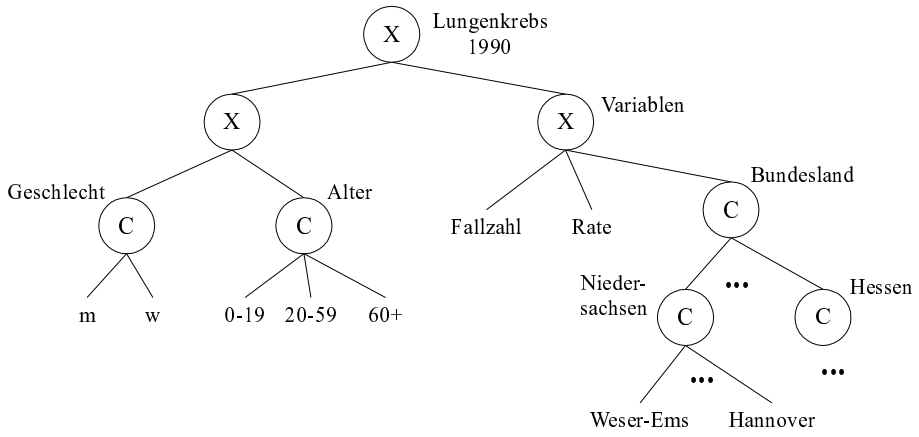

Abbildung 2.17: Modellierung des Beispiels aus Tab. 2.3 im *SUBJECT*–Modell

Während in *SUBJECT* summarische und kategorielle Attribute im wesentlichen gleich behandelt werden (in Abb. 2.17 sieht man, daß summarische Attribute unter einem „Variablen"–Knoten als Teil der Spaltenköpfe modelliert werden), führt das auf diesem Ansatz aufbauende *GRASS*–Modell [RR83] unterschiedliche Knotentypen für die beiden Attributarten ein und ermöglicht so eine klarere Modellierung. Pro statistischer Tabelle ist nur noch genau ein summarisches Attribut darstellbar.

Das *STORM*–Modell [RS90] (ein Nachfolger von *GRASS*) definiert neben der graphischen Repräsentation von Makrodaten den Begriff des *statistischen Objekts* als Bezeichnung für einen multidimensionalen Datensatz. Ein statistisches Objekt ist ein Quadrupel aus einem Namen, *einem* summarischen Attribut, einer Menge kategorieller Attribute sowie einer Abbildung vom Kreuzprodukt der Domänen der kategoriellen Attribute in die Domäne des summarischen Attributs. Die Attribute beschreiben neben den Datenstrukturen auch eine Reihe von Metadaten wie Datentypen, Maßeinheiten oder Wertebereiche. Unterschiedliche Aggregierungsebenen müssen als verschiedene Attribute modelliert werden; die Beziehung zwischen ihnen ist nur der graphischen Darstellung zu entnehmen.

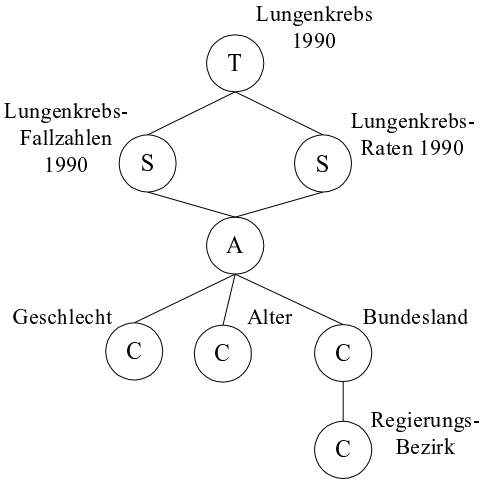

Abbildung 2.18: Modellierung im *STORM*–Modell

STORM greift einige Kritikpunkte an seinen Vorgängern für Erweiterungen auf. So wird zwischen der Beschreibung auf intensionaler und extensionaler Ebene unterschieden — Attributausprägungen werden getrennt von der Definition der kategoriellen Attribute und zwischen ihnen bestehender hierarchischer Beziehungen repräsentiert (siehe die raumbezogenen Attribute im Beispiel in Abb. 2.18 — der A–Knoten entspricht hier dem X–Knoten in *SUBJECT*, S–Knoten stehen für statistische Objekte, und T–Knoten gruppieren zusammengehörige Datenräume). Außerdem werden Kategorienhierarchien (gewissermaßen als „Domänen") zusätzlich auch unabhängig von konkreten statistischen Objekten modelliert.

Semantisch reichere Modellierung statistischer Datenbanken

Neben obigen Ansätzen, die sich vor allem auf die graphbasierte Modellierung konzentrieren, sollen hier auch noch kurz einige weitere frühe Arbeiten aus den 80er Jahren angeführt werden, die sich aus der Anwendung mächtiger semantischer Modelle auf das spezielle Gebiet statistischer Datenbanken herleiten. Teilweise werden hierbei Mikro- und Makrodaten gemeinsam modelliert.

*SAM** [SNB83] beschreibt statistische Daten auf der Basis von sogenannten „Konzepten“ sowie Beziehungen und Constraints zwischen diesen. *SDM4S* [SNFH86] legt in einer framebasierten Notation den Schwerpunkt auf die Trennung von konzeptioneller Ebene, Datenbankschema–Ebene und Instanzenebene, deren Zusammenhänge über Klassenhierarchien und Teile–Ganzes–Beziehungen modelliert werden. *CSM* [DBB88] und *STORM+* [BMR94] stützen sich in ähnlicher Weise auf objektorientierte Konzepte ab.

Speziell auf die Verwaltung verteilter, heterogener statistischer Datenbanksysteme konzentrieren sich die Arbeiten von Fröschl [Fro96, Fro97] zum System METASTASYS. Hier wird das sehr komplexe Datenmodell METAMOD entwickelt, das eine Integration (unter anderem bzgl. der jeweils unterstützten Kategorienhierarchien) heterogener multidimensionaler Datenbestände in eine einheitliche Sicht ermöglicht. Deklarativ gestellte Anfragen werden auf der Basis der jeweiligen Schemabeschreibungen, die vielfältige Metadaten vorsehen, möglichst effizient durch Ableitung von Teildatenräumen aus geeigneten Datenquellen und deren anschließende Zusammenführung beantwortet. Vor allem werden Anzahlen und Summen als Basisgrößen verarbeitet, weitere Datentypen und statistische Operationen zu deren Herleitung sind jedoch auch vorgesehen bzw. können prinzipiell in den gegebenen Rahmen integriert werden.

Modellierung dynamischer Aspekte

Während in den bisher vorstellten Modellen gängige Operationen auf aggregierten Daten (wie Aggregierung und Restriktion) nicht oder allenfalls — in den semantischen Modellierungsansätzen — als spezielle Methoden explizit modelliert wurden, konzentrieren sich einige weitere Modelle besonders auf deren adäquate Behandlung.

Mefisto [FMENR89, RR93] basiert auf einer ähnlichen Datenstruktur wie *STORM*, berücksichtigt jedoch im Modell keine Kategorienhierarchien. Auf dieser Struktur werden die schon oben vorgestellten typischen Operatoren Summierung (Eliminierung einer Dimension), Klassifikation (über jeweils spezielle Aggregationstabellen, die die Zusammenfassung von Kategorien spezifizieren), Einschränkung und Vereinigung definiert. Außerdem sind die Erweiterung um ein neues kategorielles Attribut sowie die Umbenennung von Attributen vorgesehen. In [Raf91] wird zusätzlich die Disaggregierung unter Verwendung von Schätzverfahren genannt. Speziell wird in *Mefisto* auch die automatische Wahl der korrekten Umsetzung der genannten Operatoren, insbesondere der Aggregierung, anhand des Typs der verarbeiteten Daten betrachtet.

[MERS92] motiviert die semantische Vollständigkeit einer ähnlichen Algebra, bestehend aus Projektion, Aggregierung, Vereinigung und Einschränkung, durch Definition eines Homomorphismus zwischen dieser und den Operatoren der relationalen Algebra. Hierbei wird jeweils auf eine relationale Repräsentation der kategoriellen Attribute eines Datenwürfels Bezug genommen.

In [BRT96] wird, aufbauend auf Konzepten aus dem *STORM*– und dem *Mefisto*–Modell, eine *Aggregate Data Structure (ADaS)* zur Modellierung multidimensionaler Daten vorgeschlagen. Sie erweitert obige Modelle vor allem in folgenden Punkten:

- Zusätzlich zu (expliziten) klassifizierenden kategoriellen Attributen wird ein *ADaS* auch durch *implizite Attribute* beschrieben, die lediglich eine *Einschränkung* der Daten gegenüber einem zugrundeliegenden „Universum“ beschreiben.

- Neben einfachen *ADaS* werden auch *komplexe* und *zusammengesetzte* Datenobjekte unterschieden, die mehrere einfache Objekte mit gleichem summarischen(!) Attribut über demselben Universum, aber unterschiedlichen kategoriellen Attributen bzw. mehrere beliebige *ADaS* zu einem Objekt zusammenfassen.

- Entsprechend werden neben den bekannten Operationen Summierung, Einschränkung und Vereinigung auch *Komposition* und *Dekomposition* von *ADaS* betrachtet. Kategorienhierarchien werden nicht explizit modelliert.

Das *Statistical Relational Model (SRM)* [Gho86, Gho91] wendet sich stärker der statistischen Analyse der betrachteten Fakten zu. Es lehnt sich eng an eine relationale Repräsentation multidimensionaler Daten bzw. das relationale Datenmodell an und erweitert dieses, insbesondere den Group–by–Operator, um eine Algebra auf den summarischen Attributen (u. a. Vektorproduktbildung, Berechnung statistischer Momente, Aggregations- und Ordnungsoperatoren).

Auch [ÖÖM87] definiert eine Erweiterung der relationalen Algebra, konzentriert sich jedoch auf die Aggregierung. Zum einen werden zur flexiblen Gruppierung von Kategorien mengenwertige Attribute sowie — ähnlich dem *NF2–* (*Non First Normal Form*) Modell [JS82] — Operatoren *Pack* und *Unpack* zur Zusammenfassung und Aufteilung von Ausprägungen eingeführt. Zum anderen werden die relationalen Operatoren um zwei Aggregationsoperatoren ergänzt, die ähnlich dem Group–by unter Verwendung beliebiger Aggregierungsfunktionen Tupel (disjunkt) zusammenfassen bzw. aufgrund einer separaten Relation, die über mengenwertige Attribute Gruppen von Kategorien definiert, (nicht notwendigerweise disjunkt) gruppieren können.

In diesem Kontext sei auch Gray [GBLP96] erwähnt, der die Erweiterung von SQL durch einen *CUBE–Operator* zur gleichzeitigen Durchführung einer Group–by–Operation über alle Kombinationen (einer Teilmenge) von Attributen einer Relation mit Makrodaten vorschlägt. Hierdurch können quasi synchron Randsummen eines Datenwürfels ermittelt werden. Im Ergebnis werden Attribute, über die vollständig aggregiert wurde, durch eine Konstante *ALL* repräsentiert.

Durch OLAP inspirierte Datenmodellierung

In den letzten Jahren erschienen — getrieben durch die zunehmende Verbreitung von OLAP–Systemen — wieder vermehrt neue Ansätze zur Modellierung multidimensionaler Daten (vgl. [SBHD98, BSHD98]). Auf oftmals recht formaler Ebene werden die durch das jeweilige Anwendungsgebiet geforderten Strukturen und Operationen exakt modelliert sowie darauf aufbauend flexible Anfragesprachen definiert. Letztere sollen insbesondere (im Gegensatz zu älteren Datenmodellen) die Zusammenfassung von Mengen bzw. Sequenzen von Operationen in einer deklarativen Anfragespezifikation ermöglichen. Im Vordergrund steht außerdem oft die Möglichkeit der Definition von *Ad–hoc–Aggregierungen* über benutzerdefinierte Gruppierungen der betrachteten Kategorien und beliebige Aggregierungsfunktionen. Hierbei zeichnet sich ab, daß unterschiedliche Anwendungsgebiete durchaus unterschiedliche Modellvarianten erfordern — es stellt sich die Frage, ob hier eine Vereinheitlichung der Betrachtungsweisen möglich bzw. überhaupt erstrebenswert ist. Diejenigen Modelle, die das Ziel einer allgemeinen Einsetzbarkeit anstreben, wirken oft sehr komplex und benutzungsunfreundlich. Obwohl sehr mächtig, verlieren sie zumeist die konzeptionelle Einfachheit der in Abschnitt 2.3.2 beschriebenen Struktur von Makrodaten.

Während in den Datenmodellen des vorigen Abschnitts Aspekte konzeptioneller und logischer Datenmodellierung oftmals miteinander vermischt werden (etwa im *STORM*–Modell), sind diese Aspekte im OLAP–Kontext meist klarer getrennt. Zur *konzeptionellen* (oder auch *semantischen*) Modellierung aggregierter Daten werden zumeist graphische, oft auf dem *Entity Relationship* (ER–) Modell [Che76] oder *UML* [FS98] aufbauende Notationen für Fakten, zugeordnete Dimensionen, darauf definierte Hierarchien und weitere Dimensionsattribute eingeführt [GMR98, JT98, SBHD99, HH99]. In [GG98] wird ein Überblick über einige dieser Ansätze gegeben. Zur Unterstützung des grundlegenden Entwurfs von Data Warehouses, insbesondere auch der Integration verteilter Datenbestände, steht diese konzeptionelle Sichtweise in der Regel im Vordergrund [MSR99, CDGL$^+$99, JJQV99]. Häufig wird dann auch hieraus das logische Design konzeptioneller, externer und interner Datenbankschemata abgeleitet.

In dieser Arbeit interessiert demgegenüber vor allem die Umsetzung von Datenmanagement und -analyse auf konkreten multidimensionalen Datenräumen, so daß wir uns im folgenden Arbeiten zu deren *logischer* Modellierung näher zuwenden wollen. Diese definieren in der Regel jeweils auch eine Algebra auf Makrodaten.

Einige der im weiteren Verlauf angeführten Datenmodelle basieren nicht allein auf einer logisch multidimensionalen Sicht auf Makrodaten, sondern orientieren sich an deren relationaler Repräsentation[37], die in vielen bestehenden OLAP–Systemen (vgl. Abschnitt 2.3.4) üblich ist. Makrodaten werden hierbei in einem sogenannten *Sternschema* [MMS96, Kim96] gespeichert, das pro Datenwürfel aus einer Faktentabelle und mehreren Dimensionstabellen besteht.

Die *Dimensionstabellen* enthalten die Ausprägungen der kategoriellen Attribute; neben einem eindeutigen Primärschlüssel sowie einem Namen der jeweiligen Kategorie enthalten sie auch weitere sogenannte *Dimensionsattribute*, die die Kategorie durch zugeordnete Charakteristika und Einstufungen in höhere Aggregierungsebenen näher beschreiben. Eine Dimensionstabelle zur Zeitdimension könnte so z. B. die Attribute *Zeit_ID*, *Tag*, *Monat*, *Jahr*, *Kalenderwoche*, *Wochentag*, *ist_Feiertag* etc. enthalten. Auch die Zusammenfassung mehrerer (weitgehend) unabhängiger kategorieller Attribute in einer Dimensionstabelle ist durchaus nicht unüblich (siehe z. B. [Kim96, GL97]. Die *Faktentabelle* definiert zu einer Kombination von Fremdschlüsseln aus den Dimensionstabellen die jeweiligen Werte der summarischen Attribute. Gegenüber einer vollständigen Speicherung eines Datenwürfels in Form eines mehrdimensionalen Feldes sind hier also nur die nicht–leeren Zellen (zu nicht–leeren Teilpopulationen bzw. ohne Nullwerte) gespeichert.

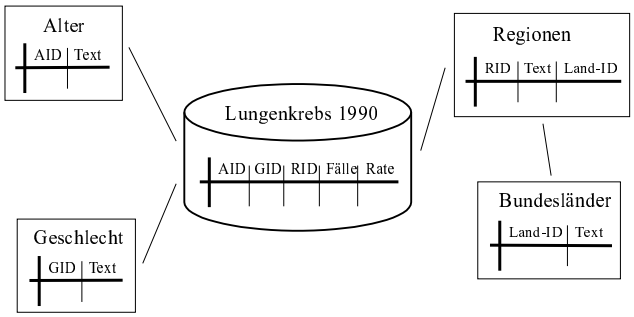

Abbildung 2.19: Schneeflockenschema zu Tab. 2.3

In einigen Implementierungen dieses Ansatzes ist nur die Modellierung *eines* summarischen Attributs pro Datenwürfel möglich. Falls eine derartige Vereinfachung zu restriktiv ist, können oftmals Gruppen von Maßzahlen als eine zusätzliche *Kennzahldimension* bzw. ein *Kennzahlattribut* modelliert werden, das als „Kategorien“ dann die verschiedenen Maße repräsentiert (vgl. etwa [LAW98]).

Wird die Restriktion auf genau eine Relation zur Repräsentation einer Dimension eines Datenwürfels im Sternschema aufgehoben, spricht man auch von einem *Schneeflockenschema*. Dieses bietet die Möglichkeit, redundante Informationen zu höheren Aggregierungsebenen im Rahmen einer Schema–Normalisierung in separaten Relationen zu speichern. In Abb. 2.19 ist ein Beispiel hierfür dargestellt: Gebiete werden durch Regionen (Regierungsbezirke in Tab. 2.3) und jeweils zugeordnete Länder unterteilt. Wären die Ländernamen nicht separat, sondern redundant im Rahmen der Regionentabelle abgelegt, erhielte man wieder ein einfaches Sternschema.

Eine *Grouping Algebra* nach Li und Wang (1996) Einen Vorschlag eines vollständigen multidimensionalen Datenmodells, das sich stark an der relationalen Datenspeicherung sowie der relationalen Algebra orientiert, machen Li und Wang in [LW96]. Eine multidimensionale Datenbank besteht hiernach aus einer Menge von Datenwürfeln in Gestalt eines Sternschemas sowie einer Menge von *Gruppierungsrelationen*, die ähnlich wie die Dimensionstabellen zusätzliche Gruppierungen auf bestehenden Dimensionsattributen definieren. Zur Ableitung neuer Gruppierungsrelationen erweitern die Autoren die relationale Algebra insbesondere um mächtige Operatoren, die auf der Grundlage einer Ordnung auf den Kategorien durch Intervallbildung neue Aggregierungsebenen (Dimensionsattribute) erzeugen, sowie solche, die durch Anwendung von Aggregierungsfunktionen auf Gruppen von Kategorien eines Datenwürfels Maßzahlen als neue Dimensionsattribute berechnen. Sie definieren weiterhin eine *Grouping Algebra* auf Datenwürfeln, die sechs Operatoren umfaßt: Hinzufügen einer neuen Dimension, Verschieben eines Dimensionsattributs zu einer anderen Dimension, Vereinigung zweier Datenwürfel (ggf. unter Verwendung von Nullwerten), Aggregierung durch Eliminieren von Dimensionen oder Dimensionsattributen, Join von Dimensionstabellen und Gruppierungsrelationen sowie die Umwandlung von (Gruppierungs–)Relationen in eindimensionale Datenwürfel mit einem ausgewählten summarischen Attribut.

[37] Diese Beeinflussung durch die physische Realisierung ist sicher auch als ein Hauptkritikpunkt zu sehen.

Datenmodell und Kalkül von Gyssens und Lakshmanan (1997) Auch das Datenmodell sowie ein hieraus abgeleitetes, äquivalentes Anfragekalkül aus [GL97] suchen die Nähe zum relationalen Modell. Die angestellten Überlegungen basieren auf einer Repräsentierbarkeit der Grunddatenstruktur des vorgeschlagenen Modells, einer *statistischen Tabelle* durch eine „äquivalente“ Relation und (ggf. unter Rückgriff auf Nullwerte) umgekehrt. Die Definition des Schemas einer statistischen Tabelle lehnt sich wiederum an Sternschemata an, in denen Kategorienhierarchien durch Dimensionsattribute repräsentiert sind. Jedoch wird die konzeptionelle Unterscheidung von summarischen und kategoriellen Attributen weitgehend aufgegeben: Die Zellinhalte der Tabelle sind Records (entsprechend komplexen Datenwürfeln), in die mittels eines *Fold*–Operators Attribute der Dimensionstabellen (Spalten- und Zeilenköpfe der statistischen Tabelle) integriert und aus denen mittels *Unfold* Attribute wiederum zu Dimensionsattributen gemacht werden können. In der relationalen Sichtweise entsprechen diese Operationen gerade einem Wechsel der Zuordnung von Relationsattributen zwischen der Menge der summarischen und der Menge der kategoriellen Attribute. Somit bilden die Dimensionsinformationen auch keinen Primärschlüssel mehr zu den Zellen der Tabelle. Dies ermöglicht die Definition sehr mächtiger Operatoren, entfernt sich jedoch stark von der Grundidee aggregierter und klassifizierter Makrodaten. Mit Hilfe der relationalen Repräsentation statistischer Tabellen werden die Operatoren des relationalen Datenmodells auf das vorgeschlagene Modell übertragen. Schließlich vervollständigen noch ein Klassifikationsoperator zur Definition von Dimensionsattributen, die Kategorien zusammenfassen, sowie ein Summierungsoperator zum Hinzufügen aggregierender Maßzahlen in die Tabellenzellen das Modell. Beide Operatoren können auf beliebige Klassifizierungs- bzw. Aggregierungsfunktionen zurückgreifen.

Agrawal, Gupta und Sarawagi (1997) In ähnlicher Weise behandeln auch Agrawal, Gupta und Sarawagi in einem in [AGS97] vorgeschlagenen Datenmodell summarische und kategorielle Attribute gleich, wobei jedoch im Gegensatz zu [GL97] die funktionale Abhängigkeit zwischen den jeweiligen Dimensionselementen und den Zelleinträgen eines Datenwürfels gewahrt bleibt. Ein Datenwürfel wird definiert durch n „flache“ Dimensionen mit jeweils atomaren Ausprägungen, eine (evtl. auch leere) Menge summarischer Attribute sowie eine Abbildung vom Kreuzprodukt der Dimensionen in das Kreuzprodukt der summarischen Attribute. Zusätzlich repräsentiert eine „0“ als Funktionswert einen Nullwert und eine „1“ die Existenz einer Kombination von Kategorien in der Datenbasis bei einer leeren Menge summarischer Attribute. Als neuartige Operatoren auf dieser Struktur werden *Push* und *Pull* eingeführt, die die Rolle eines Attributs verändern: Kategorielle Attribute werden als Zellkomponenten kopiert bzw. summarische Attribute in Dimensionen umgewandelt. Weiterhin gibt es die Restriktion, ein *Merge* als Form der Aggregierung zu höheren Aggregierungsebenen, das Verknüpfen von Datenwürfeln (über gemeinsame Dimensionen und verschiedene Hilfsfunktionen) mittels eines Joins als sehr mächtiger Operator sowie die Elimination einer Dimension mit nur einer Ausprägung. Auch in diesem Ansatz wird eine Analogie zur relationalen Algebra dargestellt, es werden jedoch keine relationalen Datenstrukturen in das Modell einbezogen. Eine klare Trennung von Schema und Extension wird nicht vorgenommen. Weiterhin werden Kategorienhierarchien nicht explizit modelliert, sondern jeweils über Gruppierungsfunktionen (entsprechend den Relationen bei [GL97]) definiert, die bei der Anwendung obiger Operatoren genutzt werden.

Das $\mathcal{MD}$–Modell nach Cabibbo und Torlone (1998) In ihrem $\mathcal{MD}$–Modell legen Cabibbo und Torlone [CT98] im Gegensatz zu den vorher genannten Ansätzen einen Schwerpunkt auf die Modellierung der hierarchischen Dimensionsstrukturen. Ein multidimensionales Datenschema besteht aus einer Menge von Dimensionen, die sich ihrerseits jeweils aus einer Menge von Aggregierungsebenen und einer Ordnung auf diesen bzw. den enthaltenen Kategorien zusammensetzen, sowie einer Menge von Datenwürfelschemata. Eine Aggregierungsebene kann mehreren Dimensionen zugeordnet sein. Datenwürfelschemata werden beschrieben durch eine Reihe kategorieller Attribute, die jeweils eine Ebene einer Dimension spezifizieren, sowie ein summarisches Attribut, dessen Wertebereich ebenfalls als Ebene einer Dimension definiert ist. Insbesondere hierzu bilden auch Zeichenfolgen und Zahlen Dimensionen mit jeweils nur einer Ebene. Weiterhin können einer Aggregierungsebene zur Beschreibung ihrer Kategorien in Form von weiteren Dimensionsattributen an-

dere Ebenen zugeordnet sein. Auf diesen Strukturen aufbauend werden Schema–Extensionen und — in Form eines deklarativen Kalküls — eine Anfragesprache mit typischem Funktionsumfang definiert [CT97]. Die Ausdrücke der Sprache basieren auf Termen aus Dimensionsausprägungen, Aggregierungsebenen, den darauf definierten Gruppierungsbeziehungen, Datenwürfelschemata sowie Aggregierungs- und skalaren Funktionen. Diese Terme können über logische Operatoren miteinander verknüpft werden. Wie auch — im Rahmen der konzeptionellen Datenmodellierung — in [GMR98] wird in [CT98] die Ableitung eines multidimensionalen Schemas aus einem ER–Schema des Basisdatenbestandes genau untersucht.

Ein weiterer Vorschlag eines multidimensionalen Datenmodells in [DT97] kombiniert im wesentlichen Konzepte der vorangegangenen vier Modelle, so daß auf diesen nicht genauer eingegangen werden soll.

***The Nested Multidimensional Data Model* von Lehner (1998)** Lehner behandelt das Problem, daß — gerade im speziell betrachteten Anwendungsgebiet der Marktforschung bzw. allgemein betriebswirtschaftlicher Anwendungen — verschiedene Kategorien einer Aggregierungsebene nach *unterschiedlichen* Kriterien feiner unterteilt werden können [Leh98]. So werden etwa in einer Dimension *Produkt* Waschmaschinen durch ihren Wasserverbrauch und Videorecorder durch ihr Videosystem näher charakterisiert. Zusätzlich zu der üblichen Kategorienhierarchie definieren in diesem Modell sogenannte *Features* (*Dimensionsattribute*) Eigenschaften von Kategorien, die diesen in einer der Kategorienhierarchie entsprechenden Klassenhierarchie zugeordnet sind. Somit kann eine Kategorie als Oberklasse aller ihr untergeordneten Kategorien über die Ausprägungen für sie speziell definierter Features lokale Teilklassifikationen definieren. Auf dieser Grundidee aufbauend ergibt sich ein zweistufiger Analyseprozeß in der Anwendung multidimensionaler Operatoren:

1. Auffinden des Analysekontextes (klassifikationsorientiert).
2. Exploration des Kontextes (merkmals- (Feature–)orientiert).

Das vorgeschlagene Datenmodell definiert entsprechend multidimensionale Strukturen, die zunächst einen Raum über die klassifizierenden Attribute aufspannen, dessen Zellen dann wiederum gemäß der jeweiligen betrachteten Features mehrdimensionale Teilräume darstellen. Neben den üblichen Operatoren Drill–down, Roll–up sowie Slicing ermöglichen *Split* und *Merge* das Hinzufügen bzw. Entfernen eines Features zur Aggregierung der Daten.

Vassiliadis (1998) Das erklärte Ziel des Modellierungsansatzes von Vassiliadis [Vas98] ist eine möglichst natürliche und einfache Modellierung der OLAP–Operationen zur Restriktion und Aggregierung in Navigations- und Anfragesequenzen explorativer Datenanalysen. Hierbei grenzt er sich von den doch oft recht komplexen kalkülbasierten Anfragesprachen aus anderen Datenmodellen ab. Der Datenwürfel als Basisdatenstruktur wird in gängiger Weise modelliert, wobei mit jedem Datenwürfel jedoch ein weiterer „Basisdatenwürfel“ assoziiert ist, der die jeweils betrachteten Daten auf der feinstmöglichen Aggregierungsebene vorhält. Somit können nacheinander angewendete Aggregierungs- und auch Disaggregierungsoperatoren intern stets unter Rückgriff auf diese Basisdaten ausgeführt werden, so daß viele Probleme der (Nicht–)Aggregierbarkeit von Maßzahlen gelöst werden. Dem steht natürlich ein erhöhter Aufwand für Speicherung und ggf. Aktualisierung der redundant gehaltenen Daten gegenüber. Weiterhin sind durch die Einführung mengenwertiger Maßzahlen sowie gruppierender *Level–Climbing*– und *Packing*–Operatoren Aggregierungen elegant definierbar.

Modellierung komplexer multidimensionaler Daten nach Pedersen und Jensen (1999) Eine Kombination von Ideen aus den Bereichen „OLAP“ und „Statistische Datenbanken“ findet sich in der Arbeit von Pedersen und Jensen [PJ99], die ein semantisch reicheres Datenmodell zum Ziel hat, um so die direkte Interaktion mit den betrachteten Daten (im Gegensatz zu vordefinierten Reports) zu erleichtern. Schwerpunkte liegen in der flexiblen Definition von (auch multiplen und nicht strikt hierarchischen) Kategorienhierarchien, der Symmetrie von Dimensionen und Maßzahlen, der Integration von Daten unterschiedlicher Granularität, der Unterstützung mengenwertiger Eigenschaften, der korrekten Aggregierung unterschiedlicher Datentypen, der Versionierung

von Daten und in der Modellierung von Unsicherheiten. Die Zellen eines Datenraums werden in relationaler Form als Menge von Paaren aus einem Untersuchungsobjekt (etwa der ID einer Person) und einem Wert einer seiner Eigenschaften oder einer daraus abgeleiteten Maßzahl repräsentiert. Diese Paare entsprechen also entweder Zellkoordinaten oder Maßzahlwerten. Hierüber werden die relationalen Operatoren sowie eine Aggregierung zu Mengen von Untersuchungsobjekten und deren Eigenschaften definiert.

Das OLAP Council (1998) Schließlich seien noch die Bestrebungen des *OLAP Council*, einer Vereinigung von Herstellern von OLAP–Produkten, zur Definition einer einheitlichen, objektorientierten Anfrageschnittstelle für OLAP–Systeme erwähnt [Ola98]. Das dieser Schnittstelle zugrundeliegende sehr allgemein und offen gehaltene[38] Modell verdeutlicht — ähnlich wie einige der oben vorgestellten Ansätze — wiederum die große Heterogenität der Anforderungen verschiedener Zielgruppen bzw. der Implementierungen verschiedener Systeme.

2.3.4 Weitere Forschungsthemen

Abschließend werden in diesem Abschnitt zur Abrundung der Vorstellung multidimensionaler Datenbanksysteme einige wichtige Forschungsthemen in diesem Bereich angerissen. Viele grundlegende Aspekte wurden bereits Anfang der achtziger Jahre (vgl. etwa [Sho82]) diskutiert. Einen umfassenden Überblick geben z. B. [Leh99, Ruf97, WB97].

Neben Fragen adäquater Datenmodellierung, daraus abgeleiteter Anfragesprachen (siehe hierzu auch [ÖÖ85]) und der Entwicklung komfortabler Benutzungsschnittstellen stand in den letzten Jahren — insbesondere aufgrund des Bedarfs an Analysemöglichkeiten auf sehr großen betriebswirtschaftlichen Datenbeständen — die Entwicklung von Verfahren zur Gewährleistung eines *effizienten* Datenzugriffs im Vordergrund der Forschung.

Vor allem werden Varianten der Speicherung und Verwaltung multidimensionaler Daten diskutiert. Hierbei werden grob drei Richtungen unterschieden:

Relationale Verwaltung (*ROLAP*): Die Daten werden, wie bereits im vorigen Abschnitt dargestellt, in relationalen Datenbanksystemen verwaltet. Als Vorteile werden hierbei vor allem der Rückgriff auf etablierte Technologien sowie die bessere Skalierbarkeit für sehr große Datenräume angeführt.

Proprietär multidimensionale Verwaltung (*MOLAP*): Multidimensionale Datenwürfel werden über geeignete Ordnungen (im einfachsten Fall zeilenweise) linearisiert und getrennt von den Dimensionsinformationen in multidimensionalen Datenbanksystemen abgelegt. Die Zuordnung einer Zelle zu den jeweiligen Kategorien ergibt sich allein über die Position im Feld. Dieses Vorgehen erlaubt eine problemadäquatere Repräsentation und Verarbeitung, spart Speicherplatz (für die Schlüsselinformationen zu den Zellen), erfordert jedoch zunehmend aufwendige Kompressionsverfahren für große, dünn besetzte Datenwürfel (mit vielen Nullwerten).

Hybrides OLAP (*HOLAP*): In letzter Zeit bewegen sich viele Anbieter auf die Verschmelzung von ROLAP (zum Zugriff auf Basisdaten) und MOLAP (zur Verwaltung voraggregierter Datenwürfel für eine effizientere Datenbereitstellung) zu.

Diskussionen dieser Varianten finden sich in [DSHB98, WKC96, Sho97, WB97]. Darüber hinaus sind auf den WWW–Seiten zahlreicher Anbieter von OLAP–Software eine Reihe von Darstellungen der Vorzüge der verschiedenen Techniken erschienen, siehe etwa [Ken95] für MOLAP, [Mic95] für ROLAP, [Arb95] für

[38] So kann z. B. eine Dimension mehrere disjunkte Hierarchien umfassen, wobei Elemente einer Aggregierungsebene auch von unterschiedlicher „Granularität“ sein dürfen, so daß etwa auch innerhalb einer Ebene Aggregierungsbeziehungen bestehen können.

eine differenzierte Gegenüberstellung der jeweiligen Vorzüge und Nachteile beider Ansätze oder [Ala97] für HOLAP.[39]

Insbesondere im ROLAP–Kontext spielen Fragen der *Indexierung* [Sar97, OG95] eine wichtige Rolle. Häufig benutzte Aggregierungen der Basisdaten, insbesondere die „Randsummen“ über bestimmte Kombinationen von Dimensionen sind zu identifizieren [GHR97, HRU96] und effizient zu berechnen und zu verwalten [AAD+96]. Derartige redundante *materialisierte Sichten* sind wiederum je nach Anforderungen der Anwendung in geeigneten Zeitintervallen auf den jeweils aktuellen Stand zu bringen (Stichwort *View Maintenance*) [AKGM96, Zac98]. Maintenance–Aufwand, Antwortzeiten für typische Anfragen und Speicherbedarf bilden die Hauptkriterien zur Auswahl zu materialisierender Sichten [TLS99]. Vor allem im Rahmen der Datenspeicherung in MOLAP–Implementierungen werden schließlich auch Zerlegungen von Datenwürfeln (*Chunking*) in bedarfsgerechte Teile [FRW+97, RZ96, SS94] sowie der Einsatz von *Kompressionsverfahren* [Bas85] diskutiert.

Bei der Diskussion um OLAP und multidimensionale Datenbanken in letzter Zeit etwas vernachlässigt, aber insbesondere im Kontext statistischer Datenbanken eingehender behandelt wurden Fragen der Datensemantik sowie der *korrekten* statistischen Verarbeitung von Makrodaten. Hierbei sind zu nennen

- die Repräsentation und Auswertung verschiedener Arten von *Nullwerten* [Sho82, Fro97],
- *Datenschutzaspekte*, vor allem was den Rückschluß von Makrodaten auf schützenswerte Mikrodaten betrifft [Mic91a], sowie
- Fragen der (automatischen) *Aggregierbarkeit* von Makrodaten (im Rahmen eines Typsystems zum *Summary Type Management*), d. h. wann, für welche Maßzahlen und Dimensionen, mit welchen Funktionen können Makrodaten höher aggregiert werden (und wie kann dies automatisch, d. h. ohne Eingriff des Benutzers geschehen) bzw. wann werden die zugrundeliegenden Mikrodaten benötigt [LS97, LAW98]?

Gerade im Zusammenhang mit der automatischen Aggregierung multidimensionaler Datensätze, aber auch etwa zur adäquaten Gestaltung komfortabler Benutzungsschnittstellen zum Datenzugriff spielt die geeignete Beschreibung der betrachteten Daten in vor allem struktureller und semantischer Hinsicht durch *Metadaten* eine große Rolle. Auf diesen Aspekt bzw. die Idee hinter der Verwendung von Metadaten soll im folgenden Abschnitt genauer eingegangen werden.

2.4 Metadaten

Mit zunehmender Größe von in rechnerbasierten Informationssystemen angesammelten Datenbeständen sowie mit der fortschreitenden Vernetzung von Rechnersystemen, die einen verteilten Zugriff auf Daten und Informationen erlaubt, steigt auch der Bedarf an zusätzlichen *Metadaten* zur näheren Beschreibung all dieser Daten, also ihrer Definition, Dokumentation, Selektion, Manipulation und Darstellung [McC82]. Metadaten erleichtern Verwaltung, Suche und korrekte Verarbeitung von Daten bzw. machen diese erst in sinnvoller Weise möglich.

Definiert werden Metadaten allgemein als *Daten über Daten* [McC82] oder auch als *Informationen, die notwendig sind, um Daten sinnvoll nutzen zu können* [BS94]. Sie geben den eigentlich zu untersuchenden Daten erst eine Bedeutung, helfen bei deren Interpretation, identifizieren, adressieren und organisieren Daten [Ole96]. Speziell dem Kontext statistischer Datenbanksysteme[40] entstammt die Definition von Lenz (siehe [Len94b]):

[39] Bezeichnenderweise sind gerade die Beiträge [Ken95, Arb95], die (zumindest teilweise) den Einsatz der MOLAP–Technologie propagieren, heute nicht mehr im WWW unter den Seiten der jeweiligen Systemanbieter zugreifbar. Auch dies weist — im Sinne von HOLAP — auf die Tendenz zur Nutzung eines unterliegenden relationalen Datenbanksystems, dessen Funktionalität durch eine MOLAP–Komponente ergänzt wird, hin.

[40] In diesem Abschnitt ist häufig von *statistischen* — im Gegensatz zu *multidimensionalen* — Datenbanksystemen die Rede, um vor allem den zentralen Bezug zur Datenanalyse hervorzuheben.

Metadaten beschreiben Mikro- und Makrodaten sowie auf diesen durchführbare oder durchgeführte Operationen auf semantischer, struktureller, statistischer und physikalischer Ebene, um eine sinnvolle Speicherung, Transformation, Abfrage und Verbreitung zu ermöglichen.

Nicht immer läßt sich klar zwischen Daten und Metadaten unterscheiden, vielmehr hängt dies oft vom Kontext bzw. der Art der Betrachtung ab. So können etwa kategorielle Attribute eines multidimensionalen Datenraums sowohl als Metadaten zur Beschreibung der Zellen des Datenraums als auch als Attributdaten, die selbst Gegenstand der Untersuchung sind, angesehen werden.

Damit Metadaten und somit auch die jeweils durch diese beschriebenen Daten einheitlich genutzt und interpretiert werden können, ist eine Standardisierung von Metadaten anzustreben [Ole96]. Einige Ansätze hierzu im Kontext der Konzeption von Metadaten–Management–Systemen für statistische Daten bieten [GAL96, KS97, Nin97, Bre94]. Einen allgemein anerkannten Metadaten–Standard gibt es derzeit zwar vor allem aufgrund der sehr heterogenen Anforderungen an Metadaten in unterschiedlichen Anwendungsszenarien noch nicht[41], aktuell werden jedoch verstärkt entsprechende Bemühungen auf der Basis erweiterbarer Informationsmodelle für objektorientierte Datenbestände forciert. Zu nennen sind hier die *Meta Object Facility* (*MOF*) der *OMG* für *CORBA*–basierte (verteilte) Umgebungen [OMG99] sowie das *Open Information Model (OIM)* der *Meta Data Coalition* [MDC99], das u. a. im *Microsoft Repository* implementiert wird [BBC+99]. Als grundlegende Modellierungssprachen werden in diesen Ansätzen *XML* (die *Extensible Markup Language*) [Bra98a] und *UML* [FS98] genutzt.

Metadaten finden sich in (fast) allen Bereichen der Daten- und Informationsverarbeitung — angefangen von Übertragungsprotokollen in Rechnernetzen über Betriebssystemtabellen bis zu Datenkatalogen in geographischen oder Umwelt–Informationssystemen. Die folgenden Betrachtungen konzentrieren sich auf das in dieser Arbeit untersuchte Gebiet statistischer bzw. multidimensionaler Datenbank- und Informationssysteme. Zunächst werden in Abschnitt 2.4.1 verschiedene Arten von Metadaten im Datenbankkontext sowie deren Klassifikationsmöglichkeiten vorgestellt, bevor dann in Abschnitt 2.4.2 speziell auf Anforderungen aus dem Bereich der Analyse multidimensionaler Datenbestände eingegangen wird.

2.4.1 Klassifikation von Metadaten

Metadaten können je nach Anwendungszweck von sehr unterschiedlicher Art und Gestalt sein. Im folgenden seien, basierend auf Vorschlägen aus [Ole96, McC82, BS94], einige mehr oder weniger grobe Möglichkeiten zu ihrer Klassifikation bzw. Kategorisierung vorgestellt. Hierbei sind Abgrenzungen zwischen verschiedenen Arten von Metadaten nur selten ganz exakt zu definieren; viele Untergruppen sind nicht disjunkt.

In Anlehnung an die Unterscheidung von externer, konzeptioneller und interner Schema–Ebene in Datenbanksystemen [TK78] können Metadaten nach der Art ihrer Verwendung differenziert werden:

Nutzer von Metadaten: Systembenutzer, Datenbankadministratoren, Anwendungssysteme oder Verarbeitungsprozeduren sowie das DBMS selbst können Zielgruppen für die Verwendung von Metadaten sein.

Aufgabe von Metadaten: Metadaten unterstützen die Datenverarbeitung in den Bereichen

- Suchen, Browsen und Abfragen (also bei der Identifikation für eine Fragestellung relevanter Daten),
- Qualitätssicherung und Reproduktion von Daten (Datendefinition und -integrität unter Einbeziehung von Hintergrundinformationen),
- Transfer zwischen Applikationen (automatische Datenintegration über geeignete Protokolle und Konvertierungen) sowie
- Basisfunktionalität eines Datenbanksystems als Archiv oder Lager (hierbei vor allem beim effizienten und sicheren Datenzugriff).

[41] Dies gilt im wesentlichen auch bereits für eingeschränkte Anwendungsdomänen, wie etwa die Verwaltung multidimensionaler Daten.

Führende vs. kontrollierende Metadaten: Metadaten können sowohl führend, als Informationsquelle für den Benutzer, als auch kontrollierend, zur Gewährleistung der internen Funktionalität, verwendet werden. Unter den kontrollierenden Metadaten kann ferner die Zuordnung zu einer Ebene der klassischen (ANSI/SPARC–)Schichtenarchitektur von Datenbanksystemen [Här87, Vos99] — von der Definition des konzeptionellen Schemas über Zugriffspfade bis hin zu Speicherungsstrukturen — vorgenommen werden.

Abbildung 2.20 faßt diese Kriterien anhand einiger ausgewählter Beispiele von Metadaten in einer Graphik zusammen. Hierbei — wie auch im Falle der meisten hier vorgestellten Klassifikationen — ist sicherlich aufgrund starker Anwendungsabhängigkeiten und Aufgabenüberschneidungen nicht immer eine eindeutige Einordnung einzelner Metadaten möglich.

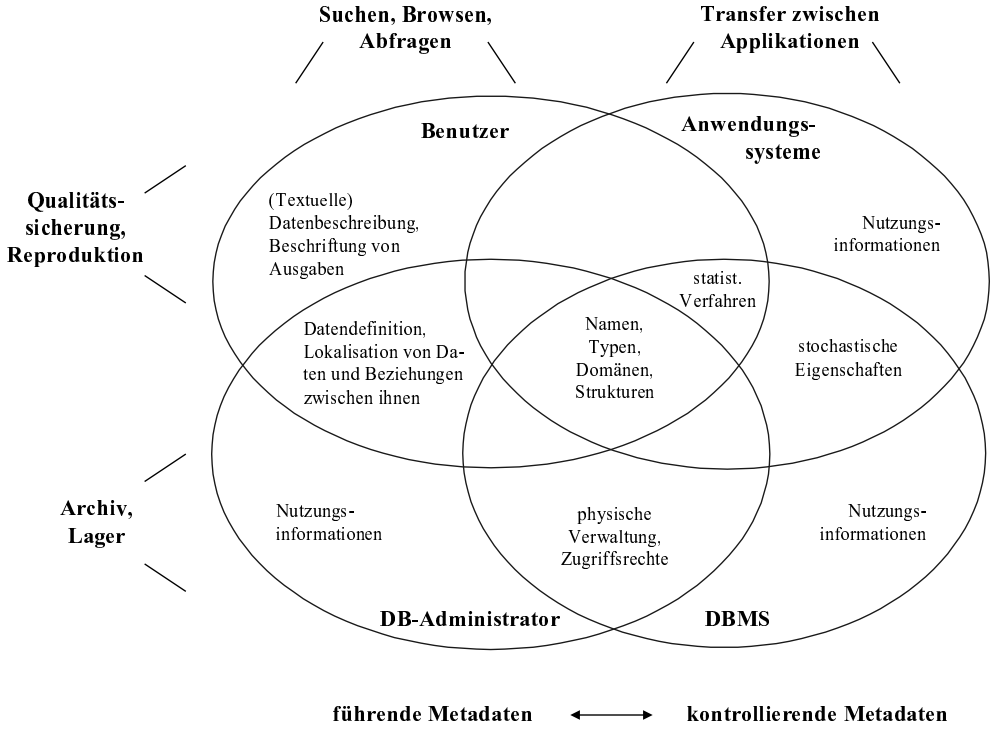

Abbildung 2.20: Aufgaben und Nutzer von Metadaten (nach [McC82] und [BS94])

Auf externer und konzeptioneller Schemaebene können ferner folgende Kriterien unterschieden werden:

Elementare vs. zusammengesetzte Metadaten: Metadaten können elementar (Bezeichner, Zahlen etc.) oder aus anderen Komponenten in komplexen Strukturen zusammengesetzt sein.

Datentypen: Viele Metadaten sind textuell, aber auch numerische oder klassifizierende Metadaten sowie — gerade in statistischen Datenbanksystemen — Metadaten in Form von Ausdrücken und Funktionen sind üblich.

Granularität beschriebener Daten: Metadaten beschreiben Daten auf unterschiedlichen Granularitätsebenen — beginnend bei einzelnen Ausprägungen über Merkmale, Entitäten, Mengen von Entitäten bis zu ganzen Datenbasen oder Kontexten. Analog werden auf der physischen Ebene etwa Dateien, Blöcke, Seiten oder Segmente beschrieben.

Interne vs. externe Metadaten: Die Nutzung von Metadaten kann auf das jeweilige Informationssystem beschränkt sein; Metadaten können aber auch gerade für den Zugriff durch externe Systeme bereitgestellt werden.

Speziell auf der internen Ebene, also im Rahmen der physischen Verwaltung, spielen schließlich weitere Aspekte eine Rolle:

Implizite vs. explizite Metadaten: Implizite Metadaten sind

- Teil der Daten (Jahresangaben in einem Datumsfeld können z. B. auch als Kontext interpretiert werden),
- Teil der Schemadefinition durch andere Metadaten (z. B. Einheiten, die in Attributbezeichnern verborgen sind) oder
- können aus den Daten abgeleitet werden (etwa stochastische Eigenschaften oder zusammenfassende Maßzahlen).

Explizite Metadaten dagegen sind klar von den jeweiligen Daten getrennt und unterscheidbar.

Von den Daten separierte Verwaltung: Die getrennte Speicherung und Verwaltung von Daten und Metadaten kann — je nachdem, ob klare Kapselung oder ein geringer Verwaltungsaufwand im Vordergrund stehen — auf der Ebene von Attributen, Entitäten, Entitätsmengen, ganzen Datenbasen, die durch ein gemeinsames DBMS verwaltet werden, separaten Subsystemen in einem Informationssystem oder auch in unabhängigen, kooperierenden Systemen erfolgen.

All diese Kriterien bilden die Grundlage für Überlegungen zur adäquaten Verwaltung von Metadaten in Datenbank- und Informationssystemen. Auf die Implementierung derartiger Verwaltungskomponenten in Form von Metadaten–Managementsystemen, Repositories oder Datenwörterbüchern (Data Dictionaries) soll hier nicht näher eingegangen werden (vgl. oben im Rahmen der Standardisierung von Metadaten genannte Ansätze sowie die allgemeine Datenbankliteratur [LS87, Vos99, EN94]). Einzelheiten der speziell auf statistische Daten ausgerichteten Verwaltung von Metadaten in einem Metadaten–Repository auf der Basis eines Metadaten–Schemas werden etwa in [GA97] oder [Len94b] betrachtet.

2.4.2 Metadaten im Kontext multidimensionaler Datenmodellierung und -analyse

Das Anwendungsgebiet der multidimensionalen Datenanalyse bzw. die Verarbeitung statistischer Daten auf der Grundlage multidimensionaler Datenbanken stellt aufgrund

- der zeitlichen Fortschreibung statistischer Datenbasen,
- der Integration heterogener Datenbestände aus verschiedenen Quellen,
- der Verwaltung komplexer Datenanalysen mit ihren jeweiligen (Zwischen–)Ergebnissen,
- des Bedarfs einer differenzierten Interpretation von Analyseresultaten sowie
- der Datennutzung in einer hochinteraktiven Umgebung durch verschiedene Benutzergruppen

besondere Anforderungen an die Nutzung von Metadaten zur Bereitstellung „selbstbeschreibender Datensätze“ [McC82]. Während Metadaten in „klassischen“ Datenbankanwendungen eher eine untergeordnete oder — besser gesagt — „unauffällige“ Rolle spielen, treten sie hier in den Vordergrund. So spricht etwa Lenz ganz explizit von einer M^3–Datenbankarchitektur, die sowohl Mikro-, Makro- als auch Metadaten adäquat verwaltet [Len94a], und auch Hand nennt in [Han92] diese drei grundlegenden Arten statistischer Daten. Datenverwaltung und -analyse bestehen entsprechend in der *expliziten* Verarbeitung von Daten *und* Metadaten; jeder Datenanalyseschritt geht stets mit einem Schritt der Metadatenanalyse einher [Fro97, Han97b].

Ähnlich wird im Bereich des konzeptionellen Entwurfs von Data Warehouses der Berücksichtigung von Metadaten zur Herstellung eines Bezugs zu anwendungsbezogenen Begriffswelten („Business Terms“) sowie

zur Sicherung der Datenqualität von aus verschiedenen Datenquellen integrierten Datenbeständen eine hohe Bedeutung beigemessen [JJQV99, MSR99, Hin99].

Zum einen spielt in der Datenanalyse gegenüber anderen Domänen der Gebrauch *führender* Metadaten eine deutlich stärkere Rolle, um den Benutzer über Quelle, genaue Berechnung und sinnvolle Verarbeitungs- und Interpretationsmöglichkeiten zu informieren. Zum anderen werden aber auch *kontrollierende* Metadaten, insbesondere Ausdrücke und Berechnungsfunktionen (etwa zur automatischen Aggregierung von Makrodaten, vgl. [CMM89]), exzessiv genutzt. Ghosh ordnet in [Gho88] sogar den gesamten Komplex der Metadaten in statistischen Datenbanken letzterer Sichtweise unter, indem er Metadaten als komplexe Datentypen mit Regeln zu ihrer Berechnung definiert.

Den vielschichtigen Charakter von Metadaten im Kontext der Datenanalyse verdeutlichen Auflistungen in [Fre91, Len94b, McC82, Mar97, Fro97]. Es werden unter anderem genannt:

- Angaben zur (physischen) Datenquelle und -qualität, zum Datenkontext sowie weitere Hintergrunddokumentation und Fußnoten,
- Details bzgl. Ort, Zeit, betrachteter Population, Urheber und Art der Datenerhebung,
- Namen, Beschriftungen und Formatierungen von Datenobjekten,
- logische Strukturen, vor allem kategorielle und summarische Attribute mit ihren jeweiligen Typen, Wertebereichen, Klassifikationen, Skalierungen, Einheiten und Nullwerten,
- Transformationen, angewendete Programme zur Datenmanipulation und -anonymisierung, Fehlerraten, stochastische Eigenschaften und unterliegende statistische Modelle,
- Spezifikation von Datenherleitung und Verarbeitungsprozeduren sowie
- Beschreibung der physischen Datenspeicherung, Zugriffspfade, -rechte und -statistiken.

Zusätzlich zu den im vorangegangenen Abschnitt genannten Kriterien bietet sich hier die Betrachtung des Datenanalyseprozesses (siehe Abb. 2.21)[42] zur Klassifikation der anfallenden Metadaten an (vgl. [vdBdF92, Gro98]). Dieser Prozeß veranschaulicht den gesamten Lebenszyklus statistischer Daten.

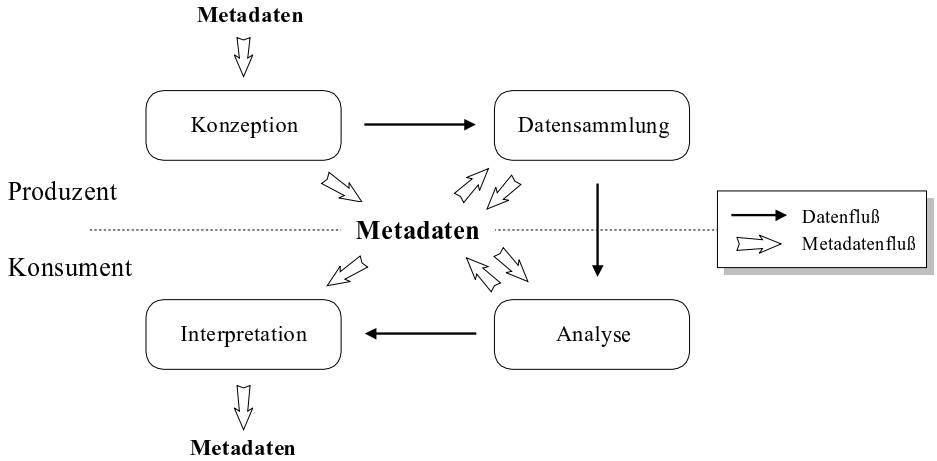

Abbildung 2.21: Metadaten im Datenanalyseprozeß

Zum einen bildet die Beschreibung des jeweiligen Analyseablaufs selbst, also der Herleitung von Analyseergebnissen, einen wesentlichen Bestandteil der benötigten Metadaten [SSW96, KBS94, Hub94]. Zum anderen

[42] Wie in Abschnitt 2.1 gesehen, ist dieser Analyseprozeß natürlich nicht streng linear; der entsprechende Kontrollfluß ist an dieser Stelle jedoch nicht weiter relevant.

werden innerhalb dieses Prozesses in vier Teilprozessen Daten verarbeitet, wobei jeweils Metadaten vorangegangener Verarbeitungsschritte genutzt und Metadaten als Beschreibung der generierten (Zwischen–)Ergebnisse für die nachfolgende Betrachtung erzeugt werden. Diese vier Teilprozesse sind

Konzeption: Definition der Problemstellung und Datenmodellierung,

Datensammlung: Durchführung der Erhebung mittels eines Fragebogens bzw. in einer Meßumgebung sowie Speicherung der Daten (in einer Datenbank),

Analyse: Selektion, Transformation und Anwendung von Analyse- und Visualisierungsverfahren in einem explorativen Prozeß sowie

Interpretation: Ziehen von Rückschlüssen aus der Datenausgabe und -visualisierung.

Die als Hintergrundwissen in die Konzeption einer Analyse einfließenden und als Ergebniswissen (etwa in Form von Berichten) durch die Analyse generierten Metadaten sind in Abb. 2.21 separat dargestellt.

Nach [Fro97] läßt sich in diesem Prozeß entsprechend der oberen bzw. unteren Hälfte in Abb. 2.21 noch zwischen (Daten–)Produzenten und (Daten–)Konsumenten differenzieren, wobei die Trennung (gerade im Hinblick auf abgeleitete Zwischenergebnisse in der Analysephase) nicht unbedingt ganz klar zwischen Phase zwei und drei verlaufen muß. Jeder Phase der Datenanalyse können nun unterschiedliche Arten von Metadaten — quasi als Sichten auf den gesamten Metadatenbestand — schwerpunktmäßig zugeordnet werden.

Weitere speziell auf die Charakteristika statistischer Datenbanken abzielende Betrachtungen von Metadaten

- unterscheiden taxonomische/terminologische Metadaten (Bedeutung aller Begriffe und Konzepte), statistische Metadaten (stochastische Eigenschaften und statistische Kenngrößen) sowie technische Metadaten (Verwaltung, Speicherung, Layout) [Fro97],
- orientieren sich (in ähnlicher Weise) an der ANSI/SPARC–Datenbankarchitektur — die konzeptionelle Ebene definiert semantische (Bedeutungen, Hintergründe, etc.) und statistische Metadaten (Typen, Verfahren etc.), die externe Ebene den Datenanalysephasen aus Abb. 2.21 entsprechend verschiedene Sichten auf die Metadaten und die interne Ebene spezielle technische Aspekte zu Datenspeicherung und Datenschutz [Len94a] — oder
- beziehen sich wiederum auf die Granularitätsebenen der beschriebenen Daten (Studien, Merkmale, Variablen etc.) [Mar97, Len94b, Han92].

Soweit soll der Einblick in die Nutzung von Metadaten bei der Datenanalyse für den Entwurf des Datenmodells *MADEIRA* in Kapitel 4 ausreichend sein. In Abschnitt 4.4 wird nochmals speziell auf die Nutzung von Metadaten in *MADEIRA* eingegangen.

Kapitel 3

Datenanalysesprachen und -umgebungen

Die Datenanalyse ist ein außerordentlich breit gefächertes Gebiet, das entsprechend auch eine sehr große Zahl von Systemen zu ihrer Unterstützung hervorgebracht hat. Unterschiedliche Anwendungsgebiete und Anwendergruppen stellen differenzierte Ansprüche an Benutzungsschnittstellen und Funktionalität, die in ihrer Breite und Tiefe sicher nicht von einem einzigen System erfüllt werden können. Im folgenden wird ein Überblick über die wichtigsten Klassen von Analysesystemen mit ihren jeweiligen Charakteristika gegeben. Ein Schwerpunkt liegt hierbei auf der Vorstellung datenflußbasierter Umgebungen zur Datenanalyse. Daneben wird auch auf einige ausgewählte Ansätze zur interaktiven Datenvisualisierung genauer eingegangen, da entsprechende Überlegungen auch eine wesentliche Rolle für die Konzeption von *VIOLA* spielen werden. Darüber hinaus werden einzelne Analysesysteme nur dann zur Darstellung herangezogen, wenn sie für die jeweilige Systemklasse typische Merkmale besonders anschaulich umsetzen oder Eigenheiten der „intelligenten“ Datenanalyse vorweisen, die im Rahmen dieser Arbeit noch von besonderem Interesse sein werden. Ergänzend wird in den einzelnen Abschnitten jeweils auf umfassende Übersichten und Systemevaluationen in der Literatur sowie im WWW verwiesen.

In Anlehnung an die in Abschnitt 2.1 vorgestellten unterschiedlichen Herangehensweisen an die Datenanalyse werden zunächst „klassische“, universell einsetzbare Statistikpakete und Analysesprachen (Abschnitt 3.1), anschließend Umgebungen zur (insbesondere interaktiven) Datenvisualisierung (Abschnitt 3.2) und Tools für OLAP und Data Mining (Abschnitt 3.3) vorgestellt, bevor dann in Abschnitt 3.4 Varianten der datenflußbasierten visuellen Programmierung in der Datenanalyse näher betrachtet werden. Abschließend wird in Abschnitt 3.5 noch kurz auf die wissensbasierte Unterstützung der Datenanalyse eingegangen, die prinzipiell in allen anderen der genannten Systemklassen zum Einsatz kommen kann, also orthogonal zu diesen anzusiedeln ist.

In diesem Kapitel wird die in Abschnitt 2.1.2 (Abb. 2.5) eingeführte Darstellung von Interaktionsmodellen zur Gegenüberstellung der durch die vorgestellten Klassen von Analysesystemen unterstützten Folgen von Datenanalyseinteraktionen genutzt. Dieses Schema ist mit der Unterscheidung von Datenwahl und -management, Statistik, Visualisierung und Navigation sowie deren Verknüpfung in einem gerichteten Graphen recht grob, reicht aber aus, um die jeweiligen Charakteristika der Untersuchungen einzelner Fragestellungen durch die entsprechenden Systeme herauszuarbeiten. Zur Verdeutlichung der Abgrenzung werden Sachverhalte hierbei etwas überzeichnet, indem nur *typische* Sitzungen repräsentiert werden, die die jeweilige Grundphilosophie der Datenanalyse in repräsentativen Systemen einer Klasse besonders gut wiedergeben.

Interaktionsfolgen, die eine untergeordnete Bedeutung haben, jedoch nicht völlig zu vernachlässigen sind, werden durch gestrichelte Pfeile angedeutet. Unter einer Navigationsinteraktion soll hier der Wechsel zu einem bereits durchgeführten Analyseschritt zur anschließenden Modifikation oder darauf aufsetzenden Fortführung der Analyse verstanden werden. Bei der Datenvisualisierung wird unterschieden, ob Graphiken lediglich erstellt werden oder auch zur direkten *visuellen* Spezifikation von Datenmanagementinteraktionen dienen können. In letzterem Fall ist diese Interaktionsklasse fett gedruckt. Wird durch derartige Visualisierungsin-

teraktionen nicht nur stets die aktuell betrachtete Graphik (wie etwa bei Dynamic Queries) modifiziert, sondern können sich diese auch (im Sinne des Linking) auf andere Darstellungen auswirken, erfolgt zusätzlich eine Schattierung.

3.1 Universelle Datenanalysesysteme

Universell einsetzbare Datenanalysesysteme decken jeweils eine breite Palette von Datenanalyseanwendungen zu wesentlichen Teilen ab, müssen sich aber in der Regel spezialisierten Werkzeugen auf deren jeweiliger Domäne hinsichtlich Funktionalität und Bedienungskomfort geschlagen geben. Es sind zum einen die „klassischen“ Statistikpakete (siehe Abschnitt 3.1.1), deren erste Versionen aus den 60er und 70er Jahren stammen, und zum anderen flexibler einsetzbare Datenanalysesprachen (siehe Abschnitt 3.1.2), die etwa seit Mitte der 80er Jahre zum Einsatz kommen, zu unterscheiden.

Zwei recht umfassende tabellarische Überblicksdarstellungen von Software zur statistischen Analyse und Datenvisualisierung finden sich in [Bra95] und [Kor93], wobei letzterer vor allem solche Systeme betrachtet, die im Sinne „visueller Datenanalyse“ die (interaktive) Visualisierung von Analysen bzw. Analyseergebnissen in den Vordergrund stellen. Während diese Papiere stichwortartig vor allem die jeweils angebotene Funktionalität charakterisieren, stellt [HHK92] einige gängige statistische Auswertungssysteme bzgl. Daten- und Methodenverwaltung, verwendeten Auswertungs- und Programmiersprachen sowie ihres jeweiligen Anwendungsgebiets gegenüber. Im WWW findet sich etwa in [Hel99] eine sehr umfassende Auflistung von Systemen mit jeweiliger Kurzbeschreibung und Links zu näheren Informationen. Etwas ausführlicher, dafür eingeschränkt auf eine geringere Menge an Systemen ist [CCS96].

3.1.1 Allgemeine Statistikpakete

Die klassischen Statistikpakete, wie *SAS*, *SPSS* oder *BMDP*, bieten im wesentlichen große Bibliotheken statistischer Analysefunktionen insbesondere der schließenden Statistik, wobei natürlich auch die Datendeskription unterstützt wird. Ein wesentlicher Vorteil ihrer Verwendung besteht in der oft großen Verbreitung und daraus resultierend einer guten Validierung der Funktionalität sowie der einfachen Portierbarkeit auf verschiedene Rechnerumgebungen. Die Pakete definieren Kommandosprachen zum sequentiellen Aufruf und zur Parametrisierung von Routinen zu Datenverwaltung, -manipulation, -analyse und -präsentation. Die zu untersuchenden Daten werden in der Regel in tabellarischer Form in ASCII–Dateien übergeben oder aus relationalen Datenbanken gelesen. Die Datenauswertung erfolgt im Batch–Betrieb durch Abarbeitung einer Befehlsdatei oder interaktiv im Dialog über ein ASCII–Terminal. Etwa seit Mitte der 80er Jahre bieten neuere Versionen der Systeme auch graphische menübasierte Benutzungsschnittstellen zur Methodenwahl oder sogar Werkzeuge zur Entwicklung von Oberflächen als anwendungsspezifische Sichten auf das System. Auswertungsergebnisse umfassen i. a. Zusammenstellungen gleich mehrerer im jeweiligen Fragekontext interessierender Maßzahlen und werden auf dem Bildschirm ausgegeben oder in eine Datei geschrieben. Das unterstützte Interaktionsmodell (siehe Abb. 3.1) beschränkt sich also im wesentlichen auf die sequentielle Spezifikation von Datenwahl, Analyse und Visualisierung.

Einen typischen Vertreter dieser Systemklasse bildet *SAS* (*Statistical Analysis System*). Seit seiner ersten Version im Jahr 1971 hat es sich vom statistischen Auswertungssystem inzwischen zu einer umfassenden Umgebung zur Verwaltung, Analyse und Darstellung von Daten in vielen Anwendungsbereichen entwickelt. Das Ziel von *SAS* ist nach [SAS93] die „Integration des unternehmensweiten Informationsflusses in einem *Information Delivery System*“. Durch Erweiterungsmodule und zusätzliche Produkte öffnet sich *SAS* zunehmend anderen „modernen“ Bereichen der Datenanalyse. So unterstützt z. B. *SAS/INSIGHT* als ein intuitiv zu bedienendes Tool die interaktive Datenexploration; *SAS/Spectraview* zielt auf die Visualisierung großer, hochdimensionaler Datenmengen; es wurde — im Sinne von OLAP — ein multidimensionales Datenbanksystem integriert, und der *SAS Enterprise Miner* bietet unter einer einfachen datenflußbasierten Oberfläche ein mächtiges Data Mining Tool [Hös97].

3.1.2 Matrixbasierte Datenanalysesprachen

Matrixbasierte Analysesprachen wie z. B. *S–Plus*[1] [VR94, Bec94] zielen deutlicher als die doch recht monolithischen und unflexiblen Statistikpakete auf die explorative Datenanalyse. Neben einer Vielzahl arithmetischer Funktionen auf Matrizen und Vektoren sind Routinen zur statistischen Analyse (besonders multivariate Verfahren) und zur Darstellung der Daten (vor allem in dynamischen, interaktiven Graphiken) die wesentlichen Elemente meist objektorientiert realisierter Sprachen.

Weiterhin werden Befehle zur Datenein- und -ausgabe, zum Datenmanagement sowie einige Kontrollstrukturen für Analyseskripte zur Verfügung gestellt. Eigene Algorithmen können erstellt und als neue Befehle der Sprache hinzugefügt werden. Meist werden bereits umfangreiche Makrobibliotheken angeboten. Hier werden Flexibilität und Offenheit für Erweiterungen als wesentliche Systemmerkmale deutlich. Die Arbeit mit derartigen Datenanalysesystemen erfolgt über eine dialogorientierte, graphische Benutzungsoberfläche, ergänzt um Menüs für typische Auswertungen. Ein Interpreter führt die gewünschten Analysen und Visualisierungsbefehle sofort aus, wodurch der Benutzer zum Experimentieren mit den Daten ermuntert werden soll.

Die Ergebnisse von Auswertungen werden als dynamisch erzeugte Datenstrukturen abgespeichert und sind im Verlauf einer Sitzung jederzeit über deren Namen wieder verfügbar. Weiterhin können auch Sitzungsprotokolle erstellt und gesichert werden. Schließlich besteht die Möglichkeit, die Systemoberfläche den speziellen Anforderungen einer Anwendung anzupassen.

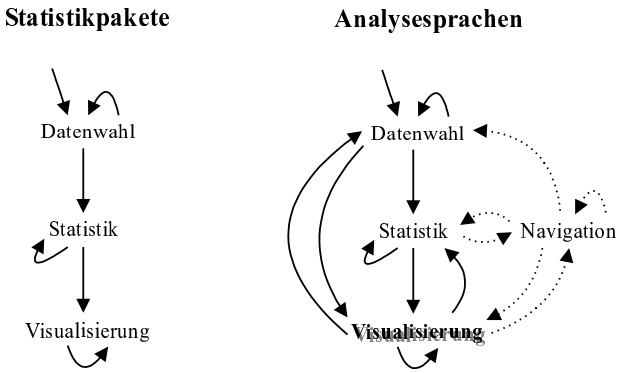

Abbildung 3.1: Interaktionsmodelle universeller Auswertungssysteme

Diese Grundphilosophie äußert sich auch in einem recht flexiblen Interaktionsmodell (siehe Abb. 3.1), das im Gegensatz zu den oben angeführten Statistikpaketen eine klar verbesserte Kontrolle über die Einzelschritte der Analyse ermöglicht. Die Navigation beschränkt sich jedoch im wesentlichen noch auf das (in einer Befehlshistorie zwar mögliche, aber doch recht mühsame) Auffinden bereits vorliegender Analyseergebnisse und deren weitere Nutzung. Ein direktes Ansprechen und Modifizieren alter Analyseschritte ist auch hier nicht möglich.

Ein Überblick über einige Analysesprachen findet sich in [CN97] sowie — speziell vor dem Hintergrund der interaktiven, visuellen Datenanalyse — in [WUT95].

3.2 Systeme zur Datenvisualisierung

Die herausragende Rolle der Datenvisualisierung für die *explorative* Datenanalyse wurde bereits in Abschnitt 2.1.3 diskutiert. Eine Vielzahl von Analysesystemen beschränkt sich auf *einfache* statistische Berechnungen (oder verzichtet sogar ganz auf diese) und konzentriert sich auf die rein graphische Darstellung von Datenbeständen. Hierunter fallen etwa auch gängige Tabellenkalkulationsprogramme wie *EXCEL* von *Microsoft*. Neben den bereits oben angeführten Übersichten [Bra95, Kor93] finden sich auch im WWW (etwa unter [KDN99]) Aufstellungen derartiger Software.

Im folgenden werden einige Beispiele für Visualisierungssysteme gegeben, die in besonderem Maße die explorative und interaktive Visualisierung von Daten ermöglichen. Ansätze hierzu kommen einerseits aus der konsequenten Umsetzung der Konzepte interaktiver statistischer Graphik als Ergänzung zu klassischen Statistikpaketen (Abschnitt 3.2.1) und andererseits aus Projekten zur datenbankbasierten Visualisierung großer

[1] *S–Plus* ist aus der bei *AT&T* entwickelten Sprache *S* entstanden [Cha93, Cha95] und wird jetzt von der Firma *Mathsoft* vertrieben.

Datenbestände (Abschnitt 3.2.2). In der aktuellen Entwicklung verschmelzen diese beiden Zweige zusehends. Abbildung 3.2 zeigt die Bandbreite der von interaktiven Visualisierungssystemen propagierten Interaktionsmodelle. Die Statistik spielt stets eine untergeordnete Rolle; in unterschiedlichem Maße werden nach einer initialen Datenwahl und -visualisierung noch nachfolgende Datenmanagementoperationen (evtl. auch auf zusätzlich herangezogenen Datensätzen) sowie ein Linking zwischen mehreren Graphiken unterstützt.

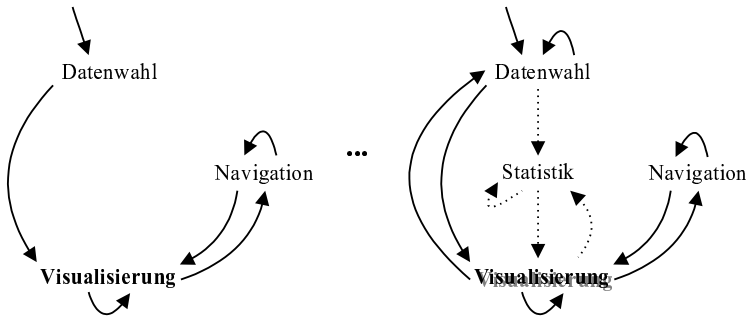

Abbildung 3.2: Interaktionsmodelle von Systemen zur Datenvisualisierung

3.2.1 Interaktive graphische Datenanalyse

Bereits die oben vorgestellten Analysesprachen bieten viele Möglichkeiten der interaktiven graphischen Datenanalyse. Die im folgenden vorgestellten Systeme *DataDesk* und *Manet* konzentrieren sich noch stärker auf diesen Aspekt. Die gesamte Analyse erfolgt fast ausschließlich durch direkte Manipulation von Graphiken und deren Komponenten; selbst die Wahl zu betrachtender Datensätze und Variablen geschieht mittels Drag–and–Drop. Alle üblichen Verfahren interaktiver Graphik (siehe Abschnitt 2.1.3) werden realisiert. Ein kurzer Überblick sowie ein Vergleich ähnlicher Systeme finden sich in [WUT95].

DataDesk Das System *DataDesk* [Vel99, The96a] wurde ursprünglich von Vellemann an der Cornell Universität entwickelt und wird inzwischen von der Firma *Data Description* vertrieben. Die einfach zu bedienende Benutzungsoberfläche von *DataDesk* besteht im wesentlichen aus einer Arbeitsfläche, auf der über ein Menü verschiedene Diagramme erzeugt werden können, die sich stets auf einen zuvor ausgewählten relationalen Datensatz beziehen. Dessen Variablen werden in einem weiteren Fenster zur Auswahl angeboten und können in die generierten Diagramme gezogen werden. Alle Graphiken sind empirisch miteinander gelinkt, ein eingeschränktes algebraisches Linking ist ebenfalls möglich. Zur Selektion stehen verschiedene Selektionswerkzeuge (Brush, Pointer, etc.) und -modi (Und–, Oder–Verknüpfung etc.) zur Verfügung. Auf den visualisierten Daten können unterschiedliche Maßzahlen und Modelle berechnet und in die Graphik einbezogen werden. Damit realisiert *DataDesk* in etwa das rechte der Interaktionsmodelle aus Abb. 3.2.

Als Besonderheit unter vergleichbaren Systemen bietet *DataDesk* wissensbasierte kontextsensitive Menüs, die zu einer Graphik Hinweise auf mögliche nachfolgende Analyseschritte und Parametrisierungsmöglichkeiten geben. Weiterhin erlaubt eine einfache datenflußbasierte visuelle Sprache die Spezifikation von Templates (Makros), also Folgen von Visualisierungen und Analyseschritten, die gespeichert und als Ganzes auf ausgewählte Datensätze angewendet werden können. Zudem können Visualisierungsschritte und -ergebnisse in Form einer „Folienshow“ gespeichert werden. Schließlich läßt sich durch eine integrierte Programmiersprache die Analysefunktionalität erweitern. Abbildung 3.3 zeigt ein Bildschirmfoto der Benutzungsoberfläche von *DataDesk* mit mehreren gelinkten Graphiken über einen Firmendatenbestand: In Histogrammen über Angestellten- und Verkaufszahlen (die zwei Fenster in der Mitte oben), einem Scatterplot dieser beiden Variablen (rechts unten) sowie einer Unternehmensliste (links oben) sind jeweils alle *Finanz*unternehmen (mittels des Balkendiagramms über Unternehmensarten links unten) selektiert.

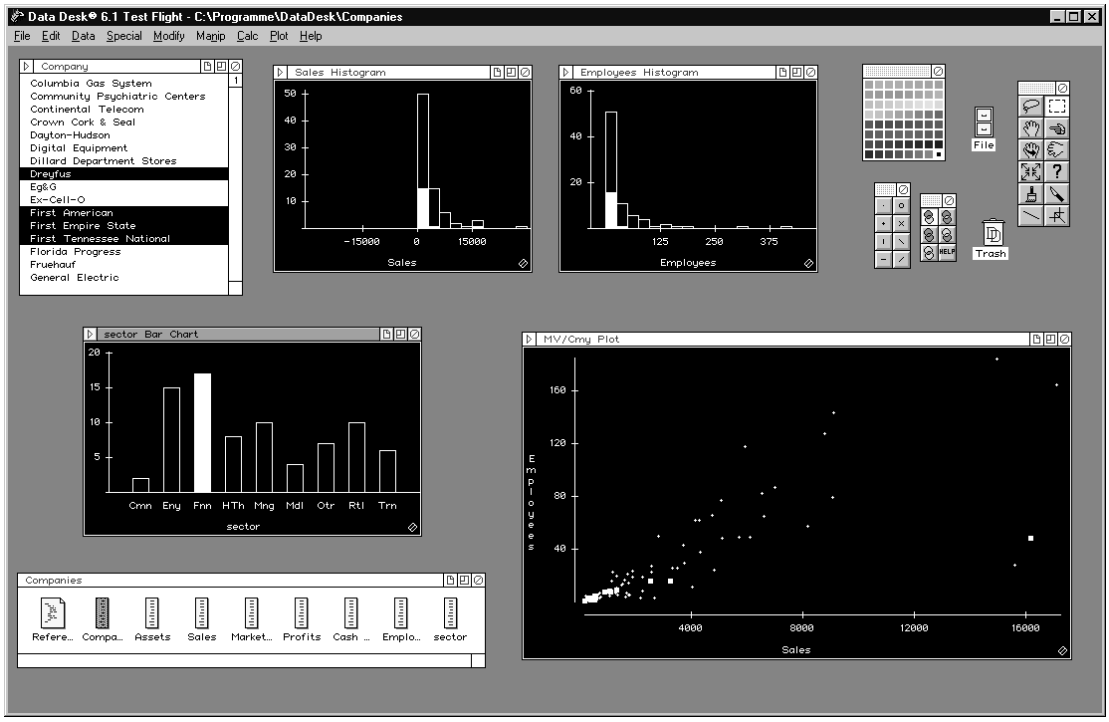

Abbildung 3.3: Die Benutzungsoberfläche von *DataDesk*

Manet In seiner Grundfunktionalität sehr ähnlich zu *DataDesk* stellt sich das an der Universität Augsburg entwickelte System *Manet* (*Missings are now equally treated*) [HTU97] dar. Zusätzlich wird hier ein Indexfenster zur Verwaltung des erzeugten „Fensterwaldes“ (vgl. auch [YS91]) angeboten; eine spezielle Rolle spielen auch Abfragen, da auf Skalen bzw. Skalenbeschriftungen oftmals verzichtet wird. Vor allem aber integriert *Manet* spezielle Konzepte zur Visualisierung von „Missing Values“, also unbekannten Ausprägungen (Nullwerten), in gängige Graphiktypen [THSU95]. Als Besonderheit sind auch noch raumbezogene Darstellungen und darauf definierte Interaktionen sowie Mosaikplots, eine Art über mehrere Variablen geschachtelter Histogramme, zu nennen. Statistische Analysefunktionen sind kaum implementiert.

3.2.2 Interaktive Visualisierung von Datenbanken

Mit der Ansammlung immer größerer Datenbestände in kommerziell oder wissenschaftlich genutzten Datenbanksystemen nahmen — etwa seit Ende der 80er Jahre verstärkt — auch die Bemühungen zur einfachen Selektion, Inspektion und Exploration dieser großen Datensammlungen mit intuitiv nutzbaren Mitteln zu. Hierzu wurden spezielle visuelle Anfragesprachen, Darstellungs- und Navigationstechniken entwickelt und in Analysesystemen implementiert. Auf speziell aus diesem Bereich stammende Software wird nun anhand einiger Beispiele eingegangen.

VisDB Das an der Universität München von Keim entwickelte System *VisDB* [KKS94] zielt auf die Visualisierung und Exploration großer relationaler Datenbestände über mehrere Attributdimensionen ab. Die Grundidee der — neben gängigen Visualisierungstechniken für mehrdimensionale Daten, wie Parallelen Koordinaten, Pixeldiagrammen und Strichmännchen [KK95] — speziell in *VisDB* implementierten Visualisierungsverfahren besteht in der gleichzeitigen Darstellung möglichst vieler Datentupel auf dem Bildschirm durch jeweils ein Pixel (siehe Abb. 3.4).

[2] Die Abbildung ist verfügbar unter http://www.dbs.informatik.uni-muenchen.de/dbs/projekt/visdb/VisDB.gif.

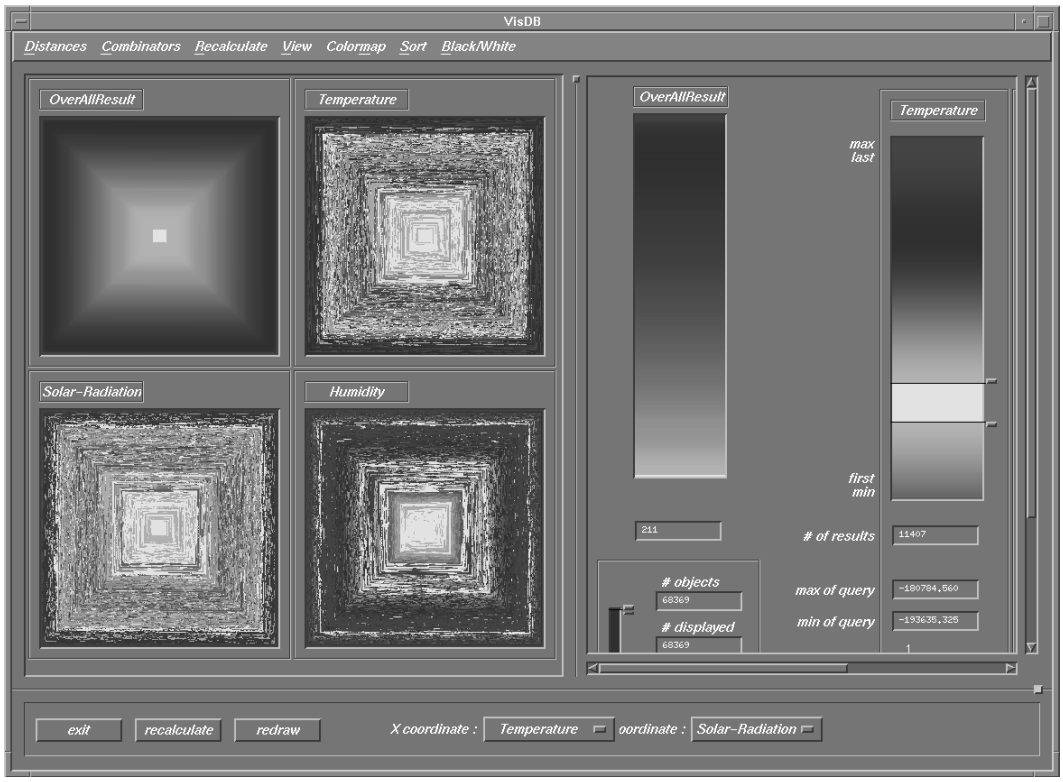

Abbildung 3.4: Pixelbasierte Datenbankvisualisierung mit *VisDB*[2]

Visualisiert wird jeweils die Tupelmenge aus dem Ergebnis einer relationalen Datenbankanfrage, in der die einzelnen Teilbedingungen gemäß ihrer Bedeutung mit Gewichtungsfaktoren versehen sind. In einer Darstellung für jeweils ein von der Anfrage betroffenes Datenbankattribut ergibt sich die Farbe eines Datentupels aus dem „Abstand“ seiner Attributausprägung zum angefragten Wert; die Position des entsprechenden Pixels wird nach einer für die jeweilige Visualisierungstechnik spezifisch definierten Formel aus dem über alle Attribute des Tupels gewichteten Gesamtabstand zur Anfrage ermittelt. Die in Abb. 3.4 dargestellte Spiraltechnik ordnet etwa alle Tupel spiralförmig von innen nach außen gemäß ihres Gesamtabstands an. Man erkennt entsprechend den gleichmäßigen Farbverlauf in der Ausgabe der Gesamtanfrage und die heterogene, Cluster bildende Färbung in den attributspezifischen Anzeigen (im Beispiel für Temperatur, Sonnenstrahlung und Feuchtigkeit).

Über eine einfache deklarative visuelle Anfragesprache können Anfragen spezifiziert und mit Hilfe von Schiebereglern auf der Basis attributspezifischer Werteverteilungen ähnlich den Dynamic Queries[3] manipuliert werden, worauf in den erzeugten Darstellungen direkt ein Feedback bzgl. der Relevanz der in der Datenbank enthaltenen Datenobjekte erfolgt. Somit können Anfrageresultate besser verstanden und Strukturen in den Daten erkannt werden. Das Interaktionsmodell von *VisDB* entspricht der linken Variante in Abb. 3.2.

Exbase *Exbase* (*Exploration of Databases*) ist ein Forschungsprojekt an der Universität von Massachusetts–Lowell, dessen zentrales Ziel in der Integration relationaler Datenbanksysteme mit Analyse und Datenvisualisierung besteht [Lee94]. Modifikationen der Ausgangsanfrage erfolgen über deren Visualisierung, die also sowohl zur Ausgabe als auch zur Eingabe dient. Typischerweise werden hier Dynamic Queries, vor allem auf Scatterplots, genutzt. Intern definiert *Exbase* Abbildungen zwischen Datenbasen, Datenbanksichten (Anfrageergebnissen), abgeleiteten Datenbanksichten (Analyseergebnissen) und Visualisierungssichten (Abbildungen

[3] Die Änderung der Darstellung erfolgt aufgrund der Datenbankgröße nicht in Echtzeit, sondern erst nach der abgeschlossenen Parametermanipulation.

von Datenattributen auf graphische Primitive sowie Visualisierungsparameter). Hierüber kann wiederum eine formale Zuordnung zwischen Benutzerinteraktionen auf Visualisierungen einerseits und auszuführenden Operationen auf den genannten Sichten, auf der Datenbankebene oder auch auf gelinkten Visualisierungen andererseits erfolgen [LG96].[4]

Als Interaktionsmodell kann wiederum der rechte Teil von Abb. 3.2 dienen, wobei die Datenwahl jedoch auf den ersten Schritt einer Untersuchung beschränkt ist. Statistische Operationen sind zwar vorgesehen, werden aber kaum näher diskutiert.

DEVise An der Universität von Wisconsin–Madison wurde das System *DEVise* (*Data Exploration and Visualisation*) zum Erstellen und interaktiven Explorieren visueller Repräsentationen verteilter, großer relationaler Datenbestände entwickelt [LRB+97, DEV97]. Mit Hilfe verschiedener Masken kann der Anwender zu lesende Datenquellen beschreiben und die Abbildung der Datenattribute auf graphische Primitive und Attribute spezifizieren. Auf erstellte Graphiken können visuelle Filter zur Selektion einer Teilmenge angewendet werden. Ferner sind Zooming, Scrolling sowie das Abfragen von Detailinformationen zu Datenwerten möglich. Basisoperationen und graphische Komponenten beschränken sich auf einfache, intuitiv nutzbare Elemente. Eine besondere Mächtigkeit erreicht *DEVise* durch verschiedene Varianten, Graphiken miteinander zu verknüpfen. Linking zwischen Filtern und Graphiken (Views) kann als gelinktes Highlighting, als Selektion von Objekten oder als Einschränkung von Attributwertebereichen sowie als algebraisches Linking erfolgen. Alle Operationen werden in SQL–Befehle übersetzt und an die zugrundeliegenden Datenbanksysteme gesandt, wobei eine Anfrageoptimierung (im Hinblick auf Caching, Indexierung, Prefetching und Vorberechnungen) auf Basis der bestehenden Verbindungen zwischen Graphiken und den jeweils repräsentierten Attributen erfolgen kann. Das Interaktionsmodell von *DEVise* ähnelt dem von *Manet*.

Visage *Visage* [RCK+97], entwickelt an der Carnegie–Mellon Universität, bietet eine umfassende Arbeitsumgebung zur Visualisierung und interaktiven Analyse von objektorientierten Datenbeständen, die — in Anlehnung an das multidimensionale Modell im OLAP–Kontext — über hierarchisch klassifizierte Attribute verfügen. Das Gesamtsystem umfaßt eine schemabasierte visuelle Anfragesprache, die Datenflußaspekte integriert, umfangreiche Möglichkeiten zur interaktiven Manipulation graphischer und tabellarischer Darstellungen sowie eine Komponente zur wissensbasierten Graphikgenerierung. Eine Grundphilosophie von *Visage* besteht darin, dem Nutzer alternative Ausdrucksmöglichkeiten über unterschiedliche Schnittstellen bzw. Oberflächenkomponenten anzubieten. Diese variieren insbesondere

- in Bezug auf Intension oder Extension (Prädikate oder Aufzählungen),
- im Abstraktionsgrad der Operatoren,
- in der Stetigkeit der Interaktion (diskrete Befehle oder kontinuierliche Parameteränderungen) sowie
- im Grad der Nutzung des Domänen–Vokabulars.

Skripte definieren die Abbildung von Daten auf Graphikelemente. Per Drag–and–Drop können Daten aus tabellarischen oder graphischen Darstellungen selektiert und in andere Graphiken eingefügt werden (vgl. Abb. 3.5, wo Standorte selektierter Einheiten der US Army in einer Karte visualisiert werden sollen). Ebenso ist über Kontextmenüs die Navigation durch Aggregierungsebenen möglich. Graphiken können gelinkt werden, Parameteränderungen werden in Form von Dynamic Queries direkt weitergeleitet. Wie schon bei *DataDesk* können Folgen von Visualisierungen auch hier als Folienshows gespeichert werden. Insgesamt sind — verglichen mit anderen Systemen — recht aufwendige Darstellungen, etwa auch komplexe thematische Karten, realisierbar.

Das Teilmodul *SDM* (*Selective Dynamic Manipulation*) definiert für Graphikelemente verschiedene Möglichkeiten zur direkten Manipulation des Graphiklayouts, zur Veränderung des Fokus einer Graphik, zur Hervorhebung, Selektion und zum Vergleich von Elementen. Das *Visual Query Environment* (*VQE*) erlaubt

[4] Leider sind hierüber keine genaueren Informationen verfügbar.

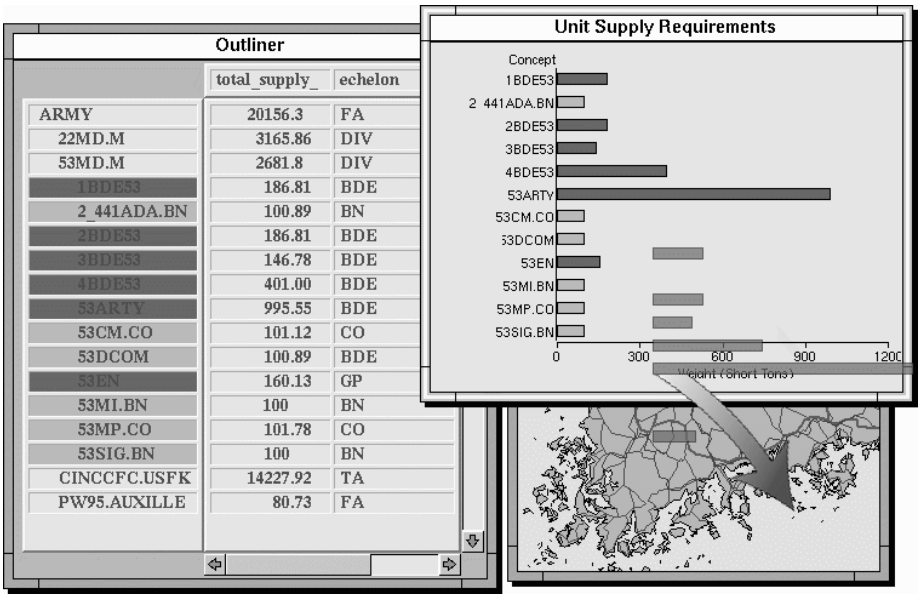

Abbildung 3.5: Visualisierung von Daten über Einheiten der US Army in *Visage* (aus [RCK+97])

über eine maskenbasierte Darstellung des Datenschemas und der Beziehungen zwischen Schemaobjekten — im Zusammenwirken mit der direkten Graphikmanipulation — eine Kombination intensionaler und extensionaler Anfragen. Über *VQE* können Verknüpfungen, Attributberechnungen, Aggregierungen und prädikatbasierte Selektionen spezifiziert werden. Die entstehenden Objektmengen können zur Darstellung in Graphiken gezogen werden, ebenso wie Darstellungen und Selektionen in Graphiken als Eingabe für *VQE* dienen können [DKR97]. Die unterliegende Modellierung aller Verknüpfungsbeziehungen in Form sogenannter „Threads“ steuert die Ausführung von Anfragen und das Linking zwischen Darstellungen.

Auf die wissensbasierte Graphikgenerierung in *Visage* wird in Abschnitt 3.5 noch kurz eingegangen werden. *Visage* basiert wie *DataDesk* auf dem rechten Interaktionsmodell in Abb. 3.2. Zusätzlich kann über die visuelle Anfragedarstellung in *VQE* auch zu vorherigen Management- und Analyseschritten navigiert werden. Da dies jedoch für Visualisierungssysteme eher untypisch ist, wurde es in Abb. 3.2 nicht eingezeichnet.

3.3 Unterstützung von OLAP und Data Warehousing

In der Regel bilden OLAP–Tools (vgl. Abschnitte 2.1.1 und 2.3) in einer Client–Server–Architektur Frontends für unterliegende OLAP–Server, die eine multidimensionale Sicht auf den Datenbestand eines Data Warehouses bereitstellen.[5] Hierbei ergibt sich eine breite Palette von Systemen, angefangen beim einfachen Werkzeug zur Berichterstellung über Systeme zur explorativen Ad–hoc–Analyse bis hin zu komplexen Umgebungen zur Entscheidungsunterstützung, die Entwicklungsumgebungen für den Aufbau unternehmensweiter Informationssysteme anbieten [Cla98].

Die Klasse explorativer Analysewerkzeuge soll hier genauer betrachtet werden. In derartigen Systemen werden auf Basis der Datenwürfel–Metapher auf den zu analysierenden Daten Selektionen, Klassifikationen und Aggregierungen mittels verschiedener, i. a. recht einfacher Aggregierungsfunktionen (Anzahl, Summe, Durchschnitt etc.) vorgenommen. Ergebnisse werden tabellarisch oder in einfachen Diagrammen dargestellt. Über diese erfolgt eine weitere flexible Navigation im Datenbestand mittels Slice–and–Dice, Roll–up und Drill–down. Abbildung 3.6 zeigt exemplarisch eine Graphik und eine Tabelle, die mit dem *ORACLE*

[5] Sogenannte Desktop–OLAP– (DOLAP–) Tools, die alle interessierenden Daten einer betrachteten Datenbank direkt auf den Client laden, stellen eine Ausnahme dar.

Express Analyzer erstellt wurden und die über die Menüs in den Kopfzeilen der Fenster sowie über weitere der Graphik zugeordnete Pop–up–Menüs interaktiv parametrisiert werden können (entnommen aus [Ora97]). Weitere aggregierende Maßzahlen können ermittelt, Zeitreihen betrachtet und auffällige (stark abweichende, minimale, maximale) Werte identifiziert werden. Oftmals können noch weiterführend Zeitreihen-, Trend- und Regressionsanalysen durchgeführt oder komplexere statistische Maßzahlen und Modelle berechnet werden. Standardisierte Analysen sind häufig auf Knopfdruck abzurufen.

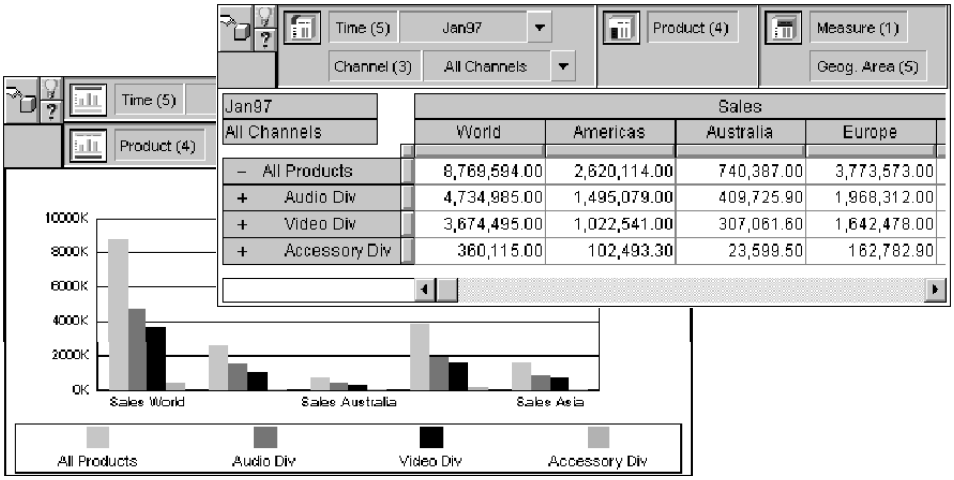

Abbildung 3.6: Graphiken und Tabellen im *ORACLE Express Analyzer*

Das diesen OLAP–Tools zugrundeliegende Interaktionsmodell (siehe Abb. 3.7) zeichnet sich dadurch aus, daß nach der erstmaligen Datenwahl alle weiteren Operationen über die Visualisierungsergebnisse erfolgen. Es kann zwischen verschiedenen Darstellungen navigiert werden, ein Linking von Graphiken wird jedoch i. a. nicht unterstützt.

In [WKC96] werden eine Reihe von OLAP–Systemen ausführlich beschrieben und anhand verschiedener Kriterien verglichen. [DSHB98] widmet sich vor allem der Betrachtung eher „technischer“ Aspekte (Architekturen, Mehrbenutzerfähigkeit, Schema–Definition und –Administration, Anfragesprachen und -optimierung, Skalierbarkeit, externe Schnittstellen, kooperative Nutzung etc.), die an dieser Stelle nicht näher interessieren. Im WWW finden sich Übersichten von OLAP–Tools etwa in [Rad97, PC99].

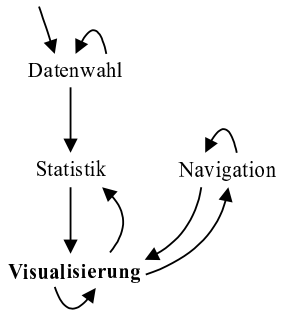

Abbildung 3.7: Das Interaktionsmodell von OLAP–Systemen

Zunehmend werden auch Verfahren des Data Mining in OLAP–Werkzeuge integriert (vgl. [WKC96]). Umgekehrt finden OLAP–Konzepte auch in Systeme zum Data Mining Einzug. So bildet z. B. eine „Drill–Engine“ zur Aggregierung und Disaggregierung von Warehouse–Daten das zentrale Basismodul des *Data Surveyor* der *Data Distilleries (DDI)* [KSHK97]. Ähnlich werden im Rahmen des an der Universität von Minnesota entwickelten *DBMiner* Varianten der Einbeziehung von OLAP–Operatoren in konkrete Mining–Algorithmen sowie die Ausnutzung der Datenwürfelstrukturen zu deren Effizienzsteigerung diskutiert [Han97a].

Einen Überblick über Mining–Software gibt etwa [WK97], Referenzen ins WWW finden sich in [FPSSU96], und im WWW selbst bietet [KDN99] einen umfassenden Überblick.

3.4 Datenflußbasierte visuelle Programmierung in der Datenanalyse

Eines der ersten Analysesysteme, das sich der datenflußbasierten visuellen Programmierung bediente, war Ende der 80er Jahre das System *ConMan* (*Connection Manager*) von Haeberli (*SiliconGraphics*) [Hae88]. Bereits hier wurden Konzepte objektorientierter Programmierung zur Realisierung der Programmbausteine und ihrer Kommunikation genutzt. Ausgehend von den mit *ConMan* vorgestellten Grundideen fand und findet die Datenflußprogrammierung in vielen Bereichen der Datenanalyse Eingang. Im folgenden werden einige bekannte und/oder für spezielle Aspekte repräsentative Analysewerkzeuge näher vorgestellt, bei denen die datenflußbasierte Programmierung von Analysen im Vordergrund steht. Hierbei beschränkt sich dieser Abschnitt auf grundlegende Charakteristika. Ein umfassender Vergleich von Systemeigenschaften findet sich im Kontext der Einordnung von *VIOLA* in Abschnitt 5.1. Die hier betrachteten Systeme lassen sich in die Klassen

- kommerzielle Systeme zur wissenschaftlichen Visualisierung und Bildverarbeitung (*AVS*, *IBM Data Explorer*, *IRIS Data Explorer* und *KHOROS*/*Cantata*; Abschnitt 3.4.1),
- „akademische“ Lösungen zur Datenbankvisualisierung (*Tioga–2*, *DFQL* und *IDEA*; Abschnitt 3.4.2),
- kommerzielle Systeme zur Meßdatenverarbeitung und Simulation (*LabVIEW* und SIMULINK; Abschnitt 3.4.3) und
- Data Mining und allgemeine statistische Analyse (*Clementine* und *ViSta*; Abschnitt 3.4.4)

einteilen. Interessant ist zu sehen, wie verschiedene Anwendungsgebiete in unterschiedlichem Maße kommerzielle und „akademische“ Analysesysteme hervorgebracht haben. Auch fällt auf, daß die überwiegende Zahl der Umgebungen (mit Ausnahme der letzten Gruppe) einen starken Bezug zur Datenvisualisierung aufweist.

Die Interaktionsmodelle datenflußbasierter Analyseumgebungen (siehe Abb. 3.8) zeichnen sich dadurch aus, daß nur hier tatsächlich zu beliebigen anderen Operationen navigiert werden kann und deren anschließende Modifikation möglich ist. Die interaktive Nutzung von Visualisierungen geht nur in wenigen Fällen über eine direkte Manipulation zur Parametrisierung von Graphiken (etwa in Form von Dynamic Queries) hinaus. Dann können Teile einer Graphik selektiert und ebenso wie aktuelle Blickwinkel auf die Darstellung als Eingabe anderer Operationen verwendet werden. Letzteres entspricht im wesentlichen dem Konzept des Linking statistischer Graphiken, wie es in Abschnitt 2.1.3 vorgestellt wurde.

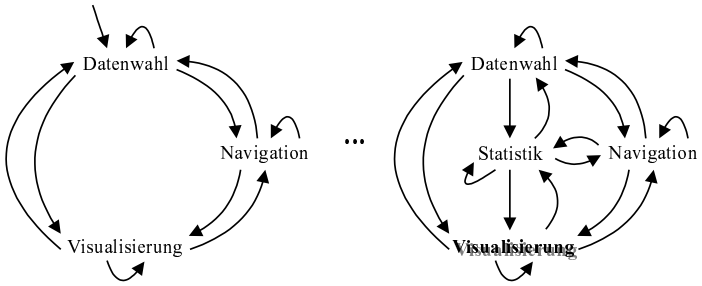

Abbildung 3.8: Interaktionsmodelle datenflußbasierter Analyseumgebungen

Zu beachten ist, daß das Interaktionsmodell die Reihenfolge der „Konstruktionsschritte“ einer Analysesitzung, nicht deren Ausführung selbst spezifiziert. Somit ist es in der Regel nicht erforderlich, mit der Datenwahl zu beginnen. Ausnahmen hiervon bilden die Analysesysteme *Tioga–2*, *IDEA* und *ViSta*, wo der Benutzer zur initialen Wahl der Datenquelle gezwungen wird. Weiterhin bieten *Tioga–2*, *DFQL* und *IDEA* kaum statistische Verfahren. In *IDEA* ist keine direkte Interaktion mit Visualisierungen möglich, und bei *ViSta* und *IDEA* beschränkt sich die Navigation auf die Wahl eines neuen Analysefokus, erlaubt jedoch keine Modifikation vorgenommener Analyseschritte. Entsprechend zeigt Abb. 3.8 links ein minimales Interaktionsmodell (verkörpert durch *IDEA*) und rechts die maximale Variante (umgesetzt in *IBM Data Explorer* und *AVS*), die nahezu beliebige Freiräume läßt.

3.4.1 Kommerzielle Systeme zur wissenschaftlichen Visualisierung/Bildverarbeitung

Systeme zur wissenschaftlichen Visualisierung dienen der Erstellung komplexer, oft dreidimensionaler (rotierbarer) Graphiken und räumlicher Körper aus Ergebnissen von Messungen, Erhebungen und Experimenten. Datenflußprogramme realisieren durch ihre Bausteine die klassische Visualisierungspipeline von Datenzugriff (oft auch im Zusammenhang mit Simulationen) über Filterung, Abbildung auf graphische Primitive und die eigentliche Visualisierung (das *Rendering*). In ähnlicher Weise verarbeiten Bildverarbeitungssysteme zweidimensionale Bilder und eindimensionale Signale durch verschiedene Filter und Transformationen und stellen die Ergebnisse dar. Über intuitiv bedienbare visuelle Programmieroberflächen sollen jeweils vor allem Wissenschaftler und Ingenieure als Systemanwender angesprochen werden. [WRH92] gibt einen kurzen, aber differenzierten Überblick über datenflußbasierte Umgebungen auf diesen beiden Gebieten.

AVS Das *Application Visualization System* (*AVS*) von *Advanced Visual Systems* bietet eine datenflußbasierte Programmierumgebung zur schnellen und einfachen Entwicklung eigenständig (außerhalb der Entwicklungsumgebung) lauffähiger Applikationen zur interaktiven Datenvisualisierung [UFK+89]. *AVS* basiert auf dem objektorientierten Paradigma: Alle Module der visuellen Programmiersprache sind Objekte, die sich ihrerseits wiederum aus verschiedenen anderen Attributobjekten zusammensetzen (bis hinunter zur Ebene von Basisdatentypen wie Integer oder String). Dieser mehrstufige hierarchische Aufbau findet sich auch bereits bei den Elementen der vordefinierten umfangreichen Funktions- bzw. Modulbibliothek.

Über die Datenflußumgebung wird die Zuweisung von Werten eines Objektattributs an ein Attribut eines anderen Objekts modelliert. Die eigentlichen Berechnungen erfolgen über Objekten zugeordnete Methoden, die in der *AVS*–internen (verbalen) Programmiersprache *V* oder durch externe Programme realisiert sein können [LP98] und bei Objekterzeugung und/oder Änderungen von Attributwerten angestoßen werden. Auf dieser Ebene wird der Datenfluß also jeweils nicht mehr visuell modelliert. Kontrollstrukturen sind nur teilweise in Form visueller Bausteine verfügbar, oftmals können bzw. müssen sie statt dessen innerhalb der Methoden und/oder mit Hilfe von Modulparametern (etwa zur (De–)Aktivierung von Modulen) realisiert werden.

Abbildung 3.9 zeigt ein einfaches, aus drei Bausteinen bestehendes *AVS*–Programm zum Einlesen eines dreidimensionalen Meßwertefeldes, der Berechnung einer Oberfläche mit konstantem Meßwert (*Isosurface*) und deren Visualisierung. Die Zusammensetzung des Isosurface–Bausteins auf erster Ebene wurde „aufgeklappt“.

Die Ausführung von Programmkomponenten kann transparent auf verteilten Plattformen erfolgen und externe Programme ansprechen. Die Erstellung von Bedienoberflächen zur Programmausführung, die aus Eingabemöglichkeiten für die Parameter der betroffenen Bausteine und den Visualisierungsergebnissen selbst bestehen, erfolgt in einem von der Datenflußoberfläche separierten Fenster. Entsprechend können Anwendungen generiert werden, in denen das unterliegende Datenflußprogramm unsichtbar ist. Erstellte Visualisierungen sind interaktiv manipulierbar (etwa im Hinblick auf Blickwinkel, Skalierung oder „Beleuchtung“ eines dreidimensionalen Objekts), insbesondere können sie auch rotiert werden.

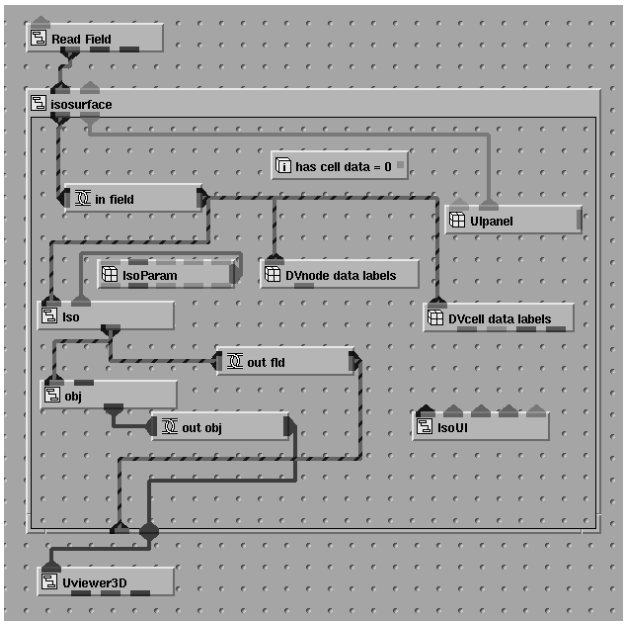

Abbildung 3.9: Ein einfaches Visualisierungsprogramm in *AVS*

Mit der Netzbeschreibungssprache *V* können Datenflußprogramme, also Objektmengen, ihre Attribute, Zustände und Verbindungen zu anderen Objekten auch außerhalb der visuellen Programmierumgebung spezifiziert bzw. über eine Kommandozeile modifiziert werden [AVS96]. Die Definition neuer Objekte, die vor allem als Programmbausteine dienen, aber auch als Attributwerte zu nutzen sind, wird über ein graphisches Tool erleichtert. Dieses unterstützt neben einfacher Spezifikation von Ein- und Ausgabeparametern sowohl Komposition von Teilobjekten (Definition von Attributen) als auch die Vererbung von Objekt–Eigenschaften.

Der *IBM Data Explorer* Der *Data Explorer* von *IBM* legt neben einer umfangreichen Funktions- und Visualisierungsbibliothek Schwerpunkte auf eine transparent verteilte und (auch innerhalb eines Bausteins) parallele Ausführung von Routinen sowie ein „intelligentes“ Ausführungsmodell [LAC+92, AT95]. Dieses unterstützt das Caching sowie die Wiederverwendung beliebiger Zwischenergebnisse und optimiert die Verarbeitung, indem auf die Evaluation von Teilen eines Analysegraphen, die zur Datenausgabe nicht benötigt werden (hinter bedingten Verzweigungen oder als Teilnetz mit Knoten ohne nachfolgende Visualisierungen), verzichtet wird.

Datenflußprogramme propagieren komplexe Daten- und Visualisierungsobjekte; in einem Netzknoten können jeweils ausgewählte Objekt–Attribute modifiziert oder neue Attribute berechnet werden. Einige einfache Kontrollstrukturen sind modellierbar [IBM97]. Ein separates Tool dient zum Import von Daten aus verschiedenen Datenquellen und -formaten. Ähnlich wie bei *AVS* erfolgen Ausführung und Parametrisierung eines Programms sowie die Ergebnisdarstellung über (hier mehrere) separate Panels und Fenster. In verbalen Programmiersprachen entwickelte Routinen können als neue Module dynamisch hinzugebunden werden.

Die Ausführung eines Datenflußprogramms erfolgt in einer Client–Server–Architektur: Der Client erzeugt aus dem Graphen ein Skript, das an den Server gesendet und dort verarbeitet wird. Die entsprechende Skriptsprache kann ebenso wie eine Programmierschnittstelle (API) auch vom Endanwender anstelle der visuellen Programmierumgebung zur Modellierung und Durchführung von Analysen genutzt werden.

Der *IRIS Data Explorer* Recht ähnlich zum *IBM Data Explorer* ist der *IRIS Data Explorer* [Wal96, Wal98], der ursprünglich von *SiliconGraphics* entwickelt wurde und seit 1992 von der *Numerical Algorithms Group* (*NAG*) weiterentwickelt und vertrieben wird [NAG96]. Gegenüber den bisher vorgestellten Umgebungen werden hier keine wesentlichen neuen Konzepte eingeführt. Im Projekt *STABLE* wurde der *IRIS Data Explorer* um eine Bibliothek statistischer Analysefunktionen erweitert [FMTW98].

KHOROS*/*Cantata Das ursprünglich an der Universität von New Mexico entwickelte und seit 1992 von der Firma *Khoral Research* kommerziell vertriebene System *KHOROS* bietet eine komplexe Programmierumgebung zur Software–Entwicklung und –Integration [RW91, YAK95]. *KHOROS* zielt insbesondere auf Bild- und Matrix- bzw. Signalverarbeitung sowie die wissenschaftliche Datenvisualisierung, ist aber durch entsprechende Erweiterungen der Funktionsbibliotheken flexibel auf anderen Anwendungsgebieten einsetzbar. Verschiedene Komponenten von *KHOROS* dienen der Spezifikation von Benutzungsschnittstellen von Anwendungssystemen sowie der Code–Generierung zur Anbindung externer Programme und deren Parametrisierung über Eingabemasken. In diesem Rahmen bildet *Cantata* als datenflußbasierte visuelle Programmierumgebung den Kern von *KHOROS* zur Verknüpfung all dieser Bausteine einer Anwendung. So werden einfache Datenauswertung, schnelles Prototyping und auch die von der Datenflußsicht losgelöste Ausführung von Anwendungen verbunden. Wiederum fördert eine objektorientierte Realisierung die einfache Systemerweiterung [YAW95].

Verglichen mit den bisher in diesem Abschnitt vorgestellten Systemen bietet *Cantata* deutlich mehr visuelle Kontrollstrukturen (vgl. Abb. 3.10: dort wird ein Bild schrittweise um jeweils 45 Grad rotiert); auch können lokale und globale Variablen genutzt werden. Letztere haben im Rahmen der Bildverarbeitung meist lediglich atomare, zur Bausteinparametrisierung genutzte Datenwerte; prinzipiell ist jedoch auch z. B. eine Verwendung globaler Variablen zur Bereitstellung ganzer Bilder möglich. Weiterhin werden eine Reihe unterschiedlicher Ausführungsmodi (schrittweise, daten- und anforderungsgetrieben) unterstützt. Ähnlich wie bei *AVS* sind einfache textuelle Programmbausteine (Formeln, Bedingungsprüfungen etc.) leicht über Masken einzubinden. Ein Linking unterschiedlicher Darstellungen über interaktiv manipulierbare Parameter ist in *Cantata* nicht möglich.

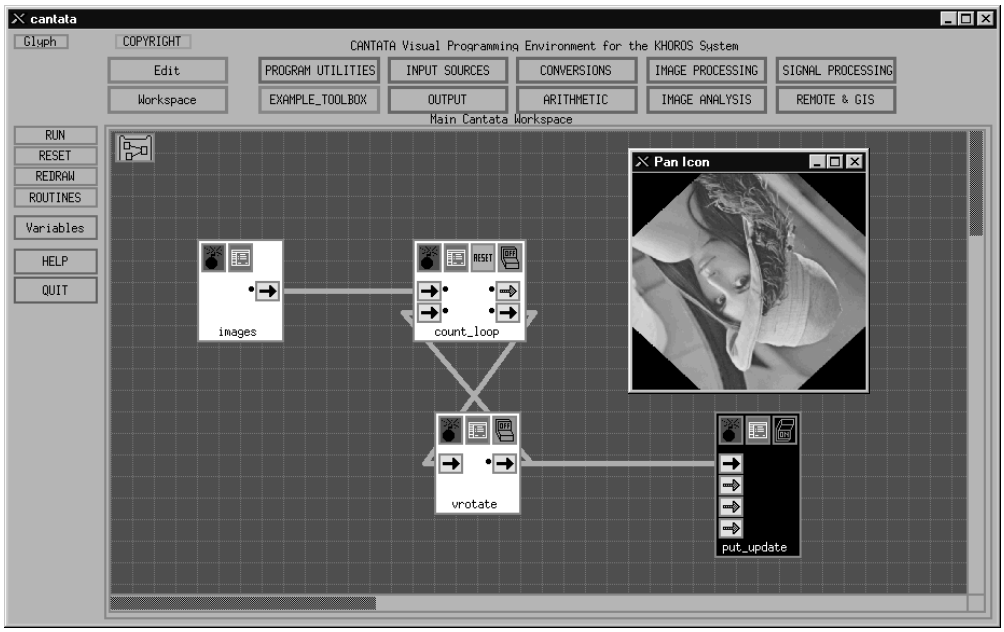

Abbildung 3.10: Eine einfache Schleife in *Cantata*

3.4.2 „Akademische“ Lösungen zur Datenbankvisualisierung

Mit ähnlicher Zielsetzung wie die in Abschnitt 3.2.2 vorgestellten Systeme versuchen auch die im folgenden aufgeführten Umgebungen dem Anwender einen schnellen und flexiblen Einblick in und Überblick über große Datenbestände zu gewähren.

Tioga–2 Im Rahmen des *SEQUOIA–2000*–Projekts [Sto94] wurde an der Universität von Kalifornien in Berkeley das datenbankzentrierte Visualisierungssystem *Tioga–2* entwickelt [WS95, ACSW96].[6] *Tioga–2* basiert auf dem erweiterbaren, objektrelationalen DBMS *POSTGRES*: Bausteine der datenflußbasierten Anfragesprache bilden die in *POSTGRES* registrierten, auf die betrachteten Daten anwendbaren (benutzerdefinierten) Funktionen. Somit ist eine einfache Erweiterbarkeit garantiert.

Neben skalaren Parametern, die über Datenflußelemente spezifizierbar sind, aber auch bei Bedarf bausteinbezogen zur Laufzeit abgefragt werden können, sind alle in *Tioga–2*–Datenflußprogrammen verarbeiteten Daten „erweiterte“ Relationen, die evtl. auch zu Gruppen zusammengefaßt sein können. Jede Relation hat stets eine visuelle Repräsentation in einer tupelbezogenen Darstellung. Hierzu enthält sie zusätzliche Attribute, die die Abbildung der Datenattribute auf benutzerdefinierte und insbesondere auch graphische Attribute (darunter immer x– und y–Koordinate sowie mindestens ein graphisches Symbol) spezifizieren. Jedem Symbol–Attribut ist ferner ein „Höhenbereich“ (*Elevation*) zugeordnet. Die Definition all dieser Eigenschaften erfolgt über eine Reihe primitiver Operationen im Datenflußprogramm, das darüber hinaus noch gängige Datenmanagement-, Transformations- und natürlich Visualisierungsbausteine enthält.

In *Tioga–2* erzeugte graphische Darstellungen sind in Form von Dynamic Queries stets mit Schiebereglern für alle nicht auf x– und y–Koordinaten abgebildeten Attribute versehen. Zudem kann mittels Zooming die „Höhe“ der Darstellung modifiziert und somit die Menge der darzustellenden Symbole ausgewählt werden (vgl. Abb. 3.11, wo die Landkarte als Umriß und die Städte repräsentierende Punkte auf jeder Höhe, die Beschriftungen jedoch erst bei näherem Zoomen ausgegeben werden). In Form sogenannter „Wurmlöcher“ können Datentupeln graphische Attribute zugeordnet werden, die selbst jeweils weitere „Detailgraphiken“ enthalten. In diese kann dann ggf. auch vollständig hineingezoomt werden, wodurch ein Kontextwechsel zur

[6] *Tioga*, der Vorläufer von *Tioga–2*, wird in [SCN+93] beschrieben.

Darstellung anderer Analyseergebnisse (aus einem anderen Bereich des Datenflußprogramms) erfolgen kann. Siehe hierzu auch Abb. 3.11: Dort wurde im rechten Programm gegenüber dem linken ein Wurmlochattribut zur Visualisierung der Stabdiagramme hinzugefügt. Weitere Gruppierungen und Kombinationen von Graphiken sind möglich, auf die hier aber nicht näher eingegangen werden soll.

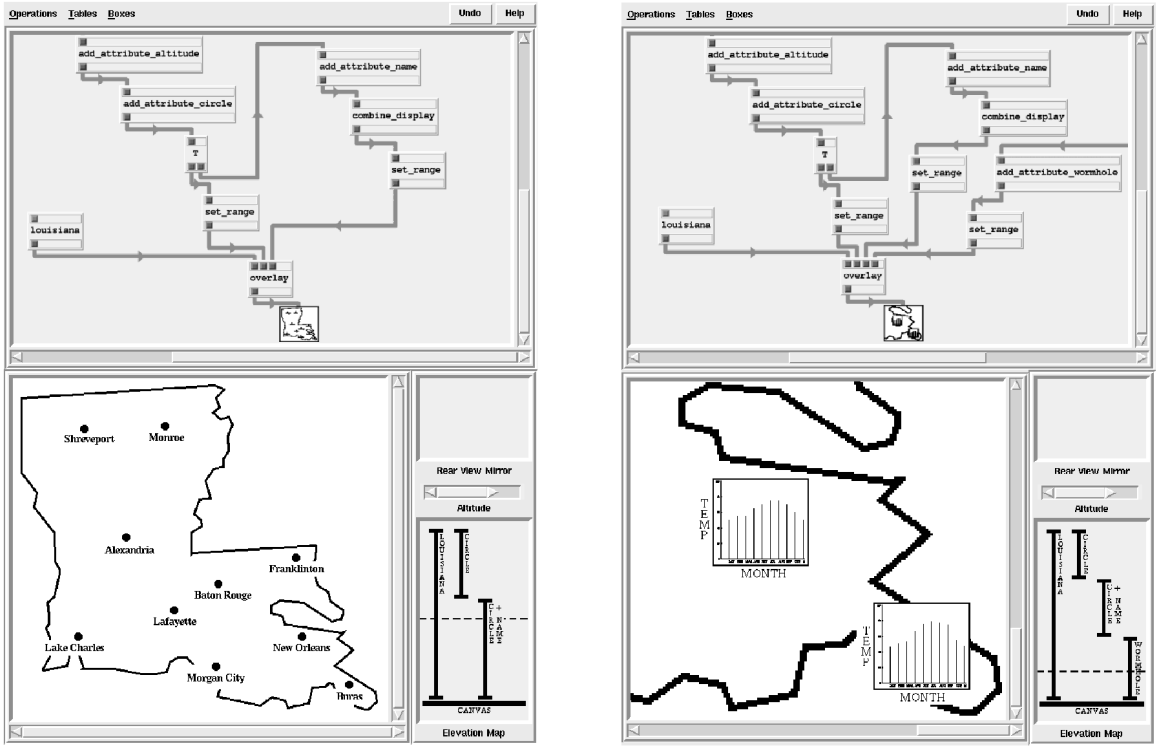

Abbildung 3.11: Wurmlöcher in *Tioga–2* (aus [ACSW96])

Erwähnenswert sind ferner die Komposition von Teilprogrammen „mit Löchern“ (quasi in Form von Funktionen höherer Ordnung), die interaktive Abfrage unbelegter Knoteneingänge zur Laufzeit, die Spezifikation von Bausteinparametern in Form von Ausdrücken einer verbalen Anfragesprache sowie die Möglichkeit, über benutzerdefinierte Masken aus dem Datenflußprogramm Updates auf betrachteten Relationen vorzunehmen.

Jeder Knoten eines Programms in *Tioga–2* stellt eine Anfrage an *POSTGRES* und die Definition einer (virtuellen) neuen Relation dar. Die Verarbeitung von Datenflußprogrammen als komplexe Anfragen wird optimiert, indem die Ausführungspläne der einzelnen Bausteine — falls dies Effizienzgewinn verspricht — verschmolzen werden sowie ein Caching ausgewählter Zwischenergebnisse erfolgt. Strategien hierzu basieren auf aus Netzstruktur und Position im Netz geschätzten Wahrscheinlichkeiten für den späteren direkten Zugriff auf einen Baustein sowie für eine Änderung seiner Parameter. Ferner fließen Größe des Datensatzes und Berechnungsaufwand ein [WS95, SCN+93].

Bereits in [ACSW96] wurde erwähnt, daß neben der direkten datenflußbasierten Programmierung auch mittels graphischer Manipulation von Visualisierungsergebnissen Anfragen spezifiziert werden können. Dies ist im Nachfolgeprodukt *DataSplash* vollständig umgesetzt, so daß der Anwender — ohne Datenflußprogrammierung — nur noch über einen Graphikeditor seine gewünschten Visualisierungen zusammenstellt [Tio97].

DFQL In [DRR+96] wird mit der Sprache *DFQL* (*Data Flow Query Language*) eine visuelle Modellierung erweiterter SQL–Anfragen vorgeschlagen. Das Ziel dieses Projekts an der Universität von Minnesota ist allgemein die Unterstützung der Auswertung wissenschaftlicher Datenbestände, die eine enge Kopplung von Datenbankzugriff, Analyse und Visualisierung verlangt. Von *DFQL* verarbeitete Daten sind Relationen; die Ope-

ratoren (und damit die Bausteine im Datenflußprogramm) entsprechen denen von SQL — zusätzlich können externe Programme als Spracherweiterung über spezielle Netzknoten angebunden werden. Über tabellarische Darstellungen hinausgehende Verfahren zur Datenvisualisierung bietet eine angebundene Funktionsbibliothek.

Ein Datenflußprogramm (mit mehreren Senken) entspricht jeweils einer Menge von SQL–Anfragen. Bei der Abarbeitung des Programms werden — soweit möglich — Operationen zu äquivalenten SQL–Ausdrücken zusammengefaßt und vom unterliegenden DBMS verarbeitet. Die Ausführung externer Operationen erfolgt über temporäre Hilfsrelationen, in denen Zwischenergebnisse abgelegt werden. Diese bleiben auch nach der Anfrageausführung bestehen und können nach Anfragemodifikationen ggf. wiederverwendet werden.

IDEA Ein ähnlicher Ansatz wird mit dem System *IDEA* (*Interactive Data Exploration and Analysis*) von Selfridge und anderen bei *AT&T* verfolgt [SSW96]. Der visuellen Programmierung wird hier jedoch ein etwas geringerer Stellenwert beigemessen: Sie dient eher zur Protokollierung von Datenanalyseschritten, die über eine maskenbasierte Benutzungsoberfläche spezifiziert werden. Entsprechend ist im Interaktionsmodell der datenflußbasierten visuellen Programmierung aus Abb. 3.8 hier die Beschränkung auf Analysen, die mit einer Datenwahlinteraktion beginnen, vorzunehmen. Auch können bereits ausgeführte Schritte später nicht modifiziert werden. Es besteht lediglich die Möglichkeit, mittels Navigation im Datenflußdiagramm den Ausgangspunkt weiterer Analyseschritte im bisherigen Untersuchungsprozeß zu wählen sowie Teilgraphen zu kopieren und vor ihrer nochmaligen Ausführung, evtl. auf anderen Basisdaten, abzuändern und neu zu parametrisieren.

Das Datenmanagement beschränkt sich — in Anlehnung an das multidimensionale Datenmodell bzw. OLAP — auf Anfrage, Segmentierung (Klassifikation) und Aggregierung relationaler Daten unter Anwendung verschiedener Aggregierungsfunktionen. Abgesehen von einfachen Diagrammen sind für weitere Analyseoperationen und Visualisierungen der Export von Daten in externe Tools sowie der anschließende Import der Berechnungsergebnisse vorgesehen.

Der gesamte Analyseprozeß, einschließlich der Beschreibung externer Aufrufe, wird in der Datenbank abgelegt. Aus [SSW96] wird nicht ganz deutlich, ob vor und/oder nach der Ausführung externer Prozeduren Datensätze (in der Datenbank) materialisiert werden. Auf jeden Fall gilt dies für zu visualisierende Analyseergebnisse. „Intelligente“ Mechanismen dienen der Anfragevereinfachung und -optimierung: Analysezweige mit gemeinsamen Zwischenschritten werden intern zusammengeführt. Ferner erfolgt ein Rückgriff auf bereits in der Datenbank abgelegte frühere Resultate, falls (Zwischen–)Ergebnisse in der aktuellen Analyse mit diesen identisch oder zumindest (leichter als aus den zugrundeliegenden Basisdaten) aus diesen ableitbar sind.

An dieser Stelle sei auch das *Exbase*–System [Lee94] noch einmal erwähnt, das die Repräsentation einer Analysesitzung in Form eines azyklischen gerichteten Graphen vor allem zur Protokollierung der einzelnen Interaktionen und ihrer Abfolge nutzt. Eine Datenbank mit so erhaltenen Interaktionsmustern soll zukünftig zur Gestaltung adäquater Analysesysteme herangezogen werden. In *Exbase* selbst interagiert der Benutzter mit dem System vor allem über die Datenvisualisierungen [LG96].

3.4.3 Kommerzielle Systeme zur Meßdatenverarbeitung und Simulation

In diesem Abschnitt steht die Verarbeitung kontinuierlich erhobener bzw. ermittelter Meßwerte auf einem relativ niedrigen Abstraktionsniveau der Programmierung im Vordergrund.

LabVIEW *LabVIEW* (*Laboratory Virtual Instruments Engineering Workbench*) ist ein von der *National Instruments Corporation* entwickeltes und kommerziell vertriebenes System zur Realisierung softwaretechnischer Lösungen im Bereich elektronischer Meß- und Steuerungssysteme [KMR91] (siehe auch [Sch98, Kapitel 6.3] sowie [GP96]). In Konzeption, Bedienung und Terminologie lehnt sich *LabVIEW* stark an die Gestalt realer Meßinstrumente an. Mit der eingängigen Metapher des „virtuellen Instruments“ zur Beschreibung eines Datenflußprogramms bzw. eines Teilmoduls wird vor allem der Ingenieur als Anwender angesprochen. Nicht zuletzt aus dieser Fokussierung begründet sich der Erfolg von *LabVIEW* in entsprechenden Anwendungsgebieten [BGL95, Kapitel 2], wenngleich das System auch anderweitig einsetzbar ist.

Virtuelle Instrumente bestehen aus zwei separat zu entwickelnden Teilen:

- der Bedienoberfläche (dies ist die Frontplatte des Instruments), die Möglichkeiten zur Parametereingabe und zur Visualisierung von Meß- und Berechnungsergebnissen enthält, sowie
- dem Blockschaltbild, das als Datenflußprogramm (oder Teil davon) die Programmlogik codiert und auf die Oberflächenkomponenten über Datenquellen und -senken zugreift.

Über dieses Konzept ist auf einfache Art und Weise auch die Integration realer Geräte möglich.

Der visuellen Programmierumgebung von *LabVIEW* liegt die visuelle Programmiersprache *G* zugrunde, die das Datenflußparadigma elegant mit Konzepten der strukturierten imperativen Programmierung kombiniert. Neben gängigen Kontrollstrukturen bietet die Sprache Sequenzen, polymorphe Operatoren auf einfachen und zusammengesetzten Datentypen sowie lokale und globale Variablen. Ablaufstrukturen sind stets durch ein Subnetz gekapselt, das abgesehen von seinen klar definierten Ein- und Ausgängen von Programmelementen außerhalb isoliert ist. Hierbei werden Schleifen durch Ports in Form von Schieberegistern unterstützt, die nach jedem Durchlauf berechnete Werte an den Schleifenanfang propagieren (vgl. Abb. 3.12). So gelingt die konsistente und nur geringfügige Erweiterung des reinen Datenflußmodells.

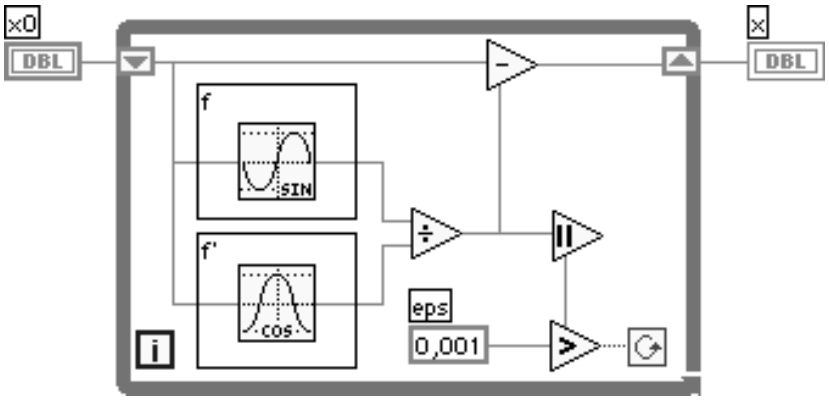

Abbildung 3.12: Schleife in *LabVIEW* zur Berechnung von Nullstellen nach dem Newton'schen Verfahren (aus [Sch98])

Simulink Ähnlich wie *LabVIEW* dient auch das Tool SIMULINK der Firma *MathWorks* zur Verarbeitung von Meßwerten in oftmals kontinuierlichen Datenströmen. SIMULINK ist eine Erweiterung der universellen Analysesprache und -umgebung MATLAB um Funktionalitäten zur Modellierung, Simulation und Analyse dynamischer Systeme [DH98]. Insbesondere können so Differenzen- und Differentialgleichungen numerisch gelöst werden. Als Programmbausteine dienen vor allem Signalgeneratoren und -transformatoren, einfache Funktionen sowie Integratoren und Darstellungen von Werteverläufen. Parameter der genutzten Module können zur Laufzeit der Simulation geändert werden. Durch die Konzentration auf ein klar abgegrenztes Problemgebiet bleiben Funktionalität und Komplexität von SIMULINK gut überschaubar, womit eine recht intuitive Nutzbarkeit gewährleistet ist. Mit der Beschränkung auf die Verarbeitung numerischer Werte und Vektoren entfallen auch aufwendige Typprüfungen. Ein ähnlicher Ansatz findet sich in [Wei98].

3.4.4 Data Mining und allgemeine statistische Analyse

Während sich alle bisher betrachteten Datenflußsysteme auf die Datenvisualisierung konzentrierten, werden im folgenden noch zwei Umgebungen betrachtet, die sich vor allem der Anwendung statistischer Verfahren bzw. von Algorithmen des Data Mining widmen.

ViSta Das *Visual Statistics System* (*ViSta*) von Young, Universität von North Carolina, stellt eine flexible Analyseunterstützung unterschiedlicher Benutzergruppen („Statistik–Laien“ und Experten in Lehre und Forschung) durch verschiedene, miteinander eng gekoppelte Benutzungsschnittstellen in den Vordergrund [YL95, You96].[7] Weiterhin soll so unterschiedlichen Analysezielen und -situationen entsprochen werden. Hierbei wird in [YS91] zwischen Explorieren, Strukturieren und Konfirmieren differenziert; auch die Ergebniskommunikation wird angesprochen:

- *Guidemaps* repräsentieren von Experten formulierte Strategien und alternative Wege zur Datenanalyse in Form zyklischer und hierarchischer gerichteter Graphen aus Klassen von Analyseschritten oder auch (auf der feinsten Ebene) konkreten Verfahren.

- *Workmaps* visualisieren in azyklischen gerichteten Graphen die bisher vorgenommenen Analyseschritte. Knoten stellen Datensätze, Analyseverfahren und Modelle (Sammlungen von Maßzahlen und Wertetabellen, aus denen neue Datensätze extrahiert werden können) dar. Datenmanagementverfahren (insbesondere Restriktionen und Vereinigungen) haben keine eigenen Icons — bei ihrer Anwendung wird direkt ein neuer Datensatz erzeugt und als Netzknoten visualisiert. Durch Anklicken eines Daten– oder Modell–Knotens werden die unterliegenden Daten in mehreren gelinkten und interaktiv manipulierbaren Graphiken visualisiert oder tabellarisch dargestellt. Über Menüs oder Navigation und Selektion in einer aktuell genutzten Guidemap lassen sich anzuwendende Verfahren spezifizieren. Außerdem kann in der Workmap der Analysefokus für weitere Schritte gesetzt werden; Modifikationen alter Schritte sind nicht möglich.

- Über eine Kommandozeile können ebenfalls Analyseschritte in der Analysesprache *ViDAL* spezifiziert werden. Eingegebene Befehle werden sofort interpretiert und in der Workmap als neue Netzbausteine visualisiert. Die Erstellung von Skripten ist ebenfalls möglich.

- Über ein insbesondere in die Guidemaps nahtlos integriertes Hilfesystem sind nähere Informationen zu Analyseschritten und -strategien abrufbar.

- Ein spezielles Tool erlaubt die visuelle Erstellung von Guidemaps.

Abbildung 3.13 zeigt ein Bildschirmfoto der Benutzungsoberfläche von *ViSta*: links die Workmap, rechts die Guidemap, oben die Kommandozeile sowie rechts unten eine Darstellung des aktuellen Datensatzes.

ViDAL und damit auch die Implementierung von *ViSta* basieren auf der Analysesprache und -umgebung *LispStat*, die funktionale und objektorientierte Konzepte kombiniert. Über den objektorientierten Ansatz erfolgt die enge Kopplung aller Systemkomponenten und wird die Menge auf einen Datensatz anwendbarer Management- und Analyseverfahren sowie darzustellender Visualisierungen spezifiziert. Zudem ist das System so leicht erweiterbar.

Clementine *Clementine* ist eine einfach zu bedienende, aber mächtige datenflußbasierte Analyseumgebung [WK97], die von der Firma *ISL* entwickelt und inzwischen von *SPSS* aufgekauft wurde. Alle Phasen des Data Mining, insbesondere auch Datenmanagement und -visualisierung bzw. -exploration auf relationalen Daten werden unterstützt. Die Sprache *CLEM* zur Definition von Ausdrücken und Bedingungsprüfungen in Form von Knotenparametern ergänzt die visuelle Programmierumgebung. Eine zentrale Rolle spielen Modellierungsknoten, die verschiedene Verfahren des Data Mining realisieren. Ergebnisse dieser Bausteine sind „gelernte Modelle“, die in Form spezieller graphischer Symbole dargestellt werden.[8] Derartige Modellsymbole können wiederum als neue Bausteine zur Vorhersage oder Klassifikation auf anderen Datensätzen im gleichen oder in weiteren Datenflußprogrammen genutzt oder auch als eigenständige *C*–Programme exportiert werden.

[7] Erste Versionen von *ViSta* finden sich in der Literatur auch unter dem Akronym *MIDAS* (*Mode–based Interface for Data Analysis and Statistics*) [YS91].

[8] Im Datenflußprogramm haben diese Symbole keine Verbindung zu ihren Erzeugern, sondern werden separat davon abgelegt.

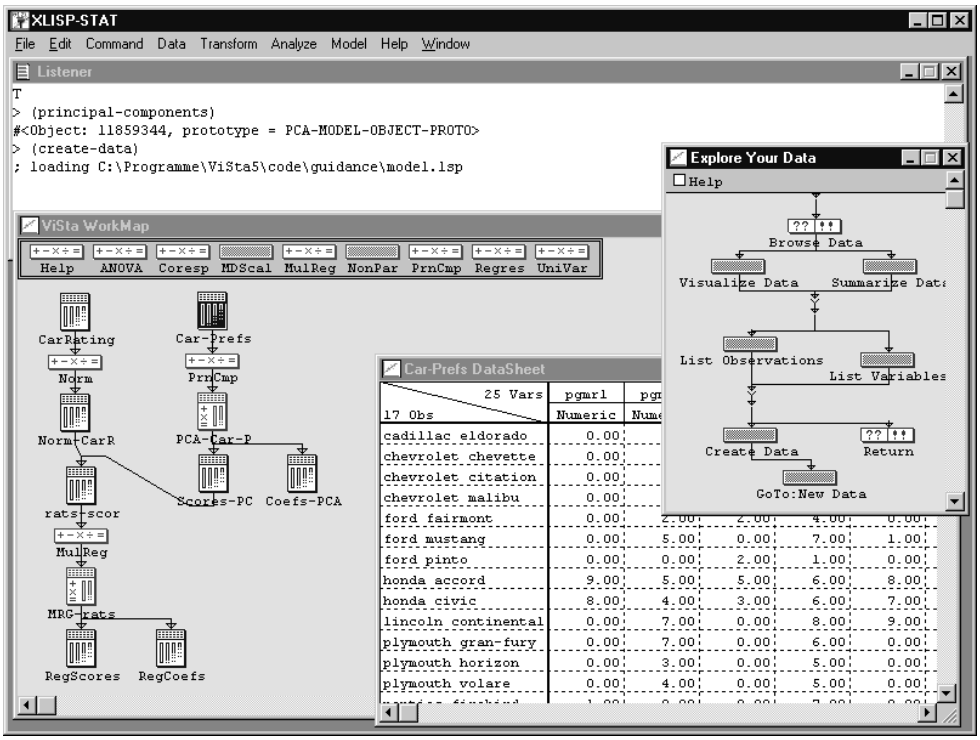

Abbildung 3.13: Alternative Benutzungsschnittstellen in *ViSta*

3.5 Wissensbasierte Systeme zur Datenanalyse

Wissensbasierte Analysesysteme (auch statistische Expertensysteme genannt) können dem Datenanalysten Hilfestellung bei der Datenanalyse geben (vgl. Abschnitt 2.1.1). Sie können prinzipiell als alleinstehende Beratungskomponenten realisiert sein, sind jedoch in der Regel an bestimmte Analysesysteme (aus den oben angeführten Systemklassen) angebunden bzw. mit diesen integriert. Entsprechend läßt sich auch kein einheitliches Interaktionsmodell für diese Systemklasse an sich definieren. Überblicksdarstellungen über wissensbasierte Datenanalysesysteme finden sich etwa in [Haa94, Mar97], Basiswerke mit grundlegenden Arbeiten bilden [Gal86, Hau86, MWS+91]. Speziell der datenflußbasierten Sicht auf Analysen widmet sich in diesem Kontext [Ama97], wo insbesondere die Darstellung der Zusammenhänge zwischen Daten, Verfahren, Analysezielen, Regeln und Strategien untersucht wird. Im folgenden seien kurz einige exemplarische Systembeispiele[9] aufgeführt, um später grundlegende Bezüge zur in dieser Arbeit propagierten „intelligenten Datenanalyse“ herstellen zu können. Hierbei interessieren vor allem die Kooperation von Mensch und Rechnersystem bei der Umsetzung von Analysestrategien, die auf Datenbeschreibungen gestützte (teil-)automatisierte Generierung von Visualisierungen, die mögliche Einbeziehung der Datenflußsicht in den Inferenzprozeß sowie die Rolle eines objektbasierten Datenmodells.

AIDE In dem Projekt *AIDE* (*Automated Intelligent Data Exploration*) an der Universität von Massachusetts steht die geeignete Kombination von Nutzer- und Rechnerinitiative im Kern der Systemkonzeption [AC96a, AC98]. Die explorative Datenanalyse wird als Such-, speziell als Planungsproblem aufgefaßt [AC96b]. Ein intelligenter Assistent sucht zum jeweils vorliegenden Analyseproblem unter Beachtung der Charakteristika zu verarbeitender Daten aus seiner Wissensbasis einen oder mehrere geeignete Pläne aus, die dem Benutzer

[9] Kommerziell haben sich wissensbasierte Systeme in der Datenanalyse (noch?) nicht durchgesetzt. Dementsprechend sind die vorgestellten Arbeiten eher „akademischer“ Natur.

vorgeschlagen werden und — in ständiger Interaktion mit diesem[10] — durch sukzessive Verfeinerung von Teilzielen in weiteren Inferenzschritten schließlich zu konkreten Analyseoperationen führen. In diesem Sinn verkörpern Pläne *Strategien* der Datenanalyse. Bereits durchgeführte Operationsfolgen sowie Alternativpfade der Analyse werden in einer einfachen graphbasierten Notation dargestellt. Diese dient zum einen zur einfachen Anforderung von Erklärungen für Systemvorschläge und Schlußfolgerungen und zum anderen zur Wahl des Analysefokus, also des Teilziels, das als nächstes verfeinert werden soll, oder eines Alternativpfades, der weiter zu verfolgen ist.

SAGE In Abschnitt 3.2.2 wurde bereits die Visualisierungsumgebung *Visage* vorgestellt. Eine Komponente von *Visage* bildet das Tool *SAGE* (*System for Automatic and Graphical Explanation*), das der wissensbasierten Komposition graphischer Darstellungen dient [RCK+97]. *SAGE* besteht aus

- dem Editor *SageBrush*, der flexible Möglichkeiten bietet, Abbildungen von Datenattributen auf Graphikkomponenten und deren Eigenschaften in komplexen Diagrammen zu spezifizieren,
- der Bibliothek *SageBook*, die einen historischen Bestand von Datenvisualisierungen enthält und hierauf Suchanfragen nach jeweils ähnlichen Fallbeispielen ermöglicht, sowie
- einer Inferenzkomponente, die aufgrund von Benutzerzielen (Visualisierungsaufgaben, vgl. Abschnitt 2.1.3, sowie Gewichtungen von Attributen in ihrer Bedeutung [GRKM94]) und Datencharakteristika (Kardinalität, Datentyp, Attributbeziehungen, vgl. [RM90]) Graphiken vollständig erstellt oder unvollständige Grobentwürfe ergänzt.

Tecate Innerhalb des *SEQUOIA–2000*–Projekts [Sto94] wurde neben *Tioga–2* an der Universität von Kalifornien in San Diego noch eine zweite Visualisierungskomponente entwickelt, nämlich das intelligente Visualisierungssystem *Tecate* [AWK94]. Wie schon bei *SAGE* wird auch in diesem System zu einer Visualisierungsaufgabe und einer Datenbeschreibung eine geeignete Graphik erstellt. Dies erfolgt in *Tecate* mittels einer Zerlegung des Gesamtproblems in attributbezogene Teilaufgaben. Auf der Basis einer framebasierten Wissensrepräsentation werden sukzessive Operatorfolgen zur Lösung dieser Teilaufgaben konstruiert und anschließend zu einem Datenflußprogramm zusammengesetzt.[11] *Tecate* nutzt *AVS* zur Modellierung und Ausführung von Visualisierungsprogrammen. Eine (mögliche) Integration von *Tecate* und *Tioga–2* wird zwar in [AWK94, SCN+93] kurz angesprochen, aber nicht näher ausgeführt.

IMACS Abschließend sei noch das bei *AT&T* entwickelte System *IMACS* (*Interactive Market Analysis and Classification System*) im Hinblick auf Möglichkeiten der adäquaten, „intelligenten“ Datenmodellierung erwähnt [BST+93]. In *IMACS* erfolgt die explorative Datenanalyse — ähnlich zu den in Abschnitt 3.1.2 vorgestellten Analysesprachen — in einem interaktiven System. Der Analyse liegt jeweils eine framebasierte Modellierung der Anwendungsdomäne zugrunde, die die Spezifikation, Beschreibung, Ermittlung und Analyse von Untergruppen mit interessierenden Eigenschaften gut unterstützt. Eine mächtige Anfragesprache gewährt flexiblen Zugriff auf diese Wissensbasis.

3.6 *CARESS* — das Auswertungssystem des Niedersächsischen Krebsregisters

Die Krebsepidemiologie bildet das zentrale Anwendungsgebiet, auf das diese Arbeit bzw. der Entwurf von *MADEIRA* und *VIOLA* abzielt. So soll *VIOLA* mittelfristig das derzeit im Epidemiologischen Krebsregister Niedersachsen (EKN) eingesetzte Auswertungssystem *CARESS* [WK98, WGG+97, RWW99] ersetzen bzw.

[10] Nur einfache Aktionen ohne Alternativen werden unter Umständen vollautomatisch ausgeführt.

[11] Hier zeigen sich Ähnlichkeiten zum *Weaves*–System, das in Abschnitt 2.2.5 bereits im Kontext der Erstellung komplexer Softwaresysteme aus Softwarekomponenten kurz vorgestellt wurde.

eine flexible Basis für die dort bereits umgesetzte Benutzungsoberfläche (vgl. Abb. 3.14) realisieren. In Abschnitt 6.3 wird hierauf noch genauer eingegangen werden; an dieser Stelle sollen lediglich einige Grundideen von *CARESS* im Kontext der anderen in diesem Kapitel vorgestellten Datenanalysesysteme skizziert werden.

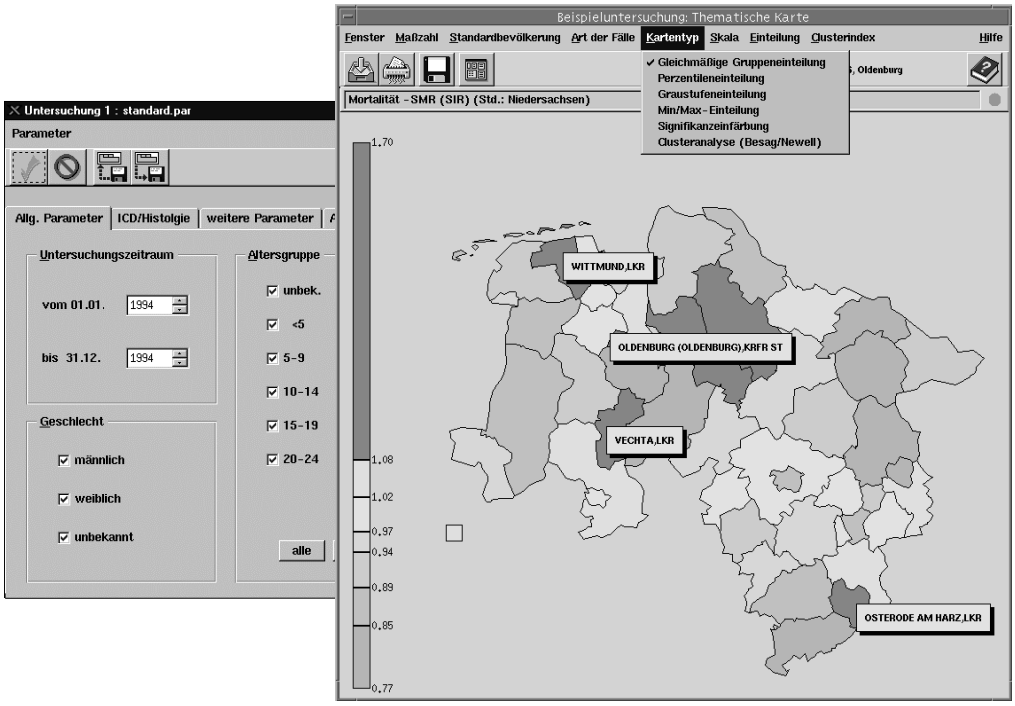

Abbildung 3.14: Das Auswertungssystem *CARESS*

CARESS ist ein menübasiertes, interaktives Informationssystem für die (Krebs–)Epidemiologie, das seinen Schwerpunkt auf die raumbezogene Datenanalyse legt. Die unterliegende relationale, durch das DBMS *ORACLE* verwaltete Datenbank enthält multidimensionale Daten zu Mortalitäts- und Inzidenzfällen sowie der Niedersächsischen Bevölkerung im zeitlichen Verlauf, weiterhin Vergleichsdaten anderer Krebsregister und künstliche Standardbevölkerungen. Im Rahmen epidemiologischen Untersuchungen, von denen parallel mehrere durchführbar sind, wird die zu betrachtende Studienpopulation jeweils über eine Niedersachsenkarte sowie verschiedene Masken mit weiteren Attributen (Geschlecht, Alter, Zeitraum, Art der Erkrankung) eingegrenzt. Weiterhin wird auch die jeweils interessierende Aggregierungsebene festgelegt.

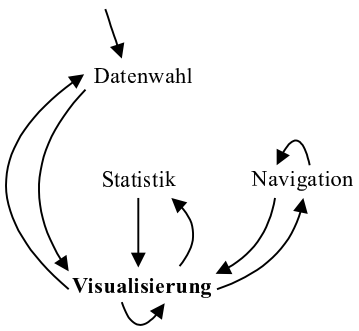

Abbildung 3.15: Das Interaktionsmodell von *CARESS*

Anschließend können auf dem so festgelegten Datenraum unterschiedliche Visualisierungen vorgenommen werden: Tabellen, thematische Karten, Altersverteilungen in Balkendiagrammen sowie zeitliche Verläufe in Kurvendiagrammen. In jeder Darstellung können interaktiv die darzustellende Maßzahl, die zu verwendende Standardpopulation sowie weitere, auswertungsspezifische Parameter eingestellt werden. So ist etwa in der in Abb. 3.14 gezeigten Karte zwischen verschiedenen Kartentypen, verschiedenen Skalen und Gruppenanzahlen zur Einfärbung zu wählen. Ausgehend von einer thematischen Karte können schließlich noch eine Reihe von Clusterindizes zur Bewertung von Heterogenität und räumlicher Autokorrelation der Erkrankungszahlen ermittelt und mit einer Reihe zugeordneter Maßzahlen angezeigt werden. Das durch *CARESS* unterstützte Interakti-

Systeme	*Multidimensionales Datenmodell*	*Flexibles Datenmanagement*	*Navigation im Explorationsprozeß*	*Visuelle Programmierung*	*Interaktive Graphik*	*Wissensbasierte Verfahrenswahl*	*Erweiterbarkeit*	*Verfahrensbibliothek*
Statistikpakete	∘	+	∘	–	∘	–	∘	++
Analysesprachen	∘	∘	+	–	+	–	++	++
(Interaktive) Visualisierung	∘	–	∘	∘	+(+)	–	–	–
OLAP–Tools	+(+)	++	∘	∘	∘	–	∘	+
Datenfluß–Umgebungen	∘	∘	++	++	+	∘	+	++
Statistik–Expertensysteme	–	–	∘	–	–	++	∘	–
CARESS	+	+	∘	–	+	–	∘	∘
VIOLA	++	++	++	++	++	+	+	∘

Tabelle 3.1: Gegenüberstellung zentraler Klassen von Datenanalysesystemen und Einordnung dieser Arbeit

onsmodell findet sich in Abb. 3.15. Es ist gegenüber der freien Navigation in datenflußbasierten Umgebungen deutlich eingeschränkt und zeichnet sich vor allem durch die Fokussierung auf die Visualisierung aus. Von dort aus können sowohl neue Maßzahlberechnungen angestoßen als auch die jeweiligen Selektionsparameter manipuliert werden, worauf sich die Darstellung dynamisch ändert. Ein Linking zwischen Graphiken ist nicht vorgesehen.

3.7 Zusammenfassung und Ableitung von Anforderungen an diese Arbeit

In diesem Kapitel wurde eine Klassifizierung von Systemen zur Datenanalyse mit ihren jeweils unterstützten Interaktionsmodellen vorgestellt. Abschließend stellen wir in Tab. 3.1 nochmals alle Systemklassen im Hinblick auf einige spezielle Charakteristika sowie den Umfang der Unterstützung der vier grundlegenden Interaktionsklassen grob gegenüber und leiten daraus zentrale Anforderungen an *VIOLA* ab. Wenn auch die vorgenommenen Einstufungen von geringer (–) über mäßige (∘) bis zu guter (+) und sehr guter (++) Berücksichtigung der betrachteten Kriterien im Einzelfall nicht immer eindeutig und klar abzugrenzen sind, zeigt Tab. 3.1 doch die jeweiligen Tendenzen der Systemausrichtung.

Zunächst einmal spielt die Unterstützung eines zugrundeliegenden multidimensionalen (matrixbasierten) Datenmodells mit seinen speziellen Eigenheiten neben einem allgemein flexibel angelegten Datenmanagement für *VIOLA* eine zentrale Rolle. Hier zeigen sich Parallelen zu OLAP–Tools, denen jedoch teilweise die formale Grundlage der Datenmodellierung fehlt. Bei *CARESS* bleibt der multidimensionale Charakter der Daten weitgehend vor dem Anwender verborgen, so daß nicht alle Möglichkeiten des Datenmodells ausgeschöpft werden können. Die meisten Analysesysteme bieten eine einfache matrixbasierte Datensicht, wobei das Datenmanagement für Statistikpakete eine überdurchschnittlich große und für Visualisierungssysteme eine geringere Rolle spielt. Statistik–Expertensysteme unterstützen i. a. lediglich einen rudimentären, relationalen Zugriff auf die Basisdaten.

VIOLA soll, wie die meisten anderen datenflußbasierten Analysewerkzeuge, eine größtmögliche Flexibilität in der Abfolge von Analyseinteraktionen bieten (dem rechten Modell aus Abb. 3.8 entsprechend). Matrixbasierte Analysesprachen ragen aufgrund der expliziten Identifizierbarkeit einzelner Analyseschritte und der hier

resultierenden Wiederverwendbarkeit von Zwischenergebnissen unter diesem Aspekt aus den übrigen Systemklassen heraus, ermöglichen jedoch keine interaktive Manipulation bereits genutzter Operationen.

Neben der Navigation in Form eines freien Manövrierens über den Explorationsprozeß wird in Tab. 3.1 auch die visuelle Programmierung, also die direkte Interaktion mit Verfahren und Daten über eine geeignete Benutzungsschnittstelle, als eigener, für *VIOLA* bedeutsamer Aspekt herausgestellt. Dies ist natürlich zugleich die Domäne der vorgestellten Datenflußumgebungen; einige Ansätze einer (nicht nur auf Visualisierungsparameter beschränkten) Programm–Manipulation zeigen auch — vor allem im Hinblick auf die Datenwahl — einige OLAP– und Visualisierungstools.

Letztere propagieren besonders die entsprechende unmittelbare Interaktion mit Analyseergebnissen, also die Nutzung von Visualisierungen zur Steuerung des Datenmanagements und (häufig) auch — in Form eines Linking — zur Verknüpfung von Graphiken. Dieser Aspekt, der in fast allen Systemklassen mehr oder weniger umfassend umgesetzt ist, bildet schließlich die letzte Säule der Konzeption von *VIOLA*. Hierbei sind datenflußbasierte Ansätze zur interaktiven Bildbearbeitung, wie sie in z. B. in den Analyseumgebungen *AVS* oder *IBM Data Explorer* realisiert wurden, auf das multidimensionale Datenmodell und darauf definierte Operatoren zu übertragen.

Insgesamt soll *VIOLA* somit im wesentlichen eine Kombination der Stärken von

- Datenflußumgebungen,
- Systemen zur interaktiven Visualisierung und
- OLAP–Tools

bieten. Die wissensbasierte Verfahrenswahl findet insofern Beachtung, als eine genaue Beschreibung der jeweils untersuchten Daten durch Metadaten/Typsysteme und somit eine syntaktische und semantische, nicht aber pragmatische Bewertung der Anwendbarkeit von Analyseverfahren erfolgen soll. Im Hinblick auf die Einbeziehung darüber hinausgehender Aspekte, wie etwa der Charakterisierung von Analysezielen oder der Berücksichtigung von Werteverteilungen bei der Einstufung der Sinnhaftigkeit einer Verfahrensanwendung, soll *VIOLA* nicht an wissensbasierte Systeme heranreichen.[12] Die Erweiterbarkeit des Systems (eine Stärke matrixbasierter Analysesprachen) wird sich auf die Definition klarer Schnittstellen zur (textuellen) Programmierung neuer Bausteine beschränken. Eine (zunächst) als Basis angestrebte Verfahrensbibliothek wird in grundlegenden Verfahren für die Krebsepidemiologie bestehen. Im Hinblick auf die Breite der angebotenen Funktionalität kann *VIOLA* somit (zumindest zunächst einmal) bei weitem nicht mit allgemeinen Statistikpaketen mithalten, erlaubt aber die Konzentration auf eine spezielle Verfahrenspalette, die von einer konkreten Anwendungsdomäne benötigt wird.

[12] Wenn auch eine Bearbeitung dieser Thematik den Rahmen dieser Arbeit sprengen würde, wäre eine entsprechende Fortführung der Konzepte von *VIOLA* nichtsdestoweniger sehr gut vorstellbar. Wir kommen hierauf in Kapitel 7 nochmals kurz zu sprechen.

Kapitel 4

MADEIRA — ein logisches Modell zur Beschreibung aggregierter Datenbestände

Die Analyseumgebung *VIOLA* zielt auf die multidimensionale Datenanalyse. Somit sind es zum einen multidimensionale Datenstrukturen, also Makrodaten im Sinne der Ausführungen in Abschnitt 2.3, die von den Verfahren der Datenanalyse verarbeitet und berechnet werden. Zum anderen ergeben sich aus den so modellierten Strukturen aber auch bereits die grundlegenden Operationen zur Navigation in Datenbeständen sowie Anforderungen an einzusetzende Kontrollstrukturen. Mehr noch, eine umfassende Algebra auf multidimensionalen Daten definiert gerade die Analysebausteine in der datenflußbasierten Umgebung. Somit wird die herausragende Bedeutung der Datenmodellierung für *VIOLA* deutlich: Das Datenmodell spezifiziert eine einheitliche „Data Language“ [Dye90], auf deren Grundlage die durchgeführten Untersuchungen genau beschrieben werden können.[1] Es bildet das Bindeglied zwischen der zugrundeliegenden Datenbank, der visuellen Programmierung der Datenverarbeitung und der Visualisierung der betrachteten Informationen.

Bisher in der Literatur vorgestellte Datenmodelle (vgl. Abschnitt 2.3.3) konzentrieren sich im wesentlichen auf die syntaktische Beschreibung multidimensionaler Daten. Sie bieten mächtige Operatoren zur Generierung komplexer Datenbankanfragen, betrachten jedoch nicht den Prozeß der nachfolgenden (oder auch vorangegangenen) Datenanalyse. Insbesondere erlauben sie keine derart genaue semantische Modellierung der Zusammenhänge zwischen den verarbeiteten Datenstrukturen bzw. zwischen diesen und den modellierten Anwendungsobjekten, wie sie nötig ist, um durchgeführte komplexe Datenanalysen in ihrer Gesamtheit wirklich genau beschreiben, nachvollziehen und mit anderen Untersuchungen vergleichen sowie auch den korrekten und sinnvollen Einsatz von Verarbeitungsverfahren auf bestimmte Datensätze begründen zu können. Gerade vielen im Kontext von OLAP entstandenen Datenmodellen mangelt es hierzu — im Gegensatz zu den in Abschnitt 2.3.3 vorgestellten Modellen älteren Datums, z. B. *STORM* [RS90] oder *Mefisto* [FMENR89] — an der Berücksichtigung von Metadaten auf der Ebene der Datensemantik. Vor allem sind hierbei eine differenzierte Betrachtung von Kategorienhierarchien sowie des Typs der summarischen Daten bzw. der Definition von Analyseverfahren hervorzuheben.

Vor diesem Hintergrund wird in der vorliegenden Arbeit das Datenmodell *MADEIRA* (*Modelling Analyses of Data in Epidemiological InteRActive studies*) entwickelt, das genau diese Aspekte in das Zentrum der Betrachtung stellt. So erfordert die flexible Unterstützung interaktiver Untersuchungen eine genaue Modellierung der betrachteten Zusammenhänge, um den Benutzer von automatisierbaren Entscheidungen befreien zu können, ihm auf seine Bedürfnisse angepaßte, intuitive Möglichkeiten zur Parametrisierung von Untersu-

[1] In ähnlicher Weise bilden verschiedene Gittermodelle die Basis für viele (gerade auch datenflußbasierte) Analysesysteme im Bereich der Bildverarbeitung und wissenschaftlichen Visualisierung (vgl. etwa [KBS94, HDP93, HLC91]).

chungen zu geben und bei alledem eine semantisch klare und eindeutige Spezifikation von Auswertungen zu gewährleisten.

Der folgende Abschnitt 4.1 faßt zunächst noch einmal wichtige Aspekte existierender multidimensionaler Datenmodelle zusammen, um auf dieser Basis den mit *MADEIRA* verfolgten Ansatz besser beschreiben, einordnen und abgrenzen zu können. Anschließend wird das Datenmodell *MADEIRA* im Detail vorgestellt. Als erstes werden in Abschnitt 4.2 Kategorien und Dimensionen als Grundelemente zur Klassifikation von Mikrodaten, anschließend Datenräume zur Darstellung von Makrodaten und schließlich in Abschnitt 4.3 Operationen auf Makrodaten formal definiert. Der Diskussion von Metadaten zur Beschreibung von Typen summarischer Daten sowie von weiteren, für das Datenmodell selbst nicht weiter relevanten, aber für die korrekte und komfortable Datenverarbeitung im Rahmen von *VIOLA* um so wichtigeren Hintergrundinformationen ist der Abschnitt 4.4 gewidmet. Abschließend folgt noch ein kurzes Fazit in Abschnitt 4.5. Konkrete Beispiele werden jeweils dem Bereich der Krebsregistrierung, wie er etwa durch das in Abschnitt 3.6 geschilderte Basisszenario für *CARESS* umrissen wurde, entnommen (vgl. auch Abschnitt 1.2).

4.1 Charakterisierung von *MADEIRA*

Es gibt inzwischen eine Vielzahl von Ansätzen zur Modellierung multidimensionaler Daten, von denen in dieser Arbeit in Abschnitt 2.3.3 nur einige wenige und auch die nur im Hinblick auf ihre herausragendsten Charakteristika vorgestellt werden konnten. Im folgenden werden einige für diese Arbeit und insbesondere für den mit *MADEIRA* verfolgten Modellierungsansatz besonders interessante Aspekte einiger ausgewählter Datenmodelle gegenübergestellt. Anschließend werden hieraus wesentliche Modellierungsziele von *MADEIRA* abgeleitet sowie zentrale Eigenarten dieses Modells anhand einer Einordnung in den angewendeten Kriterienkatalog erläutert.

4.1.1 Klassifizierung ausgewählter multidimensionaler Datenmodelle

Die nachfolgende Darstellung orientiert sich an zwei vergleichenden Übersichten über verschiedene multidimensionale Modellierungsansätze in [SBHD98] und [PJ99]. Über die dort vorgenommenen Betrachtungen hinaus wurden hier noch drei Modelle aus dem Umfeld statistischer Datenbanken aufgenommen, nämlich *Mefisto* [RR93], das *ADaS*–Modell [BRT96] sowie der METAMOD–Ansatz [Fro97]. Weiterhin wurde der Kriterienkatalog um die Unterscheidung von MOLAP und ROLAP, die Herstellung eines Mikrodatenbezugs sowie die Integration von Metadaten erweitert. Diese Aspekte dienen zum einen zur besseren Abgrenzung der hinzugekommenen drei Modelle und spielen zum anderen auch für *MADEIRA* eine besondere Rolle im Kontext der genauen Modellierung der Datensemantik.

Im einzelnen werden in Tab. 4.1 folgende Aspekte zum grundlegenden Modellierungsansatz sowie zur Repräsentation von Kategorien, Maßzahlen und Operationen gegenübergestellt („-“ steht in Tab. 4.1 jeweils dafür, daß ein Aspekt durch das betreffende Modell *nicht* berücksichtigt wird):

MOLAP vs. ROLAP Orientiert sich das Modell an einer relationalen Implementierung (R), d. h. lehnt sich die Repräsentation von Datenwürfeln an ein Sternschema an, indem etwa Würfelinstanzen als Listen besetzter Zellen in Tupelform dargestellt werden? Sind bei einer matrixbasierten Modellierung Bezüge zum relationalen Modell deutlich erkennbar (M/R)? Oder wird ein *rein*[2] multidimensionaler Ansatz verfolgt (M)?

Schema $\leftrightarrow$ Instanz Wird zwischen Datenschema und Instanz, also zwischen Metadaten und Daten klar unterschieden ($\surd$)?

[2] Sofern Mikrodatenbestände als Basisdaten zugreifbar sein sollen, müssen natürlich auch hier Beziehungen zum relationalen Modell hergestellt werden, die jedoch nicht die Gestaltung der jeweiligen zentralen Datenstrukturen berühren.

	Allgemein		**Kategorien**				**Maßzahlen**			**Verarbeitung**			
Modell	*MOLAP vs. ROLAP*	*Schema ↔ Instanz*	*Kategorienhierarchien*	*Mengenwertige Bez.*	*Ad–hoc–Gruppierung*	*Variable Granularität*	*Maße ↔ Kategorien*	*Komplexe Würfel*	*Aggregierbarkeit*	*Metadaten*	*Funktionsintegration*	*Mikrodatenbezug*	*Algebra vs. Kalkül*
STORM	M	√	S	√[2]	-	-	-	-	√[2]	(√)	-	DB[2]	-
Mefisto	M	-	-	-	√	-	-	-	√	(√)	(√)	-	A
ADaS	M	-	-	-	-	√[4]	-	√[4]	√	(√)	(√)	DB[2]	A
Fröschl	M	√	S	(√)?	√	√	DB	-	√	(√)	√	DB	K/A
Agrawal et al.	R	-	-	(√)	√	√[5]	Op.	√	-	-	√	-	A
Li & Wang	R	√	I[1]	(√)	√	-[5]	-	-	-	-	√	Op.	A
Gyssens et al.	R	√	I[1]	(√)	√	-[5]	Op.	√	-	-	√	-	A/K
Cabibbo et al.	M/R	√	S	-	√	-	Op.	-	-	-	√	-	A/K
Lehner	M/R	√	S	-	-	-	-	-	(√)	-	-	-	A
Vassiliadis	M/R	-	S	√[3]	-	-	-	-	-	-	-	-	A
Pedersen et al.	R	-	S	√	-	√	=	√	(√)	(√)	-	=	A
MADEIRA	M	√	S	√	√	√	DB	√	√	√	√	DB	A

[1] Lediglich die Aggregierungsebenen selbst sind durch das Schema, deren Beziehungen jedoch durch funktionale Abhängigkeiten der Instanzen repräsentiert. Kategorien ergeben sich allein aus den Instanzen.

[2] Der Aspekt ist nicht vollständig in das Modell integriert, wird jedoch informell diskutiert.

[3] Es werden offenbar nur mengenwertige Maßzahlen, nicht jedoch mengenwertige Kategorien unterstützt.

[4] Nur in zusammengesetzten *ADaS*.

[5] Der Begriff der „Granularität“ ist bei Agrawal undefiniert und auch bei Li bzw. Gyssens aufgrund der schemabezogenen Kategoriendefinition und sehr flexiblen Aggregierung nicht ganz eindeutig.

Tabelle 4.1: Multidimensionale Datenmodelle im Überblick

Kategorienhierarchien Werden Kategorienhierarchien explizit modelliert? Falls ja, geschieht dies als Teil von Schema (S) oder Instanz (I)?

Mengenwertige Beziehungen Werden mengenwertige Beziehungen (Eigenschaften) durch entsprechende mengenwertige Kategorien bzw. die Möglichkeit der Zuordnung *mehrerer* Kategorien einer Aggregierungsebene zu einem Maßzahlwert oder einer Kategorie einer anderen Dimension unterstützt (√)? Erfolgt dies evtl. nur durch eine flexible Modellierung der Kategorienhierarchien, ohne es aber darüber hinaus in der Datenverarbeitung speziell zu berücksichtigen ((√))?

Ad–hoc–Hierarchien Können über die modellierte Kategorienhierarchie hinausgehende Aggregierungsbeziehungen spezifiziert bzw. einzelne Kategorien beliebig zusammengefaßt werden (√)?

Variable Granularität Können in einem Datenraum gleichzeitig Daten mit bzgl. einer oder mehrerer Dimensionen *verschiedener* Granularität gehalten bzw. verarbeitet werden, wenn etwa Basisdaten mit unterschiedlich genauen Angaben vorliegen (√)?

Maße ↔ Kategorien Werden summarische und kategorielle Attribute symmetrisch behandelt? Gibt es keinen Unterschied zwischen ihnen (=), kann man sie durch spezielle Operatoren ineinander umformen (Op.), oder ist zumindest beim Zugriff auf die Basisdaten in der Datenbank wählbar, welche Mikrodatenattribute als kategorielle Attribute genutzt und aus welchen Werte summarischer Attribute berechnet werden (DB)?

Komplexe Datenwürfel Werden mehrere Maßzahlen pro Datenwürfel unterstützt ($\surd$)?

Aggregierbarkeit Werden Typen summarischer Attribute zur Unterstützung der korrekten automatischen Aggregierung modelliert ($\surd$)? Wird dieser Zusammenhang allgemein beschrieben oder sind lediglich einige wenige ausgewählte Daten- bzw. Aggregierungstypen vorgesehen (($\surd$))?

Metadaten Sind über die Beschreibung der Datenstrukturen hinausgehende Metadaten zur Unterstützung statistischer Analysen vorgesehen? Handelt es sich hierbei lediglich um einzelne Texte oder spezielle Aspekte wie Versionierung, Nullwerte, Einheiten etc. (($\surd$)), oder wird ein umfassendes Konzept vorgestellt sowie dessen Nutzung im Rahmen der Anwendung von Verarbeitungsoperatoren modelliert ($\surd$)?

Funktionsintegration Kann der Benutzer (nahezu) beliebige Berechnungsfunktionen im Rahmen des Datenmodells zur Auswertung multidimensionaler Daten einbinden ($\surd$)? Ist dies evtl. nur im Rahmen der Datenaggregierung möglich (($\surd$))?

Mikrodatenbezug Wird der Bezug der Makrodaten auf die zugrundeliegenden Mikrodaten, also die genaue Herleitung der aggregierten Werte aus den Basisdaten modelliert (DB)? Existieren Operatoren zur Verknüpfung von Makro- mit Mikrodaten (Op.) oder sind beide sowieso in einheitlicher Weise modelliert, so daß ihre Kombination keine besondere Behandlung erfordert (=)?

Algebra vs. Kalkül Wird eine Algebra (A) an Operatoren und/oder ein deklaratives Kalkül (K) beschrieben? Ggf. mit welchem Schwerpunkt: Algebra (A/K) oder Kalkül (K/A)?

Es fällt auf, daß die vier erstgenannten Datenmodelle sich im Hinblick auf die Einbeziehung von Metadaten allgemein und speziell zur Unterstützung der Datenaggregierung deutlich von den übrigen Ansätzen abgrenzen. Während ältere Arbeiten eine konzeptionell streng multidimensionale Modellierung verfolgen, bedingt die Implementierung der neueren Modelle im Rahmen von Data Warehousing und OLAP–Werkzeugen häufig eine Fokussierung auf relationale Strukturen. Vermutlich auch aus diesem Grunde wird im OLAP–Kontext häufig eine Symmetrie von Maßen und Kategorien propagiert, die die Mächtigkeit der Modelle stark erhöht, jedoch die klare multidimensionale Modellierung verschwimmen läßt.

Die letzte Zeile in Tab. 4.1 ordnet *MADEIRA* in die vorgenommene Klassifikation sowie die Menge bestehender Modelle ein: Der streng multidimensionale Ansatz aus dem Kontext statistischer Datenbanksysteme wird kombiniert mit der klareren Modellierung von Kategorienhierarchien sowie der Bereitstellung mächtigerer Klassifikations- und Analysemöglichkeiten aus dem OLAP–Umfeld. Details zur Zielsetzung von *MADEIRA* werden im nachfolgenden Abschnitt diskutiert. Insgesamt wird sich auf der Basis obiger Darstellung als Schwerpunkt der Modellierung, der *MADEIRA* aus der Menge bestehender Datenmodelle heraushebt, die umfassende Integration differenzierter Metadaten zur Datensemantik herauskristallisieren: Zum einen werden Maßzahlen, ihre Herleitung aus Mikrodaten sowie ihre Berechnung durch Analyseverfahren genau beschrieben, wobei auch eine beliebige Erweiterung um neue Maße und Berechnungswege ermöglicht wird; zum anderen werden auch Kategorien (im Gegensatz zu allen vorgestellten Modellen *unabhängig* von, aber ergänzt durch Aggregierungsebenen) in ihrer Semantik klarer und flexibler modelliert. Nur mit diesen Basiskonzepten kann eine adäquate Grundlage für die Modellierung und Durchführung explorativer Datenanalysen gelegt werden.

4.1.2 Ziele der Datenmodellierung in *MADEIRA*

Im folgenden werden nun einige grundsätzliche Überlegungen angeführt, die wesentlichen Einfluß auf die Gestaltung von *MADEIRA* haben und die vor allem auch allgemein den Entwurf eines *neuen*, gegenüber existierenden Ansätzen andersartigen Datenmodells motivieren:

- *MADEIRA* soll dem *logischen* Entwurf bzw. genauer der Beschreibung des *konzeptionellen* Schemas multidimensionaler Datenbestände auf der Basis der multidimensionalen Datenmodellierung dienen (vgl.

Standardliteratur zum Datenbankentwurf, etwa [MDL87, Vos99, HS95]). Während bei vielen bestehenden logischen multidimensionalen Datenmodellen ein enger Bezug zu einer Implementierung im relationalen Datenmodell besteht, zeichnet sich *MADEIRA* gerade durch die starke Orientierung an der grundlegenden multidimensionalen Sichtweise aus. Der konzeptionelle Entwurf, also die datenmodellunabhängige Darstellung von Sachverhalten des betrachteten Weltausschnitts, ist nicht Gegenstand von *MADEIRA*. Ebenso steht die effiziente Implementierung multidimensionaler Schemata nicht im Kern der Betrachtung. Gelegentlich werden dennoch Aspekte des physischen Entwurfs, also der sinnvollen Umsetzung von *MADEIRA*, kurz angerissen. Entsprechend treten auch interne und externe Schema–Ebene der ANSI/SPARC–Architektur [TK78] in den Hintergrund.

- Gegenüber anderen Modellen steht nicht die Definition einer mächtigen, mit Analyse- und Formatierungsoperatoren angereicherten Datenbankanfragesprache, sondern die Formalisierung des gesamten, vielschrittigen Analyseprozesses im Vordergrund. Somit sind nicht (nur) der persistente Datenbestand, sondern (gerade) auch die transienten Strukturen, die im Zuge einer Datenanalyse entstehen und verarbeitet werden, Gegenstand einer insofern anwendungsnahen Modellierung. Anders ausgedrückt: *MADEIRA* dient in erster Linie als Grundlage einer Klassenbibliothek zur Realisierung eines datenbankbasierten Analysesystems; erst an zweiter Stelle folgt die Modellierung des persistenten Datenschemas eines multidimensionalen Datenbestandes.

  Entsprechend werden im Laufe dieses Kapitels auch verschiedentlich Hinweise auf spezielle Anwendungsmöglichkeiten der Konstrukte des Datenmodells gegeben. „Anwendungsnähe“ ist hierbei als Berücksichtigung der spezifischen Informationsbedürfnisse und funktionalen Anforderungen der explorativen Datenanalyse (auch, aber nicht nur in der Krebsepidemiologie) anzusehen, nicht aber als Unterstützung eines konkreten Anwendungssystems. Bernstein spricht in diesem Kontext von der Definition eines *Informationsmodells*, das zugleich die Grundlage für die Datenverarbeitung in der Anwendung bietet [Ber97].

- *MADEIRA* beschreibt nicht einfach nur die Syntax multidimensionaler Datenbestände, sondern verfolgt in erster Linie eine möglichst genaue Spezifikation der Semantik von Maßzahldefinition und -berechnung auf Teilen einer nach verschiedenen Attributen klassifizierten Gesamtmenge von betrachteten Einzelobjekten. Es soll ein für die Formalisierung der explorativen Datenanalyse geeignetes Modell bereitgestellt werden, das die korrekte und sinnvolle Auswahl und Anwendung von Analysefunktionen unterstützt. Hierzu nötige Metadaten werden möglichst nahtlos in das Modell integriert.

- Im Hinblick auf eine einfache Handhabung des Analysesystems beschränkt sich das Modell auf *eine* grundlegende Datenstruktur, den „Datenwürfel“, und deren Komponenten, die zwischen den Bearbeitungsverfahren der Analyseumgebung ausgetauscht werden. Dies spiegelt auch den multidimensionalen Charakter der Daten unmittelbar wider. Trotz der Genauigkeit der Modellierung soll *MADEIRA* so durch direkte Umsetzung dieser Metapher möglichst verständlich und — wie auch bei Vassiliadis [Vas98] formuliert — gegenüber vielen anderen Modellen möglichst intuitiv nutzbar bleiben. Insbesondere unterscheiden sich durch Datenanalysen erzeugte Datenwürfel in ihrer Struktur nicht von persistent gespeicherten Makrodaten.

  Durch eine klare Trennung von Schema und Instanz werden explizit Metadaten über Datenwürfel definiert, die von der Extension der Daten unabhängig sind (im Gegensatz etwa zu [Leh98], wo sich die Kategorien einer Datenwürfeldimension auf in der (Mikrodaten–)Extension vorhandene Ausprägungen beschränken).

  Mikrodaten spielen keine wesentliche Rolle für Datenaustausch und -verarbeitung, müssen jedoch für einzelne Betrachtungen inspizierbar sein. Außerdem können natürlich auch Basisdaten in relationaler Form vorliegen, so daß ein entsprechender Zugriff modellierbar sein muß.

- Überlegungen zur effizienten, datenbankbasierten Implementierung von *MADEIRA* bzw. allgemein der interaktiven Bearbeitung *großer* Datenbestände stellen wir zugunsten der Umsetzung der genannten

Aspekte „Datenanalyseunterstützung“ und „umfassende Semantikdefinition“ (zunächst einmal[3]) zurück. Mindestanforderung soll die effiziente Durchführbarkeit deskriptiver epidemiologischer Studien auf der Basis von *MADEIRA* sein.

Wie bereits im einleitenden Abschnitt 1.2 dargestellt, haben die in Auswertungen der Epidemiologie untersuchten multidimensionalen Datenräume typischerweise lediglich eine mittlere Größe, also von bis zu wenigen Millionen Zellen. Ein Datenwürfel kann somit jeweils vollständig im Hauptspeicher materialisiert und verarbeitet werden. Konkret betrachtete Datenwürfel[4] sind i. a. (nahezu) voll besetzt, haben also nur wenige Nullwerte, was auch für eine konsequent multidimensionale Modellierung (im Gegensatz zu ROLAP–Ansätzen) spricht. Weiterhin kann davon ausgegangen werden, daß auch die Kategorienhierarchien aller Dimensionen während der Durchführung einer Datenanalyse im Hauptspeicher repräsentiert werden können. Somit muß — neben dem einfachen Zugriff auf in der Datenbank abgelegte Basisdaten — lediglich eine *hauptspeicherbasierte* explorative, also in Einzelschritte aufgelöste Datenanalyse effizient durch *MADEIRA* unterstützt werden können. Dies rechtfertigt die Vernachlässigung von Aspekten der effizienten Datenverwaltung.

Insgesamt definiert die Größe der zugrundeliegenden Datenbasis bzw. der jeweils untersuchten Datenräume die einzige wesentliche Eingrenzung möglicher Anwendungsdomänen von *MADEIRA*. Genauer gesagt ist *MADEIRA* zwar konzeptionell sowie im Hinblick auf die Menge unterstützter Operationen auch zur Auswertung sehr großer multidimensionaler Datenräume auf nahezu beliebigen Gebieten der Manipulation multidimensionaler, aggregierter Daten (etwa auch im Data–Warehouse–Bereich) geeignet. Nur wären vor einem entsprechenden Einsatz noch weitergehende Überlegungen zur effizienten Realisierung der Modellkonzepte anzustellen.[5] Dies beträfe vor allem die dynamische Verwaltung großer Kategorienhierarchien sowie den jeweiligen initialen Zugriff auf die Basisdaten, der die interaktiv zu analysierenden Teildatenräume bereitstellt. Zumindest bei „geschickter“ Auswahl sind diese — analog zur Epidemiologie — selten so groß, daß sie anschließend nicht hauptspeicherbasiert verarbeitet werden könnten.

Ausdrücklich *nicht* Gegenstand von *MADEIRA* ist die Integration *heterogener* Datenbestände. Es wird davon ausgegangen, daß alle Basisdaten konsistent und frei von widersprüchlichen Angaben sind und durch eine einheitliche Menge von Kategorien und Maßzahlen beschrieben werden können. Dies ist jeweils im Vorfeld durch eine umfassende Qualitätssicherung zu gewährleisten (vgl. [JJQV99, Hin99]). Zumindest sollten mögliche Inkonsistenzen bekannt sein und über geeignete Metadaten explizit modelliert werden, so daß garantiert werden kann, daß entsprechende Datenräume nicht miteinander verknüpft werden. Auch Datenschutzaspekte werden nur insofern berücksichtigt, als sie sich durch die unterschiedliche Genauigkeit von Angaben im Rahmen einer Kategorienhierarchie oder durch Nullwerte modellieren lassen. Statistische Verfahren zur Anonymisierung von Mikro- und Makrodaten werden nicht betrachtet. Weiterhin werden auch keine Überlegungen zum Maintenance aggregierter Daten, also zur Wahrung der Konsistenz zu den jeweiligen Mikrodatenbeständen, oder zur Wartung von Datenbanken, etwa zur Änderung oder Versionierung von Dimensionen und Kategorienhierarchien, angestellt. *MADEIRA* konzentriert sich allein auf die Analyse eines statischen Datenbestandes.

Als spezielle Modellierungsziele sind weiterhin folgende Aspekte zu nennen:

- Die zentrale Operation zur Navigation in den Datenbeständen ist die Aggregierung. Zu ihrer bestmöglichen Unterstützung ist eine explizite und klare semantische Modellierung

[3] In Abschnitt 6.2 kommen wir auf Möglichkeiten zur optimierten Umsetzung der Datenverarbeitung in *VIOLA*, also der Speicherung von *MADEIRA*–Datenstrukturen und der Ausführung von *MADEIRA*–Operationen, noch zu sprechen.

[4] Die Basisdatenbestände spannen natürlich mehrdimensionale Datenräume auf, die deutlich mehr Zellen enthalten als etwa einzelne Erkrankungsfälle in der Datenbank gespeichert sind — nur wird in relevanten Untersuchungen jeweils lediglich ein kleiner Teil der betreffenden Dimensionen selektiert und über alle anderen vollständig aggregiert.

[5] In diesem Zusammenhang spielt sicher auch die allgemein anerkannte Bevorzugung einer relationalen gegenüber einer multidimensionalen physischen Repräsentation großer Datenräume eine Rolle (vgl. Abschnitt 2.3.4).

  - von Aggregierungsebenen und -hierarchien kategorieller Attribute (in Grundzügen ähnlich wie bei [CT98]),
  - der Disjunktheit und Vollständigkeit, also einer Semantik von Kategorien, sowie
  - jeweils zu nutzender Aggregierungsfunktionen

  erforderlich. Dies bildet auch die Voraussetzung für die problemlose Integration von unterschiedlich granularen Datenbeständen aus verschiedenen Datenquellen, wie sie häufig in der Epidemiologie vorzunehmen ist. Die Modellierung soll unabhängig von den verwalteten Makrodatenbeständen (statisch) als Teil des Datenschemas erfolgen und auch nicht auf relational repräsentierte Hilfsinformationen, etwa zur dynamischen Spezifikation der Aggregierungsbeziehungen, zurückgreifen.

- Weiterhin müssen auch die jeweils betrachteten Objektmengen, deren Eigenschaften sowie die Rollen, in denen Objekte jeweils auftreten (über welche Populationen sollen Aussagen gemacht werden, welche liefern lediglich Referenzwerte etc.), in die Modellierung einbezogen werden, um exakte Operationsdefinitionen zu ermöglichen. Es ist also ein ausdrücklicher Bezug zu den jeweils beschriebenen Mikrodaten herzustellen.

- Auch die Untersuchung mengenwertiger Eigenschaften soll (wenn auch in begrenztem Umfang) vorgesehen werden. Die meisten bestehenden Modelle schließen diese entweder vereinfachend und in Ermangelung einer genauen Aggregierungs–Semantik aus oder aber behandeln sie — im Rahmen sehr flexibler Kategorienhierarchien und Aggregierungen — nicht gesondert, was zu unklaren oder sogar irreführenden Analyseergebnissen führen kann. In vielen Anwendungsgebieten von OLAP spielen mengenwertige Eigenschaften auch keine große Rolle, gerade aber in der Krebsepidemiologie existieren verschiedene mengenwertige Beziehungen, etwa zwischen Patienten, Tumoren, Behandlungen und Meldungen von Fällen, die in dieser Form zu berücksichtigen sind.

- Die Möglichkeit, mehrere Maßzahlen pro Datenwürfel zu führen, soll zum einen die semantische Gruppierung von Daten erleichtern und zum anderen auch die Grundlage für eine klare Definition der Verarbeitung von Datenwürfeln liefern: Erst werden Datenwürfel in ihrer Dimensionalität „passend“ gemacht und dann zu einem „komplexen“ Datenwürfel integriert, auf dem anschließend lokale Operationen ausgeführt werden können.

- Neben der Aggregierung sind weitere grundlegende, anwendungsunabhängige Operationen des Datenmanagements (wie Restriktion oder Vereinigung) zur Navigation durch mehrdimensionale Datenräume zu modellieren. Datenanalysefunktionen sollen nicht direkt Teil der in *MADEIRA* definierten Algebra sein, vielmehr spezifizieren Maßzahlen jeweils mögliche Wege zu ihrer eigenen Berechnung, auf die dann in einer allgemein definierten „Maßzahlberechnungsfunktion“ in flexibler und leicht erweiterbarer Weise zurückgegriffen werden kann. Die eindeutige und korrekte Maßzahlberechnung ist durch die umfassende Beschreibung von (Quell–)Datenräumen durch summarische und kategorielle Attribute zu ermöglichen. Eine große Rolle — auch zur adäquaten Information des Benutzers — spielen hierbei vielfältige Metadaten, die vor allem Typ und Entstehung der betrachteten Daten spezifizieren.

- In der Analyseumgebung werden durch Anwendung statistischer Funktionen Datenwürfel mit komplexen Maßzahlen erzeugt, die nicht mehr ohne Rückgriff auf die Basisdaten weiter zu aggregieren, geschweige denn zu disaggregieren sind. Um trotzdem ein effizientes Navigieren durch derartige Datenräume zu ermöglichen, aber auch im Sinne einer hohen Flexibilität der Auswertung sollen auch unterschiedlich granulare Daten in einem Datenwürfel kombiniert zu materialisieren sein. Hierauf können dann unmittelbar die gewünschten Roll–up– und Drill–down–Operationen durchgeführt werden. Weiterhin gestattet dies die Einbindung und Verarbeitung unterschiedlich detaillierter Basisdaten.

- Dem datenflußorientierten Prinzip entsprechend, wird im Kern eine Algebra von Basisoperatoren auf Datenräumen, kein deklaratives Kalkül benötigt. Darauf aufbauend ist natürlich auch eine deklarative Anfragespezifikation denkbar.

Einige weitere spezielle Anforderungen an *MADEIRA*, insbesondere motiviert anhand von Beispielen aus dem Anwendungsgebiet der Epidemiologie, werden noch im Laufe des Kapitels anhand von Beispielen vorgestellt. Insgesamt soll ein Kompromiß gefunden werden zwischen der Beschränkung auf möglichst einfache Datenstrukturen, die eine intuitive Systembedienung ermöglichen, und der Gewährleistung der jeweils nötigen Flexibilität und Ausdrucksmächtigkeit. Teilweise mögen die nachfolgenden Definitionen evtl. etwas komplex und aufwendig wirken — sie beschreiben jedoch lediglich genau das, was der Datenanalyst sonst *intuitiv* nutzt und tut, was jedoch in anderen Datenmodellen häufig nicht explizit formuliert wird, so daß Analyseergebnisse nicht immer klar und eindeutig zu interpretieren sind.

In die Konzeption von *MADEIRA* fließen Ideen aus verschiedenen der vorgestellten Datenmodelle sowie grundlegende Aspekte der relationalen und objektorientierten Modellierung ein. Vom Grundprinzip her, vor allem in der induktiven Komposition aller Entitätstypen aus Primitiven mit klar definierter Semantik, ist *MADEIRA* jedoch ein neuartiger Ansatz zur Modellierung multidimensionaler Daten.

In den Definitionen dieses Kapitels bezeichnen

- Kleinbuchstaben $(a, \ldots)$ jeweils *einzelne*, Großbuchstaben $(A, \ldots)$ *Mengen* von Entitäten sowie fettgedruckte Großbuchstaben $(\mathbf{A}, \ldots)$ *Mengen von Mengen*,
- Angaben der Form $a.a_i$ die Komponente a_i einer Struktur $a = (a_1, a_2, \ldots, a_n), i = 1, \ldots, n$,
- $\mathbb{N}$ die Menge der (positiven) natürlichen Zahlen mit $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, $\mathbb{Z}$ die Menge der ganzen und $\mathbb{R}$ die der reellen Zahlen,
- $\mathbb{B}$ die Menge der booleschen Werte *true* und *false*,
- 2^M die Potenzmenge einer Menge M, $\mathbb{N}^M$ die Menge aller Multimengen[6] über M sowie $M^* = \bigcup_{i \in \mathbb{N}_0} M^i$ die Menge aller geordneten endlichen Tupel mit Elementen aus M und $M^+ = \bigcup_{i \in \mathbb{N}} M^i$ die Menge M^* ohne das nullelementige Tupel,
- $\mathcal{F}_{A,B,\ldots}^{X,Y,\ldots}$ die Menge aller Funktionen $f : A \times B \times \ldots \to X \times Y \times \ldots$ sowie
- Punkte „$\cdot$“ als Funktionsargumente jeweils beliebige bzw. nicht näher spezifizierte Werte (z. B. ist „$f(\cdot) = g(a, \cdot)$“ je nach Kontext die Kurzschreibweise für „$\forall b\colon f(b) = g(a, b)$“ oder für „$\exists b\colon f(b) = g(a, b)$“).

Soweit nicht anders angegeben, stammen Indizes (etwa i in a_i) jeweils aus $\mathbb{N}_0$. Bei der Wahl von Funktionsnamen wurde folgende Regel zugrunde gelegt: Funktionen, die Berechnungsvorschriften spezifizieren, werden über einen kurzen Identifikator „name“ mit f^{name} bezeichnet, z. B. f^{aggr} für eine Aggregierungsfunktion. Die gleiche Gestalt haben Funktionsnamen, die andere, als statisch anzusehende Beziehungen beschreiben, etwa die Extension (die Menge der Zellwerte) eines Datenraums f^{ext}. Dagegen wird für dynamische Zuweisungen frei definierter Metadaten, wie z. B. beschreibender Texte zu einem Datenraum, ein beliebiger Funktionsbezeichner verwendet (etwa *description*()). In Einzelfällen wird von dieser Konvention abgewichen, wenn es der Verdeutlichung bestimmter Sachverhalte und Zusammengehörigkeiten dient.

Auf die Vergabe von Objektbezeichnern oder -identifikatoren wird in diesem Kapitel jeweils verzichtet. Diesbezügliche Überlegungen werden jedoch noch kurz in Abschnitt 4.4.1 diskutiert. Weiterhin wird in Strukturen $a = (a_1, \ldots, a_n)$ formal nicht unterschieden, ob eine Komponente a_i als Teil einer Komposition („a besteht aus $a_1, \ldots, a_n$“) oder nur als Referenz auf eine Entität („a hat einen Verweis auf bzw. eine bestimmte Beziehung zu $a_1, \ldots, a_n$“) anzusehen ist. Somit sind prinzipiell auch rekursive Definitionen möglich. Die jeweilige Semantik ist im Einzelfall dem Kontext zu entnehmen.

Im Laufe dieses Kapitels werden einige polymorphe Operatoren definiert. In den Definitionen wird der jeweils betroffene Wertebereich durch eine entsprechende Indizierung in Hochstellung kenntlich gemacht (etwa „$\equiv^{\text{K}}$“ für die Äquivalenz von Kategorien K). Diese wird jedoch, soweit der Bezug aus dem Zusammenhang jeweils klar ist, im Anschluß daran meist weggelassen.

[6] Eine Multimenge ist eine „Menge“, in der Elemente mehrfach vorkommen können. Elemente von Multimengen $N \in \mathbb{N}^M$ über einer Menge M werden als Tupel (m, n) mit $m \in M$ und $n \in \mathbb{N}_0$ geschrieben, wobei n die Anzahl der Vorkommen von m in N angibt, d. h. $n = |\{x \in N \mid x = m\}|$. Anstelle von $(m, 1)$ wird auch einfach m geschrieben; ein Tupel $(m, 0)$ symbolisiert $m \notin N$.

4.2 Die Entitätstypen im *MADEIRA*–Modell

Die Basis des hier vorgestellten Datenmodells bildet die objekt–orientierte Modellierung der Anwendungswelt: *Objektmengen* werden durch *Eigenschaften* beschrieben, deren Ausprägungen *Kategorien* genannt werden. Objekte können in unterschiedlichen *Rollen* betrachtet werden; Wertebereiche von Eigenschaften, also Mengen „gleichartiger" Kategorien werden als *Domänen* bezeichnet. Grundideen zu dieser Art der Betrachtung sowie Definitionen der genannten Begriffe führt Abschnitt 4.2.1 ein. Auf dieser Grundlage werden in Abschnitt 4.2.2 (relationale) *Mikrodaten*, die durch *relationale Attribute* beschrieben werden, modelliert.

Kategorien können zu *Aggregierungsebenen* zusammengefaßt werden, die wiederum die Bausteine für die Definition von *Kategorienhierarchien* liefern. Der Begriff der *Dimension* wird in dieser Arbeit weitgehend synonym zur Kategorienhierarchie verwendet. Die eindeutige Dimension zu einer Domäne kapselt eine ausgezeichnete Kategorienhierarchie, stellt somit einen für die Anwendung interessierenden Ausschnitt der Domäne sowie die Beziehungen der jeweiligen Kategorien untereinander zusammen und bildet zugleich die Grundlage für die Unterscheidung *ein-* und *mengenwertiger* Eigenschaften (Abschnitt 4.2.3).

Abschnitt 4.2.4 stellt einige spezielle Eigenschaften bzw. Varianten von Dimensionen und Aggregierungshierarchien beispielhaft vor, wobei insbesondere δ–*Kategorien* betrachtet werden, die Angaben der Form „*Kategorie k, ohne nähere Angabe*" umschreiben.

Bevor in Abschnitt 4.2.6 *Makrodaten* in Gestalt von *Datenräumen* als zentrale Datenstruktur von *MADEIRA* sowie verschiedene Sichten auf diese eingeführt werden, modelliert Abschnitt 4.2.5 *kategorielle* und *summarische Attribute* als deren Komponenten (analog zu den relationalen Attributen von Mikrodaten). Ähnlich wie kategorielle Attribute auf der Grundlage von Domänen und Dimensionen definiert sind, stützen sich summarische Attribute auf die Definition von *Maßzahlen* über *Datentypen*, denen spezifische *Aggregierungsfunktionen* zur Zusammenfassung von Maßzahlwerten auf einer jeweils gröberen Granularitätsstufe zugeordnet sind, ab.

Abschließend werden — vor dem Hintergrund der Unterscheidung zwischen einer relational basierten und der hier verfolgten streng multidimensionalen Modellierung von Makrodaten — in Abschnitt 4.2.7 die Verwendung von Tupeln von Kategorien verschiedener Dimensionen zur Beschreibung *zusammengesetzter* kategorieller Attribute — quasi ein Mittelweg beider Betrachtungsweisen — sowie in Abschnitt 4.2.8 die Abgrenzung von Datenschema und Datenbankextension diskutiert.

Ein einfaches Beispiel aus der Krebsregistrierung mag zur Verdeutlichung obiger Begrifflichkeiten dienen: Ein typischer Makrodatenraum enthält z. B. ein summarisches Attribut *Fallzahl*; die Datenwerte sind gemäß der kategoriellen Attribute *Untersuchungsgebiet* und *Untersuchungszeit* klassifiziert. Das Fallzahl–Attribut beschreibt die Maßzahl *Anzahl* mit $\mathbb{N}_0$ als Datentyp und der *Summation* als Aggregierungsfunktion über alle Eigenschaften. In diesem einfachen Fall beziehen sich alle Attribute des Datenraums sowie der Datenraum selbst lediglich auf die Objektmenge *Personen* in der Rolle *Studienpopulation*. Die beiden kategoriellen Attribute enthalten Mengen von *Landkreisen* und *Jahresangaben* als Kategorien aus der Domäne *Gebiet* bzw. *Zeit* zur Beschreibung der Personen–Eigenschaften *Wohnort zum Diagnosezeitpunkt* und *Zeitpunkt der Diagnose*. Auf diesen Domänen sind als Aggregierungsebenen *Gemeinden*, *Landkreise* und *Bundesländer* bzw. *Monate*, *Jahre* und *Jahrzehnte* definiert und jeweils durch eine entsprechende, kanonisch definierte Kategorienhierarchie in einer Dimension repräsentiert. Als Datenbasis zur Berechnung des Fallzahldatenraums kann z. B. ein Mikrodatenraum *Falldatenbasis* gedient haben, der in relationaler Form Angaben zu Einzelfällen mit den relationalen Attributen *Adresse*, *Erkrankungsdatum* etc. macht.

Im Laufe des Kapitels werden wir dieses Beispiel im Rahmen der Betrachtung einzelner Entitätstypen von *MADEIRA* noch vertiefen sowie anhand ähnlicher Szenarien weitere Teilaspekte der Datenmodellierung diskutieren. Eine umfassende Darstellung der Nutzung von *MADEIRA* in der Krebsepidemiologie findet sich in Abschnitt 6.3.

Abbildung 4.1 gibt in einem *UML*–Klassendiagramm (vgl. [FS98]) in Form eines „Meta"–Schemas von *MADEIRA* einen Überblick über die in diesem Abschnitt eingeführten Strukturen, ihre gegenseitigen Abhängigkeiten sowie die entsprechenden Definitionen. Hiermit soll ein erster Eindruck von der Art der Modellierung multidimensionaler Strukturen in *MADEIRA*, von der Beschreibung von Mikro- und Makrodaten durch unter-

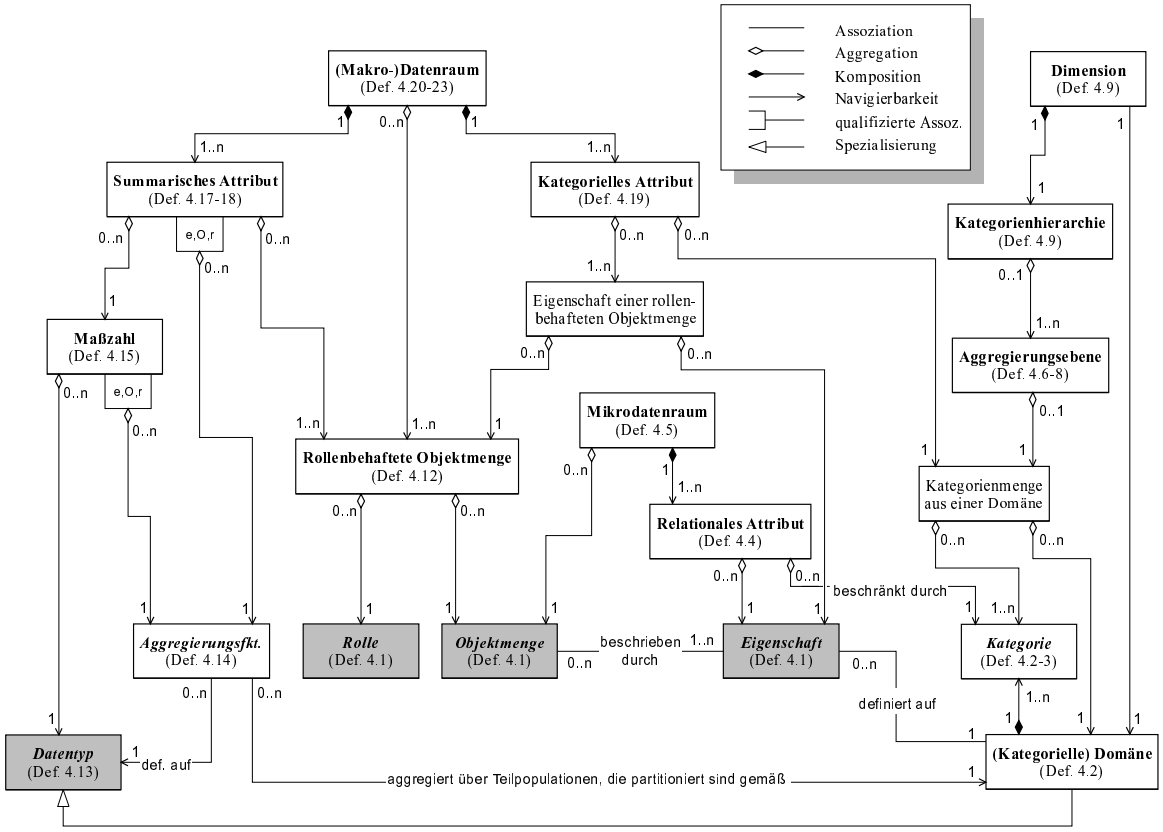

Abbildung 4.1: *UML*–Klassendiagramm der in *MADEIRA* definierten Entitätstypen

schiedlich geartete Attribute und vom gewählten Ansatz zur metadatenbasierten Definition von Datensemantik vermittelt werden.

Es fällt auf, daß Dimensionen und Aggregierungsebenen nicht in die Definition anderer Typen eingehen. Dies liegt daran, daß sie lediglich inhärente Beziehungen zwischen Kategorien für einen bestimmten, anwendungsrelevanten Teil einer Domäne explizit modellieren und so dem Benutzer von *MADEIRA* übersichtlich zugänglich machen. Wir werden hierauf im Laufe des Abschnitts noch zurückkommen.

Beziehungen zwischen Entitätsmengen werden in Abb. 4.1 unterschieden nach (exklusiver) Komposition, (nicht exklusiver) Aggregation von Teilkomponenten zu einem Ganzen und einfacher Assoziation nicht einander unterzuordnender Entitäten. Die einzige Generalisierungsbeziehung findet sich zwischen Domänen und Datentypen.[7] Maßzahlen und summarischen Attributen sind über qualifizierte Beziehungen Aggregierungsfunktionen zugeordnet, d. h. zu jeder Kombination von Eigenschaft *e*, Objektmenge *O* und Rolle *r* wird jeweils genau eine Aggregierungsfunktion referenziert. Kursive Namen repräsentieren als atomar angesehene Einheiten. Grau hinterlegt sind für *MADEIRA* charakteristische Metadaten, die jeweils als Beschreibung anderer, ihnen zugeordneter Entitäten anzusehen sind und eine wesentliche Erweiterung gegenüber existierenden Datenmodellen darstellen. Eventuelle Beschriftungen von Diagrammkanten konkretisieren Beziehungen und sind jeweils von links nach rechts bzw. von oben nach unten zu lesen.

Nicht dargestellt ist aus Gründen der Übersichtlichkeit die Definition eigenschaftsspezifischer Maßzahlen: Zu jeder Kombination von Objektmenge und hierauf definierter Eigenschaft läßt sich eine Maßzahl definieren, die die generell als Kategorien modellierten Eigenschaftsausprägungen als Maßzahlwerte auffaßt (Def. 4.16)—

[7] Um den diskreten, *kategoriellen* Charakter von Domänen zu betonen, sind diese hier als eigener Entitätstyp angeführt.

diese Möglichkeit ist jedoch auch nicht von sehr großer Bedeutung. Weiterhin fehlen die weniger zentralen, lediglich andere Entitätstypen spezialisierenden Definitionen von ein- und mengenwertigen Eigenschaften (Def. 4.10), δ–Kategorien (Def. 4.11) und speziellen Sichten auf Datenräume im Rahmen der Modellierung von Kennzahldimensionen (Def. 4.24–4.25) sowie die Betrachtung zusammengesetzter kategorieller Attribute, Eigenschaften und Domänen (Def. 4.26–4.28). Zudem sind einzelnen Entitäten zugeordnete Funktionen nur dann als eigenständige Strukturen repräsentiert, wenn sie — wie im Fall der Aggregierungsfunktionen — in *MADEIRA* speziell definiert werden.

Schließlich kommt das *UML*–Diagramm auch ohne die Angabe von (Objekt–)Attributen (entsprechend Instanzvariablen) zu Entitätsmengen aus, da deren wesentliche konzeptionelle Charakteristika bereits durch die untereinander bestehenden Beziehungen spezifiziert sind. In Abschnitt 4.4 kommen wir jedoch in diesem Sinne noch auf entsprechende Metadaten (vor allem zu Maßzahlen und Datenräumen) zu sprechen. Diese „Attribute“ zielen vor allem auf Implementierungsaspekte sowie in Gestalt zusätzlicher Strukturierungs- und Beschreibungsmittel auf die komfortable Datennutzung durch den Endanwender.

Abbildung 4.1 skizziert als Schema der Entitätstypen eines Datenmodells lediglich die Metadaten zur Beschreibung der auf der Basis von *MADEIRA* modellierten und verwalteten Daten, nicht aber die konkrete Gestalt der Datenraumextensionen: Makrodatenräume enthalten in den Zellen eines multidimensionalen Feldes Werte aus den Wertebereichen, die durch die Datentypen der jeweiligen Maßzahlen bzw. summarischen Attribute festgelegt sind; die Extension eines Mikrodatenraums dagegen hat die Form einer Relation auf den Domänen, die den jeweiligen relationalen Attributen über Eigenschaften bzw. beschränkende Kategorien zugeordnet sind.

Bezüglich der Beziehungen zwischen den in Abb. 4.1 dargestellten Entitätsmengen bestehen eine Reihe einschränkender Constraints, die im *UML*–Diagramm nicht darstellbar sind:

- Die Menge aller Kategorien wird durch Domänen partitioniert.
- Eine Kategorienmenge aus einer Domäne umfaßt (ihrem Namen gemäß) nur Kategorien *einer* Domäne.
- Alle Aggregierungsebenen einer Kategorienhierarchie beinhalten nur Kategorien derjenigen Domäne, auf der die jeweilige Dimension zur Hierarchie definiert ist.
- Durch ein relationales Attribut beschriebene Eigenschaft und beschränkende Kategorie beziehen sich auf die gleiche Domäne.
- Die Eigenschaften zu den relationalen Attributen eines Mikrodatenraums sind paarweise verschieden und beschreiben auch die Objektmengen, die dem Mikrodatenraum zugrunde liegen.
- Eine Eigenschaft einer rollenbehafteten Objektmenge beschreibt stets die jeweilige Objektmenge.
- Alle einem kategoriellen Attribut zugeordneten Eigenschaften beziehen sich auf die vom Attribut referenzierte Domäne.
- Aggregierungsfunktionen, die einer Maßzahl oder einem summarischen Attribut bzgl. einer Eigenschaft zugeordnet sind, aggregieren gerade über die Domäne, auf der auch die Eigenschaft definiert ist.
- Alle kategoriellen Attribute eines Makrodatenraums beschreiben paarweise disjunkte Mengen von Eigenschaften zu rollenbehafteten Objektmengen.
- Die Menge der rollenbehafteten Objektmengen eines Makrodatenraums ist gleich der Vereinigung der rollenbehafteten Objektmengen ihrer summarischen und auch der ihrer kategoriellen Attribute.
- Die Aggregierungsfunktionen eines summarischen Attributs ergeben sich unter Berücksichtigung von Charakteristika des jeweiligen Makrodatenraums aus denen der Maßzahl, die das Attribut beschreibt. (Der genaue Zusammenhang ist zu komplex, um ihn an dieser Stelle exakt zu spezifizieren.)

Diese Aufstellung soll zunächst einmal lediglich zur besseren Einordnung der vorgestellten Konzepte dienen. Detaillierte Motivationen und Formalisierungen der Bedingungen folgen jeweils im Rahmen der Einzeldefinitionen zur Konkretisierung und genauen Spezifikation der in Abb. 4.1 dargestellten Entitätsmengen im Laufe dieses Abschnitts.

4.2.1 Beschreibung von Objekteigenschaften durch Kategorien

Um später die Modellierung von Mikro- und vor allem Makrodaten formalisieren zu können, soll zunächst einmal der Bezug zu den jeweils repräsentierten Entitäten des Anwendungsgebiets hergestellt werden. Deren Beschreibung bzw. die Angabe von Ausprägungen ihrer Merkmale bildet die Grundlage für die Definition von zunächst Mikro- und aufbauend darauf dann auch von Makrodaten. Wir folgen einem objektorientierten Ansatz:

Definition 4.1 (Objektmengen, Eigenschaften und Rollen) *Es sei O eine nicht–leere Menge von Objekten, die im jeweiligen Diskursbereich betrachtet werden sollen und die durch* EIGENSCHAFTEN *aus einer nicht–leeren Menge $\mathcal{E}$ beschrieben werden können.*

Eine Abbildung $f^{\text{eig}}: O \to 2^{\mathcal{E}}$ ordne einem Objekt o jeweils die Menge seiner Eigenschaften zu. $(f^{\text{eig}})^{-1}(e)$, also die Menge der durch eine Eigenschaft $e \in \mathcal{E}$ beschriebenen Objekte, werde auch als O_e bezeichnet.

Eine OBJEKTMENGE *$O \subseteq O$ stehe im folgenden — als „Klasse“ im Sinne objektorientierter Modellierung — stets für eine Teilmenge des „Objektuniversums“ O, die sich durch eine Menge gemeinsamer Eigenschaften (den gleichen „Typ“) auszeichnet, also $\exists \emptyset \neq E \subseteq \mathcal{E}: O = \{o \in O \mid E \subseteq f^{\text{eig}}(o)\}$.*

In Datenbeständen bzw. Auswertungen betrachtete konkrete Teilmengen („Instanzen“) von O werden jeweils durch $\text{ext}(O) \subseteq O$ repräsentiert.

$\mathcal{R}$ bezeichne eine Menge von ROLLEN*, die Objekte oder Objektmengen bzw. die diese beschreibenden Datensätze im Rahmen einer Analyse annehmen können.*

Beispiele für typische in dieser Arbeit betrachtete Objektmengen bilden *Tumoren* oder *Personen*, die etwa bei der Berechnung standardisierter Erkrankungsraten in der Rolle einer *Studien-* oder *Standardpopulation* auftreten können. Eigenschaften dieser Objektmengen wären etwa *Alter*, *Wohnort* oder *Diagnose*.

Ausprägungen von Eigenschaften werden durch *Kategorien*[8] spezifiziert. Diese spielen eine zentrale Rolle in *MADEIRA*. Aus diesem Grund werden hier zunächst einige grundlegende Überlegungen zu ihrer Nutzung angestellt. Kategorien sollen nicht nur zur exakten Festlegung eines Wertes einer Objekteigenschaft („Person x wohnt in *Oldenburg*“) dienen, sondern auch eine Differenzierung der Genauigkeit einer Angabe erlauben:

- Kategorien können eine Eigenschaft unterschiedlich exakt spezifizieren („Person x wohnt in der *kreisfreien Stadt Oldenburg*“ oder „ ... im *Regierungsbezirk Weser–Ems*“).
- Sie können fehlendes genaueres Wissen signalisieren („ ... in *Oldenburg, ich weiß aber nicht, in welchem Ortsteil*“ im Gegensatz zu „ ... in *Oldenburg, genauer interessiert es im Moment nicht*“).
- Im Fall von mengenwertigen Eigenschaften können weitere Ausprägungen ausgeschlossen werden („Person x wohnte im Laufe seines Lebens *nur* in *Hamburg und Oldenburg*“ gegenüber „ ... *unter anderem* ... “).

Die im letzten Punkt angesprochene Differenzierung zwischen ein- und mengenwertigen Eigenschaften ist nicht so klar, wie es auf den ersten Blick scheint. Die hier vorgenommene Definition von Kategorien erlaubt verschiedene Sichtweisen auf einen Sachverhalt und basiert insbesondere nicht auf einer ausgezeichneten „feinsten Einheit“ möglicher Ausprägungen. Folgende Beispiele verdeutlichen dies:

- Das Geschlecht von Zwittern kann durch die Menge der Ausprägungen *männlich* und *weiblich*, aber auch durch eine eigene Ausprägung *Zwitter* beschrieben werden. Allgemein kann jede Menge von Kategorien durch Vergabe einer eigenen „Identifikation“ wieder als „atomare“ Kategorie gesehen werden, wodurch aus einer mengenwertigen Eigenschaft eine einwertige wird.
- Umgekehrt können auch einwertige Eigenschaften als mengenwertige codiert werden, z. B. können Altersintervalle jeweils als ein- oder zweielementige Mengen von einseitig unendlichen Altersbereichen der Form „älter (oder jünger) als x“ als Basiseinheiten spezifiziert werden.

[8] Diese Bezeichnung wurde im Hinblick auf die üblicherweise endliche, diskrete Partitionierung von Eigenschaftsausprägungen zur Beschreibung von Makrodaten gewählt, auch wenn die hier eingeführten Wertebereiche durchaus unendliche Mengen von Kategorien umfassen können.

- Die Beschreibung von Erkrankungen auf einer Betrachtungsebene kann durchaus auf der konjunktiven („mengenwertigen“) Zusammenfassung von Einzelerkrankungen auf einer anderen Ebene basieren.

Da es im Kontext einer Anwendung jedoch sicherlich wünschenswert ist, insbesondere im Hinblick auf die korrekte Aggregierung von Daten mengenwertige Attribute speziell zu behandeln, werden in Abschnitt 4.2.3 aufgrund einer jeweils in Form ausgewählter Kategorien spezifizierten Betrachtungsweise zwei Arten von Eigenschaften ($\mathcal{E}_O^\sigma$, bzgl. einer Objektmenge O einwertige, und $\mathcal{E}_O^\mu$, mengenwertige) unterschieden werden.

Im Sinne obiger Überlegungen können Kategorien k als

- Aussagen ($k \Leftrightarrow$ *„Es gilt* $\langle \ldots \rangle$ *(für eine Objekteigenschaft).“*) oder
- zweistellige Prädikate $k(o,e)$ über O und $\mathcal{E}$ ($k(o,e) \Leftrightarrow$ *„o erfüllt bzgl. e* $\langle \ldots \rangle$*.“*)

betrachtet werden. Diese aussagen- bzw. prädikatenlogische Betrachtungsweise liegt den folgenden Definitionen zugrunde, wobei je nachdem, ob der Bezug auf konkrete Objekte o oder Eigenschaften e relevant ist, mal der einen und mal der anderen Sicht der Vorzug gegeben wird.

Definition 4.2 (Kategorien und Domänen) *$\mathcal{K}$ sei eine nicht–leere Menge von* KATEGORIEN *zur Repräsentation von Aussagen über Ausprägungen von Eigenschaften betrachteter Objekte.*

$\mathcal{K}$ sei partitioniert in eine Menge nicht–leerer (KATEGORIELLER) DOMÄNEN $\mathcal{DO} \subset 2^{\mathcal{K}}$*, so daß Ausprägungen einer Eigenschaft $e \in \mathcal{E}$ jeweils durch Kategorien einer eindeutig bestimmten Domäne $D_e \in \mathcal{DO}$ beschrieben werden. Umgekehrt bezeichne $\mathcal{E}_D \subseteq \mathcal{E}$ die Menge aller Eigenschaften, denen die Domäne $D \in \mathcal{DO}$ zugeordnet ist. Die Domäne D, aus der eine Kategorie k stammt ($k \in D$), schreiben wir auch als $D = D_k$.*

Zu einer Eigenschaft $e \in \mathcal{E}$ repräsentiere die Relation „$\vdash_e$“ die Zuordnung von Eigenschaftsausprägungen zu Objekten. Gilt für ein Objekt $o \in O$ und eine Kategorie $k \in D_e$ „$o \vdash_e k$“, so sagen wir auch „k ist o (bzgl. e) ZUGEORDNET*“, „o* ERFÜLLT *k“ oder „o* FÄLLT IN KATEGORIE *k“.*

Schließlich bezeichne (für $e \in \mathcal{E}$ und $k \in D_e$) noch $O_{e,k} \subseteq O_e$ die Menge $\{o \in O \mid o \vdash_e k\}$ sowie $\mathcal{K}^\emptyset$ die Menge aller Kategorien, die nie erfüllt werden, d. h. $\mathcal{K}^\emptyset \stackrel{\text{def}}{=} \{k \in \mathcal{K} \mid \forall e \in \mathcal{E} : (k \in D_e \Rightarrow O_{e,k} = \emptyset)\}$.

Die Domäne *Gebiete* (mit Elementen wie *Deutschland* oder *Oldenburg*) oder die *Zeit*domäne (mit den Zeitkategorien *1998*, *März 1999*, *31.3.2000* etc.) sind Beispiele für Domänen aus $\mathcal{DO}$. Die Gebietsdomäne beschreibt z. B. Eigenschaften wie *aktueller Wohnort*, *Ort der Arbeitsstätte*, *bisherige Wohnorte* etc. Lebt eine Person x in Bremen, so schreibt man dies etwa als „$x \vdash_{\textit{Wohnort}} \textit{Bremen}$“.

Vor allem in Hinblick auf die Modellierung von Kategorienhierarchien und die Aggregierung von Makrodaten von einer feineren zu einer gröberen Aggregierungsebene sind die entsprechenden Beziehungen zwischen Kategorien einer Domäne sowie deren Verknüpfung von besonderem Interesse. Dies wird im folgenden formalisiert:

Definition 4.3 (Prädikate und Verknüpfungen auf Kategorien) *Für $k_1, k_2 \in D$ bezeichne „$k_1 \equiv^{\mathrm{K}} k_2$“ die* ÄQUIVALENZ *dieser Kategorien, also $(k_1 \equiv k_2) \Leftrightarrow (\forall e \in \mathcal{E}_D : O_{e,k_1} = O_{e,k_2})$.*

Eine Kategorie k' SUBSUMIERE *eine Kategorie k („k ist feiner als k'“), geschrieben „$k \preceq^{\mathrm{K}} k'$“, wenn $\forall e \in \mathcal{E}_D : O_{e,k} \subseteq O_{e,k'}$.*

Analog zu Äquivalenz und Subsumtion seien DISJUNKTION *(„$k \equiv k_1 \vee k_2$“ genau dann, wenn $\forall e \in \mathcal{E}_D : O_{e,k} = O_{e,k_1} \cup O_{e,k_2}$),* KONJUNKTION *(„$k_1 \wedge k_2$“) und* NEGATION *(„$\neg k$“) von Kategorien intuitiv definiert.*[9]

Eine Domäne D sei stets abgeschlossen gegenüber den genannten logischen Operationen, d. h.

$$\forall k \in D\ \exists k' \in D : \neg k \equiv k' \text{ sowie } \forall k_1, k_2 \in D\ \exists k', k'' \in D : (k_1 \wedge k_2 \equiv k') \wedge (k_1 \vee k_2 \equiv k'') \quad .$$

Ferner wollen wir auch davon ausgehen, daß jede Domäne „neutrale“ Elemente bzgl. der Disjunktion (aus $\mathcal{K}^\emptyset$) sowie bzgl. der Konjunktion (also Kategorien, die von allen Objekten mit entsprechenden Eigenschaften erfüllt werden[10]*) enthält.*

[9] Gelegentlich werden wir auch die Präfixnotationen $\bigvee K$ und $\bigwedge K$ mit $K \subseteq D, D \in \mathcal{DO}$ verwenden.

[10] Solche Kategorien werden wir innerhalb von Kategorienhierarchien (Def. 4.9) als *Wurzelkategorien* k_0 bezeichnen.

Zwei Kategorien k_1 und k_2 sollen VERWANDT *heißen („$k_1 || k_2$“), falls $(k_2 \preceq k_1) \vee (k_1 \preceq k_2)$, also eine lediglich eine Verfeinerung (oder gleich) der anderen ist. k_1 und k_2 gelten bzgl. einer Eigenschaft e und einer Objektmenge $O \subseteq O_e$ als* DISJUNKT *(„$k_1 \perp_{e,O} k_2$“), falls $O \cap O_{e,k_1 \wedge k_2} = \emptyset$. Die Negationen $\not\equiv, \not\preceq, \not\parallel, \not\perp$ seien kanonisch definiert.*

Schließlich gebe es eine partielle Abbildung $f^{\text{val}} : O \times \mathcal{E} \to \mathcal{K}$, die zu $e \in \mathcal{E}$ und $o \in O_e$ genau dann definiert ist, wenn $e \in f^{\text{eig}}(o)$, und die jeweils die „genaueste“ Angabe zur Ausprägung von e für O spezifiziert[11]*, d. h. es gelte stets*

$$(o \vdash_e f^{\text{val}}(o, e)) \wedge \forall k \in \mathcal{D}_e : (o \vdash_e k \Rightarrow f^{\text{val}}(o, e) \preceq k) \quad .$$

Die Subsumtion auf (oder Granularität von) Kategorien bezieht sich hier stets auf *alle* Eigenschaften aus $\mathcal{E}$, um so allgemeingültige „feiner–gröber–Beziehungen“ von Kategorien zu erfassen. Oftmals ist es sicher so, daß sich aus dem Vorhandensein einer derartigen Beziehung bei einer einzigen Eigenschaft bereits auf die allgemeine Subsumtion zweier Kategorien schließen läßt: Ist etwa eine Ortsangabe für den Wohnort „feiner“ als eine andere, so gilt dies auch für die Angabe der Arbeitsstätte. Dies muß jedoch im allgemeinen Fall nicht so sein. Zum einen seien Kategorien genannt, bei denen für einzelne Eigenschaften $O_{e,k} = \emptyset$ gilt; zum anderen modellieren auch mengenwertige Eigenschaften, wie sie bereits diskutiert wurden, keine repräsentativen Verfeinerungsbeziehungen im obigen Sinne. Vielmehr ergeben sich derartige Abhängigkeiten ($o \vdash_e k \Rightarrow o \vdash_e k'$) auch, wenn bestimmte Kategorien stets gemeinsam mit anderen in mengenwertigen Ausprägungen auftreten (etwa in der Form „der *Autopsiebefund* in der Menge von Meldungen zu einem Mortalitätsfall bedingt den *Totenschein* (als weitere Meldung)“).

Entsprechend ist es wenig sinnvoll, Disjunktheit von Kategorien unabhängig von einer konkreten Eigenschaft zu definieren, da im Falle (auf der jeweiligen Betrachtungsebene) „mengenwertiger“ Eigenschaften i. a. jede Kategorie mit (fast) jeder anderen kombinierbar ist. In der Domäne *Gebiet* gilt z. B. *Berlin* $\preceq$ *Deutschland*; die Angaben sind somit bzgl. keiner Eigenschaft disjunkt, *Bonn* und *Berlin* als Angabe des Hauptwohnsitzes einer Person schon, aber wiederum nicht als Beschreibung der *Menge* der Etappen einer Reise (weil bestimmte Personen evtl. auf einer Reise beide Orte besucht haben). Ein Beispiel für zwei nicht verwandte, nicht disjunkte Gebiete wären schließlich etwa *Norddeutschland* und das *Gebiet der neuen Bundesländer*.

Während der Begriff der Subsumtion aus einem aussagenlogischen Blickwinkel auf die Klärung der konzeptionellen Beziehungen zwischen Kategorien abzielt, folgt der Begriff der Disjunktheit einer prädikatenlogischen Betrachtungsweise von Kategorien. Er dient zum einen der Erkennung von „Überlappungen“ von Kategorien, die durch unterschiedliche disjunktive Verknüpfungen „feinerer“ Kategorien entstehen und zum anderen — darauf aufbauend — zur Vorbereitung der Unterscheidung ein- und mengenwertiger Eigenschaften im folgenden Abschnitt. So erklärt sich auch der Bezug auf eine Objektmenge in der Definition von Disjunktheit. Zum Beispiel können die Kategorien *männlich* und *weiblich* zur Beschreibung von Menschen in den meisten Anwendungsgebieten als disjunkt modelliert werden, bei der (zusätzlichen) Betrachtung bestimmter Tier- oder Pflanzenarten jedoch liegt hier eine mengenwertige Eigenschaft mit nicht disjunkten Kategorien vor. Eine Differenzierung des Disjunktheitsbegriffs bietet also eine größere Flexibilität — gerade im Hinblick auf die korrekte Aggregierung von Makrodaten, die in vielen Fällen zwischen ein- und mengenwertigen Eigenschaften unterscheiden muß.

4.2.2 Modellierung von Mikrodaten

Wie bereits in Abschnitt 2.3.2 eingeführt, werden in dieser Arbeit unter Mikrodaten Mengen von Angaben zu *einzelnen* Objekten (Ereignissen, Sachverhalten etc.) einer Teilmenge von O verstanden. Sie haben somit typischerweise relationale Gestalt: Eine Menge von Mikrodaten ist eine Relation über Wertebereichen von (relationalen) Attributen, die zur Beschreibung der jeweiligen Objekte bzw. ihrer Eigenschaften dienen.

[11] Diese Abbildung soll den jeweiligen Wissensstand über die betrachteten Objekte widerspiegeln — eine Sichtweise, die in Abschnitt 4.2.4 im Kontext der Betrachtung unspezifischer Angaben noch einmal aufgegriffen wird. Ihr Funktionswert ergibt sich als Konjunktion über alle Kategorien, die einem Objekt bzgl. einer Eigenschaft jeweils zugeordnet werden. f^{val} wird bei der Spezifikation von Mikrodatenräumen (Def. 4.5) zur Gewährleistung einer konsistenten, eindeutig definierten Datenbasis benötigt werden.

Definition 4.4 (Relationale Attribute) *Ein* RELATIONALES ATTRIBUT *a zur Beschreibung einer Eigenschaft $e \in \mathcal{E}$ sei definiert durch $a = (e, k)$, wobei $k \in D_e$ den Wertebereich des Attributs vorgebe (nämlich D_e, eingeschränkt auf von k subsumierte Kategorien).*

$\mathcal{A}$ sei die Menge aller relationalen Attribute.

Definition 4.5 (Mikrodatenräume) *Ein* MIKRODATENRAUM $dr^{\mathrm{mi}} = (O, A, f^{\mathrm{ext}})$ *sei definiert durch*

- *eine nicht–leere Menge $O \subseteq \mathcal{O}$ als durch dr^{mi} beschriebene Objektmenge,*
- *eine endliche, nicht–leere Menge von relationalen Attributen $A = \{a_1, \ldots, a_m\} \subseteq \mathcal{A}$, die sich auf paarweise unterschiedliche Eigenschaften beziehen (also $a_i \neq a_j \in A \Rightarrow a_i.e \neq a_j.e$), und*
- *eine Familie totaler Funktionen $f^{\mathrm{ext}} = (f_a^{\mathrm{ext}})_{a \in A}$ zur Spezifikation der* EXTENSION *von dr^{mi}.*

Letztere ergebe sich für jedes Attribut $a \in A$ durch die Zuordnung entsprechender Eigenschaftsausprägungen zu einer endlichen Menge $\mathrm{ext}(O) \subseteq O$ von Objekten mittels $f_a^{\mathrm{ext}} : \mathrm{ext}(O) \to D_{a.e}$. Hierbei gelte

$$\forall o \in \mathrm{ext}(O)\ \forall a \in A : (a.e \in f^{\mathrm{eig}}(o)) \wedge (f_a^{\mathrm{ext}}(o) = f^{\mathrm{val}}(o, a.e)) \wedge (f_a^{\mathrm{ext}}(o) \preceq a.k) \quad ,$$

die Mikrodaten enthalten also die „feinstmöglichen" Angaben zu den beschriebenen Objekten, wobei alle Werte aus den durch die Attribute spezifizierten Wertebereichen stammen.

Elemente der Extension (bzw. Tupel der durch $\{(o, f_{a_1}^{\mathrm{ext}}(o), \ldots, f_{a_m}^{\mathrm{ext}}(o)) \mid o \in \mathrm{ext}(O)\}$ gegebenen Relation) werden als MIKRODATEN *bezeichnet.*

$\mathcal{DR}^{\mathrm{mi}}$ sei die Menge aller Mikrodatenräume.

Mikrodaten sind somit etwa der (relational modellierbare) Tumoren- oder Patientenbestand eines Krebsregisters.

Die Forderung, daß alle Attribute eines Mikrodatenraums *unterschiedliche* Eigenschaften von $\mathrm{ext}(O)$ beschreiben, wird einige nachfolgende Betrachtungen vereinfachen und ist sicher auch intuitiv sinnvoll. Man beachte, daß Mikrodaten stets eindeutig die genauesten im Rahmen der Anwendung bekannten Angaben zu Objekten machen. Im weiteren Verlauf wird dies von entscheidender Bedeutung für die exakte Beschreibung von Makrodaten und Operationen auf diesen sein. In Abschnitt 4.2.4 werden wir Kategorien kennenlernen, mit denen auf dieser Basis auch Nullwerte in Mikrodaten modelliert werden können. Die Wertebereiche der Attribute eines Mikrodatenraums dienen lediglich dazu, explizit auszudrücken, daß der Mikrodatenraum Angaben über *genau* diejenigen Objekte, die in diese Bereiche fallen, macht. Dies wird eine Rolle für die Spezifikation von Nullwerten bei der Aggregierung aus Mikrodaten spielen (vgl. Abschnitt 4.3.1).

Die hier vorgenommene Definition bietet große Freiheiten im Hinblick auf die flexible Verwendung von Attributen eines Mikrodatensatzes als summarische oder kategorielle Attribute bei der Aggregierung zu Makrodaten. Attribute können z. B. auch mengenwertige Eigenschaften (mit entsprechenden „mengenwertigen" Kategorien) beschreiben oder rein „numerische" Werte aus einer entsprechenden Domäne, die nur zur Berechnung summarischer Attribute, nicht aber zur Klassifikation in kategoriellen Attributen verwendet werden sollen. Somit können etwa auch die natürlichen oder sogar die reellen Zahlen (evtl. mit darauf definierten Intervallen) zur Definition einer Domäne herangezogen werden.

Da sich diese Arbeit im Kern mit der Verarbeitung von Makrodaten beschäftigt und auch die Erzeugung von Makro- aus Mikrodaten nur eine untergeordnete Rolle spielen wird, soll die etwas knappe Diskussion von Mikrodaten an dieser Stelle ausreichend sein. Insbesondere soll der multidimensionale Charakter der hieraus zu gewinnenden Makrodaten im Vordergrund stehen und nicht — wie in vielen neueren der in Abschnitt 2.3.3 vorgestellten Datenmodelle — eine starke Anlehnung an das relationale Datenmodell bzw. die relationale Speicherung und Verknüpfung von Makrodaten erfolgen.

Die später in Abschnitt 4.2.6 eingeführten (Makro–)Datenräume kapseln Mikrodaten unter einer multidimensionalen Sicht auf daraus aggregierte Makrodaten. Ein Zugriff auf die jeweils hinter einer Zelle eines multidimensionalen Datenbestandes liegenden Mikrodaten sollte möglich sein, eine eventuelle weitere Verarbeitung dieser Daten wird jedoch nicht modelliert (vgl. Abschnitt 4.3.6).

4.2.3 Anwendungsspezifische Einschränkung von Domänen durch Dimensionen

Dimensionen beschreiben Kriterien zur Gruppierung von Mikrodaten über ausgewählte Attribute. Kategorienhierarchien definieren hierzu verschieden feine Granularitäten der Datenaggregierung, die durch Zusammenfassung oder Aufteilung von Kategorien zur Beschreibung der jeweiligen Attribute auseinander abgeleitet werden können. Es werden also Äquivalenzen zwischen Kategorien und Gruppen anderer Kategorien für einen bestimmten, in der jeweiligen Anwendung interessierenden Teilbereich einer Domäne explizit modelliert.

In Anlehnung an [CT98] soll hier insbesondere eine eigenständige Modellierung von unabhängigen Aggregierungsebenen mit Kategorien, die typischerweise gemeinsam in einer Datenanalyse verwendet werden, erfolgen.

Definition 4.6 (Aggregierungsebenen) *Eine* AGGREGIERUNGSEBENE *(auch* EBENE *oder* LEVEL*)* $le = (D, K)$ *auf einer Domäne* $D \in \mathcal{DO}$ *umfasse eine endliche, nicht–leere Menge von Kategorien* $K \subseteq D$*.*

Eine Aggregierungsebene le werde als (bzgl. einer Eigenschaft $e \in \mathcal{E}_D$ *und einer Objektmenge* $O \subseteq O_e$*)* EINWERTIG *bezeichnet, falls alle Kategorien aus* $le.K$ *bzgl. e und O paarweise disjunkt sind. Andernfalls heiße die Ebene* MENGENWERTIG*.*

$\mathcal{L}$ *sei die Menge aller Aggregierungsebenen,* $\mathcal{L}^{\sigma}_{e,O} \subseteq \mathcal{L}$ *die aller einwertigen und* $\mathcal{L}^{\mu}_{e,O} \subseteq \mathcal{L}$ *die aller mengenwertigen Ebenen (bzgl. e und O) sowie* $\mathcal{L}_D \subseteq \mathcal{L}$ *die Menge aller Ebenen, die auf der Domäne D definiert sind.*

Falls $\forall k_1 \neq k_2 \in le.K : k_1 \nmid k_2$*, also keine Kategorie aus le eine andere der gleichen Ebene subsumiert, heiße le auch* WOHLGEFORMT*. Einelementige Aggregierungsebenen (*$le.K = \{k\}$*) sollen nur wohlgeformt heißen, falls k auch von mindestens einem Objekt in einer Eigenschaft angenommen wird, also* $k \notin \mathcal{K}^{\emptyset}$*.*[12]

$\mathcal{L}^{\omega} \subseteq \mathcal{L}$ *sei die Menge aller wohlgeformten Aggregierungsebenen.*

Einfache Beispiele für Aggregierungsebenen sind etwa *alle Regierungsbezirke Deutschlands* in der Domäne der Gebiete oder *5–Jahres–Altersgruppen* in der Altersdomäne. Aber auch die Altersgruppen „*unter 1*“, „*1–4*“, „*5–9*“, ..., „*70–74*“, „*75 und älter*“ könnten eine andere Altersebene bilden.

Die meisten existierenden Datenmodelle, die eine derartig explizite Spezifikation von Aggregierungsebenen vornehmen, gehen davon aus, daß die Kategorien einer Ebene die „gleiche Granularität“ aufweisen. In *MADEIRA* werden jedoch keine derartigen Forderungen an Aggregierungsebenen gestellt, denn wenn man hierdurch auch viele typische Anwendungsfälle abdeckt (siehe die ersten beiden obigen Beispiele), so ist diese Sichtweise doch zu restriktiv, wie das dritte Beispiel zeigt, und nicht klar definiert. Häufig sind Kategorien nämlich auch gleichzeitig *verschiedenen* Aggregierungsebenen einer Domäne zugeordnet, z. B. die *kreisfreien Städte* sowohl der Landkreis- als auch der Gemeindeebene in der Gebietsdomäne.

Die Wohlgeformtheit von Aggregierungsebenen entspricht demgegenüber lediglich dem Gedanken, daß gerade durch eine (Multi–)Hierarchie von Aggregierungsebenen die Subsumtionsbeziehungen zwischen Kategorien *verschiedener* Ebenen repräsentiert werden sollen. Somit ist es selten sinnvoll, bereits innerhalb *einer* Ebene verwandte Kategorien zuzulassen. Die für typische Anwendungsfälle relevanten Aggregierungsebenen sind i. a. auch wohlgeformt — einwertige Ebenen dürfen offenbar lediglich keine Kategorie aus $\mathcal{K}^{\emptyset}$ enthalten. Trotzdem liegt der Begriff der Wohlgeformtheit keinen nachfolgenden Definitionen zwingend zugrunde, er soll vielmehr lediglich helfen, Gestalt und sinnvolle Verwendung von Aggregierungsebenen besser zu verstehen.

Die einzige sinnvoll anzuwendende Ausnahme nicht–wohlgeformter Ebenen wird im Rahmen der Definition zusammengesetzter Kategorien im Abschnitt 4.2.7 motiviert werden. Auch die Verwendung mengenwertiger Ebenen ist in der Regel auf Einzelfälle beschränkt. Insgesamt bleibt festzuhalten, daß typischerweise genutzte Ebenen auf einer Domäne bei weitem nicht das ganze durch das kartesische Produkt der Ebenenkomponenten gegebene Spektrum möglicher Ebenen umfassen, sondern aus einer relativ kleinen Menge *semantisch sinnvoller* Ebenen stammen.

[12] Neutrale Elemente bzgl. der Disjunktion von Kategorien werden stets von allen anderen Kategorien einer Domäne subsumiert und dürfen somit in keiner wohlgeformten Ebene enthalten sein.

Eine naive Ordnung[13] auf Aggregierungsebenen ließe sich relativ einfach aus der Subsumtion von Kategorien ableiten: Werden alle Kategorien einer Ebene *le* durch jeweils mindestens eine Kategorie einer Ebene le' subsumiert, so subsumiere le' auch *le*. Eine derartige Definition wäre jedoch gerade im Hinblick auf die Navigation in Makrodaten wenig hilfreich: Eine Ordnung auf Ebenen sollte mögliche Aggregierungen von Daten feinerer Granularität auf einer gröberen Aggregierungsebene modellieren. Hierzu ist es insbesondere nötig, daß gröbere Kategorien im Sinne folgender Definition jeweils *vollständig* in Gruppen feinerer Kategorien „zerlegt“[14] werden (vgl. auch [Sat81]):

Definition 4.7 (Ordnung auf Aggregierungsebenen) *„$\preceq^{\mathrm{L}}$“ definiere eine (partielle) Ordnung auf Aggregierungsebenen der jeweils gleichen Domäne, die durch die Subsumtion auf den Kategorien der jeweiligen Ebenen in folgender Weise induziert werde:*

Für $le, le' \in \mathcal{L}_D$ mit $D \in \mathcal{DO}$ gelte „$le \preceq^{\mathrm{L}} le'$“ („$le$ VERFEINERT *le'“) genau dann, wenn le alle Kategorien aus le' vollständig unterteilt: $\forall k' \in le'.K : k' \equiv \bigvee \{k \in le.K \mid k \preceq k'\}$.*[15]

Hierbei lassen sich zwei Varianten (vgl. Abb. 4.2) unterscheiden, je nachdem, ob

(a) alle Kategorien der Unterebene einer Kategorie der Oberebene zugeordnet sind, also $\bigvee le.K \equiv \bigvee le'.K$ (die Ebenen sind quasi „äquivalent“), oder

(b) die Oberebene nur durch einen Teil der Unterebene verfeinert wird — letztere deckt also einen größeren Bereich ab.

So wie es gemäß dieser Definition möglich ist, daß Teile einer Aggregierungsebene keinen Beitrag zur Verfeinerung einer anderen leisten (so verfeinert etwa die Ebene aller Gemeinden Deutschlands die der Landkreise Niedersachsens), so möchte man mitunter auch die Aufteilung eines *Ausschnitts* einer Ebene durch eine andere Ebene modellieren. Hierbei wollen wir uns — ähnlich der in [LAW98] vorgeschlagenen Normalform für multidimensionale Datenbestände — der Einfachheit halber auf Fälle beschränken, in denen genau eine Kategorie näher differenziert wird (vgl. Abb. 4.2). Folgende Definition faßt diese Möglichkeit mit der oben eingeführten Verfeinerung ganzer Ebenen zum Begriff der Aggregierbarkeit von Ebenen zusammen.

Definition 4.8 (Aggregierbarkeit von Aggregierungsebenen) *Seien $le, le' \in \mathcal{L}_D$ zwei Aggregierungsebenen zur Domäne D. Dann heiße le* AGGREGIERBAR *zu le' (geschrieben „$le \lhd le'$“), falls le ganz le' oder nur genau eine Kategorie $k' \in le'$ (vollständig) verfeinert, also*

$$le \lhd le' \Leftrightarrow (le \preceq le') \vee (\exists k' \in le' : k' \equiv \bigvee le.K) \quad .$$

k' definiert in letzterem Fall sozusagen den „Kontext“ für le: Ganz le wird durch k' subsumiert.

Die Relation „$\lhd$“ bildet im Gegensatz zu „$\preceq^{L}$“ keine (insbesondere transitive) Ordnung mehr auf den Ebenen einer Domäne. Zu beachten ist, daß der Begriff der „Aggregierbarkeit“ von Ebenen sich im Fall (b) aus Def. 4.7 strenggenommen nur auf diejenigen Kategorien aus der feineren Ebene *le* bezieht, die auch tatsächlich zu Kategorien der gröberen Ebene le' zusammengefaßt werden. Wir wollen trotzdem von Aggregierbarkeit der ganzen Ebene sprechen.

Zu einer im Anwendungsgebiet zur Klassifikation von Makrodaten genutzten Domäne *D* gebe es genau *eine* Dimension, die die für die Anwendung jeweils interessierende Menge von Aggregierungsebenen (also einen Ausschnitt von *D*) zu einer Kategorienhierarchie zusammenfaßt. Die Dimension dient damit als Grundlage derjenigen Aggregierungen von Mikrodaten über auf *D* basierende Attribute, die für die jeweilige Anwendung möglicherweise relevant sind.[16] Insbesondere schränkt eine Dimension die jeweilige Domäne auf eine endliche Teilmenge von Kategorien ein.

[13] Soweit nicht anders vermerkt, ist im folgenden stets von einer Ordnung im mathematischen Sinne die Rede, also einer transitiven, reflexiven, antisymmetrischen Relation.

[14] Im Falle mengenwertiger Ebenen ist diese Klassifikation i. a. natürlich nicht disjunkt.

[15] Natürlich kann eine Kategorie auch durch sich selbst verfeinert sein.

[16] Da die Menge der Dimensionen den für eine Anwendung interessierenden Weltausschnitt vollständig durch die jeweils enthaltenen Kategorienmengen umspannt, wird im folgenden auch meist anstelle von Domänen auf die jeweiligen Dimensionen Bezug genommen.

Definition 4.9 (Dimensionen und Kategorienhierarchien) *Eine* DIMENSION $d = (D, L, le_0)$ *sei definiert durch*

- *die zugehörige Domäne* $D \in \mathcal{DO}$,
- *eine endliche, nicht–leere Menge* $L \subseteq \mathcal{L}_D$ *von Aggregierungsebenen und*
- *eine* WURZELEBENE $le_0 \in L$ *mit* $le_0.K = \{k_0\}$, *für die gilt* $\forall le \in L\ \forall k \in le.K : k \preceq k_0$ *(k_0 entspricht also einer Ausprägung „Alle“ und heiße auch* WURZELKATEGORIE*).*

Als KATEGORIENHIERARCHIE *einer Dimension soll die auf L beschränkte Relation* $\lhd$ *bezeichnet werden.*

Auf einer Domäne gebe es jeweils nur genau eine Dimension. $\mathcal{DI}$ *sei die Menge aller Dimensionen,* $d_D \in \mathcal{DI}$ *die Dimension zu einer Domäne* $D \in \mathcal{DO}$ *(mit* $d_D.D = D$*) und* $d_e \in \mathcal{DI}$ *die Dimension zur Domäne einer Eigenschaft* $e \in \mathcal{E}$ *(mit* $d_e = d_{D_e}$*).*

Häufig (aber nicht immer) ist eine Dimension durch mehrere Zusammenfassungen über einer feinsten Ebene definiert, z. B. das Alter durch eine Einteilung le_3 in Jahre, eine Gruppierung le_2 in 5–, eine weitere le_1 in 10–Jahresgruppen und eine Wurzelebene le_0 mit $le_3 \preceq le_2 \preceq le_1 \preceq le_0$. Abbildung 4.2 zeigt ein Beispiel für eine einfache Gebietshierarchie in Form eines Graphen. Derartige Darstellungen der Kategorienhierarchie einer Dimension d zeigen jeweils nur diejenigen Beziehungen $le \lhd le'$, für die kein $le'' \in d.L$ mit $le \lhd le'' \lhd le'$ existiert. Typischerweise, aber nicht zwangsläufig, ergibt sich ein zusammenhängender Graph. Sofern interessant, sind Subsumtionsbeziehungen zwischen einzelnen Kategorien eingeblendet.

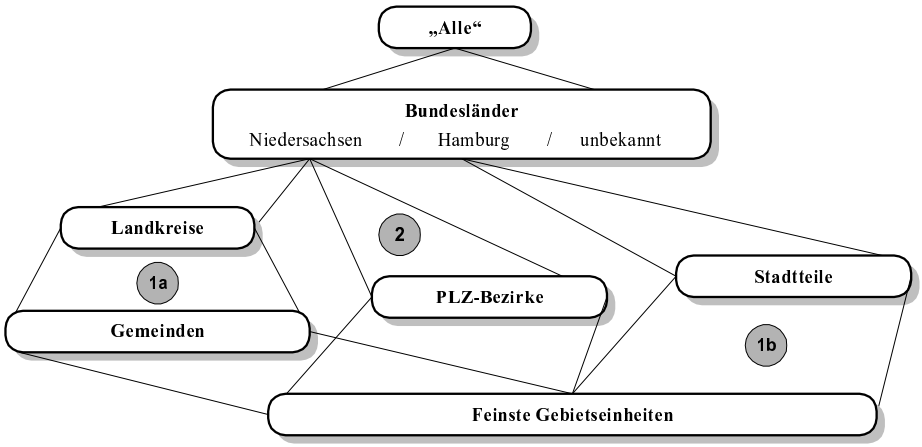

Abbildung 4.2: Beispiel einer Kategorienhierarchie

Analog zur Kategorie „*unbekannt*“ in der Ebene der Regierungsbezirke können auch auf anderen Ebenen Kategorien der Form „*x, ohne nähere Angabe*“ zur Gewährleistung der geforderten vollständigen Verfeinerung einer Kategorie *x* dienen — hierauf wird in Abschnitt 4.2.4 noch näher eingegangen. Die Varianten der Verfeinerung von Ebenen aus Def. 4.7 (Fälle (1a) und (1b)) bzw. einzelner Kategorien (Fall (2)) gemäß Def. 4.8 sind beispielhaft eingezeichnet.

Der Fall (2) entspricht etwa dem Konzept der *Features* aus [Leh98]. Gegenüber der dort vorgeschlagenen Nutzung von den Kategorien zugeordneten Dimensionsattributen[17] wird jedoch hier für die Verfeinerung *einzelner* Kategorien durch spezielle, nur „lokal“ definierte Kriterien kein neues Konzept eingeführt, das auch spezielle Operatoren zur Behandlung erfordern würde. Vielmehr wird diese Variante zugunsten von Einfachheit und intuitiver Bedienbarkeit der Analyseumgebung in eine homogene Definition von Kategorienhierarchien integriert. Diese Fokussierung ist für die meisten Anwendungsdomänen bzw. Datenbestände völlig ausreichend

[17] Diese können aber in anderen Zusammenhängen zur Repräsentation zusätzlicher Metadaten für die komfortable Datenanalyse durchaus hilfreich sein, wie wir später in Abschnitt 4.4.3 noch sehen werden.

und angemessen; so treten derartige Fälle der differenzierten Spezialisierung einzelner Kategorien auch in der Epidemiologie eher selten auf. Anders wäre die Situation evtl. bei der Modellierung sehr heterogener Produktpaletten mit einer Vielzahl jeweils unterschiedlicher Attribute in betriebswirtschaftlichen Anwendungen einzustufen, da eine *MADEIRA*–basierte Modellierung dort zu sehr komplexen und unübersichtlichen Kategorienhierarchien führen würde.

Aus ähnlichen Beweggründen und im Sinne einer klaren Abgrenzung von Fall (2) verzichten wir weiterhin darauf, eine Untergliederung einer echten Teilmenge einer Ebene mit *mehr als einer* Kategorie zuzulassen. Dies ließe sich ggf. jedoch auch über die Einführung einer zusätzlichen, „künstlichen“ Aggregierungsebene, die diese Teilmenge zu einer Kategorie zusammenfaßt, umsetzen.

Auf der Basis der Definition einer Dimension, die über die durch sie ausgewählten Kategorien eine bestimmte Betrachtungsweise auf die zugrundeliegende Domäne festlegt, ist es nun möglich und sinnvoll, ein- und mengenwertige Eigenschaften zu definieren.

Definition 4.10 (Ein- und mengenwertige Eigenschaften) *Eine Eigenschaft $e \in \mathcal{E}$ heiße (bzgl. einer Objektmenge $O \subseteq O_e$)* EINWERTIG, *falls* $\forall le \in d_e.L : le \in \mathcal{L}^{\sigma}_{e,O}$ *(geschrieben $e \in \mathcal{E}^{\sigma}_{O}$); andernfalls heiße sie* MENGENWERTIG *($e \in \mathcal{E}^{\mu}_{O}$).*

Anhand der Gebietsdomäne in Abb. 4.2 kann man sich schnell vergegenwärtigen, daß diese Definition der intuitiven Vorstellung von Ein- und Mengenwertigkeit nicht widerspricht: Zu einer einwertigen Eigenschaft, wie z. B. dem *aktuellen Wohnort*, ist jeder Person maximal eine Kategorie einer beliebigen Ebene zugeordnet — betrachtet man dagegen etwa die Menge aller Wohnorte im Laufe des Lebens einer Person, so können für diese auch mehrere Kategorien einer Ebene gelten. Allgemeiner dargestellt: Kategorien, die Mengen (also Konjunktionen) von „Basiskategorien“ repräsentieren, sind untereinander für gewöhnlich nicht „disjunkt“ (d. h. sie können gleichzeitig erfüllt sein), und somit ist jede Ebene, die mehrere dieser Kategorien enthält, mengenwertig.

Am Rande sei angemerkt, daß auch zur Beschreibung mengenwertiger Eigenschaften einwertige Aggregierungsebenen existieren können, deren Kategorien die Semantik „*exakt* Menge *x*“ haben. In Abschnitt 4.2.4 werden diesbezügliche Modellierungsmöglichkeiten noch genauer diskutiert.

Folgende zwei hinreichende, aber nicht notwendige Bedingungen für die Mengenwertigkeit einer Aggregierungsebene sind leicht herzuleiten: Eine Ebene *le* aus der Domäne *D* ist mengenwertig (bzgl. *e* und *O*), falls gilt:

1. $\exists k_1 \neq k_2 \in le.K\ \exists k \in D \setminus \mathcal{K}^{\emptyset} : (k \preceq k_1) \wedge (k \preceq k_2)$ oder
2. $\exists le' \in \mathcal{L}^{\mu}_{e,O} : (le' \lhd le) \wedge (\bigvee le'.K \preceq \bigvee le.K)$ (Fall (1b) in Abb. 4.2 also ausgeschlossen).

Abschließend wird die Aufgabe von Kategorienhierarchien noch einmal im Kontext ihrer Verwendung in einer Analyseumgebung beleuchtet. Sie dienen im wesentlichen drei Zwecken:

1. der Vorgabe typischer Navigationspfade in Datenräumen bzgl. eines kategoriellen Attributs,
2. der direkten Bereitstellung von Kategoriengruppen, die *vollständig* zu einer anderen Kategorie aggregiert werden, und
3. (für einwertige Eigenschaften bzw. Ebenen) der Sicherstellung der Disjunktheit von Kategorien.

Sie erleichtern somit die sinnvolle und korrekte Verarbeitung von Makrodaten, wobei viele Überlegungen zur Aggregierbarkeit (vgl. Abschnitt 2.4.2) durch die Interpretation von Kategorien als Aussagen über Eigenschaften stark vereinfacht werden. Aus Gründen der Flexibilität bilden Aggregierungsebenen bzw. statisch definierte Kategorienhierarchien jedoch nicht die zwingende Grundlage entsprechender Aggregierungsoperationen (siehe Abschnitt 4.3.2). Andernfalls könnte der Anwender beispielsweise keine ad–hoc definierten Gruppierungen von Kategorien (z. B. aller Gemeinden um eine Emissionsquelle) bilden und auch nicht Daten unterschiedlicher Granularität in einem Datenraum zusammenstellen.

4.2.4 Charakteristika und Arten von Dimensionen

Bereits in Abschnitt 4.2.1 wurden mögliche Arten von Kategorien diskutiert, die über die Beschreibung einer „einfachen“ Ausprägung einer Eigenschaft hinausgehen. Im Kontext der Einführung von Kategorienhierarchien sowie der Diskussion ein- und mengenwertiger Eigenschaften in Abschnitt 4.2.3 wurden weitere besondere Merkmale von Kategorien bzw. Aggregierungsebenen angesprochen.

Aufgrund der herausragenden Bedeutung, die Kategorienhierarchien als Basis für die zentrale Operation auf multidimensionalen Daten, die Aggregierung, haben, werden einige dieser Aspekte im folgenden genauer beleuchtet. Im einzelnen betrifft dies

- die Verwendung „feinster“ Aggregierungsebenen,
- Kategorien zur Beschreibung unspezifischer Angaben sowie
- die Unterstützung mengenwertiger Eigenschaften.

Es werden jeweils typische Beispiele zur Datenmodellierung angegeben, die als Richtlinie für eine spezielle Anwendung dienen können, aber trotzdem die Freiheit für andere Umsetzungen lassen.

Feinste Aggregierungsebenen

In vielen bestehenden Ansätzen zur Modellierung von Kategorienhierarchien basieren diese stets auf einer eindeutigen *feinsten* Aggregierungsebene, deren Kategorien zu gröberen Einheiten zusammengefaßt werden. Nicht immer ist jedoch eine derartige Zusammenführung von Zweigen einer Kategorienhierarchie praktisch möglich bzw. sinnvoll. Oft kann einer derartigen Ebene keine intuitive Semantik beigemessen werden, bzw. die hierzu nötigen konjunktiven Verknüpfungen von Kategorien sind in der Anwendungswelt nicht klar definiert.

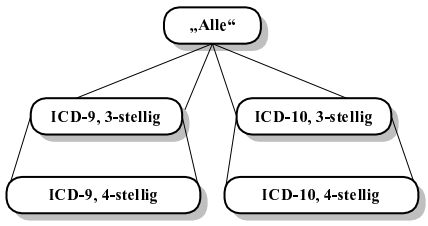

Abbildung 4.3: Mehrere „feinste“ Aggregierungsebenen

Ein Beispiel zeigt Abb. 4.3. Dort besteht eine Kategorienhierarchie zur Beschreibung von Krankheiten gemäß der ICD–Klassifikation aus zwei unabhängigen Zweigen, die jeweils eine ICD–Version auf verschiedenen Granularitätsstufen darstellen. Diese Versionen basieren auf keiner gemeinsamen feinsten Einteilung, da bei weitem nicht alle Codes einer Version in Codes der anderen eindeutig zu übersetzen sind. Ein ähnliches Beispiel wären Hierarchien, in denen spezifische Untergliederungen der Kategorien einer groben Ebene (wie etwa die Ebene der *Bundesländer* in Abb. 4.2) nicht wieder auf einer feineren gemeinsamen Ebene zusammengeführt werden können (im Beispiel war dies die Ebene der *feinsten Gebietseinheiten*).

MADEIRA stellt keinerlei Anforderungen an das Vorhandensein bzw. die Eindeutigkeit feinster Aggregierungsebenen. Nichtsdestoweniger können diese natürlich Datenverarbeitung und Navigation erleichtern sowie eine effiziente Verarbeitung unterstützen.

Kategorien zur Beschreibung unspezifischer Angaben

Die Spezifikation von feiner–gröber–Beziehungen zwischen Aggregierungsebenen (Def. 4.7 und 4.8) erfordert eine vollständige Zerlegung einer jeweils auf der gröberen Ebene betrachteten Kategorie k durch Kategorien der feineren Ebene. Diese kann in vielen Fällen jedoch nur gewährleistet werden, wenn Angaben der Form „k, ohne nähere Angabe“ als eigene, von k subsumierte Kategorie modelliert werden. Zum Beispiel werden Wohnortangaben oftmals auf unterschiedlichen Granularitätsebenen erhoben, so daß die Menge aller Personen, die etwa in einem Landkreis x wohnen, gegeben ist durch die Menge aller Personen, die einer Teilgemeinde von x zugeordnet sind, *plus* die Menge aller Personen, von denen man nur weiß, daß sie in x leben, denen aber keine konkrete Wohngemeinde zugewiesen werden konnte. Die Krankheitsklassifikation nach ICD integriert sogar Werte der Form „*Lungenkrebs, nicht näher spezifiziert*“ explizit in ihre Codierungstabelle.

Somit kann es sinnvoll sein, zu einer „gewöhnlichen“ Kategorie k eine δ–*Kategorie*[18] k^{δ} einzuführen, die die übrigen „Söhne“ von k komplettiert. Einen Sonderfall bildet die Wurzelkategorie k_0 einer Dimension. Neben einer allgemeinen δ–Kategorie k_0^{δ} mit der Semantik „unbekannt“ sollen hier oftmals weitere Arten von Nullwerten für qualifizierende Eigenschaften spezifiziert werden, die den näheren Grund für das Fehlen einer Merkmalsausprägung angeben, z. B. im Rahmen einer Erhebung „Aussage verweigert“, „weiß nicht“, „Person nicht angetroffen“ etc. Diese Kategorien (inklusive k_0^{δ}) werden auch als *NULL–Kategorien* bezeichnet.

Abbildung 4.4 zeigt ein Beispiel für die Anwendung von δ–Kategorien in einer Kategorienhierarchie für regionale Angaben. Das Bundesland Bayern zerfällt hier (der Einfachheit halber) zum einen in zwei Landkreise *KA* und *KB*, und zum anderen in zwei Postleitzahlbezirke *P1* und *P2*. Basisgebiete bilden die feinsten Einheiten, aus denen sich sowohl Landkreise als auch Postleitzahlbezirke zusammensetzen (hier *Gxy* als Teil von *Kx* und *Py*).

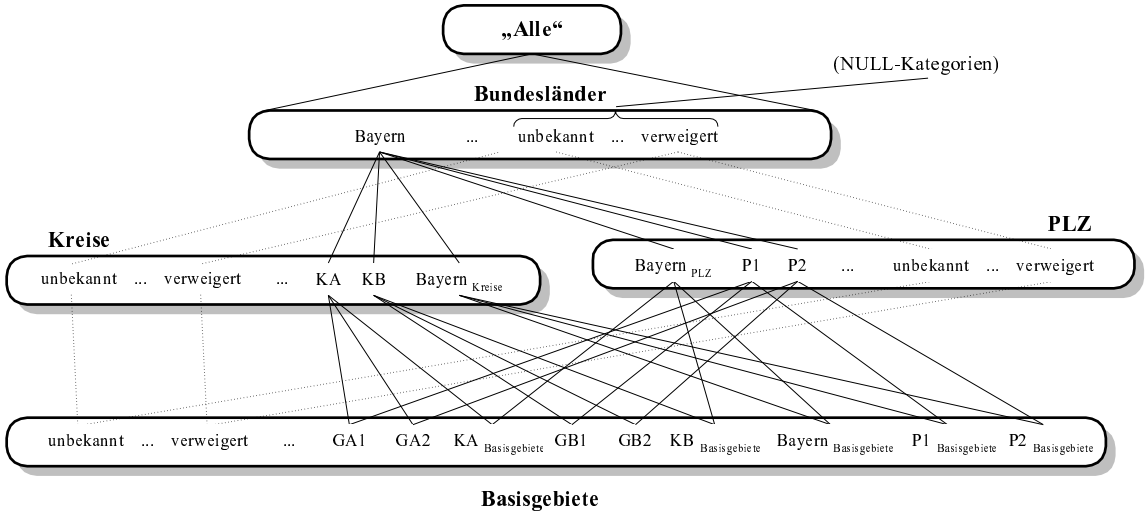

Abbildung 4.4: δ–Kategorien zur Beschreibung unspezifischer Angaben

δ–Kategorien sind jeweils als zusätzliche „Söhne“ der vervollständigten Kategorie in den entsprechenden Sohnebenen eingefügt. Man sieht, daß NULL–Kategorien — neben anderen „normalen“ Kategorien bereits der obersten Ebene unterhalb der Wurzelebene zugeordnet sind. Sie finden sich jedoch auch identisch zur Gewährleistung der jeweiligen Vollständigkeit in allen feineren Ebenen wieder. Allgemein werden auch die übrigen δ–Kategorien zwischen sich verfeinernden Ebenen dupliziert.

In dem hier dargestellten Fall nebeneinanderstehender Sohnebenen (*Landkreise* und *Postleitzahlbezirke*), die wieder auf einer gemeinsamen Basisebene (*Basisgebiete*) zusammengeführt werden, ist die Situation etwas komplizierter. Zum einen sind dem Bundesland Bayern auf allen genannten Ebenen verschiedene δ–Kategorien zugeordnet, zum anderen werden die δ–Kategorien der Kreis– und PLZ–Ebene auf der Basisebene auch noch weiter differenziert. Die genaue Semantik einer δ–Kategorie zur Kategorie k ist also im allgemeinen Fall „k, auf der Ebene ... nicht genauer spezifizierbar (evtl. aber auf ‚parallelen‘ Ebenen)“.

Folgende Definition faßt die angestellten Überlegungen zusammen:

Definition 4.11 (δ–Kategorien) *Es seien $\mathcal{K}^{\delta} \subset \mathcal{K}$ und $\mathcal{K}^{\mathrm{NULL}} \subset \mathcal{K}^{\delta}$ zwei ausgezeichnete Mengen von Kategorien zur Angabe unspezifischen Wissens, die* δ– *bzw.* NULL–KATEGORIEN.

Zu $k \in D$ ($D \in \mathcal{DO}$) und $le \in \mathcal{L}_D$ bezeichne $k_{le}^{\delta} \in \mathcal{K}^{\delta} \cap le.K$ die Kategorie mit der Semantik „k, ohne nähere Angabe in Ebene le“ bzw. (da diese Kategorien oftmals unabhängig von der Ebene identisch sind) $k^{\delta} \in \mathcal{K}^{\delta} \cap D$ allgemein die Kategorie „k, ohne nähere Angabe“. Es gelte jeweils $k_{le}^{\delta} \preceq k$ und $k^{\delta} \preceq k$.

[18] „δ“ stehe hier für „Rest“, „Sonstige“ bzw. „Differenz zum Gesamtwert“.

Als NULL–Kategorien werden δ–Kategorien zur Wurzelkategorie k_0 einer Dimension mit der Semantik „unbekannt (aus dem Grund ...)“ bezeichnet.

Bereits in dem noch einfachen Beispiel aus Abb. 4.4 zeigt sich, daß die konsequente Modellierung von δ–Kategorien eine große Anzahl zusätzlicher Kategorien produziert. Diese kann jedoch je nach Anwendungsgebiet und dem dort möglichen Auftreten unspezifischer Angaben unter Gewährleistung der vollständigen Partitionierung von Kategorien oftmals stark eingeschränkt werden. Somit legt auch obige Definition nicht exakt fest, wann und wie δ–Kategorien genutzt werden *müssen*, sondern dient lediglich dem Anwendungsentwickler zur Begriffsbildung und als Hilfestellung im Rahmen der Auswahl und Gestaltung von Kategorienhierarchien.

Weiterhin wird nochmals eine allgemeine Anforderung an Kategorien deutlich, die sich direkt aus der in *MADEIRA* verfolgten aussagen- bzw. prädikatenlogischen Modellierung von Kategorien ergibt: Die Granularität möglicher Beschreibungen von Eigenschaften durch Kategorien einer Dimension muß *exakt* möglichen Ebenen des Wissens im jeweiligen Anwendungsgebiet entsprechen, d. h. zu jeder Stufe der Genauigkeit einer Angabe muß auch eine entsprechende Kategorie existieren. Andernfalls wäre die in Def. 4.8 geforderte Vollständigkeit der Zerlegung von Kategorien in untergeordneten Aggregierungsebenen nicht gewährleistet. So dürfte es im Beispiel etwa nicht vorkommen, daß ein Objekt sowohl einem konkreten Landkreis als auch einem Postleitzahlbezirk zugeordnet werden kann, jedoch keiner Gebietseinheit der feinsten Ebene — es gäbe hier keine einzelne Kategorie, die dem Objekt zugewiesen werden könnte. Das Problem wäre nur zu lösen, indem oberhalb der Basisebene eine Ebene mit Gebietskategorien, die Schnittmengen von Regionen aus Kreisen und Postleitzahlbezirken entsprechen, eingeführt würde.

Schließlich wird auch deutlich, daß die in Def. 4.3 eingeführte Abbildung f^{val} in vielen Fällen — wenn nicht explizite Angaben auf einer nicht zu verfeinernden Ebene vorliegen — δ–Kategorien als genaueste Eigenschaftsausprägungen spezifiziert. Entsprechend enthalten auch Mikrodatenräume (vgl. Def. 4.5) häufig δ–Kategorien.

Kategorienhierarchien speziell für mengenwertige Eigenschaften

Eine Besonderheit, die *MADEIRA* gegenüber anderen multidimensionalen Datenmodellen auszeichnet, ist sicherlich auch die Einbeziehung mengenwertiger Eigenschaften. Eine typische Kategorienhierarchie speziell zur Beschreibung mengenwertiger Eigenschaften (etwa die Menge der Therapien, mit der eine Krebserkrankung therapiert wurde)[19] wird im folgenden vorgestellt.

Einer derartigen Kategorienhierarchie (vgl. Abb. 4.5) liegt meist eine Menge M von „normalen“, als einwertig betrachteten Basiskategorien zugrunde, hier einmal $M = \{A, B, C\}$ genannt. Alle möglichen Kombinationen dieser Kategorien bilden eine *Basisebene* le_{b}, deren Elemente k (*Basiskategorien*) die Semantik „*exakt* Menge $x \subseteq M$“ aufweisen, genauer $k \equiv \bigwedge\{k' \in x\} \wedge \bigwedge\{\neg k' \mid k' \in M \setminus x\}$. Insofern ist für diese Ebene „künstlich“ die Disjunktheit aller ihrer Kategorien hergestellt worden — sie ist somit eine einwertige Ebene im Sinne von Def. 4.10. Entsprechend wurden diese Kategorien in Abb. 4.5 als x^{σ} bezeichnet.

Alle anderen Ebenen sind mengenwertige Ebenen bzgl. der betrachteten Eigenschaft(en). Typischerweise enthalten diese Ebenen Kategorien k, die jeweils gleich großen Mengen $x \subseteq M$ der zugrundeliegenden einwertigen Basiskategorien entsprechen ($k \equiv \bigwedge\{k' \mid k' \in x\}$). Für alle solchen Ebenen $le^{(i)}$ (i stehe für die Größe der jeweiligen Mengen) gilt $le_{\mathrm{b}} \preceq le^{(i)}$.[20] Untereinander besteht jedoch keine Aggregierbarkeit, wenn auch die Kategorien der Ebenen mit zunehmender Mengengröße „feiner“ werden. Die Ebene $le^{(1)}$ mit den einelementigen Mengen (ergänzt um eine leere Menge) verfeinert natürlich die Wurzelebene le_0, was für die Ebenen mit mehrelementigen Mengen nicht gilt, da hier keine Vollständigkeit der Zerlegung gewährleistet ist.

Soll eine Kategorienhierarchie sowohl zur Beschreibung ein- als auch mengenwertiger Eigenschaften genutzt werden, sind entsprechende Ebenen zwischen Wurzelebene und der Ebene $le^{(1)}$ einzufügen. Natürlich

[19] Oftmals sind alle betrachteten Eigenschaften einer Domäne entweder ein- oder mengenwertig. Es kann jedoch durchaus auch sein, daß Kategorienhierarchien, die — entsprechend bereits gegebener Beispiele — auf die Modellierung einwertiger Eigenschaften abzielen, zur Beschreibung mengenwertiger Eigenschaften um einige Aggregierungsebenen erweitert werden.

[20] Diese Beziehung ist insbesondere wichtig für die disjunkte Aggregierung von entsprechend vorliegenden Basisdaten.

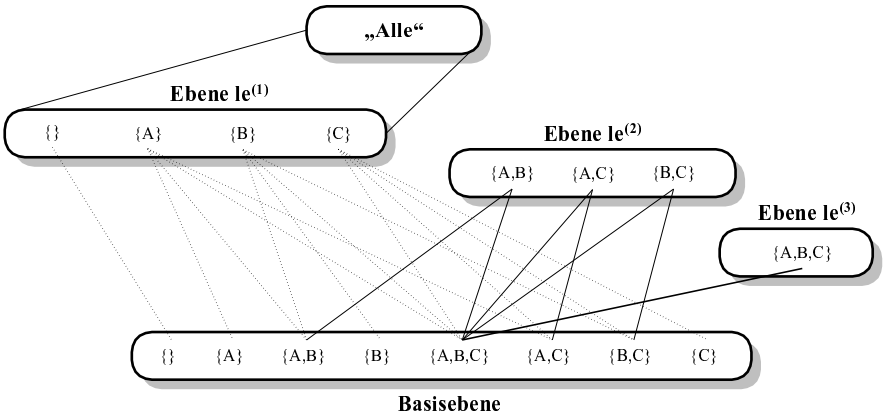

Abbildung 4.5: Eine typische Kategorienhierarchie für mengenwertige Eigenschaften

sind auch Szenarien mit unterschiedlich granularen oder sogar aus mehreren „feinsten Ebenen“ stammenden Basis–Mengenelementen sowie (falls Angaben nicht immer in der zugrundegelegten Basisgranularität vorliegen) die Einbeziehung von δ–Kategorien in die Modellierung denkbar und sinnvoll. Dies läßt sich alles aus den vorgestellten Ansätzen entwickeln und soll hier deshalb nicht weiter ausgeführt werden.

Mengenwertige Ebenen sowie auch die entsprechenden Kategorienhierarchien insgesamt, die gemäß der hier beschriebenen Vorgehensweise vollständig konstruiert werden, können natürlich mit zunehmender Größe der zugrundeliegenden Menge an Basiskategorien sehr schnell sehr groß werden. Oftmals ist es jedoch je nach Anwendungsgebiet relativ einfach möglich, eine Einschränkung nach Mengengröße und Auswahl relevanter Kategoriegruppen unter Ausschluß von nicht auftretenden Kombinationen vorzunehmen.

Auch erfolgen in vielen Fällen Betrachtungen mengenwertiger Eigenschaften über sehr kleine Mengen von Basiskategorien. Ein Beispiel ist etwa für ein Krebsregister der Meldemodus, der über Gruppen von Meldern, die einen jeweils betrachteten Fall an das Register gemeldet haben, spezifiziert wird und somit nicht mehr als zehn Basisausprägungen, etwa *Pathologen*, *Kliniken*, *Tumorzentren*, *Gesundheitsämter* etc., aufweist.

Sind die betrachteten Basismengen größer, treten mengenwertige Angaben oftmals nur in den zugrundeliegenden Mikrodaten auf — nach Kombinationen von Basiskategorien wird jedoch nicht ausgewertet. So werden z. B. für Krebspatienten Mengen ausgeübter Berufe erfaßt, für Auswertungen interessiert es jedoch lediglich, ob eine Person eine bestimmte *einzelne* Tätigkeit ausgeübt hat oder nicht. In einer derartigen Situation könnte die Implementierung der konzeptionell zumindest unter Einschluß der Basisebene le_b modellierten Kategorienhierarchie auf die (immer noch mengenwertige) Ebene $le^{(1)}$ und gröbere Ebenen beschränkt werden, wenn die Mikrodaten unter Ausnutzung mengenwertiger Datentypen verwaltet werden und bei deren Auslesen bereits die Betrachtung lediglich einelementiger Mengen erfolgt.

4.2.5 Summarische und kategorielle Attribute eines Datenraums

Makrodaten bilden multidimensionale Datenräume, deren Dimensionalität durch kategorielle Attribute beschrieben wird und die durch summarische Attribute spezifizierte Maßzahlwerte in ihren Zellen enthalten. Der Wert einer Zelle errechnet sich hierbei aus Eigenschaften von Teilgruppen der durch den Datenraum beschriebenen Objektmengen. Diese Teilgruppen werden jeweils durch die Kategorien der kategoriellen Attribute eingegrenzt, die einer Zelle zugeordnet sind. Die Gestalt der summarischen und kategoriellen Attribute wird im folgenden genauer betrachtet, bevor dann im nächsten Abschnitt Datenräume modelliert werden.

Es sei hier noch einmal an die in Abschnitt 4.2.1 eingeführten Rollen von Objektmengen erinnert: In einem Makrodatenraum können — anders als in Mikrodatenräumen — Daten zu mehreren Objektmengen in unterschiedlichen Rollen (typisches Beispiel: Personen und Tumoren aus Studien- und Standardpopulationen)

zusammengeführt werden. Somit können auch alle Attribute eines Datenraums mit Teilmengen seiner Objektmengen (und deren Rollen) behaftet sein. So kann ein kategorielles Attribut das Erhebungsjahr der Daten zur Standardbevölkerung und ein anderes das zur Studienbevölkerung repräsentieren. Ebenso kann ein summarisches Attribut Daten zur Standardpopulation, ein anderes Daten zur Studienpopulation und ein drittes Daten, die beide Populationen einbeziehen, beschreiben. Schließlich beziehen sich in der Regel bestimmte Attribute eines entsprechenden Datenraums auf die Tumordaten, andere wiederum auf personenbezogene Angaben. Diese Zusammenhänge sollen in den nachfolgenden Definitionen besonders berücksichtigt werden.

Lediglich eine vereinfachende Schreibweise bietet folgende Definition rollenbehafteter Objektmengen:

Definition 4.12 (Rollenbehaftete Objektmengen) *Die Menge $\mathcal{OR}$ zur Beschreibung von Objektmengen in bestimmten Rollen sei definiert als $\mathcal{OR} \stackrel{\text{def}}{=} 2^{\mathcal{O}} \times \mathcal{R}$.*

Kategorielle und summarische Attribute sowie Datenräume beziehen sich jeweils auf eine bestimmte endliche, nicht–leere Teilmenge $OR = \{or_1, \ldots, or_n\} \subseteq \mathcal{OR}$ mit $or_i = (O_i, r_i), i = 1, \ldots n$.

Beschreibung statistischer Größen durch Maßzahlen

Maßzahlen definieren statistische Größen, die durch Zellinhalte eines Datenwürfels instantiiert werden können. Während sich summarische Attribute eines Datenraums stets auf konkrete, durch sie beschriebene Objektmengen beziehen und die exakte Herleitung der Zellinhalte eines Datenraums aus zugrundeliegenden Mikrodaten repräsentieren, stellen Maßzahlen lediglich mehr oder wenige grobe Bausteine dieser Berechnungen dar. Beispiele für Maßzahlen reichen

- von Basisgrößen wie *Anzahl*, *Fall-* oder *Bevölkerungszahl* (wie man an diesen Beispielen sieht, können bestimmte Maßzahlen „genauer“ als andere sein und meist in diesem Zusammenhang auch schon den Bezug auf ausgewählte Objektmengen implizieren)
- über allgemeine, fast beliebig gleichermaßen auf Basisdaten sowie auch auf andere Maßzahlen anwendbare Größen wie *Summe*, *Minimum*, *Maximum*, *Durchschnitt*, *Quotient* oder universelle Teststatistiken
- und spezielle, aus anderen Maßen zusammengesetzte Maßzahlen wie „*Minimum des Durchschnitts*“, (standardisierte) Raten, Risiken oder Clusterindizes
- bis zu (durch Dimensionen, Eigenschaften, Objektmengen oder Rollen) spezialisierten Varianten der genannten Maßzahltypen wie „*Rate der Studienpopulation*“, „*regionaler Durchschnitt*“ oder „*zeitliches Minimum*“.

Gegenüber summarischen Attributen abstrahieren Maßzahlen also von Teilen ihrer Herleitung und repräsentieren in diesem Sinne — je nach Bedarf — lediglich deren letzte(n) Teilschritt(e). Näheres hierzu, insbesondere zur Betrachtung von Abhängigkeiten zwischen Maßzahlen, deren Komposition sowie Generalisierungs- und Spezialisierungsbeziehungen zwischen ihnen, findet sich in Abschnitt 4.4.2.

Maßzahlen definieren Wertebereiche, Berechnungsvorschriften und grundlegende Default–Operationen zu ihrer Behandlung im Rahmen von Datenaggregierungen, also einer Vergröberung der Betrachtungsweise durch Zusammenfassung von Kategorien bzw. der jeweiligen Objektmengen. Im folgenden werden zunächst einige entsprechende Komponenten der Maßzahlbeschreibung eingeführt, bevor dann der Maßzahlbegriff formal definiert werden kann.

Wertebereiche von Maßzahlen, also mögliche Ausprägungen der Zellen eines Datenwürfels, sollen als *Datentypen* bezeichnet werden, die Teilmengen eines „Werteuniversums“ darstellen. Einem Datentyp sind *Nullwerte* und *neutrale Elemente* zugeordnet, die fehlendes Wissen über die konkrete Ausprägung einer Maßzahl bzw. Maßzahlwerte zu einer leeren Objektmenge spezifizieren. Weitere spezielle Ausprägungen, wie sie z. B. benötigt werden, wenn Ergebnisse von Berechnungen nicht definiert sind (in Folge einer Division durch 0 o. ä.) werden hier nicht gesondert modelliert. Um die Berechnung von Maßzahlen aus anderen Maßzahlen sowie aus Kategorien in Mikrodaten einheitlich beschreiben zu können, werden auch Domänen von Kategorien als Datentypen angesehen.

Definition 4.13 (Maßzahlwerte und Datentypen) *$\mathcal{W}$ sei eine Menge von* (MASSZAHL–)WERTEN. *Ein* DATENTYP *sei eine Teilmenge $T \subseteq \mathcal{W}$. Zu T definiere $T_0 \subset T$ eine nicht–leere Menge* NEUTRALER ELEMENTE *und $T^{\mathrm{N}} \subset T \setminus T_0$ eine nicht–leere Menge von* NULLWERTEN.

$\mathcal{T}$ sei die Menge aller Datentypen. Es gelte $\mathcal{K} \subseteq \mathcal{W}$ sowie $\mathcal{DO} \subseteq \mathcal{T}$.

Typische Beispiele für Datentypen sind etwa die reellen oder natürlichen Zahlen zuzüglich „künstlicher“ Nullwerte und mit 0 oder 1 als neutralem Element.

Wie aus obiger Definition ersichtlich, werden Nullwerte, also unvollständiges Wissen bezeichnende Angaben (vgl. Abschnitt 2.3.2), im Prinzip genau wie „echte“ Maßzahlwerte repräsentiert. Eine etwaige spezielle Behandlung wird durch die jeweils verarbeitenden Funktionen gekapselt. Dies steht in Analogie zu den in Abschnitt 4.2.4 eingeführten δ–Kategorien, die „ganz normale“ Kategorien zur Repräsentation von Nullwerten für kategorielle Attribute darstellen. Jeder Datentyp T sieht zumindest für drei Arten von Nullwerten spezielle Ausprägungen $t_{\mathrm{nexist}}, t_{\mathrm{unknown}}, t_{\mathrm{navail}} \in T^{\mathrm{N}}$ vor, und zwar zur Spezifikation

- eines *strukturellen Nullwertes* (vgl. [Fro96]), d. h. ein Maßzahlwert existiert nicht, weil es die durch ihn zu beschreibende Objektmenge *prinzipiell* nicht gibt,
- von im zugrundeliegenden Basisdatenbestand unbekannten Werten (aufgrund von Erhebungslücken) oder
- von Lücken im gerade bearbeiteten Datensatz (eine Ausprägung wurde nicht aus den Basisdaten abgefragt und ist somit nicht verfügbar — dieser Nullwert ist insofern der semantisch „schwächste“, als keine Information darüber gegeben wird, *warum* der „wahre“ Wert nicht vorliegt).

Darüber hinaus kann es natürlich — insbesondere zur genaueren Spezifikation der Gründe für obigen zweiten Fall — noch weitere Nullwerte geben.

Während also Nullwerte für unbekannte Ausprägungen stehen, repräsentieren Elemente von T_0 eine (im Gegensatz zu strukturellen Nullwerten) in der jeweiligen Erhebung *„zufällig“* leere Menge betrachteter Objekte. Der Begriff „neutrales Element“ ergibt sich daraus, daß die zentrale Operation auf Datentypen im Kontext von *MADEIRA* die Aggregierung von Maßzahlwerten zu verschiedenen Objektmengen darstellt. Wird nun eine leere Objektmenge mit einer anderen zusammengefaßt, so darf sich deren Maßzahlwert nicht ändern, d. h. der Maßzahlwert zur leeren Objektmenge bildet ein im allgemein üblichen Sinne „neutrales“ Element bzgl. der Aggregierung (etwa im Fall von Anzahlen oder Summen auf den natürlichen oder reellen Zahlen der Wert 0). Typischerweise ist $T_0 = \{t_0\}$ einelementig; T_0 soll aber auch mehrere Werte umfassen dürfen, um Maßzahlen, die mehrere andere Maße (mit evtl. unterschiedlichen neutralen Elementen) verallgemeinern und somit deren Mengen T_0 vereinigen, modellieren zu können (etwa Verhältniszahlen wie Raten und Risiken mit neutralen Elementen 0 oder 1, vgl. Abschnitt 4.4.2, sowie bei der Definition von Kennzahldimensionen in Abschnitt 4.2.6).

Man beachte, daß unter Umständen auch nicht–leere Objektmengen durch Werte aus T_0 beschrieben werden können. So steht z. B. ein Erkrankungsrisiko von 1 (mit 1 als neutralem Element) sowohl für gleich hohe Erkrankungsraten in (nicht–leerer) Standard- und Studienpopulation als auch für gleichermaßen leere Populationen.[21] Ein neutrales Element spezifiziert einfach einen „Normalwert“, der auch für leere Objektmengen, über die also keine weitere Aussage möglich ist, angenommen wird.

Nähere Anforderungen an Nullwerte und neutrale Elemente werden im Laufe dieses Kapitels im Kontext anderer Begriffsdefinitionen spezifiziert werden. Insbesondere definiert Abschnitt 4.3 genau, wann Nullwerte im Laufe von Datenanalysen vorkommen können und wie sie weiterverarbeitet werden.

Kategorien treten als Maßzahlwerte in der Regel nicht in Makrodaten auf, sondern dienen lediglich als Quelle zur Berechnung anderer Maßzahlen.[22] Nullwerte und neutrale Elemente bilden jeweils ausgewählte Kategorien — die konkrete Wahl (etwa von $t_{\mathrm{unknown}} = k_0^{\delta}$) braucht hier nicht näher zu interessieren.

[21] Typischerweise sind derartige Maßzahlen nicht aggregierbar, d. h. aus Maßzahlwerten zweier Objektmengen kann der Maßzahlwert zur Gesamtmenge nicht direkt berechnet werden. Andernfalls hätten nicht–leere Objektmengen mit Maßzahlwerten aus T_0 keinen Einfluß auf den Wert zur Gesamtmenge, was kaum sinnvoll wäre.

[22] Hier gibt es einen Ansatzpunkt zur symmetrischen Behandlung von summarischen und kategoriellen Attributen, wie sie etwa in [AGS97] oder [GL97] vorgeschlagen wird. Diese Möglichkeit soll hier jedoch — insbesondere um das Datenmodell möglichst intuitiv und einfach zu halten — nicht näher betrachtet werden.

Neben der genauen Spezifikation der jeweils betrachteten Objektmengen und Eigenschaften steht die Unterstützung der automatischen Aggregierung im Vordergrund der Modellierung von summarischen (und auch kategoriellen) Attributen eines Datenraums. Navigation in Datenräumen besteht zu großen Teilen im interaktiven Wechsel zwischen verschiedenen Granularitätsstufen der Daten über ausgewählte Dimensionen. Dimensions- bzw. domänenbezogene Aggregierungsfunktionen dienen als Vorgabe von Berechnungsvorschriften zur Zusammenfassung der Werte in den Zellen des Datenraums.[23] Unter Umständen ist hierbei auch von Interesse, über *welche* Kategorien jeweils aggregiert wird. Folgende Situationen bzw. Maßzahltypen sollen unterscheidbar sein:

1. Die Zusammenfassung von Maßzahlwerten zu beliebigen Objektmengen („Teilpopulationen“, die durch die jeweils gruppierten Kategorien definiert sind) ist immer möglich, z. B. werden Maxima oder Minima wiederum über die Bildung von Maximum bzw. Minimum über alle Teilpopulationen aggregiert.

2. Die betreffenden Teilpopulationen müssen disjunkt zueinander sein, damit aggregiert werden kann, z. B. werden Anzahlen nur über disjunkte Gruppen aufsummiert. Diese Art der Aggregierung spielt naturgemäß eine besondere Rolle für die Behandlung mengenwertiger Eigenschaften. Aus personenbezogenen Fallzahlen bzgl. einzelner Krebsdiagnosen kann z. B. nicht (mittels Summation) die Gesamtzahl *aller* an Krebs erkrankter Personen gebildet werden, da die Zuordnung einer Krebsdiagnose in der Objektmenge der Personen eine mengenwertige Eigenschaft ist.

3. Es wird über eine Dimension aggregiert, die für die Objektmengen, auf denen die jeweilige Maßzahl bestimmt wurde, keine „Relevanz“ hat. Aufgrund dieser „Unabhängigkeit“ sind alle Maßzahlwerte entlang dieser Dimension identisch und ändern sich auch durch die Aggregierung nicht. Werden z. B. Fall- und Bevölkerungszahlen über die Art der Erkrankung aggregiert, sind und bleiben die jeweiligen Bevölkerungsangaben (sowohl zu *Leberkrebs* als auch zu *Lungenkrebs* als auch zu *„Alle Diagnosen“*) gleich.

4. Eine Aggregierung ist gar nicht möglich; dies gilt für die meisten komplexeren Maßzahlen wie Raten, Risiken, Clusterindizes etc. Aber auch Bevölkerungszahlen, wenn sie zu einem Stichtag definiert sind, erlauben keine Aggregierung über die Zeit, während sie über andere Dimensionen analog zu anderen Anzahl–Maßen problemlos „disjunkt“ zu aggregieren sind. Wir kommen auf diesen Fall später noch einmal zu sprechen.

Weitere Konstellationen spielen in *MADEIRA* keine Rolle.

Definition 4.14 (Aggregierungsfunktionen) *Sei $T \subseteq \mathcal{W}$ ein Datentyp mit neutralen Elementen $T_0 \subset T$ sowie $D \in \mathcal{DO}$ eine Domäne.*

$\mathcal{F}_{T,D}^{\text{aggr}}$ bezeichne die Menge aller (partiellen) (KATEGORIENBEZOGENEN) AGGREGIERUNGSFUNKTIONEN *$f^{\text{aggr}} : \mathbb{N}^{T \times D} \to T$ auf T zur Zusammenfassung jeweils mehrerer Maßzahlwerte, die sich auf Kategorien aus D beziehen.*

Es gelte $\forall f^{\text{aggr}} \in \mathcal{F}_{T,D}^{\text{aggr}}\ \forall x \in T\ \forall k \in D\colon f^{\text{aggr}}(\{(x,k)\}) \stackrel{\text{def}}{=} x$ sowie $f^{\text{aggr}}(\emptyset) \in T_0$. Darüber hinaus werden genau vier Spezialfälle unterschieden:

1. *$f^{\text{aggr}}(X)$ ist für <u>alle</u> $X \in \mathbb{N}^{T \times D}$ definiert, wobei für alle neutralen Elemente gilt: $\forall X \subseteq T \times D$ $\forall Y \subseteq T_0 \times D\colon f^{\text{aggr}}(X \cup Y) \stackrel{\text{def}}{=} f^{\text{aggr}}(X)$[24] ($f^{\text{aggr}}$ heiße dann* BELIEBIG AGGREGIEREND*),*

2. *zu einer Eigenschaft $e \in \mathcal{E}_D$ und einer Objektmenge $O \subseteq O_e$ sowie für alle $x_h \in T, k_h \in D$ für $h = 1, \ldots, q$ ist $f^{\text{aggr}}(\{(x_1,k_1), \ldots, (x_q,k_q)\})$ — bei gleicher Behandlung der neutralen Elemente wie im ersten Fall — nur definiert, falls alle k_h paarweise bzgl. e und O disjunkt sind (f^{aggr} heiße dann* DISJUNKT AGGREGIEREND *über e und O),*

[23] Vergleiche hierzu auch die formale Betrachtung der Definition von Aggregierungsfunktionen in [CT99].

[24] Dies ergibt sich gemäß der oben dargelegten Bedeutung neutraler Elemente aus der korrekten Behandlung von Maßzahlen zu leeren Objektmengen, die hier mit anderen zusammengefaßt werden.

3. $\forall x \in T\ \forall q \in \mathbb{N}\ \forall k_1, \ldots, k_q \in D\colon f^{\text{aggr}}(\{(x, k_1), \ldots, (x, k_q)\}) = x$ *— ansonsten ist* f^{aggr} *undefiniert (*f^{aggr} *heiße dann* IDENTISCH AGGREGIEREND *und werde auch als* f^{id} *bezeichnet) — oder*
4. $f^{\text{aggr}}(X)$ *ist für kein* $X \in \mathbb{N}^{T \times D}$ *mit* $|X| > 1$ *definiert (*f^{aggr} *heiße dann* NICHT AGGREGIEREND*).*

Durch die Aggregierung werden Maßzahlwerte auf einer gröberen Granularitätsebene bzgl. einer Dimension *exakt* neu berechnet. Ist dies ohne zusätzliches Wissen allein aus den Werten auf der feineren Ebene und den betroffenen Kategorien nicht möglich, ist der Funktionswert für nicht–einelementige Kategorienmengen undefiniert. Heuristiken zur eventuellen Schätzung eines aggregierten Wertes werden — ebenso wie ein interner Rückgriff auf die zugrundeliegenen Basisdaten zur vollständigen Neuberechnung von Maßzahlwerten zu der gewünschten Zielgranularität — in *MADEIRA* nicht unterstützt.

Die *Disaggregierung* als „Inverse“ der Aggregierung wird in *MADEIRA* ebenfalls nicht speziell betrachtet. Im Gegensatz zur Aggregierung kann sie keine *exakten* Werte auf einer feineren Aggregierungsebene liefern, sondern lediglich eine „sinnvolle“ Näherung feinergranularer Daten aufgrund vorliegender Maßzahlwerte einer groben Ebene schätzen, falls trotz fehlenden Wissens eine spezifische Unterteilung nötig sein sollte. Hierdurch entsteht jeweils eine *neue* Maßzahl mit anderer Semantik („Schätzung von ...“) als die Ausgangsmaßzahl. Da kein anwendungsspezifisches Wissen bzw. keine näheren Informationen über die betroffenen Teilgruppen von Objektmengen einbezogen werden können, muß hierbei jeder Teilmenge der gleiche Maßzahlwert zugeordnet werden. Etwa könnte man aus gegebenen Jahresbevölkerungen mittlere Quartalsbevölkerungen über die Gleichverteilung schätzen.

Diese Art der Bestimmung von Maßzahlwerten widerspräche der zu Beginn des Abschnitts vorgestellten Grundidee von *MADEIRA*, durch Ausprägungen von Würfelzellen jeweils genau die Objektmengen zu beschreiben, die durch diejenigen Kategorien definiert sind, die die jeweiligen „Zellkoordinaten“ bilden. Im Falle der Disaggregierung müßten auch Obermengen dieser Objektmengen in die Maßzahlberechnung miteinbezogen werden — dies ließe sich nur sehr umständlich modellieren und würde die Komplexität von *MADEIRA* nur unnötig erhöhen, zumal Disaggregierungen relativ selten genutzt werden. Der Verzicht auf diese Operation läßt sich leicht verschmerzen: Es müssen lediglich die jeweiligen Basisdaten bereits disaggregiert bereitgestellt werden, wobei im Regelfall einfache Disaggregierungsfunktionen wie Identität oder Gleichverteilung völlig ausreichend sind. Das Wissen um die vorgenommene Disaggregierung läßt sich in den Basisdaten dann leicht durch entsprechende Eigenschaften modellieren.

Auf der Basis der Definitionen von Datentypen und Aggregierungsfunktionen können wir nun den Begriff der „Maßzahl“ definieren. Maßzahlen, wie sie bereits zu Beginn dieses Abschnitts exemplarisch vorgestellt wurden, können mit bestimmten Funktionen aus anderen Maßzahlen berechnet und selbst wiederum zu gröberen Granularitätsebenen aggregiert werden.[25] Berechnungen können auf einzelne Maßzahlwerte (aus einer Datenraumzelle) beschränkt sein, wie z. B. bei der Berechnung von Erkrankungsraten einfach zusammengehörige Fallzahlen und Bevölkerungszahlen durcheinander dividiert werden. Aber auch die Zusammenfassung (Aggregierung) von Werten zu verschiedenen Teilen einer Objektmenge (entlang einer Dimension eines Datenwürfels, die durch Eigenschaften und Objektmengen spezifiziert ist) ist möglich. Ein Beispiel hierfür bilden etwa standardisierte Raten, die alters- und geschlechtsspezifische Raten über Alter und Geschlecht in einer gewichteten Summe zusammenfassen. Außerdem können im Rahmen der Berechnung einer Maßzahl unter Umständen auch noch weitere Maßzahlen als Zusatzinformationen anfallen.

Definition 4.15 (Maßzahlen) *Eine* MASSZAHL $mz = (T, T_0, T^{\text{N}}, F, f^{\text{aggr}})$ *werde beschrieben durch*

- *einen Datentyp* $T \subseteq \mathcal{W}$ *mit neutralen Elementen* $T_0 \subset T$ *und Nullwerten* $T^{\text{N}} \subset T \setminus T_0$*,*
- *eine Menge* F *von* BERECHNUNGSVORSCHRIFTEN*, die die Berechnung von mz aus anderen Maßzahlen definieren —* F *bestehe aus Quintupeln*[26] *mit jeweils*

[25] Der Begriff der „Aggregierung“ soll im folgenden stets verwendet werden, wenn Maßzahlwerte zu verschiedenen, durch Kategorien klassifizierten Teilgruppen einer Objektmenge zu einem neuen Wert zusammengefaßt werden. Dies kann sowohl im Rahmen der Berechnung einer *neuen* Maßzahl als auch bei der Bestimmung von Werten *derselben* Maßzahl auf einer gröberen Granularitätsebene erfolgen.

[26] Im „Normalfall“ handelt es sich hier um (rechtseindeutige) Abbildungen, die MZ^{q}, MZ^{z}, **EOR** und dem Flag *disjunkt*? eine Funkti-

- *einer endlichen, nicht–leeren Multimenge* $MZ^q = \{mz_1, \ldots, mz_n\} \subseteq \mathcal{M}$ *von „Quellmaßzahlen“,*
- *einer endlichen Multimenge* $MZ^z \subseteq \mathcal{M}$ *von „Zielmaßzahlen“, wobei* $mz \in MZ^z$ *sei*[27]*,*
- *einer Reihe von Eigenschaftsspezifikationen* $\mathbf{EOR} \subseteq 2^{\mathcal{E} \times \mathcal{OR}}$*,*
- *einem booleschen Flag disjunkt*? *aus* $\mathbb{B}$ *und*
- *einer Familie* $(f_{mz'})_{mz' \in MZ^z}$ *von totalen oder (alle gleichermaßen) auf einelementige Mengen beschränkten* BERECHNUNGSFUNKTIONEN $f_{mz'} : \mathbb{N}^{mz_1.T \times \ldots \times mz_n.T} \to mz'.T$

(„mz ist BERECHENBAR *aus* $mz_1, \ldots, mz_n$ *unter (disjunkter) Aggregierung über* **EOR***“), sowie*

- *eine Familie* $f^{\text{aggr}} = (f^{\text{aggr}}_{e,or})_{e \in \mathcal{E}, or \in \mathcal{OR}}$ *von (partiellen)* AGGREGIERUNGSFUNKTIONEN *der Gestalt* $f^{\text{aggr}}_{e,or} : \mathbb{N}^{T \times D_e} \to T$ *mit* $f^{\text{aggr}}_{e,or} \in \mathcal{F}^{\text{aggr}}_{T,D_e}$ *zur Definition der Aggregierung von Werten von mz über Kategorien, die bestimmte Eigenschaften von Objektmengen in den jeweiligen Rollen definieren.*

Definition 4.14 entsprechend heiße mz BELIEBIG, DISJUNKT, IDENTISCH *oder* NICHT AGGREGIERBAR *über Eigenschaft e und Objektmenge O in Rolle r, falls* $f^{\text{aggr}}_{e,(O,r)}$ *total, disjunkt (bzgl. e und O), identisch bzw. nicht aggregierend ist.*

$\mathcal{M}$ *sei die Menge aller Maßzahlen.*

Die gleichzeitige Berechnung zusätzlicher Maßzahlen im Rahmen der gegebenen Berechnungsvorschriften zielt bereits auf eine effiziente Implementierung von *MADEIRA*: Typischerweise können „zusammengehörige“ Größen, wie etwa eine Maßzahl mit ihrem Konfidenzintervall, deutlich effizienter gemeinsam als separat ermittelt werden, da die zugehörigen Formeln ähnlich aufgebaut sind und Zwischenergebnisse gemeinsam genutzt werden können.

Sind Berechnungsfunktionen $f_{mz'}$ aus $mz.F$ auf einelementige Mengen von Maßzahlwerten beschränkt, so schreiben wir auch $f_{mz'} : mz_1.T \times \ldots \times mz_n.T \to mz'.T$. In diesem Fall ist die jeweilige Menge **EOR** stets leer und das boolesche Flag unerheblich. Es wird einfach die Verknüpfung von jeweils *einem* Wert unterschiedlicher Quellmaßzahlen zur neuen Maßzahl *mz* beschrieben.

Oftmals aggregieren Maßzahlberechnungen jedoch über spezielle Dimensionen, also über Eigenschaften der betrachteten Objekte und verarbeiten somit auch Mengen von Tupeln verschiedener Maßzahlwerte — in diesem Fall sind die Funktionen $f_{mz'}$ auf Multimengen von Werten definiert. So ergibt sich etwa die alters- und geschlechtsstandardisierte Rate als gewichtete Summe über ein Alters- und ein Geschlechtsattribut; ein Maßzahlwert berechnet sich aus einer Menge von Tupeln mit jeweils Fall- und Bevölkerungszahl einer Studienpopulation und Bevölkerungszahl einer Standardpopulation. Um diesen Zusammenhang beschreiben zu können, definiert **EOR** Mengen von zu aggregierenden Eigenschaftsgruppen — eine Menge $\mathbf{EOR} = \{EOR_1, EOR_2 \ldots\}$ hat somit die Semantik „Aggregiere über *eine oder mehrere* Eigenschaften aus EOR_1 *und eine oder mehrere* Eigenschaften aus EOR_2 *und* ...“. Je nach dem Flag *disjunkt*? hat die Aggregierung ggf. über disjunkte Zerlegungen der betrachteten Objektmenge zu erfolgen — was der Regelfall ist — oder kann über beliebige Unterteilungen vorgenommen werden.[28] Im Beispiel der standardisierten Rate wäre etwa EOR_1 eine Menge von (meist, aber nicht notwendigerweise allen) Eigenschaften zur Altersdimension und EOR_2 entsprechend eine Menge von Geschlechtseigenschaften, über die disjunkt zu aggregieren ist. Entsprechend der Unterscheidung zwischen einzelnen Eigenschaften in **EOR** muß die jeweilige Maßzahl *mz* i. a. auch bezüglich dieser Eigenschaften differenzierende Aggregierungsfunktionen $f^{\text{aggr}}_{e,or}$ vorhalten (siehe auch die Beispiele weiter unten).

Durch eine derartige Definition von Maßzahlen werden — zugunsten einer relativ einfachen einheitlichen und exakten Beschreibung der Maßzahlberechnung — dem Benutzer von *MADEIRA* sicherlich gewisse Einschränkungen bei der Spezifikation komplexer Berechnungsfunktionen auferlegt: Es fließen für alle Quellmaßzahlen gleich viele Werte in das Ergebnis ein, und alle Maßzahltupel sind aufgrund der mengenbasierten

onsfamilie $f_{mz'}$ zuordnen. Dies gilt insbesondere nicht mehr, wenn allgemeine Maßzahlen als Oberklasse anderer Maßzahlen betrachtet werden — vgl. Abschnitt 4.4.2.

[27] In Abschnitt 4.4.2 werden wir diese Bedingung darauf einschränken können, daß nur *mz oder* eine „speziellere“ Maßzahl enthalten sein muß.

[28] Das Flag bezieht sich jeweils auf die Disjunktheit bzgl. *aller* betrachteten Eigenschaften, da auch durch $f_{mz'}$ nicht zwischen verschiedenen Klassifikationskriterien unterschieden wird.

Definition der Argumente einer Berechnungsfunktion $f_{mz'}$ gleichberechtigt. Aufwendige Berechnungen sind somit über mehrere Zwischenschritte aus „Hilfsmaßzahlen“ zusammenzusetzen, wie es auch der schrittweisen, datenflußbasierten Grundidee von *VIOLA* entspricht.

Die Aggregierbarkeit von Maßzahlen entspricht dem Begriff der (Semi–)Additivität[29] statistischer Maße, wie er etwa von Chen [CM89] verwendet wird. Wir kommen darauf weiter unten noch genauer zu sprechen.

Wie bereits oben angedeutet, können auch Kategorien als Maßzahlwerte interpretiert werden:

Definition 4.16 (Maßzahlen zur Darstellung von Objekteigenschaften) *Einer Eigenschaft $e \in \mathcal{E}$ und einer Objektmenge $O \subseteq \mathcal{O}_e$ sei $mz_{e,O} \in \mathcal{M}$ als Maßzahl zugeordnet, wobei $mz_{e,O}.T \stackrel{\text{def}}{=} D_e$ mit spezifisch gewählten Nullwerten und neutralen Elementen, $mz_{e,O}.F \stackrel{\text{def}}{=} \emptyset$ und $mz_{e,O}$ über keine Eigenschaft aggregierbar ist.*

Personenbezogene Mortalitätszahlen bilden ein einfaches Beispiel einer Maßzahl: Ihr Datentyp sind die natürlichen Zahlen (zzgl. Nullwerte und mit der 0 als neutralem Element), die Berechnung erfolgt aus (der Maßzahl zu) einer Eigenschaft der Objektmenge *Personen*, die das Versterben repräsentiert, über eine Funktion, die lediglich die betroffenen Personen (über die betrachtete Eigenschaft disjunkt aggregierend) zählt — hier also gleich ein Beispiel für die Nutzung von Objekteigenschaften als „Basismaßzahlen“. Fallzahlen können über alle Eigenschaften durch Summenbildung disjunkt[30] aggregiert werden.

Während im Fallzahlbeispiel die Menge von Berechnungsmöglichkeiten stark eingeschränkt ist, wird im Regelfall die Menge $mz.F$ möglicher Berechnungswege sehr groß sein. Insbesondere bei so allgemeinen Maßzahlen wie *Summe* oder *Minimum* können mehr oder weniger beliebige Maßzahlen als Quelle dienen, wobei jedoch die jeweiligen eigentlichen Berechnungsfunktionen $f_{mz'}$ identisch oder zumindest auf wenige (in der Regel kaum mehr als drei) Varianten beschränkt sind. Diese repräsentieren dann nur noch prinzipiell alternative Berechnungspfade (etwa je nachdem, ob nötige Zwischenergebnisse bereits vorliegen oder erst aus grundlegenden Basisgrößen ermittelt werden müssen). Gleiches gilt für die Aggregierungsfunktionen, die typischerweise auch nur nach sehr wenigen (ein bis zwei) unterschiedlich zu behandelnden Dimensionen bzw. „Klassen“ von Eigenschaften und Objektmengen differenzieren.

Ein anderes Beispiel motiviert nochmals die Notwendigkeit einer genauen Spezifikation von Maßzahlen im Sinne obiger Definition: Die Aggregierung von Bevölkerungszahlen über die Zeit wirft ein Problem auf. Während über (einwertige) Eigenschaften anderer Dimensionen problemlos summiert werden kann, erscheint für die Zeitdimension eher die Durchschnittsbildung angebracht. Folgende drei Varianten einer exakten Maßzahldefinition sind möglich und ergeben jeweils unterschiedliche Aggregierungsfunktionen:

1. Bevölkerungszahlen zu einem Stichtag (etwa zur Jahresmitte) erlauben keine wohldefinierte Aggregierung über die Zeit.
2. Durchschnittliche Jahresbevölkerungen (quasi als Durchschnitt über Anzahlen zu jedem Tag im Jahr) legen als Aggregierung eine mit der Länge der jeweiligen Zeitintervalle gewichtete Durchschnittsbildung nahe.
3. Wesentlich einfacher (nämlich analog zu Fallzahlen) zu behandeln und in epidemiologischen Untersuchungen oft auch angemessener sind Personenjahre, also mit der Länge des betrachteten Zeitraums multiplizierte Bevölkerungszahlen. Diese sind auch über die Zeit einfach summierbar.

Eine Maßzahl ist (insbesondere durch ihre Berechnungsvorschriften gemäß der Menge F) also stets so genau zu spezifizieren, daß alle Aggregierungsfunktionen wunschgemäß definiert werden können, bzw. umgekehrt sind diese Funktionen auf die jeweilige (intendierte) Maßzahldefinition abzustimmen.

So wie das Beispiel der Bevölkerungszahlen die Differenzierung der Aggregierung nach verschiedenen Dimensionen (hier die Zeit als „Sonderfall“) plausibel macht, gelten ähnliche Überlegungen für Eigenschaften,

[29] Semi–additive unterscheiden sich von additiven Maßen dadurch, daß der betrachtete Datentyp keine inversen Elemente bzgl. der jeweiligen Aggregierungsfunktion enthält, also nur ein Monoid und keine Gruppe bildet.

[30] Somit ist natürlich eine Aggregierung über mengenwertige Eigenschaften nur sehr eingeschränkt, d. h. nur für „ausnahmsweise“ disjunkte Kategorien, möglich.

Objektmengen und Rollen: Immer wenn ein Datensatz entstanden ist, indem eine bestimmte Funktion über ein spezielles kategorielles Attribut (das innerhalb eines Datenraums durch die beschriebene Eigenschaft, die betrachtete Objektmenge und deren Rolle spezifiziert wird, wie wir weiter unten sehen werden) angewendet wurde (im Beispiel die Durchschnittsbildung über die Zeit), hat dies Konsequenzen auf die weiteren Aggregierungsmöglichkeiten.

Neben den Betrachtungen zur Disjunktheit der Aggregierung kann auch näheres Wissen über die von einer Aggregierung betroffenen Kategorien selbst mit in die entsprechende Aggregierungsfunktion einfließen, wie man etwa im zweiten Fall obiger Beispielaufstellung sieht: Hier liefert die Länge des jeweils repräsentierten Zeitraums die entscheidende Information, ohne die eine korrekte Aggregierung nicht möglich wäre. Demgegenüber soll in Fällen, in denen zur Aggregierung noch weitere „externe“, nicht durch die Kategorien selbst repräsentierte Daten nötig wären, darauf verzichtet werden, diese — falls überhaupt möglich — automatisch aus dem Prozeß der Herleitung des jeweiligen Datenraums zu extrahieren. Hier muß ggf. eine explizite Aggregierung bzw. Maßzahlverknüpfung (vgl. Abschnitt 4.3.5) durch „manuelle“ Zusammenführung mit den benötigten Daten und unter Angabe der zu nutzenden Berechnungsvorschrift vorgenommen werden. So lassen sich dann auch Erkrankungsraten aggregieren, wenn man ausdrücklich die zugrundeliegenden Bevölkerungszahlen heranzieht — die automatische Einbeziehung derartiger Daten wäre nur sehr aufwendig allgemein zu definieren und berechnungsintensiv zu realisieren.

Während etwa in [CM89] die Menge im Datenbestand gespeicherter Maßzahlen ausdrücklich auf aggregierbare (semi–additive) Maße beschränkt ist und darüber hinaus nur Maßzahlen betrachtet werden, die aus derartigen semi–additiven Maßen berechnet werden können (wie z. B. der Durchschnitt aus Summe und Anzahl)[31], werden dem Anwender von *VIOLA* größere Freiheiten gewährt bzw. eine größere Verantwortung zugewiesen. Er hat die Kontrolle über die Repräsentation einer Datenanalyse in Form eines Datenflußprogramms und muß ggf. selbständig Aggregierungen auf Ebene vorhandener aggregierbarer Maßzahlen vornehmen, um hieraus ableitbare nicht–aggregierbare Maße auf jeweils höheren Aggregierungsebenen zu berechnen. Die automatische Nutzung solcher Hilfsmaßzahlen zur Aggregierung „komplexerer“ Maßzahlen ist in *MADEIRA* nicht vorgesehen. Derartige Maße werden, ebenso wie Maßzahlen, zu deren Berechnung prinzipiell immer die jeweiligen Basis- (Mikro-)daten vorliegen müssen (in [GBLP96] *holistische Aggregierungsfunktionen* genannt, z. B. der Median), als nicht aggregierbar eingestuft.

Ein weiterer Aspekt der Aggregierbarkeit ist beachtenswert: Ihre Definition macht keinen grundsätzlichen Unterschied zwischen ein- und mengenwertigen Eigenschaften. Auch über letztere können durchaus sinnvolle Aggregierungsfunktionen zur *beliebigen* (nicht nur disjunkten) Aggregierung definiert werden. So bezieht sich etwa das betrachtete Zeitintervall im zweiten Fall in obiger Diskussion von Bevölkerungsdaten auf eine mengenwertige Eigenschaft von Personen: jede Person ist mehreren Zeiträumen zugeordnet. Ein weiteres Beispiel für die *beliebige* Aggregierung sind auch die Bildung von Minimum und Maximum.

Summarische Attribute

Wie bereits oben angedeutet, präzisiert ein summarisches Attribut *sa* über eine Summierungsfunktion die genaue Berechnung von Maßzahlwerten zu gegebenen Objektmengen unter Berücksichtigung der gesamten Historie der Entstehung des jeweiligen Datenraums. In diesem Sinne instantiiert es eine assoziierte Maßzahl, die nur einen durch die letzten Berechnungsschritte gegebenen Datentyp vorgibt. Außerdem kann ein summarisches Attribut aus dem Kontext des jeweiligen Datenraums sowie der jeweils zugrundeliegenden Objektmengen heraus gegenüber seiner Maßzahl zusätzliche Aggregierungsmöglichkeiten zulassen, sofern von der Aggregierung Objektmengen betroffen sind, die nicht vom Attribut beschrieben werden. Dies dient der Konsistenzerhaltung in Datenräumen und wird im nächsten Abschnitt im Rahmen der Datenraumdefinition (in Def. 4.23) konkretisiert.

[31] In [GBLP96] werden derartige Maßzahlen bzw. ihre Aggregierungsfunktionen, die sich unter Abstützung auf die Aggregierung anderer, „einfacherer“ Maße definieren lassen, als *algebraisch* bezeichnet.

Definition 4.17 (Summarische Attribute) *Ein* SUMMARISCHES ATTRIBUT *sei definiert als* $sa = (mz, OR, f^{\text{sum}}, f^{\text{aggr}})$ *durch*

- *die dargestellte Maßzahl* $mz \in \mathcal{M}$*,*
- *die durch sa in ihren jeweiligen Rollen beschriebenen Objektmengen* $OR = \{(O_1, r_1), \ldots, (O_n, r_n)\} \subseteq \mathcal{OR}$ *mit* $n > 0$*,*
- *eine (totale)* SUMMIERUNGSFUNKTION $f^{\text{sum}} : 2^{O_1} \times \ldots \times 2^{O_n} \to mz.T$*, die angibt, wie die zugeordnete Maßzahl direkt aus Teilen der zugrundeliegenden Objektmengen abzuleiten wäre, sowie*
- *eine Familie* $f^{\text{aggr}} = (f^{\text{aggr}}_{e,or})_{e \in \mathcal{E}, or \in \mathcal{OR}}$ *von (partiellen)* AGGREGIERUNGSFUNKTIONEN *der Gestalt* $f^{\text{aggr}}_{e,or} : \mathbb{N}^{T \times D_e} \to T$ *mit* $f^{\text{aggr}}_{e,or} \in \mathcal{F}^{\text{aggr}}_{T,D_e}$*, die im wesentlichen von der Maßzahl mz übernommen werden (vgl. später Def. 4.23).*

Wie schon bei den Maßzahlen, spreche man auch hier analog von BELIEBIGER, DISJUNKTER, IDENTISCHER *oder* KEINER AGGREGIERBARKEIT *über Eigenschaft e und Objektmenge O in Rolle r. Allgemein heiße sa* AGGREGIERBAR *über e und* (O, r)*, falls sa beliebig, disjunkt oder identisch aggregierbar über e und* (O, r)*. Weiterhin bezeichne* $sa.T \stackrel{\text{def}}{=} sa.mz.T$*.*

$\mathcal{SA}$ *sei die Menge aller summarischen Attribute.*

Ein summarisches Attribut repräsentiert also eine Maßzahl in einem Datenraum. Die Summierungsfunktion f^{sum} ist in dieser Definition prinzipiell lediglich als Metadatum anzusehen, das evtl. einfach als beschreibender Text implementiert wird. Die Berechnungsfunktionen einer Maßzahl sind die eigentlichen Träger der Durchführung einer Datenanalyse. Sie beschreiben *und* realisieren Analyseschritte. Demgegenüber dient f^{sum} nur zur (statischen) Spezifikation der Semantik eines vorliegenden Datenraums bzw. seiner Zellinhalte. Der Zusammenhang zwischen Maßzahlberechnung und Summierungsfunktion besteht darin, daß man letztere schreiben kann als $f^{\text{sum}} = f_{mz} \circ f$, wobei f_{mz} eine Berechnungsfunktion der jeweiligen Maßzahl (also aus $mz.F$) ist und f diejenigen Datenbasen und Berechnungen repräsentiert, die der Maßzahl *mz* im Rahmen der jeweiligen Datenanalysesitzung als Grundlage zur Erzeugung des betrachteten Datenraums dienten. Unter Umständen kann jedoch auch die Nutzung so spezifischer Maßzahlen nötig sein, daß f die Identität ist, also alle Details der Herleitung bereits durch die Maßzahl repräsentiert werden. Auf diese Zusammenhänge wird in Abschnitt 4.4.2 im Rahmen der Parametrisierung von Maßzahlen noch etwas genauer eingegangen.

Die Nullwerte t_{navail} und t_{nexist} treten i. a. nicht als Ergebniswerte von f^{sum} auf[32], da zum einen alle nötigen Angaben (also die betrachteten Objektmengen) vorliegen und zum anderen durch diese Funktion nicht unterschieden werden kann, *warum* die Objektmengen evtl. leer sind. Somit gilt für Teilmengen der betrachteten Objektmengen $U_i \subseteq O_i, i = 1, \ldots, n$, immer $U_1 = \emptyset \wedge \ldots \wedge U_n = \emptyset \Rightarrow f^{\text{sum}}(U_1, \ldots, U_n) \in mz.T_0$.

Die Aggregierung von Maßzahlwerten über verschiedene Kategorien mittels f^{aggr} entspricht der Berechnung eines Wertes des summarischen Attributs für die Vereinigung entsprechender Objektmengen mittels f^{sum}. Somit müssen diese beiden Funktionen miteinander verträglich sein, was in der folgenden Definition konkretisiert wird. Es wird davon ausgegangen, daß über eine Menge *EOR* von Objekteigenschaften (die später durch ein kategorielles Attribut spezifiziert werden, vgl. Def. 4.19) aggregiert wird und die Aggregierungsfunktionen des summarischen Attributs *sa* zu all diesen Eigenschaften identisch sind (was Def. 4.23 garantieren wird). Im Prinzip wird mit Def. 4.18 erst formal spezifiziert, was eigentlich *inhaltlich* unter „Aggregierbarkeit“ von Maßzahlen bzw. summarischen Attributen eines Datenraums zu verstehen ist: Maßzahlwerte (in den Datenraumzellen) zu feineren Teilen einer Objektmenge müssen (unter Verwendung einer Aggregierungsfunktion) zu Werten gleicher Semantik (gemäß der Summierungsfunktion des summarischen Attributs) zur Beschreibung von gröberen Objektgruppen zusammengefaßt werden können. Gray bezeichnet diese aggregierbaren Maßzahlen bzw. ihre Aggregierungsfunktionen in [GBLP96] als *distributiv*.

[32] Auf die einzigen Ausnahmen hiervon werden wir bei der Definition von Datenräumen mit Kennzahldimensionen in Abschnitt 4.2.6 sowie bei der Berechnung von Maßzahlwerten zu Kategorien in Abschnitt 4.4.3 treffen.

Definition 4.18 (Aggregierbarkeit: Verträglichkeit von Summierung und Aggregierung) *Sei* $sa = (mz, OR, f^{\text{sum}}, f^{\text{aggr}})$ *ein summarisches Attribut mit* $OR = \{(O_1, r_1), \ldots, (O_n, r_n)\}$*; ferner sei* $EOR \subseteq \mathcal{E} \times \mathcal{OR}$ *eine nicht–leere Menge von Objekteigenschaften mit identischer Domäne D, über die aggregiert werden soll, wobei für alle* $(e, or) \in EOR$ *gelte, daß sa über e und or mittels der gleichen Aggregierungsfunktion* f^{aggr} *aggregierbar ist.*

Sei $k \in D$ *eine beliebige Kategorie zur Beschreibung der betrachteten Eigenschaften. k lasse sich schreiben als* $k \equiv k_1 \vee \ldots \vee k_q$*. Falls sa lediglich disjunkt über die betrachteten Eigenschaften, Objektmengen und Rollen aggregierbar ist, seien diese Kategorien paarweise disjunkt bzgl. all dieser Objektmengen und Eigenschaften.*

Für $i = 1, \ldots, n$ *seien ferner* $\overline{O_i} \subseteq O_i$ *beliebig*[33] *sowie*

$$U_i \stackrel{\text{def}}{=} \{o \in \overline{O_i} \mid \forall e \in \mathcal{E} : ((e, O_i, r_i) \in EOR \Rightarrow o \vdash_e k)\} \ ,$$
$$U_i^h \stackrel{\text{def}}{=} \{o \in \overline{O_i} \mid \forall e \in \mathcal{E} : ((e, O_i, r_i) \in EOR \Rightarrow o \vdash_e k_h)\},\ h = 1, \ldots, q,$$

die Teilmengen von $\overline{O_i}$*, die k bzw.* k_h *erfüllen.*

Insbesondere bedeutet dies für Objektmengen O_i*, die von sa beschrieben werden, über die aber nicht aggregiert werden soll (also* $\nexists e \in \mathcal{E} : (e, O_i, r_i) \in EOR$*), daß* U_i *und* U_i^h *gleich* $\overline{O_i}$ *sind.*

Dann gelte stets $f^{\text{sum}}(U_1, \ldots, U_n) = f^{\text{aggr}}(\{(f^{\text{sum}}(U_1^1, \ldots, U_n^1), k_1), \ldots, (f^{\text{sum}}(U_1^q, \ldots, U_n^q), k_q)\})$.

Aus einem anderen Blickwinkel betrachtet, sagt diese Definition gerade aus, daß ein summarisches Attribut genau dann (disjunkt oder beliebig) aggregierbar ist, wenn eine Aggregierungsfunktion f^{aggr} existiert, die die genannte Bedingung erfüllt. Spezifiziert f^{sum} z. B. einfach die Anzahl $|O|$ von Objekten einer einzelnen Objektmenge O, so kann zur Aggregierung über beliebige *disjunkte* (bzw. durch *disjunkte* Kategorien repräsentierte) Teilmengen $U^1, \ldots, U^q \subseteq O$ mit $U \stackrel{\text{def}}{=} U^1 \cup \ldots \cup U^q$ einfach die Summation als Aggregierungsfunktion f^{aggr} gewählt werden, denn es gilt stets $|U| = |U^1| + \ldots + |U^q|$. Anzahlen sind also (mittels Summation) disjunkt, aber nicht beliebig aggregierbar, da die Gleichung für nicht–disjunkte Objektmengen nicht gilt (und auch keine andere, anstelle der Summation verwendbare Aggregierungsfunktion existiert, die die Gleichung im allgemeinen Fall erfüllen kann).

Mit Def. 4.18 wird bereits die Art der Nutzung von Aggregierungsfunktionen bei der Aggregierung von Datenräumen angedeutet. Im Rahmen der Formalisierung des Aggregierungs- bzw. Ableitungsoperators in Abschnitt 4.3.2 werden wir anhand obiger Definition die Korrektheit der Aggregierung, d. h. die Konsistenz der Definition des jeweiligen summarischen Attributs mit den Aggregierungsergebnissen beweisen.

Schon in Def. 4.18 sieht man, daß Aspekte der „(Un–)Vollständigkeit“ der Aggregierung, wie sie etwa in [LS97] diskutiert werden, in *MADEIRA* bereits durch die zugrundeliegende Definition der genutzten Kategorien behandelt werden und bei der Berechnung selbst keiner speziellen Betrachtung bedürfen. Das Aggregierungsergebnis wird stets einer Kategorie zugeordnet, die sich *genau* („vollständig“) als Disjunktion über alle zur Aggregierung herangezogenen Teilkategorien ergibt. Aggregiert man also z. B. Maßzahlwerte zu den drei Alterskategorien *„0–39“*, *„40–69“* und *„70–99“*, so erhält man als Ergebnis einen Wert zur Alterskategorie *„0–99“*. Je nach Anwendungsgebiet kann diese Alterskategorie evtl. als äquivalent zu einem „vollständigen“ Gesamtwert über das Alter modelliert werden. Dies ist genau dann der Fall, wenn kein Objekt (keine Person) im betrachteten Anwendungsszenario 100 Jahre oder älter ist. Dieses Wissen fließt speziell bei der Modellierung der jeweils für die betrachtete Dimension gültigen Kategorienhierarchie ein.

Auf einen Aspekt, der auch für die Definition von Datenräumen im nächsten Abschnitt noch relevant sein wird, muß hier weiterhin noch kurz eingegangen werden: Sind die Objektmengen, über die aggregiert werden soll, und die durch das summarische Attribut beschriebenen Objektmengen durchschnittsleer, ergibt sich aus Def. 4.18, daß für die zu verwendende Aggregierungsfunktion f^{aggr} (mit $x \in sa.T$ beliebig) die Bedingung „$f_{e,or}^{\text{aggr}}(\{(x, k_1), \ldots, (x, k_q)\}) = x$“ erfüllt sein muß. Der Maßzahlwert ist also (wie auch intuitiv klar) unabhängig von der Aggregierung; *sa* ist *identisch* aggregierbar über die entsprechenden Objekteigenschaften.

[33] Gemäß der im nächsten Abschnitt folgenden Definition von Datenräumen kann $\overline{O_i}$ als diejenige Teilmenge einer durch den jeweiligen Datenraum beschriebenen Objektmenge O_i angesehen werden, die durch die Koordinaten (Kategorien) einer Datenraumzelle *außer k* eingegrenzt wird.

Kategorielle Attribute

Kommen wir nun zu den kategoriellen Attributen, die die „Dimensionen“[34] eines Datenraums beschreiben, also die Menge der betrachteten Kategorien dieser Dimension spezifizieren.

Definition 4.19 (Kategorielle Attribute) *Ein* KATEGORIELLES ATTRIBUT $ka = (D, K, EOR)$ *umfasse eine endliche, nicht–leere Menge $K \subseteq D$ von Kategorien zur Beschreibung einer endlichen, nicht–leeren Menge $EOR \subseteq \mathcal{E} \times \mathcal{OR}$ von Eigenschaften rollenbehafteter Objektmengen.*

Für alle $(e, O, r) \in EOR$ gelte $D_e = D$ sowie $O \subseteq O_e$. $ka.OR$ bezeichne die Menge rollenbehafteter Objektmengen, deren Eigenschaften durch ka beschrieben werden, also $\{(O, r) \mid \exists e \in \mathcal{E} : (e, O, r) \in EOR\}$.

$\mathcal{KA}$ sei die Menge aller kategoriellen Attribute.

Aufgrund praktischer Erfahrungen bei der Datenanalyse im Krebsregister verzichtet diese Definition im Gegensatz zu vielen in der Literatur vorgeschlagenen multidimensionalen Datenmodellen darauf, Kategorien eines Attributs auf eine gemeinsame Aggregierungsebene[35] zu beschränken. Dies ermöglicht — auf Kosten einer aufwendigeren Modellierung und geringer Effizienzverluste bei der Implementierung der rein ebenenbezogenen Fälle — eine anwendungsnähere Repräsentation und eine deutlich vereinfachte Durchführung typischer Auswertungen (in Tabellen sind z. B. oftmals Zwischensummen auf verschiedenen Aggregierungsebenen oder speziell interessierende Zusammenfassungen von Gruppen vorgesehen). Weiterhin dient eine Zusammenfassung unterschiedlich granularer Angaben in einem Datenraum einer effizienteren Bereitstellung von Gesamtwerten, die nachträglich (aufgrund fehlender Aggregierbarkeit) oftmals nur sehr aufwendig bzw. unter abermaligem Rückgriff auf Basisdaten berechnet werden können.

Es sei hier nochmals darauf hingewiesen, daß die Kategorien eines Attributs nicht unbedingt aus der Kategorienmenge der entsprechenden Dimension stammen müssen, sondern beliebige Elemente einer Domäne (insbesondere auch dynamische Zusammenfassungen anderer Kategorien) sein können. Die Dimension liefert jeweils nur eine hilfestellende Grundlage, einen „Basisrahmen“ für Untersuchungen, der jedoch flexibel erweitert werden kann.

Die Aufnahme von Rollen in die Definition eines kategoriellen Attributs ermöglicht dessen richtige Einordnung im Kontext eines Datenraums, insbesondere die Zuordnung zu dessen summarischen Attributen (siehe folgender Abschnitt). Für die durch das kategorielle Attribut beschriebenen Eigenschaften haben sie keine Bedeutung. Ein kategorielles Attribut kann durchaus gleichzeitig mehrere Objektmengen mit evtl. unterschiedlichen Rollen beschreiben. So werden im Laufe einer Datenanalyse über den Vereinigungsoperator (siehe Abschnitt 4.3.4) Daten zu unterschiedlichen Objektmengen verknüpft, wobei oftmals Eigenschaften zur gleichen Domäne miteinander identifiziert werden.

Als Beispiel diene etwa die Domäne mit Altersangaben zur Beschreibung von Tumorerkrankungen. Zur Berechnung von Inzidenzraten werden Daten zu Tumorfällen auf zugrundeliegende Bevölkerungsdaten bezogen. Beim Tumor (erste Objektmenge) beschreibt die Altersangabe „*das Alter der betroffenen Person zum Zeitpunkt der Erstdiagnose*“, während die Bevölkerungsdaten auf Personenbasis (zweite Objektmenge) hiermit „*das Alter einer gezählten Person zur Jahresmitte*“ repräsentieren. Das Altersattribut des Ergebnis–Datenraums integriert beide Eigenschaften.

Die hier erfolgte Definition summarischer und kategorieller Attribute weist eine beachtenswerte Symmetrie auf: So wie Domänen und Dimensionen den von konkreten Datenräumen unabhängigen Rahmen für qualifizierende Eigenschaften bilden, die durch kategorielle Attribute beschrieben werden, so definieren Maßzahlen anwendungsunabhängige quantifizierende Eigenschaften, die in summarischen Attributen konkretisiert werden. Hieraus motiviert sich auch die doppelte Modellierung von Aggregierungsfunktionen: Maßzahlen geben allgemeine Verarbeitungsrichtlinien vor, die von summarischen Attributen im Kontext konkret beschriebener

[34] Soweit Verwechslungen mit dem in Def. 4.9 eingeführten Dimensionsbegriff ausgeschlossen sind, wird auch im folgenden der Begriff „Dimension eines Datenraums“ gelegentlich zur Bezeichnung eines kategoriellen Attributs verwendet, um die Betrachtung eines Datenraums als ein multidimensionales Feld zu unterstreichen.

[35] Natürlich ist es durchaus häufig der Fall, daß ein Attribut eine gesamte oder durch eine Kategorie einer höheren Ebene eingeschränkte Aggregierungsebene (vgl. [LAW98]) repräsentiert.

Objekte und Eigenschaften sowie im Rahmen eines Datenraums, der Anforderungen an die Konsistenz zwischen verschiedenen summarischen und kategoriellen Attributen stellt, verfeinert werden.

4.2.6 Modellierung von Makrodaten durch Datenräume

Auf der Grundlage des vorangegangenen Abschnitts können nun Datenräume, die zentrale Datenstruktur von *MADEIRA* zur Repräsentation von Makrodaten, modelliert werden. Wie schon die im vorigen Abschnitt eingeführten Attribute beschreibt auch ein Datenraum eine Menge von rollenbehafteten Objektmengen, wobei sich jedes summarische und jedes kategorielle Attribut des Datenraums jeweils auf einen Teil dieser Menge bezieht.

Definition 4.20 (Datenräume) *Ein* (MAKRO–)DATENRAUM $dr = (OR, KA, SA, f^{\text{ext}})$ *sei definiert durch*

- *eine endliche, nicht–leere Menge* $OR = \{(O_1, r_1), \ldots, (O_n, r_n)\} \subseteq \mathcal{OR}$ *von durch den Datenraum beschriebenen, rollenbehafteten Objektmengen,*
- *eine endliche, nicht–leere Menge* $KA \subseteq \mathcal{KA}$ *von kategoriellen Attributen,*
- *eine endliche, nicht–leere Menge* $SA \subseteq \mathcal{SA}$ *von summarischen Attributen und*
- *eine Extensionsfunktion* f^{ext}.

Sei $KA^{\text{k}} = \{ka_1, \ldots, ka_m\} \subseteq KA$ *die Menge aller mindestens zweielementigen Attribute ka aus KA (mit* $|ka.K| > 1$*), auch* KLASSIFIZIERENDE ATTRIBUTE *genannt. Diese leisten also einen echten Beitrag zur Dimensionalität des durch dr verwalteten multidimensionalen Feldes. Entsprechend soll die Anzahl* $|KA^{\text{k}}|$ *auch als* DIMENSIONALITÄT *von dr bezeichnet werden. Demgegenüber grenzen die* EINSCHRÄNKENDEN ATTRIBUTE *des Datenraums,* $KA^{\text{r}} = KA \setminus KA^{\text{k}} = \{ka_{m+1}, \ldots, ka_{m+v}\}$*, lediglich die zugrundeliegenden Objektmengen bzw. Basisdaten ein.*

Zwei kategorielle Attribute eines Datenraums unterscheiden sich entweder in der beschriebenen Eigenschaft oder in ihren Objektmengen, genauer: $\forall ka_1 \neq ka_2 \in KA\colon ka_1.EOR \cap ka_2.EOR = \emptyset$.

Weiterhin sei $OR = \bigcup_{sa \in SA} sa.OR = \bigcup_{ka \in KA} ka.OR$ *durch die von den Attributen des Datenraums beschriebenen Objektmengen bestimmt.*

Die EXTENSION *eines Datenraums sei gegeben durch eine Familie* $f^{\text{ext}} = (f_{sa}^{\text{ext}})_{sa \in SA}$ *totaler Abbildungen* $f_{sa}^{\text{ext}}\colon ka_1.K \times \ldots \times ka_m.K \to sa.T$*, die jeweils die „Ausprägungen" eines summarischen Attributs definieren.*[36] *Analog zur Definition von Mikrodatenräumen lassen sich die Funktionswerte von* f^{ext} *aus Instanzen* $\text{ext}(O_i, r_i) \subseteq O_i, i = 1, \ldots, n$*, der betrachteten Objektmengen herleiten.*[37] *Ein Tupel* $(k_1, \ldots, k_m)$ *mit* $k_j \in ka_j.K$ *für* $j = 1, \ldots, m$ *definiert eine* ZELLE *eines Datenraums, deren Inhalte die entsprechenden Werte von* $(f_{sa}^{\text{ext}})_{sa \in SA}$ *bilden. Die Kategorien* $k_1, \ldots, k_m$ *werden auch* KOORDINATEN *dieser Zelle genannt.*

Zu ihrer einfacheren Handhabung sei f_{sa}^{ext} *auf die einschränkenden Attribute erweitert durch* $f_{sa}^{\text{ext}}(k_1, \ldots, k_{m+v}) \stackrel{\text{def}}{=} f_{sa}^{\text{ext}}(k_1, \ldots, k_m)$ *mit* $k_j \in ka_j.K$ *für* $j = 1, \ldots, (m+v)$*. Tupel der durch* f^{ext} *definierten Relationen oder auch nur die Funktionswerte werden als* MAKRODATEN *bezeichnet.*

$\mathcal{DR} = \mathcal{DR}^{\text{ma}}$ *sei die Menge aller (Makro–)Datenräume.*

Die Forderung disjunkter Objekteigenschaften für verschiedene kategorielle Attribute ist intuitiv sinnvoll, denn andernfalls wäre eines dieser Attribute redundant und würde evtl. die Dimensionalität des Datenraums unnötig erhöhen. Auch ist die Disjunktheit der Eigenschaften für die eindeutige Aggregierung über ein kategorielles Attribut nötig (vgl. Def. 4.23).

Die geforderte Übereinstimmung der Objektmengen zu kategoriellen und summarischen Attributen läßt sich ebenfalls leicht motivieren: Über die kategoriellen Attribute sollen *genau* die Objektmengen der summarischen Attribute auf die im jeweiligen Datenraum interessierenden Teilgruppen eingegrenzt werden. Der Klarheit halber wird auch eine Angabe (nämlich dann die entsprechende Wurzelkategorie(n)) gefordert, wenn eine Objektmenge gar nicht eingeschränkt werden soll.

[36] Hierdurch wird implizit eine Reihenfolge auf den kategoriellen Attributen aus KA^{k} festgelegt.

[37] Dies wird weiter unten in Def. 4.21 konkretisiert.

Eine Implementierung eines Datenraums wird sicherlich die Instanzen $\mathrm{ext}(O_i, r_i)$ der beschriebenen Objektmengen nicht explizit speichern. Vielmehr spezifizieren sie als Gesamtheit die jeweils zugrundeliegenden Basisdaten (etwa als Identifikator einer Datenbankrelation) und bilden die Grundlage für die genaue Modellierung der Semantik von Datenmengen und Operationen auf diesen.

Die Zellwerte eines Datenraums, d. h. die Funktionswerte der Abbildungen f_{sa}^{ext}, sind — bis auf Nullwerte — eindeutig durch die Summierungsfunktionen der jeweiligen summarischen Attribute bestimmt. Hierbei ist zu beachten, daß nicht immer alle Koordinaten einer Zelle für den entsprechenden Wert eines summarischen Attributs relevant sind, wenn einzelne kategorielle Attribute Eigenschaften anderer Objektmengen beschreiben als die derjenigen, auf denen das summarische Attribut definiert ist. Somit werden gemäß nachfolgender Definition die Maßzahlwerte entlang der entsprechenden „Würfeldimension“ identisch dupliziert (ein kleines Beispiel folgt weiter unten). Dieser Aspekt steht auch in engem Zusammenhang mit Def. 4.18 zur Verträglichkeit von Aggregierung und Summierungsfunktion eines summarischen Attributs, die ebenfalls eine derartige Werteverdopplung fordert.

Definition 4.21 (Herleitung der Datenraumextension) *Sei $dr = (OR, KA, SA, f^{\mathrm{ext}})$ ein Datenraum mit den klassifizierenden Attributen $ka_1, \ldots, ka_m$ und den einschränkenden Attributen $ka_{m+1}, \ldots, ka_{m+v}$. Zu einem summarischen Attribut $sa \in SA$ seien $sa.OR = \{(O_1, r_1), \ldots, (O_n, r_n)\}$ die jeweils beschriebenen Objektmengen. Ferner seien für $j = 1, \ldots, m+v$ Kategorien $k_j \in ka_j.K$ als Koordinaten einer Zelle gegeben sowie $U_i \subseteq \mathrm{ext}(O_i, r_i), i = 1, \ldots, n$, die durch diese Eigenschaftsausprägungen eingegrenzten Objektmengen von sa, also*

$$U_i \stackrel{\mathrm{def}}{=} \{o \in \mathrm{ext}(O_i, r_i) \mid \forall j = 1, \ldots, m+v \, \forall e \in \mathcal{E} : ((e, O_i, r_i) \in ka_j.EOR \Rightarrow o \vdash_e k_j)\} \quad .$$

Dann gelte

- $f_{sa}^{\mathrm{ext}}(k_1, \ldots, k_m) = sa.f^{\mathrm{sum}}(U_1, \ldots, U_n)$ *oder*
- $f_{sa}^{\mathrm{ext}}(k_1, \ldots, k_m) \in sa.T^{\mathrm{N}}$.

Ist $f_{sa}^{\mathrm{ext}}(k_1, \ldots, k_m) \in T_0 \cup \{t_{\mathrm{nexist}}\}$, so soll die entsprechende Zelle (bzgl. sa) LEERE ZELLE *genannt werden.*

Leere Objektmengen werden durch leere Zellen eines Datenraums repräsentiert, genauer (in obiger Notation):

- Gilt $U_1 = \emptyset \wedge \ldots \wedge U_n = \emptyset$, so ergibt sich ein neutrales Element aus T_0. Man beachte, daß unter Umständen auch nicht–leere Objektmengen durch Werte aus T_0 beschrieben werden können.
- Ist sogar mindestens eine der Mengen leer, die man erhält, wenn man oben in der Definition der U_i anstelle von $\mathrm{ext}(O_i, r_i)$ ganz O_i gemäß der der Zelle zugeordneten Kategorien eingrenzt, erhält man einen strukturellen Nullwert t_{nexist}.

Im Gegensatz zur Definition neutraler Elemente werden also Maßzahlwerte bereits dann als „nicht existent“ erachtet, wenn nur *eine* der betrachteten Objektmengen nicht existiert.

In Def. 4.21 wurden zur Definition des Inhalts einer Zelle nicht unbedingt alle ihre Koordinaten herangezogen, sondern nur diejenigen, die für das jeweilige summarische Attribut relevante Objektmengen klassifizieren. Diese Abhängigkeit zwischen kategoriellen und summarischen Attributen konkretisiert die folgende Definition.

Definition 4.22 (Für ein summarisches Attribut relevante kategorielle Attribute) *Zu einem Datenraum $dr = (OR, KA, SA, f^{\mathrm{ext}})$ und einem summarischen Attribut $sa \in SA$ bezeichne $KA_{sa} \stackrel{\mathrm{def}}{=} \{ka \in KA \mid sa.OR \cap ka.OR \neq \emptyset\}$ die Menge aller kategoriellen Attribute von dr, die für sa (also zur Beschreibung seiner Objektmengen) relevant sind.*

Für ein Attribut $ka \in KA \setminus KA_{sa}$ ist der Wert von f_{sa}^{ext} unabhängig von der entsprechenden Koordinate — er wird entlang dieser Würfeldimension jeweils für alle Kategorien des kategoriellen Attributs identisch dupliziert. Zur Vereinfachung der Definition einiger Operatoren auf Datenräumen in Abschnitt 4.3 sei deshalb f_{sa}^{ext} für derartige kategorielle Attribute ka jeweils kanonisch von $ka.K$ auf ganz $ka.D$ erweitert.

Neben der oben vorgestellten (bis auf Nullwerte) eindeutigen Definition von Würfelzellen gibt es noch eine weitere Konsistenzanforderung an Datenräume. Da ein kategorielles Attribut gleichzeitig mehrere verschiedene Objektmengen und Eigenschaften beschreiben kann und die Aggregierung eines Datenraums natürlich (wie wir in Abschnitt 4.3.2 im Detail sehen werden) gerade über seine kategoriellen Attribute erfolgt, müssen auch die Aggregierungsfunktionen jedes summarischen Attributs jeweils für alle Eigenschaften eines kategoriellen Attributs identisch sein. Anders gesagt: Ein summarisches Attribut *sa* kann nur dann die Aggregierungsfunktionen, die sich auf die Menge der (e, O, r)–Tripel eines bestimmten kategoriellen Attributs *ka* des jeweiligen Datenraums beziehen, aus der Definition seiner Maßzahl *sa.mz* übernehmen, wenn diese Funktionen alle gleich sind. Andernfalls kann *sa nicht* über all diese Objekteigenschaften bzw. über *ka* aggregiert werden. Somit wird deutlich, warum ein summarisches Attribut *sa* seine eigene Menge von Aggregierungsfunktionen definieren muß und nicht einfach auf die Funktionen der jeweiligen Maßzahl zurückgreifen kann. Hinzu kommt hierbei noch der Aspekt der identischen Aggregierung über für *sa* nicht relevante kategorielle Attribute. Folgende Definition von auf kategorielle Attribute bezogenen Aggregierungsfunktionen eines summarischen Attributs faßt diese Überlegungen zusammen:

Definition 4.23 (Aggregierungsfunktionen eines summarischen Attributs) *Seien $dr = (OR, KA, SA, f^{\text{ext}})$ ein Datenraum, $ka \in KA$ ein kategorielles und $sa \in SA$ ein summarisches Attribut des Datenraums. Folgendermaßen werde (zu sa) eine Aggregierungsfunktion f_{ka}^{aggr} ausgewählt, so daß $\forall (e, or) \in ka.EOR$: $sa.f_{e,or}^{\text{aggr}} \stackrel{\text{def}}{=} f_{ka}^{\text{aggr}}$ einheitlich definiert werden kann:*

1. *Falls $ka \notin KA_{sa}$, sei $f_{ka}^{\text{aggr}} \stackrel{\text{def}}{=} f^{\text{id}}$. Da ka für sa nicht relevant ist, werden Maßzahlwerte, wie in Def. 4.20 bzw. 4.21 gefordert, identisch dupliziert.*
2. *Falls dagegen $ka \in KA_{sa}$ und $\exists f^{\text{aggr}} \in \mathcal{F}_{sa.T, ka.D}^{\text{aggr}} \forall (e, or) \in ka.EOR$: $sa.mz.f_{e,or}^{\text{aggr}} = f^{\text{aggr}}$, so sei $f_{ka}^{\text{aggr}} \stackrel{\text{def}}{=} f^{\text{aggr}}$. Eine einheitliche Aggregierung ist also möglich.*
3. *Sonst sei f_{ka}^{aggr} nicht aggregierend, da keine gemeinsame „echte“ Aggregierungsfunktion definierbar ist.*[38]

Wiederum heiße (analog zu Def. 4.17) sa je nach Art von f_{ka}^{aggr} BELIEBIG, DISJUNKT, IDENTISCH *oder* NICHT AGGREGIERBAR *über ka.*

sa sei identisch aggregierbar über alle $(e, O, r) \in \mathcal{E} \times \mathcal{OR} \setminus \bigcup_{ka \in KA} ka.EOR$.

Somit werden alle Aggregierungsfunktionen eines summarischen Attributs eindeutig definiert. Da gemäß Def. 4.20 nicht zwei kategorielle Attribute eines Datenraums das gleiche Tripel (e, O, r) beschreiben, können sich gemäß obigem Algorithmus aus den Aggregierungsfunktionen f_{ka}^{aggr} nicht mehrere widersprüchliche Definitionen einer Aggregierungsfunktion $f_{e,(O,r)}^{\text{aggr}}$ ergeben.

Mit der eindeutigen Definition von f_{ka}^{aggr} erfolgt auch die Behandlung mengenwertiger Eigenschaften (bzw. die „disjunkte Aggregierbarkeit“) für alle Eigenschaften eines kategoriellen Attributs gleich. Ist über ein kategorielles Attribut lediglich disjunkt aggregierbar, so müssen auch jeweils zwei Kategorien dieses Attributs bzgl. aller beschriebenen Objektmengen und Eigenschaften gleichermaßen entweder disjunkt sein oder nicht.

Zu den vorangegangenen Definitionen zur Beschreibung von Datenräumen nun ein Beispiel eines typischen, in der Epidemiologie verarbeiteten vierdimensionalen Datenraums *dr* mit den kategoriellen Attributen $KA = \{$*Region, Jahr, Krebsart, Standardpopulation*$\}$ und den summarischen Attributen $SA = \{$*Bevölkerungsgröße, Fallzahl, Standardisierte Rate*$\}$: Sowohl das Attribut *Region* also auch die Angabe der *Standardpopulation* beziehen sich auf die gleiche Dimension *Gebiete*. Der Datenraum beschreibt Personen (P) und Tumoren (T) in den Rollen Studienbevölkerung (r^{S}) und Vergleichs- (Standard-)population (r^{V}), also $OR = \{(P, r^{\text{S}}), (T, r^{\text{S}}), (P, r^{\text{V}})\}$. Die beschriebenen Objektmengen der Attribute verteilen sich wie in Tab. 4.2 dargestellt.

[38] Spezialfälle, in denen ausgenutzt wird, daß die betrachteten Aggregierungsfunktionen $f_{e,or}^{\text{aggr}}$ zwar nicht auf dem gesamten Definitionsbereich, aber zumindest auf einer Teilmenge übereinstimmen, sollen hier keine Sonderbehandlung erfahren.

	(P, r^{S})	(T, r^{S})	(P, r^{V})
Region	•	•	
Jahr	•	•	•
Krebsart		•	
Standard			•

	(P, r^{S})	(T, r^{S})	(P, r^{V})
Bevölkerung	•		
Fallzahl		•	
Std. Rate	•	•	•

Tabelle 4.2: Durch kategorielle bzw. summarische Attribute beschriebene Objektmengen und ihre Rollen

Somit sind etwa die Angaben der Bezugsbevölkerung für unterschiedliche Erkrankungen und Standardpopulationen identisch dupliziert — die Bevölkerung zu (*Oldenburg*, *1998*, *Lungenkrebs*, *Weltstandard*) ist genauso groß wie zu (*Oldenburg*, *1998*, *Brustkrebs*, *Europastandard*). Die Daten der Standardbevölkerung stammen jeweils aus dem gleichen Jahr wie die der Studienpopulation, was man an den Objektmengen des Zeitattributs erkennt. Würde dort die Standardpopulation fehlen, wären auch ihre Daten über diese Dimension dupliziert. Dieses Attribut ist außerdem wieder ein gutes Beispiel für die Nutzung mehrerer Eigenschaften in einem kategoriellen Attribut — in diesem Fall werden gleichzeitig die Erhebungsdaten der Wohnbevölkerung und die Erstdiagnosedaten von Tumoren repräsentiert. Somit hat etwa das summarische Attribut *Fallzahl* die gleiche Aggregierungsfunktion *Summe* über (*Diagnosezeitpunkt*, (*Tumoren*, r^{S})) *und* (*Erhebungszeitpunkt*, (*Personen*, r^{S})). Letztere gehen zwar nicht in die Fallzahl ein, sind aber in diesem Datenraum untrennbar mit der Beschreibung der Diagnosezeitpunkte von Tumoren verknüpft.

Maßzahlwerte sind jeweils eindeutig durch das summarische Attribut und die durch die jeweilige Zelle des Datenraums repräsentierte(n) Objektmenge(n) definiert; insbesondere sind sie unabhängig von anderen Zellen im gleichen Datenraum. Eine derartige Abhängigkeit wäre etwa denkbar zur Angabe von Rängen oder Anteilen des Wertes einer Zelle an der Summe aller Werte einer Dimension oder des gesamten Datenraums (vgl. etwa [FMENR89]). Von dieser Möglichkeit soll hier jedoch abgesehen werden, um eine einfachere Semantik der Werte in den Würfelzellen definieren zu können. Weiterhin wird so eine Restriktion von Datenräumen ohne Neuberechnung von Werten ermöglicht. Schließlich bietet sich auf diese Weise auch eine größere Flexibilität, andere Bezugsgrößen als gerade den vorliegenden Datenraum zu wählen. Trotzdem kann auch das hier vorgestellte Modell derartige Abhängigkeiten modellieren: Zur Darstellung von Anteilen ist ein zusätzliches einschränkendes kategorielles Attribut zur gleichen Objektmenge einzuführen, das die jeweilige Gesamtmenge modelliert; Rangzahlen können durch eine zusätzliche Kapselung des Wissens über die Unterteilung dieser Gesamtmenge und die jeweiligen Maßzahlwerte der anderen Gruppen in der definierenden Funktion des summarischen Attributs spezifiziert werden. Dieses Wissen wird i. a. aus der bisherigen Berechnungshistorie des Datenraums zu gewinnen sein. Die Maßzahlwerte, auf denen die Rangbildung beruht, müssen schließlich vorher bereits berechnet worden sein.

Dadurch, daß ein Datenraum *mehrere* summarische Attribute umfassen kann, die sich vor allem auf *unterschiedliche Teilmengen* der kategoriellen Attribute des Datenraums beziehen können, erhöht sich zweifellos die Komplexität der Spezifikation von *MADEIRA*. Dies erscheint jedoch angemessen, da so eine viel klarere Definition von Operatoren auf multidimensionalen Datenräumen (siehe Abschnitt 4.3) sowie der Zusammengehörigkeit von verarbeiteten Daten im Datenfluß einer Untersuchung (vgl. Kapitel 5) möglich ist. Die nötige redundante Werteverdoppelung läßt sich speichereffizient durch Kapselung von Teilwürfeln als Unterräume über die jeweils relevanten kategoriellen Attribute implementieren. Dieser Ansatz kann ebenfalls dazu beitragen, Bereiche mit leeren Zellen oder Nullwerten in Datenräumen, die typischerweise zusammenhängend auftreten, als Teilwürfel zu abstrahieren, ohne eine redundante Materialisierung nötig zu machen.

Datenbasen und Datenwürfel

An Varianten von Makrodatenräumen werden *Datenbasen* — das sind im Datenbanksystem abgelegte Basisdaten — und *Datenwürfel (Data Cubes)*, also von Datenbasen oder anderen Datenwürfeln im Rahmen einer

Analyse abgeleitete Datensätze, unterschieden.

Datenbasen liefern eine jeweils widerspruchsfreie Datengrundlage, d. h. ihre summarischen Attribute sind wohldefiniert — sie haben für jedes kategorielle Attribut jeweils einheitliche Aggregierungsfunktionen und gewähren die Verträglichkeit von Summierungs- und Aggregierungsfunktion, so daß aus diesen Basisdaten mittels Aggregierung keine widersprüchlichen Daten abzuleiten sind. Darüber hinaus können Datenbasen auch auf Redundanzen verzichten, wenn Daten über die jeweils betrachteten Dimensionen mittels der zugeordneten Aggregierungsfunktionen für gröbere Kategorien abgeleitet werden können. Meistens, aber nicht notwendigerweise, liegen Daten bzgl. einer Dimension (eines kategoriellen Attributs) auf einer (vollständigen) Aggregierungsebene oder zumindest zu disjunkten Kategorien vor. Ist ein summarisches Attribut nicht über das betreffende kategorielle Attribut (bzw. die jeweilige Eigenschaft) aggregierbar, können Datenbasen natürlich auch sinnvollerweise gleichzeitig Daten auf unterschiedlichen Aggregierungsebenen — auch zu verwandten Kategorien — zur Verfügung stellen.

Datenwürfel werden im Verlauf einer Analysesitzung aus Datenbasen und anderen Datenwürfeln abgeleitet. Während die Datenbasen meist fundamentale Maßzahlen wie z. B. Anzahlen oder Summen von Attributwerten aus Mikrodaten enthalten, die also bereits die Summierungsfunktionen der jeweiligen summarischen Attribute eindeutig definieren, ergeben sich diese Funktionen im Fall von Datenwürfeln oft erst aus dem Analyseprozeß. Mit der Widerspruchsfreiheit der zugrundeliegenden Datenbasen und natürlich der korrekten Definition und Durchführung verarbeitender Operationen ist auch die konsistente Beschreibung abgeleiteter summarischer Attribute bzw. entsprechender Datenwürfel auf der Basis repräsentierter Objektmengen möglich. Ein Angebot entsprechender Schemata ermöglicht eine Identifikation gleicher oder überlappender Datenräume ohne Kenntnis der enthaltenen Datenwerte und somit eine effiziente Verarbeitung unter Ausnutzung bereits etwa in einem Cache vorliegender Berechnungsergebnisse (siehe Abschnitt 6.2.3).

Vereinheitlichung von Datenräumen durch Kennzahldimensionen

Unter anderem im Kontext einer einfachen Datenvisualisierung, die summarische und kategorielle Attribute einheitlich behandelt, aber auch zur Vereinfachung der Struktur eines Datenraums etwa für den Datenaustausch mit anderen Systemen[39] möchte man gelegentlich eine Sicht auf einen Datenraum erhalten, die nur ein einzelnes summarisches Attribut erlaubt. Hierzu sollen unter Beibehaltung der Modellierungsmöglichkeiten von *MADEIRA* mehrere summarische Attribute als eine eigene *Kennzahl- (Maßzahl-)dimension* (vgl. [LAW98]) zusammengefaßt werden können. Diese erweitert also den Datenraum um eine zusätzliche Dimension, ein weiteres kategorielles Attribut (das *Kennzahlattribut*), das als Ausprägungen die verschiedenen summarischen Attribute repräsentiert. Jede einzelne Zelle des Datenraums enthält somit nur noch einen Wert; das einzige, „künstliche“ summarische Attribut des Datenraums hat quasi nur noch „Verteilerfunktionalität“. Folgende Definition einer *homogenisierten Sicht* auf einen Datenraum, die zunächst die bestehenden Mengen von Objekten, Domänen und Dimensionen erweitert, konkretisiert diesen Ansatz. Insbesondere werden summarische Attribute sowohl als durch eine Kennzahleigenschaft beschriebene Objekte als auch als Kategorien zur Beschreibung von Ausprägungen dieser Eigenschaft modelliert.

Definition 4.24 (Kennzahldimension und Kennzahlmaßzahl) *Die Objektmenge O enthalte Repräsentanten $O^{\text{kz}} \subset O$ für alle summarischen Attribute, o. B. d. A. $O^{\text{kz}} \stackrel{\text{def}}{=} \mathcal{SA}$ (summarische Attribute sind* KENNZAHLOBJEKTE*). Weiterhin sei $r^{\text{kz}} \in \mathcal{R}$ eine* KENNZAHLROLLE*.*

$D^{\text{kz}} \in \mathcal{DO}$ mit $\mathcal{SA} \subseteq D^{\text{kz}}$ sei der Abschluß von $\mathcal{SA}$ bzgl. logischer Verknüpfungen von Kategorien und diene als KENNZAHLDOMÄNE*. $e^{\text{kz}} \in \mathcal{E}$ sei eine* KENNZAHLEIGENSCHAFT *mit $\mathcal{E}_{D^{\text{kz}}} \stackrel{\text{def}}{=} \{e^{\text{kz}}\}$ und $O_{e^{\text{kz}}} \stackrel{\text{def}}{=} O^{\text{kz}}$, d. h. die Kennzahldomäne diene genau zur Beschreibung der Kennzahleigenschaft, die wiederum nur auf den Kennzahlobjekten definiert sei. Außerdem gelte für $o_{sa} \in O^{\text{kz}}$ und $k_{sa} \in D^{\text{kz}}$ gerade $(o_{sa} \vdash_{e^{\text{kz}}} k_{sa}) \Leftrightarrow (o_{sa} \preceq k_{sa})$.*

Es sei $le^{\text{kz}} \stackrel{\text{def}}{=} (D^{\text{kz}}, \mathcal{SA}) \in \mathcal{L}_{D^{\text{kz}}}$ eine KENNZAHLEBENE*, die alle summarischen Attribute als Kategorien umfaßt, sowie $d^{\text{kz}} \in \mathcal{DI}$ die* KENNZAHLDIMENSION *mit $d^{\text{kz}} \stackrel{\text{def}}{=} (D^{\text{kz}}, \{le_0, le^{\text{kz}}\}, le_0)$, kanonischer Wurzelebene le_0 und $le^{\text{kz}} \preceq le_0$.*

[39] Implementierungen multidimensionaler Datenmodelle sehen oft nur ein summarisches Attribut pro Datenraum vor.

Sei ferner $mz^{kz} \stackrel{def}{=} (T^{kz}, T_0, T^N, F, f^{aggr}) \in \mathcal{M}$ *die in Datenräumen mit Kennzahldimension für das einzige summarische Attribut genutzte* KENNZAHLMASSZAHL *mit dem* KENNZAHLTYP T^{kz}. $T^{kz} \setminus T^N$ *und* T_0 *seien als Vereinigungen der entsprechenden Wertemengen über alle (anderen) Maßzahlen aus* $\mathcal{M}$ *definiert;* T^N *sei disjunkt zu allen anderen Datentypen, spezifiziere aber jeweils für jeden Nullwert dieser Typen einen Repräsentanten mit gleicher Semantik, also insbesondere wiederum die drei Werte* $t_{nexist}, t_{unknown}$ *und* t_{navail}. F *enthalte für alle anderen Maßzahlen als Berechnungsfunktion die Identität, die zusätzlich Nullwerte auf die jeweils entsprechenden Werte aus* T^N *abbilde;* mz^{kz} *sei nie aggregierbar.*[40]

Eine homogenisierte Sicht eines Datenraums entsteht nun aus einem „normalen“ Datenraum, indem alle seine summarischen Attribute in einer Kennzahldimension zusammengefaßt werden und an ihre Stelle ein einzelnes summarisches Attribut, das auf der Kennzahlmaßzahl basiert, gesetzt wird.

Definition 4.25 (Homogenisierte Sichten auf Datenräume) *Zu* $dr = (OR, KA, SA, f^{ext}) \in \mathcal{DR}$ *sei eine* HOMOGENISIERTE SICHT $f^{hom}(dr) \stackrel{def}{=} (OR \cup \{(O^{kz}, r^{kz})\}, KA \cup \{ka^{kz}\}, sa^{kz}, f^{ext\prime}) \in \mathcal{DR}$ *definiert durch ein zusätzliches kategorielles* KENNZAHLATTRIBUT $ka^{kz} \stackrel{def}{=} (D^{kz}, SA, \{(e^{kz}, O^{kz}, r^{kz})\})$ *sowie ein übergeordnetes summarisches Attribut* $sa^{kz} \stackrel{def}{=} (mz^{kz}, OR \cup \{(O^{kz}, r^{kz})\}, f^{sum}, mz^{kz}.f^{aggr})$ *als „Verteiler“–Attribut.*

$f^{sum}(\cdot, \ldots, \cdot, O^{kz}), O^{kz} \subseteq O^{kz}$, *sei für* $|O^{kz}| \neq 1$ *definiert als* t_{nexist} *— es werden nur einzelne Maßzahlen, nicht Gruppen davon beschrieben.*[41] *Ansonsten gelte für* $OR = \{(O_1, r_1), \ldots, (O_n, r_n)\}$, *ein* $sa \in SA$ *mit* $sa.OR = \{(O_{n(1)}, r_{n(1)}), \ldots, (O_{n(v)}, r_{n(v)})\} \subseteq OR$ *und* $U_i \subseteq O_i, i = 1, \ldots, n$*:*

$$f^{sum}(U_1, \ldots, U_n, \{sa\}) \stackrel{def}{=} sa.f^{sum}(U_{n(1)}, \ldots, U_{n(v)}) \quad ,$$

wobei Nullwerte jeweils durch ihre entsprechenden Repräsentanten ersetzt werden.

Die Extension $f^{ext\prime}_{sa^{kz}}$ *von* $f^{hom}(dr)$ *ist somit — unter Beachtung der Nullwertumbenennung — gegeben durch* $f^{ext\prime}_{sa^{kz}}(\cdot, \ldots, \cdot, sa) \stackrel{def}{=} f^{ext}_{sa}(\cdot, \ldots, \cdot)$. *Die Instanzen der beschriebenen Objektmengen bleiben unverändert, ergänzt um SA als Instanzenmenge der Kennzahlobjekte.*

Es sei nochmals darauf hingewiesen, daß hier kein Operator des multidimensionalen Datenmodells definiert wurde, der aus einem Datenraum einen anderen mit Kennzahldimension *erzeugt*. Vielmehr wurde eine weitere Modellierungsvariante als alternative *Sicht* auf einen Datenraum eingeführt, die die Beschreibung bestimmter Operationen erleichtern wird.

4.2.7 Zusammengesetzte Dimensionen — ein Kompromiß zwischen relational basierter und streng multidimensionaler Modellierung von Makrodatenräumen

Das bisher vorgestellte Datenmodell geht implizit davon aus, daß Datenräume weitgehend vollständig gefüllt sind, d. h. wenig leere Zellen enthalten. Ein Makrodatenraum umfaßt also Daten zum vollständigen kartesischen Produkt der Kategorienmengen seiner kategoriellen Attribute. Auch wenn diese Annahme im hier als Leitbeispiel verwendeten Anwendungsgebiet der Epidemiologie meist gerechtfertigt ist, so gibt es doch auch Fälle, in denen in diesem Sinne dünn besetzte Datenräume zu einem hohen Speicheraufwand sowie einer unangemessen Datenvisualisierung (etwa in weitgehend „leeren“ Tabellen) führen können.

Beispiele hierfür bilden zum einen Fälle, in denen Ausprägungen zweier Dimensionen in den zugrundeliegenden Basisdaten in Abhängigkeit voneinander auftreten, z. B. schließen sich bestimmte Kombinationen von Lokalisation (Ort des Erkrankungsherdes) und Histologie (also befallenem Gewebetyp) einer Krebserkrankung

[40] Da rekursive Schachtelungen von Kennzahldimensionen bzw. homogenisierte Sichten auf homogenisierte Sichten von Datenräumen wenig sinnvoll erscheinen, seien summarische Attribute, die auf mz^{kz} basieren, von obigen Kennzahlobjekten sowie der Kennzahldomäne ausgenommen. Es gelte also $O^{kz} \stackrel{def}{=} \{sa \in \mathcal{SA} : sa.mz \neq mz^{kz}\}$, und diese Menge sei auch die Grundlage für die Definition von D^{kz}.

[41] Während der Nullwert t_{nexist} i. a. aussagt, daß keine entsprechenden Objekte existieren, soll er hier bedeuten: „die *Objektmenge* existiert nicht“.

gegenseitig aus.[42] Zum anderen kann es auch erwünscht sein, nur einzelne Kategorien eines kategoriellen Attributs durch Kategorien einer anderen Dimension zu unterteilen, während alle anderen Kategorien bzgl. dieser Dimension einen Gesamtwert repräsentieren sollen. Etwa werden häufig in Aufstellungen über Krebserkrankungen, die sonst nicht nach dem Geschlecht unterscheiden, lediglich Brustkrebsfälle entsprechend differenziert, da hier doch zwei zwischen den Geschlechtern sehr unterschiedliche und unterschiedlich häufige Erkrankungsarten vorliegen.

Eine relational basierte Modellierung von Makrodaten im Sinne von ROLAP „löst“ dieses Problem, indem sie sowieso nur die belegten Zellen als „Makrodaten–Tupel“ einer Makrodatenrelation betrachtet (siehe Abschnitt 2.3.3). Hier soll jedoch eine Lösung vorgestellt werden, die sowohl die streng multidimensionale Betrachtungsweise beibehält als auch eine Konzentration auf besetzte bzw. interessierende Zellen oder Teilbereiche eines Datenraums erlaubt.[43]

Um aus den Kombinationen von Kategorien zweier im obigen Sinne voneinander abhängiger kategorieller Attribute jeweils nur eine interessierende Teilmenge auswählen zu können, müssen Paare von Kategorien wieder als eigenständige Kategorien einer neuen Domäne modelliert werden, die die Elemente eines „zusammengesetzten“ kategoriellen Attributs bilden können. Derartige zusammengesetzte Kategorien grenzen jeweils gleichzeitig zwei Eigenschaften der betrachteten Objektmenge konjunktiv ein. In Abb. 4.6 ist ein Beispiel anhand eines zweidimensionalen Datenraums dargestellt: Es sollen zum einen einzelne Zellen, also Kategorienkombinationen ausgewählt werden, und zum anderen interessiert für die Kategorie *d* aus Dimension D_2 nicht die Aufteilung gemäß der Kategorien aus D_1.

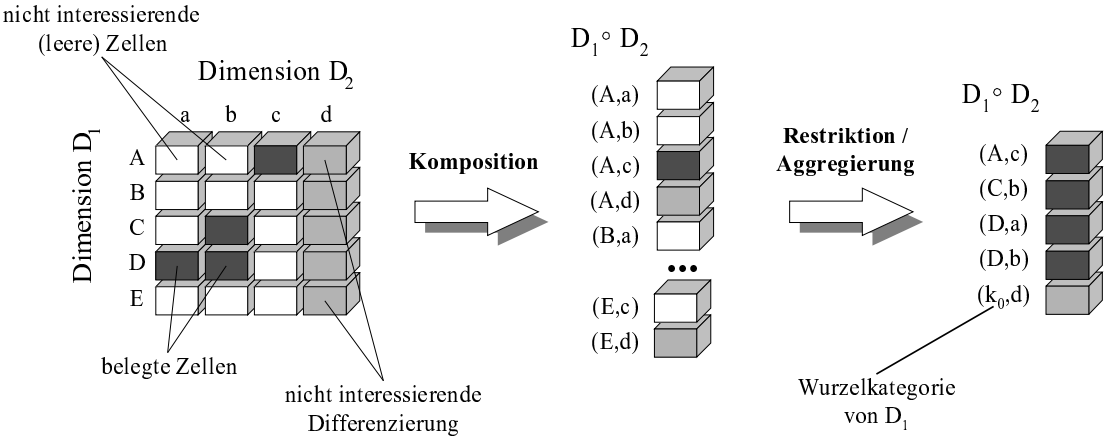

Abbildung 4.6: Bildung und Nutzung kategorieller Attribute mit zusammengesetzten Kategorien

Wie man leicht sieht, bleibt allein bei einem derartigen Zusammensetzen zweier kategorieller Attribute unter Bildung aller möglicher Paare von Kategorien die Menge an Zellen eines Datenraums sowie deren Semantik exakt gleich, nur die Dimensionalität des Datenraums verringert sich um eins.[44] Würde man diese Komposition von Dimensionen fortsetzen, erhielte man schließlich eine eindimensionale, relationale Repräsentation von Makrodaten. Insofern stellt die Bildung zusammengesetzter Dimensionen einen Kompromiß zwischen relationaler und multidimensionaler Modellierung dar.

Die beiden obigen Beispiele lassen sich mit diesen Mitteln einfach bearbeiten: Im ersten Fall werden einfach die interessierenden zusammengesetzten Kategorien ausgewählt; im zweiten Fall werden für alle nicht zu

[42] Hier, wie auch in vielen ähnlichen Fällen, kann man eigentlich nicht vom völligen Ausschluß einer Ausprägungskombination (einem strukturellen Nullwert entsprechend) reden; diese ist meist nur äußerst unwahrscheinlich.

[43] In Abschnitt 4.2.8 werden wir in ähnlichen Überlegungen im Kontext der Unterscheidung von Datenschema und Extension noch einmal auf diese zwei Sichtweisen der multidimensionalen Datenmodellierung, ROLAP vs. MOLAP, zurückkommen.

[44] Erst im Anschluß an die Komposition kann man den betrachteten Datenraum in weiteren Operationen näher eingrenzen.

untergliedernden Kategorien einfach Kompositionen mit der Wurzelkategorie der zweiten betrachteten Dimension gebildet.

Folgende Definition faßt die Formalisierung der nötigen Strukturen zusammen:

Definition 4.26 (Zusammengesetzte Kategorien und Dimensionen) *Eine aus den Domänen $D_1, D_2 \in \mathcal{DO}$* ZUSAMMENGESETZTE DOMÄNE *$D = D_1 \circ^D D_2 \in \mathcal{DO}$ sei definiert als der Abschluß von $D_1 \times D_2$ gegenüber logischen Verknüpfungen von Kategorien (gemäß Def. 4.3). Im folgenden sollen nur die Kategorien $k \in D$ interessieren, die sich als (k_1, k_2) mit $k_1 \in D_1, k_2 \in D_2$ schreiben lassen.*[45]

D beschreibt ZUSAMMENGESETZTE EIGENSCHAFTEN *$\mathcal{E}_D \stackrel{\text{def}}{=} \{e_1 \circ^E e_2 \mid e_1 \in \mathcal{E}_{D_1} \wedge e_2 \in \mathcal{E}_{D_2}\} \subset \mathcal{E}$, wobei für $e = e_1 \circ e_2$ gilt: $O_e \stackrel{\text{def}}{=} O_{e_1} \cap O_{e_2} \wedge \forall k = (k_1, k_2) \in D\, \forall o \in O_e : (o \vdash_e k) \Leftrightarrow (o \vdash_{e_1} k_1 \wedge o \vdash_{e_2} k_2)$. k heiße dann auch* ZUSAMMENGESETZTE KATEGORIE.

Eine ZUSAMMENGESETZTE AGGREGIERUNGSEBENE *$le \in \mathcal{L}_D$ sei durch $le = le_1 \circ^L le_2 \stackrel{\text{def}}{=} (D, K_1 \times K_2)$ mit $le_1 = (D_1, K_1) \in \mathcal{L}_{D_1}$ und $le_2 = (D_2, K_2) \in \mathcal{L}_{D_2}$ definiert.*

Entsprechend sei die ZUSAMMENGESETZTE DIMENSION *$d = d_1 \circ^d d_2 \stackrel{\text{def}}{=} (D, L, le_0)$ auf D eindeutig über die Dimensionen d_1 und d_2 zu D_1 bzw. D_2 definiert durch $L \stackrel{\text{def}}{=} \{le_1 \circ le_2 \mid le_1 \in d_1.L \wedge le_2 \in d_2.L\}$, $le_0 \stackrel{\text{def}}{=} d_1.le_0 \circ d_2.le_0$.*

Schließlich bezeichnen wir als ZUSAMMENGESETZTES KATEGORIELLES ATTRIBUT *ein kategorielles Attribut über einer zusammengesetzten Dimension.*

Zwei Ebenen $le = (le_1 \circ le_2), le' = (le'_1 \circ le'_2)$ einer zusammengesetzten Domäne *verfeinern* sich gegenseitig ($le \preceq le'$), falls dies auch für le_1 und le'_1 sowie le_2 und le'_2 gilt. le ist zu le' *aggregierbar* ($le \lhd le'$), falls le ganz le' verfeinert oder sowohl le_1 als auch le_2 nur jeweils eine Kategorie aus le'_1 bzw. le'_2 unterteilen.

Meistens wird man gemäß obiger Definition *unterschiedliche* Dimensionen bzw. deren Kategorien miteinander verknüpfen, aber auch die Betrachtung unterschiedlicher Eigenschaften über der gleichen Dimension ist auf diese Weise durchaus sinnvoll. Auch können mehr als zwei Komponenten komponiert werden — die Zusammensetzung ist, wie man leicht sieht, assoziativ und kommutativ (modulo Äquivalenz von Kategorien).

Enthält ein Datenraum ein kategorielles Attribut über zusammengesetzten Kategorien, so unterscheidet sich dieses in der Regel bzgl. aller seiner „Komponenteneigenschaften“ von allen anderen kategoriellen Attributen, da sonst wiederum zwei Attribute redundante Informationen repräsentieren würden. Entsprechend ist auch die Komposition einer Eigenschaft mit sich selbst selten (höchstens für mengenwertige Eigenschaften) sinnvoll. Beides braucht jedoch nicht explizit gefordert bzw. ausgeschlossen zu werden — die Formalisierung der entsprechenden Bedingung in Def. 4.20 ist für unsere Zwecke ausreichend.

Im weiteren Verlauf dieses Abschnitts werden nun Charakteristika zusammengesetzter Aggregierungsebenen näher untersucht. Wiederum trifft man hier — wie schon in Abschnitt 4.2.3 — auf zwei grundlegene Aspekte:

1. die *Disjunktheit* von Kategorien bzw. die *Mengenwertigkeit* von Eigenschaften, deren Berücksichtigung wesentlich für eine korrekte und differenzierte Durchführung von Datenaggregierungen ist, sowie
2. die *Subsumtion* von Kategorien bzw. die *Wohlgeformtheit* von Aggregierungsebenen, deren Betrachtung auf eine „schöne“ Modellierung der Anwendung, klare Strukturen und Navigationspfade abzielt.

Bemerkenswert ist, daß diese Sichtweise entsprechend der Definition zusammengesetzter Kategorien auch Aufschluß über Eigenschaften von Zellen eines beliebigen Datenraums gibt. So sind Disjunktheit und Verwandtschaft von Kategorien unmittelbar übertragbar auf das Ineinander–Enthaltensein von Objektmengen, die durch zwei Würfelzellen repräsentiert werden, deren Koordinaten sich lediglich hinsichtlich eines kategoriellen Attributs unterscheiden.

Zunächst kurz zum erstgenannten Aspekt, der relativ schnell abzuhandeln ist (vgl. auch [Sat81]): Aus einwertigen Ebenen werden trivialerweise wiederum einwertige Aggregierungsebenen zusammengesetzt, und die

[45] Dies stellt keine Einschränkung dar, gemäß der Definitionen aus Abschnitt 4.2.1 lassen sich — soweit nötig — alle getroffenen Aussagen auf ganz *D* übertragen.

Kombinationen einer mengenwertigen mit einer ein- oder mengenwertigen Ebene ist dann auf jeden Fall wieder mengenwertig, wenn letztere (genauer: die Disjunktion ihrer Kategorien) äquivalent zur Wurzelkategorie ist. Über andere Fälle von Ebenenkombinationen ist keine allgemeingültige Aussage möglich.

Die Wohlgeformtheit zusammengesetzter Aggregierungsebenen verdient dagegen eine ausführlichere Betrachtung: Wann sind aus wohlgeformten Ebenen zusammengesetzte Ebenen wiederum wohlgeformt? Diese Fragestellung führt recht schnell zum Begriff der „Unabhängigkeit“ von Eigenschaften bzw. Domänen, wie auch schon das eingangs des Abschnitts gegebene Beispiel der Kombination von Lokalisation und Histologie von Krebserkrankungen nahegelegt hat. Wenn bestimmte Kombinationen von Kategorien niemals gemeinsam auftreten, hier also eine Abhängigkeit zwischen zwei Eigenschaften besteht, so ist eine Aggregierungsebene, die diese zusammengesetzte Kategorie enthält, per Definition nicht mehr wohlgeformt. Während dieser Fall bereits bei der Kombination einwertiger Ebenen bzw. Eigenschaften auftreten kann, lassen sich darüber hinaus im mengenwertigen Fall relativ leicht Situationen konstruieren, in denen aufgrund bestehender Abhängigkeiten in aus wohlgeformten Ebenen zusammengesetzten Ebenen verwandte Kategorien entstehen, ohne daß die zugeordneten Objektmengen leer sind.

Konkretisieren wir also zunächst die Vorstellung von Abhängigkeiten zwischen (nicht notwendigerweise verschiedenen) Domänen sowie Eigenschaften:

Definition 4.27 (Unabhängigkeit von Domänen) *Gegeben seien zwei Domänen $D_1, D_2 \in \mathcal{DO}$, zwei Eigenschaften $e_1 \in \mathcal{E}_{D_1}, e_2 \in \mathcal{E}_{D_2}$ sowie $\emptyset \neq O \subseteq \mathcal{O}_{e_1} \cap \mathcal{O}_{e_2}$ eine Menge durch beide Eigenschaften beschreibbarer Objekte. Dann heißen D_1 und D_2 bzgl. O, e_1 und e_2* UNABHÄNGIG *voneinander, falls*

$$\forall k_1 \in D_1, k_2 \in D_2 : (O \cap \mathcal{O}_{e_1,k_1} \neq \emptyset \wedge O \cap \mathcal{O}_{e_2,k_2} \neq \emptyset) \Rightarrow (O \cap \mathcal{O}_{e_1 \circ e_2,(k_1,k_2)} \neq \emptyset) \ ,$$

mögliche Ausprägungen also auch in beliebiger Kombination auftreten können.[46]

Bevor nun der Zusammenhang zwischen der Unabhängigkeit von Domänen und der Wohlgeformtheit zusammengesetzter Aggregierungsebenen über diesen untersucht werden kann, ist zunächst der Begriff der Wohlgeformtheit (und damit auch die zugrundeliegende Subsumtion bzw. Verwandtschaft von Kategorien) auf einzelne Eigenschaften und bestimmte Objektmengen einzuschränken. So wie schon die Unabhängigkeit von Domänen entsprechend spezifisch definiert wurde, ist es in diesem Kontext angesichts der offenbar hiermit bestehenden Zusammenhänge ebenfalls nicht sinnvoll, Wohlgeformtheit wie bisher allgemeingültig formulieren. Je nach betrachteter Eigenschaft und Objektmenge ergeben sich nämlich aus der diesbezüglichen Abhängigkeit der Domänen unterschiedliche Beziehungen zwischen den Kategorien zusammengesetzter Ebenen.

Definition 4.28 (Eingeschränkt wohlgeformte Aggregierungsebenen) *Beziehen sich Subsumtion oder Verwandtschaft von Kategorien $k, k' \in D$ (mit $D \in \mathcal{DO}$) lediglich auf einzelne Eigenschaften $e \in \mathcal{E}_D$ sowie eine Teilmenge $\emptyset \neq O \subseteq \mathcal{O}_e$, werde dies durch eine Indizierung mit diesen bezeichnet, also $(k \preceq_{e,O} k') \Leftrightarrow \forall o \in O$: $(o \vdash_e k \Rightarrow o \vdash_e k')$ („k'* SUBSUMIERT *k bzgl. e und O“) bzw. $(k \|_{e,O} k') \Leftrightarrow (k \preceq_{e,O} k') \vee (k' \preceq_{e,O} k)$ („k' und k sind bzgl. e und O* VERWANDT“*).*

Analog heiße eine Aggregierungsebene $le \in \mathcal{L}_D$ WOHLGEFORMT *bzgl. e und O, falls*

$$(le.K = \{k\} \wedge O \cap \mathcal{O}_{e,k} \neq \emptyset) \vee (|le.K| > 1 \wedge \forall k_1 \neq k_2 \in le.K : k_1 \nparallel_{e,O} k_2) \ .$$

$\mathcal{L}^{\omega}_{e,O} \subseteq \mathcal{L}$ sei die Mengen aller bzgl. e und O wohlgeformten Aggregierungsebenen.

Auf der Grundlage dieser Definitionen läßt sich nun formalisieren und beweisen, was auch intuitiv einleuchtend ist: Aus wohlgeformten Aggregierungsebenen unabhängiger Domänen werden wiederum wohlgeformte zusammengesetzte Ebenen.

[46] Betrachtet man Kategorien k_1, k_2 als Ausprägungen zweier durch die Eigenschaften e_1, e_2 definierter Zufallsvariablen in einem auf Grundlage der Objektmenge O definierten Wahrscheinlichkeitsraum, so entspricht obiger Ausdruck einer wahrscheinlichkeitstheoretischen Aussage der Form $\forall k_1, k_2 : P(k_1) \neq 0 \wedge P(k_2) \neq 0 \Rightarrow P(k_1, k_2) \neq 0$. Dies ist offenbar eine schwächere Forderung als die allgemeine Definition der Unabhängigkeit von e_1 und e_2: $\forall k_1, k_2 : P(k_1, k_2) = P(k_1) \cdot P(k_2)$. Somit kann hier, auch ohne in *MADEIRA* einen Wahrscheinlichkeitsbegriff zu formalisieren, eine für unsere Zwecke ausreichende und mit dem üblichen Verständnis von „Unabhängigkeit“ verträgliche Definition formuliert werden.

Lemma 4.1 (Wohlgeformtheit zusammengesetzter Aggregierungsebenen) *Seien $D_1, D_2 \in \mathcal{DO}$ zwei Domänen, $e_1 \in \mathcal{E}_{D_1}, e_2 \in \mathcal{E}_{D_2}$ zwei Eigenschaften auf diesen Domänen sowie $\emptyset \neq O \subseteq O_{e_1} \cap O_{e_2}$ eine Menge durch beide Eigenschaften beschreibbarer Objekte, so daß D_1 und D_2 unabhängig sind bzgl. O, e_1 und e_2. Weiterhin seien $le_1 \in \mathcal{L}^{\omega}_{e_1,O}$, $le_2 \in \mathcal{L}^{\omega}_{e_2,O}$ zwei bzgl. dieser Eigenschaften und O wohlgeformte Aggregierungsebenen.*

Dann ist (mit $e = e_1 \circ e_2$) $le = le_1 \circ le_2 \in \mathcal{L}^{\omega}_{e,O}$ wohlgeformt bzgl. e und O.

Beweis: Für $le.K = \{(k_1, k_2)\}$ einelementig folgt die Behauptung direkt aus der Wohlgeformtheit von le_1 und le_2 sowie der Unabhängigkeit der Domänen. Sei also im folgenden $|le.K| > 1$.

Annahme: le nicht wohlgeformt, d. h. $\exists k \neq k' \in le.K : k \,||_{e,O}\, k'$ (sei o. B. d. A. $k \preceq_{e,O} k'$). Es soll gezeigt werden, daß in diesem Fall zumindest eine der Ausgangsebenen nicht wohlgeformt sein kann. Sei $k = (k_1, k_2), k' = (k'_1, k'_2)$ und o. B. d. A. $k_1 \neq k'_1$. Es gilt:

$$
\begin{array}{ll}
 & k \preceq_{e,O} k' \\
\Longleftrightarrow & (k_1, k_2) \preceq_{e,O} (k'_1, k'_2) \\
\Longleftrightarrow & O \cap O_{e,(k_1,k_2)} \subseteq O \cap O_{e,(k'_1,k'_2)} \\
\Longleftrightarrow & O \cap O_{e_1,k_1} \cap O_{e_2,k_2} \subseteq O \cap O_{e_1,k'_1} \cap O_{e_2,k'_2} \\
\Longleftrightarrow & O \cap O_{e_1,k_1} \cap O_{e_2,k_2} \cap (O_{e_1,\neg k'_1} \cup O_{e_2,\neg k'_2}) = \emptyset \\
\Longrightarrow & O \cap O_{e_1,k_1} \cap O_{e_2,k_2} \cap O_{e_1,\neg k'_1} = \emptyset \\
\Longleftrightarrow & (O \cap O_{e_1 \circ e_2,(k_1 \wedge \neg k'_1,k_2)} = \emptyset \\
\overset{\text{Unabh.}}{\Longrightarrow} & (O \cap O_{e_1,k_1} \cap O_{e_1,\neg k'_1} = \emptyset) \vee (O \cap O_{e_2,k_2} = \emptyset) \\
\Longleftrightarrow & (O \cap O_{e_1,k_1} \subseteq O \cap O_{e_1,k'_1}) \vee (O \cap O_{e_2,k_2} = \emptyset) \\
\Longleftrightarrow & (k_1 \preceq_{e_1,O} k'_1) \vee (O \cap O_{e_2,k_2} = \emptyset) \\
\Longrightarrow & \text{Widerspruch zur Wohlgeformtheit von } le_1 \text{ und } le_2.
\end{array}
$$

q.e.d.

Gezeigt wurde hier nur eine hinreichende Bedingung für die Wohlgeformtheit zusammengesetzter Aggregierungsebenen. Der Umkehrschluß, also die Implikation der Unabhängigkeit von D_1 und D_2 aus der Wohlgeformtheit von *le* gilt natürlich nicht, da die betrachteten Ebenen nur Teile der gesamten Domäne umfassen.[47] Als einfache notwendige Bedingung läßt sich (trivialerweise) die Wohlgeformtheit der beiden Ausgangsebenen anführen.

Die dargestellten Zusammenhänge sollten im Anwendungsfall jeweils die Entscheidungsgrundlage dafür bieten, inwiefern aus zwei bestehenden Dimensionen zusammengesetzte Dimensionen genutzt werden sollen, bzw. zur Charakterisierung der entstehenden zusammengesetzten Ebenen dienen.

Für Mikrodaten spielen zusammengesetzte Kategorien üblicherweise keine Rolle. Aber auch zur Beschreibung von Makrodaten sollten sie — wenn sie auch in Einzelfällen bestimmte Untersuchungen entscheidend vereinfachen können — eher als Ausnahmefall angesehen werden, da sie vom intuitiv klaren, grundlegenden Konzept der multidimensionalen Datenmodellierung abweichen und außerdem auch aufgrund ihrer komplexen Struktur nicht so effizient zu verarbeiten sind.[48] Aus diesem Grund wurde *MADEIRA* auch erst in diesem Abschnitt um die Definition zusammengesetzter Kategorien ergänzt, anstatt entsprechende Konstrukte (Tupel von Kategorien und Eigenschaften) gleich als Grundlage der Modellierung in Abschnitt 4.2.1 einzuführen. Stets ist

[47] Folgern ließe sich dagegen die Unabhängigkeit zweier Domänen, die auf den Abschluß der Kategorienmengen der betrachteten Ebenen bzgl. logischer Verknüpfungen reduziert wären. Dies soll hier jedoch nicht näher ausgeführt werden, da die Betrachtung der gesamten Domänen normalerweise ausreichend ist.

[48] Eine Implementierung zusammengesetzter Dimensionen wird im allgemeinen darauf verzichten, diese wie die einfachen Dimensionen vollständig im Hauptspeicher zu materialisieren, da mit dem Kreuzprodukt der Kategorien und Ebenen der Speicheraufwand beträchtlich (quadratisch) steigt. Statt dessen werden lediglich Konstrukte für Tupel von Kategorien bereitgestellt, die jeweils die zugrundeliegenden Basisdimensionen referenzieren.

bei der Definition von Domänen und Dimensionen bzw. Datenräumen zu überlegen, ob anstelle der Nutzung zusammengesetzter Kategorien aus zwei Domänen

- Eigenschaften nicht auch — unter Einführung zusätzlicher Aggregierungsebenen, die auch ggf. nur einzelne Kategorien verfeinern — durch eine „gewöhnliche“ gemeinsame Dimension *adäquat* darzustellen sind. So ist etwa die Dignität (der Grad des bösartigen Verhaltens) eines Tumors keine eigenständige Domäne, sondern definiert nur zusammenfassende Kategorien und eine entsprechende Aggregierungsebene in der Dimension der Histologie von Tumoren.

- „geschachtelt“ definierte Eigenschaften bzw. kategorielle Attribute (etwa „der Wohnort des Arztes des Tumorfalles“) nicht auch als unabhängige, ggf. mit einer entsprechenden Rolle gekennzeichnete Attribute (hier also „Wohnort“ neben dem Attribut zum „behandelnden Arzt“) modelliert werden können. Diese Entscheidung hängt sicher auch mit der Anzahl der so entstehenden leeren Zellen im Datenraum zusammen.

- in einem Fall, der dem zweiten zum Anfang dieses Abschnitts genannten Beispiel entspricht (nur einzelne Kategorien werden unterteilt), nicht auch eine Modellierung als gemeinsam verarbeitete „Vereinigung“ zweier getrennter Datenräumen in Frage kommt (vgl. Abschnitt 4.3.4). Von den zwei Datenräumen würde der eine die zu unterteilenden Kategorien des ersten kategoriellen Attributs und das zweite Attribut enthalten und der andere alle anderen Kategorien des ersten Attributs umfassen, wobei das zweite Attribut als einschränkendes Attribut einen Gesamtwert repräsentiert.

Eine weitere Überlegung zielt auf eine mögliche Implementierungsvariante von *MADEIRA*: Anstatt zusammengesetzte Dimensionen lediglich dazu zu nutzen, dünn besetzte Datenräume zu komprimieren, könnten alternativ auch die Datenmatrizen transparent für den Benutzer mit gängigen Kompressionstechniken, etwa der Lauflängencodierung [Gol66], oder Chunking–Verfahren [SS94] bearbeitet werden. Diesen Techniken kommt zugute, daß leere bzw. nicht–leere Zellen in einem Datenraum oftmals nicht einzeln, sondern (ggf. nach geeigneter Sortierung der Würfeldimensionen) in vollständigen mehrdimensionalen Unterräumen auftreten, die (ab einer bestimmten Mindestgröße) als eigene Nullwert–Teilwürfel implementiert werden können (vgl. etwa [Fro96]).

4.2.8 Unterscheidung von Datenschema und Datenbankextension

Abschließend soll noch einmal ein Aspekt angesprochen werden, der häufig im Kontext multidimensionaler Datenmodellierung diskutiert wird: die Trennung von Schema und Extension, insbesondere im Hinblick auf Dimensionen und kategorielle Attribute eines Datenraums. Das Schema einer Datenbank beschreibt mittels der Sprache des jeweiligen Datenmodells Objekte einer gegebenen Anwendung, über die Daten gespeichert werden sollen, ihre Eigenschaften und Beziehungen zueinander [Vos99]. Es wird im Rahmen eines Datenbankentwurfs spezifiziert und ist während des Bestehens der durch die Metadaten des Schemas beschriebenen Datenbank weitgehend stabil [EN94]. Demgegenüber bildet die Extension, also die Menge der Ausprägungen der beschriebenen Objekte, Eigenschaften und Beziehungen, den Zustand der Datenbank, der laufenden Änderungen während des Systembetriebs unterworfen ist.

Im Sinne dieser Begriffsbildung definieren folgende zentrale Komponenten der hier vorgenommenen multidimensionalen Modellierung das Schema eines Datenbestandes:

- Objekte, Eigenschaften und Rollen bezeichnen die modellierten Einheiten der realen Welt. Sie werden als atomare Entitätstypen eingeführt, um mit ihrer Hilfe die Semantik der weiteren Modellkonstrukte spezifizieren zu können.

- Domänen, Kategorien, Aggregierungsebenen und Dimensionen definieren Typen bzw. Wertebereiche für qualifizierende Eigenschaften, einschließlich Angaben zur Semantik dieser Werte gemäß Kategoriensubsumtion sowie Kategorienhierarchien.

- Datentypen, Aggregierungsfunktionen und Maßzahlen dienen entsprechend zur Spezifikation von Typen und semantischen Angaben für quantifizierende Eigenschaften.
- Attribute definieren Einschränkungen auf den allgemein definierten Wertebereichen (insbesondere die kategoriellen Attribute) sowie eine anwendungsspezifisch konkretisierte Semantik (vor allem im Fall der summarischen Attribute) für modellierte Datenbestände.
- Datenräume schließlich modellieren jeweils zusammengehörige, einheitlich aufgebaute Datenbestände aus Mikro- bzw. Makrodaten.

Die Extension einer Datenbank bilden nun die jeweiligen Datenwerte in den Mikro– und Makro–Datenräumen zu den Instanzen der betrachteten Objektmengen. Indem die durch einen Datenraum beschriebenen Objektmengen durch konkrete Objekte (mit konkreten Eigenschaftswerten, den vorliegenden Basisdaten entsprechend) instantiiert werden, wird für einen Mikrodatenraum bereits die Extension definiert. Werden hieraus gemäß der Summierungsfunktionen von summarischen Attributen unter Berücksichtigung der durch kategorielle Attribute gegebenen Klassifizierungen Maßzahlwerte bestimmt, ergibt sich die Extension eines Makrodatenraums.

Kategorielle und summarische Attribute nehmen in gewisser Weise eine Sonderstellung zwischen Schema und Extension ein: Auch die Kategorienmenge eines kategoriellen Attributs könnte als „Extension einer Dimension“ bzw. die konkrete Summierungsfunktion eines summarischen Attributs als „Extension einer Maßzahl“ angesehen werden. In dieser Arbeit steht jedoch der durch solche Attribute modellierte Datenraum im Vordergrund — zur Modellierungszeit werden Struktur und Größe eindeutig festgelegt, nur die Zellwerte sind dynamisch. So ist etwa der Falldatenbestand eines Krebsregisters gegeben durch einen *festen* regionalen Bezug (also eine Menge von Gebietskategorien, z. B. alle Gemeinden Niedersachsens), eine *feste* Menge erfaßter Erkrankungen (etwa alle vierstelligen ICD–Codierungen für Krebserkrankungen) etc. Entsprechend sieht *MADEIRA* die Attribute als Teil der Modellierung — nur die Zellinhalte ändern sich mit dem Einfügen beispielsweise neuer Falldaten (also dem Erweitern der zugrundeliegenden Objektmengen).

Natürlich kann sich auch in einem derartig modellierten Datenbestand von Zeit zu Zeit die Menge der interessierenden Attribute bzw. Wertebereiche ändern. Gerade eine Fortschreibung über die Zeit macht früher oder später eine Erweiterung der Zeitdimension nötig. In diesem Fall ist — analog zur Änderung von Datentypen in relationalen Datenbanksystemen — das Datenschema zu modifizieren und die Extension entsprechend anzupassen.

In diesem Kontext muß auch der Aspekt der Schema–Evolution allgemein angesprochen werden. Im Laufe einer Datenanalysesitzung werden viele neue Datenwürfel erzeugt. Diese sind zwar oftmals flüchtig und werden nur temporär benötigt, unter Umständen sollen sie jedoch auch (etwa zur Effizienzsteigerung nachfolgender Analysen im Rahmen eines Caching) persistent gemacht werden. In diesem Fall erweitert sich also das Datenschema um neue Datenräume — ein recht einfacher Fall von Schema–Evolution (vgl. Abb. 4.7, die auch noch einmal den Zusammenhang von Basisdaten, Mikrodatenräumen, Datenbasen und Datenwürfeln veranschaulicht). Mit abgeleiteten Datenwürfeln entstehen häufig, der neuen Datenverknüpfung entsprechend, auch neue summarische Attribute, während in vielen Fällen die jeweils verbleibenden kategoriellen Attribute von den Ausgangsdaten übernommen werden können und nur ggf. neue einschränkende Attribute als Zusammenfassung zugrundeliegender klassifizierender Attribute definiert werden. In Abschnitt 4.3 werden diese Zusammenhänge genauer betrachtet.

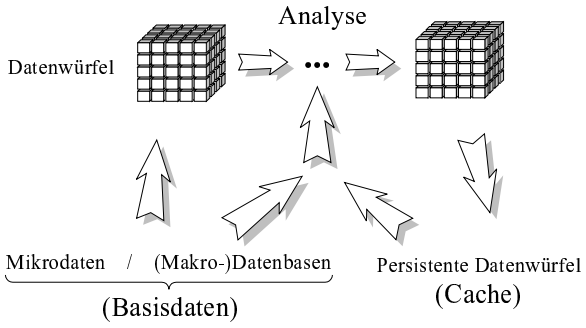

Abbildung 4.7: Basisdaten, Datenbasen und -würfel

Gegenüber der mit *MADEIRA* verfolgten streng multidimensionalen Sicht der Datenmodellierung ergibt

sich aus einer auf einer relationalen Datenspeicherung basierenden Modellierung multidimensionaler Daten ein etwas anderes Verständnis für die Trennung von Datenschema und Extension. In einem Sternschema (vgl. Abschnitt 2.3.3) ist eine Relation von Makrodaten gegeben durch eine Teilmenge des kartesischen Produkts über den Domänen von summarischen *und* kategoriellen Attributen, wobei als Domäne der kategoriellen Attribute die gesamte zugrundeliegende Dimension angesehen wird. Somit zählt die Menge der in einem Datenbestand vorliegenden Kategorien eines Attributs mit zur Datenbankextension (siehe etwa [GL97] oder die Diskussion verschiedener Datenmodelle in [SBHD98]).

Die relationale Art der Betrachtung orientiert sich an einer tupelbasierten Sicht auf Makrodaten, die gerade in Anwendungsgebieten mit vielen extrem dünn besetzten Datenräumen, die also viele strukturelle Nullwerte enthalten, einer speichereffizienteren Datenspeicherung entspricht. Auch konzeptionell mag dies in solchen Fällen die geeignetere Sicht sein, wenn Datenräume eben nicht als vollständige mehrdimensionale Felder, sondern als Sammlung von Zellen angesehen werden. Gerade auf dem speziellen Anwendungsgebiet der Epidemiologie erscheint jedoch der bisher verfolgte Ansatz geeigneter, wie auch obiges Beispiel verdeutlicht haben sollte.

4.3 Operationen auf multidimensionalen Daten

Nachdem im vorangegangenen Abschnitt die Strukturen der von *VIOLA* zu verarbeitenden Daten genau spezifiziert wurden, werden im folgenden die Operatoren vorgestellt, die die Basis für die Verarbeitungsbausteine der Analyseumgebung bilden werden. Hierbei werden allgemeine Klassen von Operationen auf multidimensionalen Daten formalisiert, die anhand von Parametern aus dem multidimensionalen Datenmodell durch konkrete Datenmanagementschritte und (Maßzahl–)Berechnungen instantiiert werden können. Im einzelnen sollen folgende Operatoren von *MADEIRA* definiert werden:

- die *Konsolidierung* (Aggregierung) $f^{\text{kons}}: \mathcal{DR}^{\text{mi}} \times \mathcal{R} \times 2^{\mathcal{KA}} \times \mathcal{F}^{\mathcal{A}}_{\mathcal{KA}} \times 2^{\mathcal{SA}} \times \mathcal{F}^{\mathcal{A}}_{\mathcal{SA}} \to \mathcal{DR}^{\text{ma}}$ von Makrodaten aus Mikrodaten unter Abbildung ausgewählter relationaler Attribute auf summarische und kategorielle Attribute des Zieldatenraums (Abschnitt 4.3.1) sowie — quasi invers dazu — die *Abfrage* $f^{\text{mik}}: \mathcal{DR} \times 2^{\mathcal{K}} \times \mathcal{OR} \times 2^{\mathcal{E}} \to \mathcal{DR}^{\text{mi}}$ der jeweils einer Zelle eines Datenraums zugrundeliegenden Mikrodaten hinsichtlich interessierender Eigenschaften einer Objektmenge (*Drill–through*, Abschnitt 4.3.6),
- die *Ableitung* $f^{\text{abl}}: \mathcal{DR} \times \mathcal{KA} \times \mathcal{KA} \to \mathcal{DR}$ von Datenräumen aus anderen Datenräumen über ein ausgewähltes kategorielles Quellattribut (Abschnitt 4.3.2) zu einem Zielattribut mittels *impliziter Aggregierung* (*Roll–up*) oder *Restriktion* (*Slicing*),
- die *Selektion* $f^{\text{sel}}: \mathcal{DR} \times 2^{\mathcal{SA}} \to \mathcal{DR}$ summarischer Attribute eines Datenraums (Abschnitt 4.3.3),
- die *Vereinigung* $f^{\cup}: 2^{\mathcal{DR}} \times 2^{2^{\mathcal{KA}}} \to \mathcal{DR}$ mehrerer Datenräume unter Identifikation bestimmter kategorieller Attribute (Abschnitt 4.3.4),
- verschiedene Varianten der Maßzahlberechnung auf einem Datenraum (*zellenweise Verarbeitung* $f^{\text{cell}}: \mathcal{DR} \times \mathcal{M} \to \mathcal{DR}$, *explizite Aggregierung* $f^{\text{eaggr}}: \mathcal{DR} \times \mathcal{KA} \times \mathcal{KA} \times \mathcal{M} \to \mathcal{DR}$ und beliebige *Maßzahlverknüpfung* $f^{\text{join}}: \mathcal{DR} \times \mathcal{M} \to \mathcal{DR}$ — siehe Abschnitt 4.3.5),
- die Erzeugung und Auflösung zusammengesetzter kategorieller Attribute in *Split*– und *Merge*–Operationen $f^{\text{split}}: \mathcal{DR} \times \mathcal{KA} \to \mathcal{DR}$ bzw. $f^{\text{merge}}: \mathcal{DR} \times \mathcal{KA} \times \mathcal{KA} \to \mathcal{DR}$ (Abschnitt 4.3.7) — darüber hinaus erfordern zusammengesetzte Attribute im Rahmen anderer Operationen keine gesonderte Behandlung — sowie
- die *Umbennennung* von Rollen der durch einen Datenraum beschriebenen Objektmengen mittels $f^{\text{role}}: \mathcal{DR} \times \mathcal{R} \times \mathcal{R} \to \mathcal{DR}$ (Abschnitt 4.3.8).

Wie man sieht, orientieren sich die angebotenen Operatoren eng an den Strukturen der verarbeiteten Entitätstypen. Bewußt wurde *MADEIRA* auf grundlegende Verarbeitungsschritte in einigen wenigen Operationsklassen beschränkt. Spezielle statistische Berechnungsverfahren (wie sie z. B. in [Gho86] in ein statistisches

Datenmodell integriert werden) sind nicht Bestandteil des Modells. Vielmehr können diese flexibel in Gestalt neuer Maßzahlen in die Verarbeitung integriert werden. Homogenisierte Sichten auf Datenräume werden nicht gesondert betrachtet, da sie nur eine spezielle Sichtweise auf Datenräume, aber keine neue Datenstruktur definieren. Ebenso sind für mengenwertige Eigenschaften bzw. kategorielle Attribute zu deren Beschreibung i. a. keine speziellen Überlegungen nötig. Operationen zur Modifikation des grundlegenden Datenschemas, die über die Verarbeitung von Datenräumen hinausgehen, also die (Neu–)Definition von Domänen, Dimensionen, Maßzahlen, Aggregierungsfunktionen und ähnlichem, sind nicht Gegenstand von *MADEIRA*. Das Datenschema wird diesbezüglich als statisch angesehen.[49]

Abbildung 4.8 gibt anhand eines Beispiels einen Überblick über die Aufgaben der angeführten Operatoren. Die Änderung der Rollen betrachteter Objektmengen könnte hier noch integriert werden, wenn et-

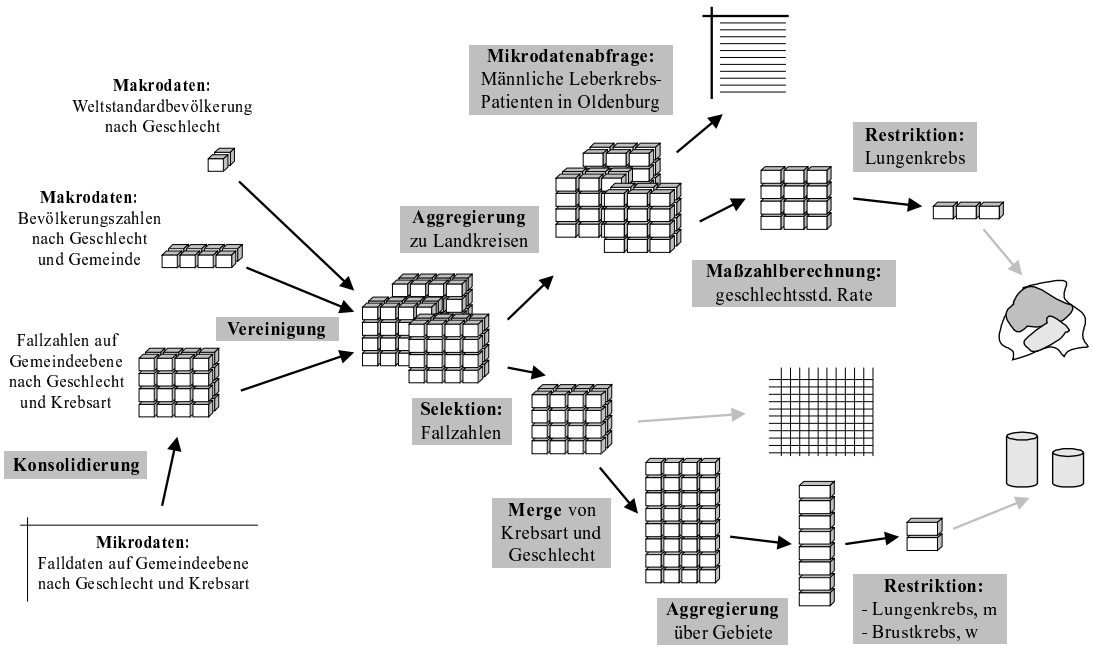

Abbildung 4.8: Operationen auf multidimensionalen Daten

wa die Bevölkerungsdaten der betrachteten Population (standardmäßig in der Rolle einer *Studienpopulation*) zusätzlich nach entsprechender Aggregierung über alle Teilgebiete anstelle des Weltstandards in der Rolle einer *Standardpopulation* verwendet werden sollten.

Sämtliche Operatoren sind lediglich partiell definiert, da jeweils bestimmte Anwendbarkeitsbedingungen (oftmals das Vorhandensein von zu bestimmten Maßzahlen bzw. Eigenschaften „passenden“ summarischen oder kategoriellen Attributen) erfüllt sein müssen, damit der jeweilige Zieldatenraum „ordnungsgemäß“ spezifiziert werden kann. Insbesondere ist zu beachten, daß die in Abschnitt 4.2.6 geforderten Konsistenzeigenschaften von Datenräumen gewährleistet werden:

- Übereinstimmung der von summarischen und kategoriellen Attributen beschriebenen Objektmengen (Def. 4.20),
- Disjunktheit der von den kategoriellen Attributen beschriebenen Eigenschaften (Def. 4.20) sowie
- Herleitung der Extension aus den Summierungsfunktionen der summarischen Attribute (Def. 4.21).

[49] Man beachte, daß der Benutzer trotzdem frei in der Nutzung beliebig „ad–hoc“ gruppierter Kategorien einer Domäne ist, d. h. er ist bei allen Operationen nicht auf die Kategorienhierarchie einer Dimension beschränkt.

Im Einzelfall werden diese Aspekte jeweils diskutiert, sofern ihre Einhaltung nicht bereits aus der Darstellung der Operatordefinition und der jeweiligen Anwendbarkeitsbedingungen klar zu ersehen ist. In diesem Sinne wird „Anwendbarkeit“ von Operatoren also stets als die Gewährleistung einer exakten und vollständigen Definierbarkeit eines im Sinne von *MADEIRA* gültigen Berechnungsergebnisses verstanden. Betrachtet werden allein Syntax und Semantik von Datenräumen und ihrer Komponenten. Die Pragmatik der Operationsdurchführung, also ihre Sinnhaftigkeit im Hinblick auf die Beantwortung bestimmter Fragestellungen, sowie die Berücksichtigung von Charakteristika der Datenwerte, aufgrund derer bestimmten statistischen Verfahren der Vorzug gegenüber anderen zu geben ist, spielen in *MADEIRA* keine Rolle.[50] In Kapitel 7 werden jedoch Ansatzpunkte zur entsprechenden Erweiterung von *VIOLA* in Richtung eines statistischen Expertensystems skizziert.

Auf die Laufzeitkomplexität der Anwendung einzelner Operatoren kommen wir in Abschnitt 6.2 im Kontext von Möglichkeiten zur Optimierung der Verarbeitung von *VIOLA*–Programmen noch zu sprechen.

Eine besondere Behandlung bei der Verarbeitung von Datenräumen erfahren jeweils Nullwerte in Datenraumzellen, wobei insbesondere die spezielle Semantik der Ausprägungen $t_{\text{nexist}}, t_{\text{unknown}}$ und t_{navail} berücksichtigt wird: Ist ein Eingabewert einer Berechnungsfunktion t_{nexist}, so wird auch dem Ergebnis dieser Nullwert zugewiesen. Andernfalls wird aus einem Argument t_{unknown} stets wieder t_{unknown}, und schließlich führt in den übrigen Fällen t_{navail} wieder zu t_{navail} als Ergebnis. Diese intuitive Priorisierung von Nullwerten, die in diesem Sinne einen unterschiedlichen Grad des Nicht–Wissens spezifizieren („es ist bekannt, daß der Wert nicht existiert“ — „es ist bekannt, daß der Wert nicht bekannt ist“ — „gar nichts ist bekannt“), ist ggf. für weitere Nullwerte zu ergänzen. In den folgenden Unterabschnitten, insbesondere bei der Aggregierung aus Mikrodaten in Abschnitt 4.3.1 und der Vereinigung von Datenräumen in Abschnitt 4.3.4, wird jeweils auf Besonderheiten bei der Nullwertbehandlung eingegangen.

Da alle Operatoren als Grundlage der Definition von Instanzen von Verarbeitungsoperatoren in der datenflußbasierten Umgebung *VIOLA* dienen sollen, werden sie auch entsprechend als Funktionen $f^{\cdots}$ definiert, die einen oder — im Fall der Vereinigung — mehrere „eingehende“ Datenräume $dr = (OR, KA, SA, f^{\text{ext}}) \in \mathcal{DR}$ unter Beachtung verschiedener Parameter zu einem Ergebnisdatenraum $dr' = (OR', KA', SA', f^{\text{ext}\prime}) \in \mathcal{DR}$ verarbeiten.[51] Es fließen also jeweils nur genau die vorliegenden Funktionsargumente in eine Berechnung ein — ein interner („verborgener“) Rückgriff auf die zugrundeliegenden Basisdaten oder zusätzliche Hilfsdaten, um anderweitig nicht bestimmbare Maßzahlwerte zu ermitteln, ist ausgeschlossen (vgl. auch die Anmerkung zur Definition von Aggregierungsfunktionen in Abschnitt 4.2.5). Diese Konvention begründet sich aus der Zielsetzung von *VIOLA*, dem Anwender die Kontrolle des Explorationsprozesses in all seinen operationalen Einzelschritten zu überlassen und ihm diese Schritte auch wiederum direkt für anschließende Manipulationen zugänglich zu machen.

Parameter von Operatoren haben in *MADEIRA* zumeist die Gestalt kategorieller oder summarischer Attribute. Da diese jeweils nur einem einzigen Datenraum zugeordnet sind, erlaubt eine derartige Vorgehensweise eine auf konkret zu verarbeitende Datenräume abgestimmte, elementare und prozedurale (an Quelldaten und darauf auszuführenden Analyseschritten orientierte) Spezifikation der Datenverarbeitung. So können die hinter den jeweiligen Operatoren stehenden Ideen gut motiviert und erläutert werden. In *VIOLA* werden diese Parameterattribute durch ihre Komponenten (Objektmengen, Kategorien etc.) ersetzt werden, um eine flexiblere Datenexploration zu gestatten, bei der deklarativ Charakteristika von Komponenten der Zieldatenräume vorgegeben werden. Indem Parameter unabhängig von konkreten Datenräumen definiert werden, können durch ein gegebenes, parametrisiertes Netz von Operatorknoten unterschiedliche Mengen von Basisdaten nebeneinander (quasi „parallel“) oder nacheinander untersucht werden.

Die Instanzen $\text{ext}(O, r)$ der Objektmengen zu verarbeiteten Datenräumen bleiben jeweils im Ergebnis unverändert, sofern (O, r) in $dr'.OR$ erhalten bleibt. Hierauf wird somit im folgenden jeweils nicht speziell ein-

[50] Zu dieser Thematik siehe z. B. [LS97], wo am Beispiel des Simpson–Paradoxons die Problematik der Aggregierung von Daten diskutiert wird.

[51] Konsolidierung und Mikrodatenabfrage weichen hiervon insofern ab, als sie sich auf einen Mikrodatenraum als Argument bzw. Ergebnis beziehen.

gegangen. Weiterhin werden auch die Aggregierungsfunktionen f^{aggr} neu erzeugter summarischer Attribute (bzw. der zugeordneten Maßzahlen) jeweils nicht explizit spezifiziert, da sie implizit bereits gemäß Def. 4.18 durch die Summierungsfunktionen der Attribute sowie durch die Zuordnung von Aggregierungsfunktionen zu kategoriellen Attributen, wie in Def. 4.23 beschrieben, festgelegt sind.

4.3.1 Konsolidierung von Makro- aus Mikrodaten

Nicht immer liegen zu analysierende Daten bereits als Makrodaten vor, sondern müssen unter Umständen erst aus Mikrodaten aggregiert werden. Darüber hinaus liegt konzeptionell, gemäß dem von *MADEIRA* verfolgten Modellierungsansatz, jedem Makrodatensatz eine vorherige Zusammenfassung von Mikrodaten (Objekten bzw. deren Eigenschaften) zugrunde. Aus diesem Grund spielt die nachfolgend präzisierte Konsolidierung, also die Aggregierung von Mikrodaten, eine besondere Rolle.

Definition 4.29 (Konsolidierung aus Mikrodaten) *Die* KONSOLIDIERUNG *eines Makrodatenraums aus Mikrodaten sei definiert durch die (partielle) Funktion*

$$f^{\text{kons}}\colon \mathcal{DR}^{\text{mi}} \times \mathcal{R} \times 2^{\mathcal{KA}} \times \mathcal{F}^{\mathcal{A}}_{\mathcal{KA}} \times 2^{\mathcal{SA}} \times \mathcal{F}^{\mathcal{A}}_{\mathcal{SA}} \to \mathcal{DR}^{\text{ma}} \quad .$$

Sei $dr = dr^{\text{mi}} = (O, A, f^{\text{ext}}) \in \mathcal{DR}^{\text{mi}}$ ein Mikrodatenraum mit Objektinstanzenmenge $\text{ext}(O)$. Ferner seien folgende weitere Konsolidierungsparameter gegeben:

- *Eine Rolle $r \in \mathcal{R}$, in der O betrachtet werden soll.*
- *Eine endliche, nicht–leere Menge kategorieller Attribute $KA = \{ka_1, \ldots, ka_m\} \subseteq \mathcal{KA}$ sowie eine injektive Abbildung $g_{\text{k}}\colon KA \to A$, die jedem kategoriellen Attribut das Mikrodaten–Attribut zuordnet, aus dem es hergeleitet werden soll. Somit gelte für alle $ka \in KA$:*
  - *$ka.D = D_{g_{\text{k}}(ka).e}$ und*
  - *$ka.EOR = \{(g_{\text{k}}(ka).e, O, r)\}$.*
- *Eine endliche, nicht–leere Menge summarischer Attribute $SA \subseteq \mathcal{SA}$ sowie eine Abbildung $g_{\text{s}}\colon SA \to A$, die jedem summarischen Attribut ein zugrundezulegendes Mikrodaten–Attribut zuordnet. Es gelte für alle $sa \in SA$:*
  - *$sa.OR = \{(O, r)\}$,*
  - *eine Berechnungsfunktion von $sa.mz$ (im folgenden f_{sa} genannt) faßt Werte des Attributs $g_{\text{s}}(sa)$ zusammen, d. h. $(\{mz_{g_{\text{s}}(sa).e,O}\}, \{sa.mz\}, \{\{(g_{\text{s}}(sa).e, O, r)\}\}, true, f_{sa}) \in sa.mz.F$, und*
  - *$sa.f^{\text{sum}}(U) = f_{sa}(\{f^{\text{val}}(o, g_{\text{s}}(sa).e) \mid o \in U\})$ für $U \subseteq O$.*

Dann sei $f^{\text{kons}}(dr, r, KA, g_{\text{k}}, SA, g_{\text{s}}) = dr' \stackrel{\text{def}}{=} (\{(O, r)\}, KA, SA, f^{\text{ext}\prime})$ definiert, und es gelte für alle $sa \in SA$ sowie $k_j \in ka_j.K$, $j = 1, \ldots, m$:

$$f^{\text{ext}\prime}_{sa}(k_1, \ldots, k_m) \stackrel{\text{def}}{=} \begin{cases} t_{\text{unknown}}, & \text{falls } \exists j \in \{1, \ldots m\}\colon k_j \not\preceq g_{\text{k}}(ka_j).k, \\ t_{\text{nexist}}, & \text{falls } \nexists o \in O\, \forall j = 1, \ldots, m\colon o \in O_{ka_j.e,k_j}, \\ f_{sa}(\{\, (k, |\{o \in dr.\text{ext}(O) \mid (f^{\text{ext}}_{g_{\text{s}}(sa)}(o) = k) \wedge \\ \qquad (\forall j = 1, \ldots, m\colon f^{\text{ext}}_{g_{\text{k}}(ka_j)}(o) \preceq k_j)\}|) \text{ mit } k \in D_{g_{\text{s}}(sa).e}\,\}) & \text{sonst,} \end{cases}$$

wobei $dr'.\text{ext}(O, r) \stackrel{\text{def}}{=} dr.\text{ext}(O)$.[52]

Es entstehen somit aus Einzelfalldaten aggregierte Makrodaten, indem die Werte von Mikrodatenattributen gemäß der aggregierenden Berechnungsfunktionen ausgewählter Maßzahlen zusammengefaßt, also z. B. die

[52] Man beachte, daß hier — wie prinzipiell bei der Anwendung aller Aggregierungs- und Berechnungsfunktionen — Multimengen von Werten (Kategorien) aggregiert werden.

Summe oder der Durchschnitt über ein relationales Attribut berechnet werden. Auch die einfache Anzahl–Berechnung auf Mikrodaten, etwa die Zusammenfassung eines personenbezogenen Datenbestandes mit Krebspatienten zu Neuerkrankungszahlen, erfolgt problemlos gemäß obiger Definition. Ein entsprechendes summarisches Attribut ist auf einem speziellen Attribut der Mikrodaten (etwa einer ID) zu definieren, das die Zugehörigkeit zur betrachteten Objektmenge symbolisiert (vgl. auch das Beispiel zur Maßzahldefinition in Abschnitt 4.2.5). Die Maßzahlberechnung erfolgt bei der Konsolidierung stets disjunkt (das Flag *disjunkt*? der Berechnungsvorschrift zu f_{sa} ist *true*), da sich per Definition jeder Wert der Quellmaßzahl (also ein Attributwert in den Mikrodaten) auf ein einzelnes Objekt bezieht und jedes Objekt jeweils maximal einmal in die Berechnung einfließt.

Das Wissen um das Auftreten von strukturellen Nullwerten (t_{nexist}) kann — da es sich im Gegensatz zu Werten aus T_0 auf die ganze Objektmenge O und nicht nur $\text{ext}(O)$ bezieht — nicht direkt aus der Mikrodaten–Extension abgeleitet werden, sondern ist in Implementierungen von *MADEIRA* in geeigneter Form separat zur Verfügung zu stellen. Zum Beispiel können dem Mikrodatenraum bzw. den betrachteten Objektmengen entsprechende Metadaten zugeordnet werden. Zusätzlich zur Repräsentation nicht–berechenbarer Maßzahlwerte zu Kategorien außerhalb der Wertebereiche der Mikrodaten–Attribute kann die Summierungsfunktion den Nullwert t_{unknown} auch aus der Aggregierung über bestimmte NULL–Kategorien liefern. Der Nullwert t_{navail} schließlich wird seiner Semantik entsprechend nie im Rahmen der Konsolidierung erzeugt. Existieren im Wertebereich der betrachteten Maßzahl weitere Nullwerte, so ist obige Definition ggf. um deren spezielle Behandlung zu ergänzen.

Die zu Beginn von Abschnitt 4.3 aufgestellten Konsistenzbedingungen für Datenräume (siehe Seite 137) sind für dr' offenbar alle erfüllt.

Definition 4.29 wurde bewußt einfach gehalten, indem nur eine Rolle der modellierten Objekte sowie nur jeweils ein zugrundeliegendes Mikrodatenattribut für jedes kategorielle und summarische Attribut vorgesehen wurde. Die Erzeugung komplexerer Datenräume bleibt nachfolgenden Verarbeitungsschritten (vgl. insbesondere Abschnitt 4.3.4) vorbehalten. Außerdem könnten auch ggf. bereits in den Mikrodaten Attribute angeboten werden, die aus mehreren anderen Attributen berechnet sind. In diesem Sinne wird auch darauf verzichtet, kategorielle Attribute mit zusammengesetzten Kategorien direkt aus Mikrodaten zu erzeugen, sofern nicht bereits die Mikrodatenbasis entsprechende „zusammengesetzte“ Attribute enthält. Gegebenenfalls ist — als eigenständige atomare Operation — im Anschluß an die Konsolidierung eine Merge–Operationen (siehe Abschnitt 4.3.7) vorzunehmen.

Auch mengenwertige Mikrodatenattribute werden nicht speziell behandelt. Eine Implementierung des Mikrodatenzugriffs wird, wie bereits in Abschnitt 4.2.4 angedeutet, für mengenwertige Attribute im Regelfall über eine geeignete Schnittstelle von einer separaten Speicherung der jeweiligen Mengenelemente abstrahieren.

4.3.2 Ableitung von Datenräumen durch Aggregierung und Restriktion

Die zentrale Operation im Umgang mit Datenräumen ist die Aggregierung. So wie durch die Konsolidierung aus Mikrodaten Makrodaten gewonnen werden, werden im Laufe einer Analyse durch Zusammenfassung von Kategorien einer Domäne gemäß der jeweiligen Aggregierungsfunktionen gröbere Datenwürfel generiert. Diese Verarbeitung beschränkt sich i. a. nicht auf den Wechsel zwischen vorgegebenen Aggregierungsebenen, ja nicht einmal auf die in der jeweiligen Dimension enthaltenen Kategorien. So sollen häufig Gesamtwerte über alle vorhandenen Kategorien bestimmt oder auch Gruppierungen vorgenommen werden, die ad–hoc über beliebige Prädikate[53] auf den Kategorien definiert werden (z. B. eine ringförmige Einteilung der Gebiete um eine Emissionsquelle).

Zunächst wird einmal betrachtet, was es heißt, Kategorien eines kategoriellen Attributs bzw. allgemein einer beliebigen Kategorienmenge zu vorgegebenen Kategorien „zusammenzufassen“.

[53] Derartige Prädikate würden Zusatzwissen über Kategorien einbeziehen — dieser Aspekt wird in Abschnitt 4.4.3 noch etwas genauer betrachtet.

Definition 4.30 (Ableitung von Kategorien) *Eine Kategorie k' aus der Domäne D heiße aus einer Menge $K \subseteq D$ von Kategorien* ABLEITBAR, *falls* $\exists K_{k'} \subseteq K : k' \equiv \bigvee K_{k'}$.

Seien $e \in \mathcal{E}_D$ eine Eigenschaft und $O \subseteq O_e$ eine durch e beschreibbare Objektmenge. Gibt es eine Menge $K_{k'}$, die gemäß obiger Definition äquivalent zu k' ist und deren Kategorien bzgl. e und O paarweise disjunkt sind, so heiße k' DISJUNKT ABLEITBAR *aus K (bzgl. e und O).*

Der Begriff der Ableitbarkeit entspricht also gerade der Äquivalenz von Vater– und Sohn–Kategorien in einer Kategorienhierarchie. Der Betrachtung einwertiger Aggregierungsebenen entsprechend, interessiert auch hier in den meisten Fällen die disjunkte Ableitbarkeit, also das „überlappungsfreie Zusammensetzen“ von Zielkategorien. Speziell möchte man in der Regel keine Kategorie doppelt in der Menge $K_{k'}$ verwenden.

Auf obiger Definition aufbauend, definieren wir nun die Ableitung von Datenwürfeln. Hierbei wird zu einer Dimension, genauer einem kategoriellen Attribut eines gegebenen Datenraums, ein gröberes kategorielles Attribut spezifiziert, und Maßzahlwerte werden gemäß der Ableitbarkeit von Kategorien und der durch das jeweilige summarische Attribut bestimmten Aggregierungsfunktion aggregiert.

Definition 4.31 (Ableitung von Datenräumen) *Die* ABLEITUNG *eines Datenraums aus einem anderen sei definiert durch eine (partielle) Abbildung $f^{\text{abl}} : \mathcal{DR} \times \mathcal{KA} \times \mathcal{KA} \to \mathcal{DR}$. Seien $dr = (OR, KA, SA, f^{\text{ext}})$ ein Datenraum sowie $ka \in KA$ ein Quell- und $ka' \in \mathcal{KA}$ ein Zielattribut, über die die Ableitung erfolgen soll.*

$f^{\text{abl}}(dr, ka, ka') = dr' \stackrel{\text{def}}{=} (OR, KA \setminus \{ka\} \cup \{ka'\}, SA, f^{\text{ext}\prime})$ sei definiert („dr' ist ABLEITBAR *aus dr“), falls $ka.EOR = ka'.EOR$ und entweder*

1. *$ka'.K \subseteq ka.K$ (nur Einschränkung des Wertebereichs, keine Zusammenfassung von Kategorien) oder*
2. *$\forall sa \in SA$: sa ist aggregierbar[54] über ka (mittels einer Funktion f^{aggr}_{ka} gemäß Def. 4.23) und alle Kategorien von ka' sind ableitbar[54] aus $ka.K$.*

Für die Extension von dr' gelte dann (mit $sa \in SA$ und $k' \in ka'.K$ beliebig):

$$f^{\text{ext}\prime}_{sa}(\ldots, k', \ldots) \stackrel{\text{def}}{=} sa.f^{\text{aggr}}_{ka}(\{(f^{\text{ext}}_{sa}(\ldots, k, \ldots), k) \mid k \in K_{k'}\}) \quad .$$

Hierbei bezeichne $K_{k'}$ wie in Def. 4.30 — nicht notwendigerweise eindeutig — jeweils die Menge zur Kategorie k' zusammengefaßter Kategorien aus dem Attribut ka von dr.

Gilt unter (1) „$ka'.K \subseteq ka.K$“, spreche man auch von RESTRIKTION *von dr, andernfalls von* (IMPLIZITER) AGGREGIERUNG.

Zum Beispiel können mit dem Ableitungsoperator aus einem Datenraum mit Fallzahlen nach Altersgruppen und betrachteten Landkreisen die Angaben zu Jugendlichen extrahiert oder die landkreisbezogenen Werte zu Summen über die jeweiligen Regierungsbezirke zusammengefaßt werden.

Falls nicht lediglich eine Restriktion auf dem Ausgangsdatenraum *dr* erfolgt, wird stets gleichzeitig über *alle* summarischen Attribute von *dr* aggregiert. Dabei ist gemäß Def. 4.23 zu beachten, daß nur für diejenigen summarischen Attribute *sa* tatsächlich *neue* Maßzahlwerte berechnet werden, deren Objektmengen auch durch das betrachtete kategorielle Attribut beschrieben werden ($ka \in KA_{sa}$). Alle anderen Maßzahlen bleiben unverändert, sie werden *identisch* aggregiert.

Der Begriff der Ableitbarkeit definiert eine partielle, reflexive, transitive, nicht jedoch antisymmetrische Relation auf Datenräumen (also keine Ordnung). Die Ableitung eines Datenraums aus einem anderen muß — im Hinblick auf die genutzten Gruppierungen von Kategorien — nicht auf einem eindeutigen Weg erfolgen, da Datenräume durchaus auch redundante Informationen auf unterschiedlichen Aggregierungsniveaus enthalten können. Somit wird deutlich, daß die den Analysen zugrundeliegenden Mikrodaten bzw. persistente Makrodaten einen konsistenten, also insbesondere mit den Kategorienhierarchien und den Aggregierungsfunktionen

[54] Sind alle Kategorien sogar (bzgl. aller Eigenschaften in $ka.EOR$) *disjunkt* ableitbar aus $ka.K$, so ist *disjunkte* Aggregierbarkeit von *sa* ausreichend — andernfalls ist entweder *beliebige* oder *identische* Aggregierbarkeit erforderlich.

der summarischen Attribute verträglichen Zustand der Daten gewährleisten müssen, um eindeutige Ergebnisse der Ableitung zu garantieren. Dies ergibt sich bereits aus der Definition der Summierungsfunktion eines summarischen Attributs (Def. 4.17) bzw. ihrer Nutzung zur Herleitung der Datenraumextension in Def. 4.21.

Wir verzichten an dieser Stelle darauf, auch Restriktionen oder Gruppierungen anhand von *Zellinhalten* eines Datenraums durch *MADEIRA* direkt zu unterstützen. So könnten etwa die Erkrankungsraten für eine Reihe von Regionen dazu dienen, Gruppen von Regionen mit hohen, mittleren oder niedrigen Raten zu definieren bzw. für die weitere Betrachtung zu selektieren. Ein derartiges Vorgehen wäre nur für eindimensionale Datenräume klar zu definieren. Weiterhin würde das Datenmodell — ohne tatsächliche Erweiterung seiner Berechnungsmöglichkeiten — unnötig komplex, denn wie schon im Fall der Disaggregierung (vgl. Abschnitt 4.2.5) müßte zur Beschreibung einer Objektmenge Wissen über Obermengen von Objekten einbezogen werden. Letztlich kann auch für alle anderen Selektionen bzw. Ableitungen der *Grund* für die jeweilige Kategorienwahl nicht exakt formalisiert werden — dies bleibt der formlosen Angabe beschreibender Metadaten als Hintergrundinformation vorbehalten (vgl. Abschnitt 4.4.1). Es wird Aufgabe einer geeigneten Parameterbehandlung in *VIOLA* (vgl. Abschnitt 5.2.4) sein, anhand von Datenrauminhalten Kategorien für einen Ableitungsschritt zu selektieren bzw. zu gruppieren.

Weiterhin sollen Attribute über zusammengesetzten Eigenschaften nicht speziell behandelt werden. Aggregierungsfunktionen über diese sind in einheitlicher Weise unabhängig von denen der Komponenten–Eigenschaften definiert.[55]

Schließlich wird auch die Möglichkeit, im Rahmen der Ableitung Disaggregierungsschritte vorzunehmen, nicht näher verfolgt. Liegen etwa nur Daten zu Kategorien $(a \vee b)$ und b vor, so könnte unter Umständen bei geeigneter Aggregierungsfunktion hieraus auch ein Maßzahlwert zur Kategorie b, z. B. — bei Aggregierung durch Summation — über die Subtraktion, berechnet werden. Der Bedarf für einen derartigen Automatismus ist gering, entsprechende Fälle können i. a. auf anderem Wege mit den ansonsten geboten Möglichkeiten von *MADEIRA* gleichermaßen verarbeitet werden, indem etwa gleich alle für die weitere Aggregierung benötigten Gruppen selektiert werden.

Eine Implementierung des Ableitungsoperators erfordert, vor allem durch die potentielle Vereinigung unterschiedlich granularer Kategorien in einem kategoriellen Attribut und daraus resultierende Überlappungen, sicherlich genauere Überlegungen zur effizienten Suche „passender“ Kategorienmengen. Aufgrund des angenommenen nur mittleren Datenumfangs bestünde die Möglichkeit, zu jedem erzeugten kategoriellen Attribut jeweils ableitbare Kategorien im voraus zu ermitteln. Inwiefern der sich ergebende Verwaltungs- und initiale Berechnungsoverhead noch zu vertreten wäre, bleibt zu evaluieren.

Verträglichkeit der Ableitung mit der Definition von Datenräumen

Während sich die Einhaltung aller übrigen Konsistenzbedingungen für Datenräume gemäß der Aufstellung auf Seite 137 per Definition direkt aus dem jeweiligen Quelldatenraum ergibt, verdient die eindeutige Definition der Datenraumextension anhand der Summierungsfunktion der summarischen Attribute eine genauere Betrachtung: Datenräume beschreiben gemäß Def. 4.20 in ihren Zellen jeweils die Teilgruppen von Objektmengen, die durch die Koordinaten einer Zelle spezifiziert sind. Zur jeweiligen Teilgruppe geben die summarischen Attribute des Datenraums gemäß ihrer beschreibenden Funktion f^{sum} Maßzahlwerte an. Im folgenden wird gezeigt, daß dieser Zusammenhang auch durch die oben eingeführte Ableitungsoperation erhalten bleibt.

Lemma 4.2 (Verträglichkeit von Ableitung und Datenraumdefinition) *Die Ableitungsoperation gemäß Def. 4.31 ist in dem Sinne mit der Definition von Datenräumen verträglich, daß für alle Datenräume $dr, dr' \in \mathcal{DR}$ und kategorielle Attribute $ka, ka' \in \mathcal{KA}$ mit $f^{\text{abl}}(dr, ka, ka') = dr'$ gilt: Die Extension von dr' ist — wie in Def. 4.21 beschrieben — eindeutig durch die Summierungsfunktionen der summarischen Attribute von dr' bestimmt, falls das gleiche bereits für dr galt.*

[55] Eine entsprechende Herleitung von Aggregierungsfunktionen wäre theoretisch denkbar und möglich, würde aber das Datenmodell in einigen Bereichen deutlich komplizierter machen, wobei nur relativ wenig an Funktionalität für die ohnehin nur in Ausnahmefällen genutzten zusammengesetzten Eigenschaften gewonnen wäre.

Beweis: Seien $dr, dr' \in \mathcal{DR}$ zwei Datenräume und $ka, ka' \in \mathcal{KA}$ zwei kategorielle Attribute mit $f^{\text{abl}}(dr, ka, ka') = dr'$. sa sei ein summarisches Attribut von dr' (und dr) mit $sa.OR = \{(O_1, r_1), \ldots, (O_n, r_n)\}$. $\text{ext}(O_i, r_i), i = 1, \ldots, n$, bezeichne die durch die Datenräume beschriebene Instanz der jeweiligen Objektmenge. f_{sa}^{ext} und $f_{sa}^{\text{ext}'}$ seien die Extensionen der Datenräume bzgl. des Attributs sa. Es seien $ka'_1, \ldots, ka'_m$ die kategoriellen Attribute von dr', ferner (für $j = 1, \ldots, m$) $k'_j \in ka'_j.K$ beliebig sowie (für $i = 1, \ldots, n$)

$$U_i \stackrel{\text{def}}{=} \{o \in \text{ext}(O_i, r_i) \mid \forall j = 1, \ldots, m \, \forall e \in \mathcal{E} : ((e, O_i, r_i) \in ka'_j.OR \Rightarrow o \vdash_e k'_j)\}$$

die durch die k'_j jeweils beschriebenen Teilmengen von $\text{ext}(O_i, r_i)$.

Zu zeigen ist nun gemäß Def. 4.21: $f_{sa}^{\text{ext}'}(k'_1, \ldots, k'_m) = sa.f^{\text{sum}}(U_1, \ldots, U_n)$.

Sei $1 \leq u \leq m$, so daß $ka' = ka'_u$. Es sei (die oben ausgewählte Kategorie) $k'_u \equiv k^1 \vee \ldots \vee k^q$ mit $\{k^1, \ldots, k^q\} \subseteq ka.K$. (Dies sei gerade die Menge $K_{k'_u}$ aus der Ableitungsdefinition.) Ferner seien

$$U_i^h \stackrel{\text{def}}{=} U_i \cap \{o \in \text{ext}(O_i, r_i) \mid \forall e \in \mathcal{E} : ((e, O_i, r_i) \in ka'_u.OR \Rightarrow o \vdash_e k^h)\}$$

(für $i = 1, \ldots, n$ und $h = 1, \ldots, q$) die durch die Quellkategorien k^h in dr eingeschränkten Objektmengeninstanzen. Dann ist

$$\begin{aligned} f_{sa}^{\text{ext}'}(k'_1, \ldots, k'_u, \ldots, k'_m) &\stackrel{\text{Def. 4.31}}{=} sa.f_{ka}^{\text{aggr}}(\{(f_{sa}^{\text{ext}}(k'_1, \ldots, k^h, \ldots, k'_m), k^h) \mid h = 1, \ldots, q\}) \\ &\stackrel{\text{Def. 4.21}}{=} sa.f_{ka}^{\text{aggr}}(\{(sa.f^{\text{sum}}(U_1^h, \ldots, U_n^h), k^h) \mid h = 1, \ldots, q\}) \\ &\stackrel{\text{Def. 4.18}}{=} sa.f^{\text{sum}}(U_1, \ldots, U_n). \end{aligned}$$

q.e.d.

Zur Erinnerung: Die im letzten Beweisschritt verwendete Definition 4.18 beschreibt die Verträglichkeit von Summierungs- und Aggregierungsfunktion eines summarischen Attributs. Die dort zugrundeliegenden Objektmengen $\overline{O_i}$ entsprechen hier gerade den Mengen

$$\{o \in \text{ext}(O_i, r_i) \mid \forall j \in \{1, \ldots, m\} \setminus \{u\} \, \forall e \in \mathcal{E} : ((e, O_i, r_i) \in ka'_j.OR \Rightarrow o \vdash_e k'_j)\} \quad .$$

4.3.3 Selektion summarischer Attribute eines Datenraums

Mit der Einführung des Ableitungsoperators in Abschnitt 4.3.2 ist es zwar möglich, die kategoriellen Attribute eines Datenraums einzuschränken, die summarischen Attribute bleiben jedoch unberührt. Um auch einzelne summarische Attribute zur weiteren Betrachtung auswählen zu können, wird nun der Selektionsoperator eingeführt. Im Gegensatz zur Ableitung werden bei der Selektion *Mengen* von Zielattributen übergeben, um den jeweils betrachteten Datenraum nicht notwendigerweise auf nur ein Attribut einschränken zu müssen. Bei der Verringerung der Menge summarischer Attribute eines Datenraums ist zu beachten, daß ggf. mit den rollenbehafteten Objektmengen der eliminierten summarischen Attribute auch diejenigen kategoriellen Attribute entfernt werden, die für keines der verbliebenen summarischen Attribute relevant sind.

Definition 4.32 (Selektion summarischer Attribute) *Die* SELEKTION *summarischer Attribute eines Datenraums sei definiert durch eine (partielle) Abbildung $f^{\text{sel}} : \mathcal{DR} \times 2^{\mathcal{SA}} \to \mathcal{DR}$. Seien $dr = (OR, KA, SA, f^{\text{ext}})$ ein Datenraum sowie $\widetilde{SA} \subseteq \mathcal{SA}$ eine (nicht notwendigerweise endliche) Menge summarischer Attribute. Sei ferner $SA' \stackrel{\text{def}}{=} SA \cap \widetilde{SA}$.*

Genau dann, wenn $SA' \neq \emptyset$, sei $f^{\text{sel}}(dr, \widetilde{SA}) = dr' \stackrel{\text{def}}{=} (OR', KA', SA', f^{\text{ext}'})$ definiert mit

- $OR' \stackrel{\text{def}}{=} \bigcup\{sa.OR \mid sa \in SA'\}$ *und*
- $KA' \stackrel{\text{def}}{=} \{ka' = (ka.D, ka.K, ka.EOR \cap \mathcal{E} \times OR') \mid (ka \in KA) \wedge (ka.OR \cap OR' \neq \emptyset)\}$.

Sei $KA = \{ka_1, \ldots, ka_m\}$ *und* $KA' = \{ka'_1, \ldots, ka'_n\}$*, so daß* ka'_i *aus* $ka_{p(i)}$ *mit* $p(i) \in \{1, \ldots, m\}$*,* $i = 1, \ldots, n$*, hervorgegangen ist. Dann gelte (mit* $sa \in SA'$ *und* $k_j \in ka_j.K$ *für* $j = 1, \ldots, m$*) für die Extension* $f^{\text{ext}'}$ *von* dr'*:*

$$f_{sa}^{\text{ext}'}(k_{p(1)}, \ldots, k_{p(n)}) \stackrel{\text{def}}{=} f_{sa}^{\text{ext}}(k_1, \ldots, k_m) \quad .$$

Die Extension von dr' ist gemäß obiger Definition eindeutig spezifiziert, da die verbliebenen summarischen Attribute unabhängig von den entfallenen kategoriellen Attributen (bzw. von deren Objektmengen) und somit ihre Maßzahlwerte in *dr* entlang dieser Attribute identisch sind. Auch die anderen Konsistenzbedingungen für Datenräume sind weiter eingehalten.

4.3.4 Zusammenführung von Datenräumen durch Vereinigung

Zur Vereinfachung und Vereinheitlichung der Operatordefinitionen in *MADEIRA* sollen alle Verknüpfungen unterschiedlicher Maßzahlen nur *innerhalb* eines Datenraums möglich sein. So werden z. B. bei der Berechnung von Erkrankungsraten nicht ein Fallzahl- und ein Bevölkerungsdatenwürfel durcheinander dividiert, sondern zunächst werden beide Quelldatenräume in einen Datenraum integriert, um dann erst anschließend den eigentlichen Verarbeitungsschritt zu realisieren.

Der im folgenden eingeführte Vereinigungsoperator implementiert gerade diese Zusammenführung von Datenräumen, bei der die in den Quelldatenräumen enthaltenen Maßzahlwerte als Würfelzellen stets erhalten bleiben. Im Zuge einer Vereinigung wird überprüft, ob die jeweiligen Datenräume überhaupt zueinander „passen“, es werden geeignete Zuordnungen sich entsprechender summarischer und kategorieller Attribute definiert und hieraus die Extensionsfunktion des Zieldatenraums festgelegt. Die Kapselung dieser Schritte in einer Vereinigungsoperation entbindet somit alle anderen Operatoren von derartigen Überlegungen. Folgende Aspekte werden im einzelnen berücksichtigt:

- Kategorielle Attribute zu unterschiedlichen Objektmengen bzw. unterschiedlichen Eigenschaften über der gleichen Domäne können miteinander identifiziert werden (vgl. Def. 4.19). So werden etwa auf der Objektmenge der Tumoren definierte Fall- und personenbezogene Bevölkerungszahlen hinsichtlich Alters-, Geschlechts-, Gebietsangaben etc. vergleichbar gemacht.

- Datenräume unterschiedlicher Dimensionalität können unter Duplizierung von Maßzahlwerten über Eigenschaften (also kategorielle Attribute), die nicht zur Beschreibung einer Maßzahl beitragen, verschmolzen werden (vgl. Def. 4.20). Zum Beispiel sollen bei der Vereinigung von Fall- und Bevölkerungszahlen letztere für alle Erkrankungen einer Diagnose–Dimension identisch sein.

- Unter Verwendung von Nullwerten (insbesondere t_{navail}) für undefinierte Maßzahlwerte im Vereinigungsergebnis können auch kategorielle Attribute zusammengeführt werden, deren Kategorienmengen unterschiedlich sind. Um beim gleichen Beispiel zu bleiben: Fallzahlen für Niedersachsen können mit Bevölkerungszahlen zu ganz Norddeutschland verschmolzen werden, wobei die Falldaten für Schleswig–Holstein, Hamburg usw. als nicht verfügbar markiert werden. In ähnlicher Weise kann ein nach dem Alter klassifizierter Falldatenraum für ganz Niedersachsen mit einem nach Gemeinden klassifizierten Datenraum mit Fallzahlen über alle Altersgruppen vereinigt werden. Das zweidimensionale Ergebnis enthält nur die beiden Randsummen; innere Zellen (für beliebige Kombinationen von Alter und Gebiet) sind mit Nullwerten belegt.[56]

- Schließlich können natürlich auch Vereinigungen von Kategorienmengen bzgl. einer Datenraumdimension erfolgen, z. B. bei der Zusammenführung zweier Datenräume mit Angaben zu männlichen bzw. weiblichen Erkrankungsfällen.

[56] Etwaige Unabhängigkeitsannahmen bzgl. der betrachteten Eigenschaften, die zu einer Disaggregierung über eine Gleichverteilung und somit zu einem Füllen der inneren Zellen genutzt werden könnten, sind — wie bereits in Abschnitt 4.2.5 diskutiert — in *MADEIRA* nicht modellierbar und können deshalb hier keine Berücksichtigung finden.

Somit bietet die Vereinigung eine mächtige Grundlage zur korrekten bzw. semantisch exakt definierbaren Verknüpfung multidimensionaler Daten. Lediglich zwei Situationen verhindern prinzipiell eine Vereinigung einer gegebenen Datenraummenge, falls zwei oder mehrere der Datenräume die gleichen Objektmengen in den gleichen Rollen beschreiben:

1. Falls sich die Datenräume auf unterschiedliche Instanzenmengen dieser Objektmengen beziehen, kann die entsprechende Menge $\text{ext}(O, r)$ des Ergebnisdatenraums nicht eindeutig definiert werden. In diesem Fall wird also auf unterschiedliche, nicht notwendigerweise konsistente Basisdaten Bezug genommen.

2. Falls die betreffenden Datenräume nicht genau die gleichen Eigenschaften dieser Objektmengen beschreiben, sind sie aufgrund fehlender Informationen nicht vergleichbar. Werden z. B. in einem Datenraum *dr* in einem kategoriellen Attribut Ausprägungen einer Eigenschaft *e* einer Objektmenge *O* spezifiziert und gibt es in einem hiermit zu vereinigenden Datenraum dr', der sich auch auf *O* bezieht, kein entsprechendes kategorielles Attribut, so kann nicht davon ausgegangen werden, daß dr' die Objekte aus *O* völlig unabhängig von *e* beschreibt.[57] In *MADEIRA* wird — der Klarheit und Eindeutigkeit halber — in dem Fall, daß *e* nicht irrelevant für die Analyse von Objekten aus *O* ist, verlangt, daß ein Datenraum über *O* auf die Eigenschaft *e* zumindest in einem einschränkenden Attribut Bezug nimmt.

In allen anderen Fällen, insbesondere auch wenn gänzlich verschiedene Objektmengen beschrieben werden, können beliebige Datenräume — unter Angabe einer im Sinne folgender Definition gültigen Partitionierung und Zuordnung aller beteiligten kategoriellen Attribute — innerhalb eines gemeinsamen „Hüllen"–Datenraums gekapselt und zur gemeinsamen Weiterverarbeitung bereitgestellt werden.

Definition 4.33 (Vereinigung von Datenräumen) *Die* VEREINIGUNG *von Datenräumen sei definiert durch eine (partielle) Abbildung* $f^{\cup}: 2^{\mathcal{DR}} \times 2^{2^{\mathcal{KA}}} \to \mathcal{DR}$. *Sei* $DR = \{dr_1, \ldots, dr_n\} \subseteq \mathcal{DR}$ *eine nicht–leere, endliche Menge von Datenräumen mit* $dr_i = (OR_i, KA_i, SA_i, f^{\text{ext}(i)}), i = 1, \ldots, n$, *wobei* $\mathbf{KA} \subseteq 2^{\mathcal{KA}}$.

Folgendermaßen ergeben sich die Komponenten von $dr' = f^{\cup}(DR, \mathbf{KA})$ *bzw. Anwendbarkeitsbedingungen für die Durchführung der Vereinigung:*

- *Beschreiben zwei Quelldatenräume eine Objektmenge in derselben Rolle, so muß diese jeweils über dieselben Eigenschaften klassifiziert sein:*

$$\forall dr_1 \neq dr_2 \in DR\, \forall or \in dr_1.OR \cap dr_2.OR\, \forall e \in \mathcal{E}: \\ (\exists ka_1 \in dr_1.KA: (e, or) \in ka_1.EOR) \Rightarrow (\exists ka_2 \in dr_2.KA: (e, or) \in ka_2.EOR) \quad .$$

- $OR' \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} OR_i$.
- *Die Instanzenmengen* $\text{ext}(O_i, r_i)$ *müssen jeweils in den verschiedenen Quelldatenräumen (wo vorhanden) identisch sein und werden so ins Ergebnis übernommen.*
- **KA** *definiere eine disjunkte und vollständige Partitionierung von* $KA \stackrel{\text{def}}{=} \bigcup_{i=1}^{n} KA_i$, *die angibt, welche Attribute der Ausgangsdatenwürfel miteinander identifiziert und verschmolzen werden. Es gelte:*
  - *Es werden nur kategorielle Attribute über der gleichen Domäne zusammengefaßt, d. h.* $\forall \widetilde{KA} \in \mathbf{KA}\, \exists D_{\widetilde{KA}} \in \mathcal{DO}\, \forall ka \in \widetilde{KA}: ka.D = D_{\widetilde{KA}}$.
  - *Alle zu verschmelzenden Attribute einer Attributmenge* $\widetilde{KA} \in \mathbf{KA}$ *stammen jeweils aus paarweise verschiedenen Datenräumen* dr_i, *was Voraussetzung dafür ist, daß alle Ausgangsdatenräume als „Teilräume" des Vereinigungsergebnisses erhalten bleiben.*
  - *Attribute zur gleichen Eigenschaft fallen stets in die gleiche Menge in der Partition:* $\forall ka_1, ka_2 \in KA$ *mit* $ka_1.EOR \cap ka_2.EOR \neq \emptyset\ \exists \widetilde{KA} \in \mathbf{KA}: (ka_1 \in \widetilde{KA}) \wedge (ka_2 \in \widetilde{KA})$.

[57] Ein derartige implizite Annahme mag in vielen Fällen korrekt sein, muß es aber nicht.

Es sei $KA' \stackrel{\text{def}}{=} \{(D_{\widetilde{KA}}, \bigcup_{ka \in \widetilde{KA} \cap KA} ka.K, \bigcup_{ka \in \widetilde{KA} \cap KA} ka.EOR) \mid \widetilde{KA} \in \mathbf{KA}\} \stackrel{\text{def}}{=} \{ka'_1, \ldots, ka'_m\}$. *Für ein Attribut* $ka_j^{(i)} \in KA_i = \{ka_1^{(i)}, \ldots, ka_{m_i}^{(i)}\}$ *ergebe* $ka'_{p_i(j)}$ *das entsprechende Attribut in* KA' *— umgekehrt sei jeweils* ka'_j *(unter anderem) aus* $ka_{q_i(j)}^{(i)} \in KA_i$ *hervorgegangen.*[58]

- SA' *sei entsprechend definiert als Vereinigung über alle* SA_i.[59] $sa^{(i)}$ *sei jeweils zu einem Attribut* $sa \in SA'$ *das zugehörige Attribute in* SA_i *(oder undefiniert, falls keine Entsprechung im Datenraum* dr_i *existiert). Die Aggregierungsfunktionen der Attribute in* SA' *werden definiert, wie in Def. 4.23 beschrieben.*

Konnten alle Komponenten wie angegeben spezifiziert werden, sei $dr' \stackrel{\text{def}}{=} (OR', KA', SA', f^{\text{ext}\prime})$ *als Ergebnis der Vereinigung definiert. Zu einem Attribut* $sa \in SA'$ *mit relevanten kategoriellen Attributen* $KA'_{sa} = \{ka'_{t(1)}, \ldots, ka'_{t(v)}\}$ *gelte dann für die Extension* $f^{\text{ext}\prime}$ *(mit* $k_j \in ka'_j$ *für* $j = 1, \ldots, m$*)*[60]*:*

$$f_{sa}^{\text{ext}\prime}(k_1, \ldots, k_m) \stackrel{\text{def}}{=} \begin{cases} f_{sa^{(i)}}^{\text{ext}(i)}(k_{p_i(1)}, \ldots, k_{p_i(|KA_i|)}) & \text{falls } \exists dr_i \in DR\colon sa^{(i)} \text{ ist def. und} \\ & \forall j = 1, \ldots, v\colon k_{t(j)} \in ka_{q_i(t(j))}^{(i)}.K, \\ t_{\text{navail}} & \text{sonst.} \end{cases}$$

Existieren zu einer Zelle mehrere „passende“ Quelldatenräume, von denen mindestens einer einen Nullwert enthält, so ergebe sich das Ergebnis in absteigender Priorität gemäß: („kein Nullwert“ oder t_{nexist}*)* $-$ t_{navail} $-$ t_{unknown}.

Zwei Beispiele sollen diese Definition verdeutlichen:

- Gegeben seien ein Datenraum mit Fallzahlen (Objektmenge: Tumoren) über alle Landkreise Norddeutschlands, verschiedene Tumorarten und eine Reihe von Altersgruppen sowie ein Datenraum mit Bevölkerungszahlen (Objektmenge: Personen) für alle Landkreise Niedersachsens und die gleichen Altersgruppen. Die Vereinigung dieser beiden Datenräume ergibt einen Datenraum mit beiden summarischen Attributen sowie den kategoriellen Attributen des Falldatenraums. Die Bevölkerungszahlen für alle Gebiete außerhalb Niedersachsens sind t_{navail}, über die Erkrankungsdimension sind sie jeweils identisch dupliziert. Zu beachten ist, daß sich beide Quelldatenräume auf *verschiedene* Objektmengen beziehen müssen, da sonst (falls die Tumorart ein personenbezogenes Attribut wäre) aufgrund der unterschiedlichen Eigenschaftsmengen eine Vereinigung nicht möglich wäre: Bevölkerungszahlen wären nicht nach Tumorart differenzierbar.

- Zwei Falldatenräume, einmal für alle Landkreise Niedersachsens und beide Geschlechter (als Summe über die Jahre 1990–95) und zum anderen über die gleichen Landkreise und für die Jahre 1990, 1991 und 1992 (als Summe über beide Geschlechter), werden zu einem drei–dimensionalen Datenraum vereinigt. Die Gebietsdimension wird übernommen, die Geschlechtsdimension enthält beide Geschlechter und deren Vereinigung, die Zeitdimension analog die drei Einzeljahre und den Gesamtwert 1990–95. Fallzahlen für Kombinationen eines spezifischen Geschlechts mit einem Einzeljahr sind wiederum t_{navail}. Inkonsistenzen in den sich überlappenden Bereichen beider Quelldatenräume können aufgrund der eindeutigen Definition der Datenraum–Extension nicht auftreten.

Auf die Angabe der Partitionierung der kategoriellen Attribute der an einer Vereinigung beteiligten Datenräume kann unter Umständen auch verzichtet werden, wenn die oben aufgelisteten Vorbedingungen für den Parameter **KA** bereits zu einer eindeutigen Aufteilung führen. Ansonsten bleibt dem Anwender die Freiheit,

[58] Hier kann $q_i(j)$ natürlich ggf. undefiniert sein.

[59] Die Identifikation „gleicher“ summarischer Attribute beschränkt sich bzgl. der jeweiligen Aggregierungsfunktionen auf rollenbehaftete Objektmengen, die auch von den betrachteten Attributen beschrieben werden, da bzgl. anderer Objektmengen gemäß Def. 4.23 eine datenraumspezifische Definition erfolgt.

[60] Hierbei wird auf die gemäß Def. 4.21 erweiterte Extension der Quelldatenräume Bezug genommen.

Attribute über der gleichen Domäne zu identifizieren oder unabhängig zu belassen. Hierbei können die angeführten Anforderungen an die Partitionierung **KA** durchaus zu Konflikten führen, die eine Vereinigung verhindern — etwa wenn in einem Quelldatenraum dr_1 ein kategorielles Attribut zwei Eigenschaften beschreibt, die in einem anderen Quelldatenraum zwei verschiedenen kategoriellen Attributen zugeordnet sind.

Wesentlich für die Gewährleistung der auf Seite 137 angeführten Basisconstraints auf Datenräumen ist die erste Bedingung in obiger Definition, die die einheitliche Klassifikation von Objektmengen in verschiedenen Quelldatenräumen fordert. Hierdurch wird garantiert, daß die Menge für ein summarisches Attribut relevanter kategorieller Attribute unverändert bleibt. Unter Heranziehung der anderen genannten Bedingungen sind die geforderten Charakteristika leicht nachzuweisen.

Weitere mengenbezogene Operatoren neben der Vereinigung werden in *MADEIRA* nicht vorgesehen. Insbesondere sind Durchschnitt und Differenz bereits durch die Restriktion im Rahmen der Ableitung abgedeckt. Außerdem wird das „Erweitern“ von Datenräumen um neue Dimensionen oder Kategorien bewußt auf die Vereinigungsoperation beschränkt und nicht als eigener Operator definiert. Nur um zwei Datenräume für die anschließende Verknüpfung „passend“ zu machen, ist eine Erweiterung eines Datenraums sinnvoll — für sich genommen, bringt sie keine neuen Informationen.

Bei der Vereinigung von Datenräumen können durch die Einführung von Nullwerten sowie die Duplizierung von Werten über irrelevante Dimensionen große „homogene“ Teildatenräume entstehen. Wie schon in Abschnitt 4.2.6 dargestellt, müssen diese natürlich in einer Implementierung von Datenräumen nicht materialisiert werden, sondern sind geeignet zu komprimieren.

4.3.5 Verarbeitungsverfahren zur Berechnung neuer Maßzahlen

Während durch die bisher vorgestellten Operationen die in einem Datenraum enthaltenen Maßzahlen jeweils unverändert bleiben, werden in diesem Abschnitt nun Operatoren zur Berechnung *neuer* Maßzahlen aus den Zellwerten eines Datenraums eingeführt. Diese Operatoren stützen sich im wesentlichen auf die Berechnungsvorschriften F einer vorgegebenen Zielmaßzahl ab. Drei grundlegende Fälle sollen unterschieden werden:

1. In einer zellenweisen Verarbeitung bleibt die Dimensionalität des Datenraums unverändert; aus den verschiedenen Werten *einer* Zelle wird der Wert einer neuen Maßzahl in einer entsprechenden Zelle des Zieldatenraums ermittelt.

2. Wie bei der Aggregierung im Rahmen der Ableitungsoperation werden Kategorien gruppiert, und zu den so erhaltenen Wertegruppen wird jeweils durch explizite Angabe einer Aggregierungsfunktion (unabhängig von den Aggregierungsfunktionen der betroffenen summarischen Attribute) ein neuer Maßzahlwert auf einer gröberen Granularitätsebene berechnet. Der Einfachheit halber werden hierbei jeweils nur Werte *eines* summarischen Attributs aggregiert.

3. Bei der freien Verknüpfung von Maßzahlwerten eines Datenraums werden einige Dimensionen des Datenraums eliminiert, indem über die Kategorien der entsprechenden kategoriellen Attribute durch die Berechnungsfunktion iteriert und aggregiert wird; die übrigen Dimensionen bleiben unverändert.

Fall (1) ist lediglich ein einfach zu definierender Spezialfall von (3), der hier aber, sozusagen als „Einstieg“ in die Thematik, separat definiert werden soll. Zudem wird er auch in *VIOLA* zur klareren Darstellung und Strukturierung von Datenanalysen durch einen eigenständigen Operator realisiert werden. Der Verzicht auf die Kombination von (2) und (3), also die *teilweise* Aggregierung über *mehrere* Dimensionen unter Einbeziehung *verschiedener* Maßzahlen erspart uns sehr komplexe (und aufwendig zu implementierende) Definitionen und birgt zugleich keine große Einschränkung hinsichtlich der Palette durch *MADEIRA* unterstützbarer Datenanalysen. Zum einen werden entsprechende Operationen recht selten sinnvoll angewendet und zum anderen ließe sich das gewünschte Ergebnis jeweils auch unter Einbeziehung von Restriktion und Datenraumvereinigung sowie Nacheinanderausführung von (2) und (3) durch Nutzung der gegebenen Operatoren erreichen.

Wie schon in Abschnitt 4.2.6 diskutiert, fließen auch in den Ergebnis–Datenräumen jeweils in die Werte einer Zelle nur Angaben zu denjenigen Teilen der betrachteten Objektmengen ein, die durch die entsprechenden Zellkoordinaten eingegrenzt werden. Somit können also bei der Berechnung nur Maßzahlwerte aus Zellen berücksichtigt werden, die zu *einer* Ergebniszelle zusammengefaßt werden. Aus diesem Grund werden hier auch keine Disaggregierungsoperationen betrachtet (vgl. Abschnitt 4.2.5).

Alle drei Typen von Analyseoperationen sind (wie auch die anderen *MADEIRA*–Operationen) in dem Sinne objekterzeugend und nicht objekterweiternd, als durch sie jeweils ein *neuer* Datenraum entsteht, der nur die neu berechnete(n) Maßzahl(en) enthält. Die summarischen Attribute des Quelldatenraums werden vollständig durch zu den Zielmaßzahlen neu definierte ersetzt. Dadurch können — wie schon bei der Selektion summarischer Attribute — evtl. rollenbehaftete Objektmengen und kategorielle Attribute, die nicht mehr relevant sind, im Berechnungsergebnis wegfallen. Der Quelldatenraum wird nicht verändert. Dies entspricht der typischen Anwendung von Datenanalyseoperationen; ggf. können Quell- und Ergebnisdatenraum nachträglich wieder vereinigt werden.

In den nachfolgenden Definitionen wird jeweils eine Funktion $f \in \{f^{\text{cell}}, f^{\text{eaggr}}, f^{\text{join}}\}$ angegeben, die einen Datenraum mit einem summarischen Attribut zur gewünschten Maßzahl sowie evtl. weiteren summarischen Attributen zu im gleichen Berechnungsvorgang ergänzend ermittelten Größen berechnet. Streng genommen handelt es sich bei dieser Funktion f jeweils lediglich um eine (nicht notwendigerweise rechtseindeutige) *Relation*, da unter Umständen auch mehrere alternative Berechnungswege zur Ermittlung von *verschiedenen* Spezialisierungen der betreffenden Zielmaßzahl (etwa aus unterschiedlichen summarischen Attributen des Quellwürfels) bestehen können. Dies ist insbesondere dann der Fall, wenn es sich um allgemeine Maßzahlen, wie z. B. Summe oder Quotient, handelt. Aus Gründen der Einheitlichkeit mit den übrigen Operatoren von *MADEIRA* soll hier jedoch auch von einer (in diesem Fall *nichtdeterministischen*) Funktion gesprochen werden. Der Nichtdeterminismus kann jeweils entweder durch „zufällige“ Selektion bzw. Einführung einer Ordnung auf den Berechnungsalternativen oder interaktiv vom Benutzer der Analyseumgebung *VIOLA* durch Auswahl einer Alternative aufgelöst werden.[61] Oftmals läßt sich eine derartige Situation jedoch bereits vermeiden, indem der Quelldatenraum vorher auf die interessierenden Attribute eingeschränkt wird.

Die Einhaltung der Konsistenzbedingungen für Datenräume ergibt sich direkt aus den Operatordefinitionen, wobei die Spezifikation der Summierungsfunktionen der neu bestimmten summarischen Attribute jeweils — auf Basis der Berechnungsfunktionen der zugehörigen Maßzahlen — aus Komplexitätsgründen nur angedeutet wird. Die Analogie zur Festlegung der Datenraumextension sollte jedoch stets ersichtlich sein. Auch auf die Aggregierungsfunktionen der neuen summarischen Attribute wird jeweils nicht näher eingegangen; diese werden (unter Berücksichtigung von Def. 4.23) direkt von den vorgegebenen Maßzahlen übernommen.

Sei im folgenden $dr = (OR, KA, SA, f^{\text{ext}})$ ein Quelldatenraum mit $KA = \{ka_1, \ldots, ka_m\}$ gegeben.

Definition 4.34 (Zellenweise Verarbeitung von Datenräumen) *Die zellenweise Verarbeitung von Datenräumen sei definiert durch eine (partielle) Funktion* $f^{\text{cell}} : \mathcal{DR} \times \mathcal{M} \to \mathcal{DR}$. *Sei* $mz' \in \mathcal{M}$ *die Zielmaßzahl.*

Existieren eine Attributmenge $\widetilde{SA} = \{sa_1, \ldots, sa_s\} \subseteq SA$ *sowie eine Menge zusätzlicher Maßzahlen* $MZ = \{mz_1, \ldots, mz_u\} \subseteq \mathcal{M}$ *und eine Familie* $(f_{mz})_{mz \in MZ}$ *von Berechnungsfunktionen, so daß*

$$(\{sa_1.mz, \ldots, sa_s.mz\}, MZ, \emptyset, \cdot, (f_{mz} : sa_1.T \times \ldots \times sa_s.T \to mz.T)_{mz \in MZ}) \in mz'.F \quad ,$$

dann sei $f^{\text{cell}}(dr, mz) = dr' \stackrel{\text{def}}{=} (OR', KA', \{sa'_g \mid g = 1, \ldots, u\}, f^{\text{ext}\prime})$ *definiert, wobei gelte:*

- $sa'_g.mz \stackrel{\text{def}}{=} mz_g$,
- *die* sa'_g*–Summierungsfunktion ergibt sich gemäß* $f^{\text{sum}}(\cdot) \stackrel{\text{def}}{=} f_{mz_g}(sa_1.f^{\text{sum}}(\cdot), \ldots, sa_s.f^{\text{sum}}(\cdot))$ *aus den Summierungsfunktionen der Quellattribute,*

[61] Durch weitere spezifische Funktionsparameter wäre natürlich auch eine Eindeutigkeit der Funktion zu erreichen. Hiervon wollen wir jedoch absehen, da eine derartige Parameterspezifikation zu stark auf den jeweils betrachteten Datenraum abgestimmt wäre und somit der flexiblen Nutzung von Analyseoperatoren zuwiderlaufen würde, wie sie eingangs von Abschnitt 4.3 motiviert wurde. Auf die Behandlung des Nichtdeterminismus kommen wir in Abschnitt 5.2.2 noch kurz zu sprechen.

- $sa'_g.OR \stackrel{\text{def}}{=} OR' \stackrel{\text{def}}{=} \bigcup_{sa \in \widetilde{SA}} sa.OR$ *und*
- $KA' \stackrel{\text{def}}{=} \{ka' = (ka.D, ka.K, ka.EOR \cap \mathcal{E} \times OR') \mid (ka \in KA) \wedge (ka.OR \cap OR' \neq \emptyset)\}$.

Sei $KA' = \{ka'_1, \ldots, ka'_n\}$*, so daß* ka'_i *aus* $ka_{p(i)}$ *mit* $p(i) \in \{1, \ldots, m\}, i = 1, \ldots, n$*, hervorgegangen ist. Dann gelte (mit* $k_j \in ka_j.K$ *für* $j = 1, \ldots, m$*):*

$$f^{\text{ext}\prime}_{sa'_g}(k_{p(1)}, \ldots, k_{p(n)}) \stackrel{\text{def}}{=} f_{mz_g}(f^{\text{ext}}_{sa_1}(k_1, \ldots, k_m), \ldots, f^{\text{ext}}_{sa_l}(k_1, \ldots, k_m)) \quad .$$

Ein Beispiel für diese Operation wäre etwa die Berechnung von rohen Inzidenzraten durch Division von Neuerkrankungszahlen durch die entsprechenden Bevölkerungszahlen (und Multiplikation mit 100.000, um die Rate auf jeweils 100.000 Einwohner zu beziehen). Eventuell kommen noch obere und untere Grenze von Konfidenzintervallen als zusätzliche Maßzahlen hinzu.

Die explizite Aggregierung wird ähnlich definiert wie die Ableitungsoperation, nur daß diesmal explizit die zu nutzende Aggregierungsfunktion vorgegeben und somit eine *neue* Maßzahl (in eher seltenen Fällen mit weiteren zusätzlichen Größen) berechnet wird.

Definition 4.35 (Explizite Aggregierung) *Die explizite Aggregierung sei definiert durch eine (partielle) Funktion* $f^{\text{eaggr}} : \mathcal{DR} \times \mathcal{KA} \times \mathcal{KA} \times \mathcal{M} \to \mathcal{DR}$*. Seien* $mz' \in \mathcal{M}$ *eine Zielmaßzahl sowie* $ka \in KA, ka' \in \mathcal{KA}$ *ein kategorielles Quell- bzw. Zielattribut zur Spezifikation der Granularität der Aggregierung.*

Folgende Bedingungen müssen erfüllt sein, damit $f^{\text{eaggr}}(dr, ka, ka', mz) = dr'$ *definiert ist:*

- $ka.EOR = ka'.EOR$*,*
- *mz ist aus allen summarischen Attributen von dr berechenbar, d. h. für alle* $sa \in SA$ *gibt es jeweils eine Menge* $EOR \subseteq \mathcal{E} \times \mathcal{OR}$*, die alle durch ka klassifizierten Eigenschaften der sa–Objektmengen umfaßt* $(\emptyset \neq \mathcal{E} \times sa.OR \cap ka.EOR \subseteq EOR)$*, sowie eine Menge* $MZ = \{mz_1, \ldots, mz_u\}$ *von Zielmaßzahlen und eine Familie* $(f_{sa,mz})_{mz \in MZ}$ *von Berechnungsfunktionen*[62]*, so daß*

$$(\{sa.mz\}, MZ, \{EOR\}, disjunkt?, (f_{sa,mz} : \mathbb{N}^{sa.T} \to mz.T)_{mz \in MZ}) \in mz'.F \quad , \text{ und}$$

- *alle Kategorien von* ka' *sind gemäß Def. 4.30 ableitbar aus* $ka.K$*. Gilt „*$disjunkt? = true$*“ für die Berechnung von mz aus mindestens einem summarischen Attribut aus SA, so müssen die Kategorien sogar disjunkt ableitbar sein.*

Dann sei $SA' \stackrel{\text{def}}{=} \bigcup_{sa \in SA} \{sa'_g \stackrel{\text{def}}{=} (mz_g, sa.OR, f^{\text{sum}\prime}_g, f^{\text{aggr}\prime}_g) \mid g = 1, \ldots, u\}$*, wobei sich* $f^{\text{sum}\prime}_g$ *aus* $sa.f^{\text{sum}}$*,* f_{sa,mz_g} *und der konkreten Ableitung der Kategorien von* ka' *ergibt.*[63]

Mit $KA' \stackrel{\text{def}}{=} KA \setminus \{ka\} \cup \{ka'\}$ *erhalten wir — ähnlich zur Ableitung in Def. 4.31 — als Ergebnis der Aggregierung* $dr' \stackrel{\text{def}}{=} (OR, KA', SA', f^{\text{ext}\prime})$*.*

Hierbei gelte (mit $sa'_g \in SA'$*, hervorgegangen aus* $sa \in SA$ *über* $mz_g \in MZ$*, und* $k' \in ka'.K$ *beliebig) für die Extension*

$$f^{\text{ext}\prime}_{sa'_g}(\ldots, k', \ldots) \stackrel{\text{def}}{=} f_{sa,mz_g}(\{f^{\text{ext}}_{sa}(\ldots, k, \ldots) \mid k \in K_{k'}\}) \quad ,$$

wobei $K_{k'}$ *jeweils wie in Def. 4.30 die Menge zur Kategorie* k' *zusammengefaßter Kategorien aus dem Attribut ka von dr bezeichne.*

Während die implizite Aggregierung unmittelbar auf den einem summarischen Attribut zugeordneten Aggregierungsfunktionen basiert, kann die explizite Aggregierung prinzipiell jede Aggregierungsfunktion auf jedes summarische Attribut anwenden. Zum Beispiel könnte man so gleichzeitig die Werte aller summarischen Attribute in bestimmten Gruppen aufsummieren oder deren Durchschnitt bilden. Ein konkretes Beispiel wäre

[62] Diese Funktionen haben i. a. ähnliche Gestalt wie die in Def. 4.14 eingeführten Aggregierungsfunktionen.

[63] Dies ist direkt herzuleiten, wird hier aber nicht explizit ausformuliert, da eine geschlossene Formel recht umfangreich wäre.

etwa die Extraktion der nach Regierungsbezirken klassifizierten minimalen oder maximalen Erkrankungsraten der jeweils enthaltenen Gemeinden in Niedersachsen.[64]

Der Einfachheit halber wird mit einer Anwendung des expliziten Aggregierungsoperators jeweils nur über *ein* kategorielles Attribut auf einmal aggregiert. Im Gegensatz zur impliziten Aggregierung stellt dies eine gewisse Einschränkung der Mächtigkeit des Operators dar, da die Nacheinanderausführung mehrerer Aggregierungsschritte über verschiedene kategorielle Attribute i. a. (falls es sich nicht um eine distributive Aggregierungsfunktion handelt) nicht kommutativ ist und zudem eine andere Semantik als die gleichzeitige Aggregierung über diese Attribute hätte. Typische Anwendungen der Aggregierung über mehrere Attribute fassen jedoch in der Regel Dimensionen vollständig zusammen und sind somit durch die weiter unten definierte freie Verknüpfung von Maßzahlen abgedeckt.

Wie man an den angeführten Anwendbarkeitsbedingungen sieht, ist die explizite Aggregierung nur sinnvoll definierbar, wenn *alle* summarischen Attribute des Datenraums durch das betreffende kategorielle Attribut *ka* klassifiziert werden. Nur so kann die in der jeweiligen Berechnungsvorschrift implizierte Aggregierung über Eigenschaften von *ka* auf das jeweilige summarische Attribut bzw. seine Objektmengen bezogen werden. Wie auch in der nachfolgenden Definition freier Maßzahlverknüpfung müssen jeweils alle Eigenschaften von *ka*, die sich auf Objektmengen eines bestimmten summarischen Attributs beziehen, *vollständig* in der durch die jeweilige Berechnungsvorschrift gegebenen Eigenschaftsmenge enthalten sein. Diese hat also stets die Semantik „Aggregiere über diese Eigenschaften *und keine anderen* . . . “.

Das Ergebnis der expliziten Aggregierung kann sowohl in der Wahl der Berechnungsvorschrift als auch im Hinblick auf die Zerlegung der Kategorien aus ka' nichtdeterministisch sein. Gegebenenfalls sollte vor Anwendung des Operators der Ausgangsdatenraum (etwa auf Kategorien einer Aggregierungsebene) eingeschränkt werden.

Verknüpfungen von Maßzahlen definieren ihre Berechnungen oftmals nicht zellenweise, sondern „aggregieren“ über bestimmte kategorielle Attribute eines Datenraums vollständig.

Definition 4.36 (Verknüpfung von Maßzahlen) *Die Verknüpfung von Maßzahlen sei definiert durch eine (partielle) Funktion* $f^{\text{join}}\colon \mathcal{DR} \times \mathcal{M} \to \mathcal{DR}$. *Sei* $mz' \in \mathcal{M}$ *eine Zielmaßzahl.*

Existieren eine Attributmenge $\widetilde{SA} = \{sa_1, \ldots, sa_s\} \subseteq SA$ *sowie eine Menge von (zusätzlichen) Zielmaßzahlen* $MZ = \{mz_1, \ldots, mz_u\}$ *und eine Familie* $(f_{mz})_{mz \in MZ}$ *von Berechnungsfunktionen, so daß*

$$(\{sa_1.mz, \ldots, sa_s.mz\}, MZ, \mathbf{EOR}, \textit{disjunkt}?, (f_{mz}\colon 2^{sa_1.T \times \ldots \times sa_s.T} \to mz.T)_{mz \in MZ}) \in mz'.F \quad ,$$

und gibt es mit $OR' \stackrel{\text{def}}{=} \bigcup_{sa \in \widetilde{SA}} sa.OR$ *für alle Eigenschaftsmengen aus* **EOR** *jeweils mindestens ein kategorielles Attribut in KA, das eines der summarischen Zielattribute bzw. dessen Objektmengen beschreibt und über das gemäß* $mz'.F$ *aggregiert werden kann, d. h.* [65]

$$\forall EOR \in \mathbf{EOR}\colon \widetilde{KA}_{EOR} \stackrel{\text{def}}{=} \{ka \in KA \mid \emptyset \neq ka.EOR \cap \mathcal{E} \times OR' \subseteq EOR\} \neq \emptyset \quad ,$$

wobei die Kategorien all dieser kategoriellen Attribute jeweils paarweise disjunkt sein müssen, falls $\textit{disjunkt}? = \textit{true}$, *dann sei* $f^{\text{join}}(dr, mz) = dr' \stackrel{\text{def}}{=} (OR', KA', \{sa'_g \mid g = 1, \ldots, u\}, f^{\text{ext}\prime})$ *definiert, wobei gelte:*

- *mit* $\widetilde{KA} \stackrel{\text{def}}{=} \cup_{EOR \in \mathbf{EOR}} \widetilde{KA}_{EOR}$ *ergibt sich die Menge der kategoriellen Attribute von* dr' *als Vereinigung der beibehaltenen Attribute (unter Beachtung der Einschränkung der Objektmengen zu* OR'*) und der mit der Maßzahlberechnung zu einer Gesamt–Kategorie aggregierten Attribute aus* $\widetilde{KA}$, *also*

$$KA' \stackrel{\text{def}}{=} \{(ka.D, ka.K, ka.EOR \cap \mathcal{E} \times OR') \mid (ka \in KA \setminus \widetilde{KA}) \wedge (ka.OR \cap OR' \neq \emptyset)\} \cup \{(ka.D, \{\bigvee ka.K\}, ka.EOR \cap \mathcal{E} \times OR') \mid ka \in \widetilde{KA}\} \quad ,$$

[64] Man beachte, daß der Zieldatenraum als Raumbezug die Regierungsbezirke erhält; die Information, *welche* Gemeinde jeweils die niedrigste bzw. höchste Rate hat, geht verloren.

[65] Hierbei sei es unerheblich, ob die Mengen $\widetilde{KA}_{EOR}$ paarweise disjunkt sind.

- $sa'_g.mz \stackrel{\text{def}}{=} mz_g$,
- $sa'_g.OR \stackrel{\text{def}}{=} OR'$ *und*
- *die Summierungsfunktion von* sa'_g *ergibt sich ähnlich wie bei der expliziten Aggregierung aus der Maßzahlberechnungsfunktion* f_{mz_g}*, den Summierungsfunktionen der summarischen Quellattribute und den Kategorienmengen der kategoriellen Attribute in* $\widetilde{KA}$*.*

Sei $KA' = \{ka'_1, \ldots, ka'_n\}$*, so daß* ka'_i *aus* $ka_{p(i)}$ *mit* $p(i) \in \{1, \ldots, m\}$, $i = 1, \ldots, n$*, hervorgegangen ist. Dann gelte (mit* $k_j \in ka_j.K$ *für die beibehaltenen Attribute, also* $j \in \{1, \ldots, m\}$ *und* $ka_j \notin \widetilde{KA}$*) für die Extension* $f^{\text{ext}\prime}$ *von* dr' *bzgl. eines summarischen Attributs* sa'_g*:*

$$f^{\text{ext}\prime}_{sa'_g}(k_{p(1)}, \ldots, k_{p(n)}) \stackrel{\text{def}}{=} f_{mz_g}(\{(f^{\text{ext}}_{sa_1}(k_1, \ldots, k_m), \ldots, f^{\text{ext}}_{sa_s}(k_1, \ldots, k_m)) \mid \forall j = 1, \ldots, m \text{ mit (die aggregierten Attribute:) } ka_j \in \widetilde{KA}: k_j \in ka_j.K\}) \ .$$

Ein Beispiel für eine derartige Maßzahlberechnung wäre etwa die alters- und geschlechtsstandardisierte Rate, die über Alters- und Geschlechtsattribute disjunkt aggregiert. Vor ihrer Ermittlung ist es i. a. sinnvoll, die Kategorienmengen der entsprechenden kategoriellen Attribute auf bestimmte Aggregierungsebenen einzuschränken. Üblicherweise werden z. B. 5–Jahres–Altersgruppen verwendet. Wiederum können alternative Berechnungswege gemäß $mz.F$ existieren, zwischen denen zu wählen ist. Etwa können als Ausgangsdaten alternativ alters- und geschlechtsspezifische Raten oder entsprechende Fall- und Bevölkerungszahlen zur Studienpopulation vorliegen.

Man beachte, daß stets über alle „möglichen“ kategoriellen Attribute aggregiert wird, d. h. die Mengen $\widetilde{KA}_{EOR}$ werden auch dann weiter aufgefüllt, wenn bereits ein zu betrachtendes Attribut gefunden wurde. Dies läßt sich nur verhindern, indem eine andere Maßzahl zur Berechnung gewählt bzw. definiert wird, die restriktiver in der Wahl zu aggregierender Attribute ist.

4.3.6 Drill–through: Abfrage von Mikrodaten aus Würfelzellen

Im Zuge einer Datenanalyse kann es gelegentlich hilfreich sein, die hinter einer Zelle eines Datenraums liegenden, als Basisdaten zugreifbaren Mikrodaten zu inspizieren. Diese Mikrodatenabfrage erfolgt mittels einer „Drill–through“–Operation, die die Eigenschaftsausprägungen einer interessierenden Objektmenge auf feinstmöglicher Granularitätsebene liefert.

Die Wertebereiche der Attribute des Mikrodatenraums sind hierbei durch die jeweiligen Koordinaten der inspizierten Zelle eingeschränkt, sofern Eigenschaften abgefragt werden, die auch durch kategorielle Attribute des betrachteten Makrodatenraums repräsentiert werden. Ein derartiges Attribut ist (falls existent) eindeutig, da gemäß der Definition von Makrodatenräumen (Def. 4.20) jede Eigenschaft nur durch maximal ein kategorielles Attribut beschrieben wird. Relationalen Attributen zu nicht im Datenraum referenzierten Eigenschaften wird die jeweilige Wurzelkategorie als Wertebereich zugewiesen.

Definition 4.37 (Mikrodatenabfrage (Drill–through)) *Die Mikrodatenabfrage sei definiert durch eine (partielle) Funktion* $f^{\text{mik}}: \mathcal{DR} \times 2^{\mathcal{K}} \times \mathcal{OR} \times 2^{\mathcal{E}} \to \mathcal{DR}^{\text{mi}}$.

Seien $dr = (OR, \{ka_1, \ldots, ka_m\}, SA, f^{\text{ext}})$ *ein Datenraum,* $K = \{k_1, \ldots, k_m\} \subseteq \mathcal{K}$ *mit* $k_j \in ka_j.K$ *für* $j = 1, \ldots, m$ *Koordinaten einer Zelle von dr und* $(O, r) \in OR$ *eine rollenbehaftete Objektmenge. Ferner sei* $E \subseteq f^{\text{eig}}(O)$ *eine endliche, nicht–leere Menge von Eigenschaften der Objekte aus O.*

Dann sei $f^{\text{mik}}(dr, K, (O, r), E) = dr^{\text{mi}} \stackrel{\text{def}}{=} (O, A, f^{\text{ext}\prime})$ *definiert, wobei*

$$A \stackrel{\text{def}}{=} \{(e, k) \in E \times \mathcal{K} \mid k = \begin{cases} k_j & \text{falls } \exists ka_j \in KA: (e, O, r) \in ka_j.EOR, \\ k_0 & \text{sonst} \end{cases}\}$$

und $\quad \text{ext}(O) \stackrel{\text{def}}{=} \{o \in dr.\text{ext}(O, r) \mid \forall j = 1, \ldots, m \, \forall e \in \mathcal{E}: ((e, O, r) \in ka_j.EOR \Rightarrow o \vdash_e k_j)\} \ .$

Die Extension $f^{\text{ext}\prime}$ des Mikrodatenraums ergebe sich, wie in Def. 4.5 beschrieben, aus der Abbildung f^{val} als genaueste bekannte Angabe zur jeweiligen Eigenschaft. Insbesondere erhält man einen Nullwert k_0^{δ}, falls die Angaben zu den angefragten Eigenschaften nicht verfügbar sind.

Der Drill–through–Operator kann etwa sinnvoll im Rahmen raumbezogener Clusteranalysen genutzt werden, wenn die Patienten eines gefundenen Clusters mit ihren Eigenschaften für die korrekte Interpretation der Analyse dargestellt oder mit ihren genauen Wohnortkoordinaten in eine Karte eingezeichnet werden sollen. Eine Weiterverarbeitung dieser Mikrodaten ist im Rahmen von *MADEIRA* jedoch nicht vorgesehen.

Eine analoge Abfrage zugrundeliegender Basis–*Makrodaten* wird durch *MADEIRA* nicht explizit unterstützt, da die auszugebende Granularitätsebene nicht eindeutig definiert wäre. Außerdem bleiben im Rahmen des Datenmodells vielfältige Möglichkeiten, direkt auf die jeweils interessierenden Makrodaten zuzugreifen.

Bei der Implementierung des Drill–through–Operators ist natürlich — ähnlich wie bei der Bestimmung des Nullwertes t_{nexist} bei der Konsolidierung von Makro- aus Mikrodaten — zu beachten, daß die abgefragten Angaben nicht aus der Extension des untersuchten Makrodatenraums herzuleiten sind. Über die durch den Datenraum beschriebenen Objektmengen bzw. deren Instanzen ist jeweils eine Verbindung zum relevanten Mikrodatenbestand herzustellen, um aus diesem alle angeforderten Angaben auszulesen. Sind angefragte Eigenschaften nicht im Mikrodatenbestand beschrieben, sind geeignete Nullwerte (entsprechend t_{navail}) zu liefern.

4.3.7 Einbeziehung zusammengesetzter Kategorien durch Split und Merge

Schließlich wird die *MADEIRA*–Algebra noch um Operatoren zur Erzeugung und Auflösung kategorieller Attribute über zusammengesetzten Eigenschaften ergänzt.[66] Die Merge–Operation „verschmilzt“ hierbei jeweils ein Paar kategorieller Attribute (sowie Paare von Werten ihrer Kategorienmengen), die paarweise Eigenschaften der gleichen Objektmengen beschreiben.

Definition 4.38 (Merge von kategoriellen Attributen) *Das Mergen zweier kategorieller Attribute sei definiert durch die (partielle) Funktion $f^{\text{merge}} : \mathcal{DR} \times \mathcal{KA} \times \mathcal{KA} \to \mathcal{DR}$. Es seien $dr = (OR, KA, SA, f^{\text{ext}})$ ein Datenraum mit $KA = \{ka_1, \ldots, ka_m\}$ sowie $ka_i \neq ka_j \in KA$ zwei zusammenzuführende Attribute.*

Beschreiben diese beiden Attribute paarweise Eigenschaften derselben Objektmengen, also

$$\exists\, f_{ij} : ka_i.EOR \to ka_j.EOR \text{ bijektiv, so daß } f_{ij}((e,O,r)) = (e',O',r') \Rightarrow O = O' \wedge r = r' \quad ,$$

und existiert in $dr.KA$ noch kein Attribut, das Kompositionen dieser Eigenschaftspaare beschreibt, so sei $f^{\text{merge}}(dr, ka_i, ka_j) = dr' \stackrel{\text{def}}{=} (OR, KA', SA, f^{\text{ext}\prime})$ definiert, wobei

$$KA' \stackrel{\text{def}}{=} KA \setminus \{ka_i, ka_j\} \cup \{(ka_i.D \circ ka_j.D, ka_i.K \times ka_j.K, \{(e_1 \circ e_2, O, r) \mid (e_1,O,r) \in ka_i.EOR \wedge f_{ij}((e_1,O,r)) = (e_2,O,r)\})\} \quad .$$

Die Extension von dr' ergebe sich einfach aus der von dr, indem (mit $k_v \in ka_v$ für $v = 1, \ldots, m$ sowie $sa \in SA$) zusammengesetzte Zellkoordinaten in der Form

$$f_{sa}^{\text{ext}\prime}(\ldots,(k_i,k_j),\ldots) \stackrel{\text{def}}{=} f_{sa}^{\text{ext}}(k_1,\ldots,k_m) \quad .$$

wieder aufgesplittet werden.

Die Anpassung der Aggregierungsfunktionen der summarischen Attribute des Datenraums hinsichtlich der bearbeiteten kategoriellen Attribute wird in obiger Definition (wie auch in der folgenden Spezifikation des Split–Operators) nicht explizit thematisiert. Lediglich im Sinne von Def. 4.23 ist eine Anpassung bezüglich

[66] Diese Operatoren Split und Merge kann man sich im Hinblick auf die durch Komposition erfolgende „Kapselung“ von Attributen in gewisser Weise analog zum Nest und Unnest des *NF2*–Modells [JS82] vorstellen.

der identischen Aggregierung über jeweils nicht relevante Objektmengen und Eigenschaften vorzunehmen, die jedoch an der prinzipiellen Attributsemantik nichts verändert.

Im wesentlichen invers zum Merge zerlegt die Split–Operation ausgewählte zusammengesetzte kategorielle Attribute wieder in ihre Komponenten.

Definition 4.39 (Split kategorieller Attribute über zusammengesetzten Dimensionen) *Das Aufsplitten von zusammengesetzten kategoriellen Attributen sei durch eine (partielle) Funktion* $f^{\text{split}} : \mathcal{DR} \times \mathcal{KA} \to \mathcal{DR}$ *definiert. Seien* $dr = (OR, KA, SA, f^{\text{ext}})$ *ein Datenraum mit* $KA = \{ka_1, \ldots, ka_m\}$ *sowie* $ka \in KA$ *ein aufzusplittendes Attribut über einer zusammengesetzten Domäne* $ka.D = D_1 \circ D_2$.

Es bezeichne $K^{(1)} \stackrel{\text{def}}{=} \{k_1 \in D_1 \mid \exists k_2 \in D_2 : (k_1, k_2) \in ka.K\}$ *die enthaltenen Kategorien der ersten Komponentendomäne; analog sei* $K^{(2)} \stackrel{\text{def}}{=} \{k_2 \in D_2 \mid \exists k_1 \in D_1 : (k_1, k_2) \in ka.K\}$ *definiert. Hieraus ergeben sich als Ergebnis der Zerlegung von ka die zwei Attribute*

$$\begin{aligned} ka^{(1)} &\stackrel{\text{def}}{=} (D_1, K^{(1)}, \{(e_1, O, r) \in \mathcal{E} \times \mathcal{OR} \mid \exists e_2 \in \mathcal{E}_{D_2} : (e_1 \circ e_2, O, r) \in ka.EOR\}) \text{ und} \\ ka^{(2)} &\stackrel{\text{def}}{=} (D_2, K^{(2)}, \{(e_2, O, r) \in \mathcal{E} \times \mathcal{OR} \mid \exists e_1 \in \mathcal{E}_{D_1} : (e_1 \circ e_2, O, r) \in ka.EOR\}) \ . \end{aligned}$$

Existieren in $dr.KA$ *noch keine Attribute über Eigenschaften dieser beiden Attribute, sei* $f^{\text{split}}(dr, ka) = dr' \stackrel{\text{def}}{=} (OR, KA', SA, f^{\text{ext}\prime})$ *definiert, wobei* $KA' \stackrel{\text{def}}{=} KA \setminus \{ka\} \cup \{ka^{(1)}, ka^{(2)}\}$.

KA' *werde geschrieben als* $\{ka'_1, \ldots, ka'_n\}$*; seien* $k_j \in ka'_i.K$ *für* $i = 1, \ldots, n$*. Dann ergebe sich die Extension von* dr' *zu einem summarischen Attribut* $sa \in SA$ *in der Form*

$$f_{sa}^{\text{ext}\prime}(k_1, \ldots, k_n) \stackrel{\text{def}}{=} \begin{cases} f_{sa}^{\text{ext}}(\ldots, (k_i, k_j), \ldots) & \text{falls } (k_i, k_j) \in ka.K, \\ t_{\text{navail}} & \text{sonst.} \end{cases}$$

Der Split bildet die inverse Operation zu einem Merge. Andersherum gilt dies jedoch nicht, da durch eine Split–Operation Nullwerte zu im Ausgangsdatenraum *dr* nicht vorhandenen Kategorienkombinationen entstehen, wenn ein zusammengesetztes Attribut aufgelöst wird, das nicht dem vollständigen Kreuzprodukt zweier Kategorienmengen entspricht. Durch ein anschließendes Merge erhält man also einen gegenüber *dr* um einige Nullwerte und die neuen Kategorienkombinationen erweiterten Datenraum.

4.3.8 Umbenennung der Rolle einer Objektmenge

Die Rolle einer Objektmenge dient im Rahmen der Beschreibung von Makrodaten insbesondere zur korrekten Identifikation zueinander passender Attribute bei der Verknüpfung von Maßzahlen bzw. Datenräumen sowie der richtigen Verwendung von Quellmaßzahlen bei der Durchführung von Berechnungen. Die Maßzahlwerte selbst sind jedoch unabhängig von den konkreten Rollen der beteiligten Objektmengen.

Aus dieser Überlegung heraus spricht nichts dagegen, Rollen von Objektmengen in Datenräumen zu ändern, also quasi einfach „umzubenennen“, sofern weiterhin zwischen verschieden zu nutzenden Objektmengen differenziert werden kann. Eine derartige Umbenennung ist dann sinnvoll, wenn ein Datenraum in unterschiedlichen Rollen in verschiedene Berechnungen einfließen soll. So kann etwa, wie schon eingangs von Abschnitt 4.3 am Beispiel erläutert, ein Fall- oder Bevölkerungsdatensatz einmal zur Beschreibung einer Studienpopulation und ein anderes Mal als Vergleichsdatensatz in Form einer Standardpopulation für eine andere Studienpopulation verwendet werden. Wie man an diesem Beispiel sieht, erfolgt die Rollenumbenennung typischerweise für Datenräume mit eher einfachen Maßzahlen, die sich auf Objektmengen in nur einer oder zumindest nur sehr wenigen verschiedenen Rollen beziehen.

Definition 4.40 (Umbenennung von Rollen) *Die Umbenennung der Rolle einer Objektmenge in einem Datenraum sei definiert durch eine (partielle) Funktion* $f^{\text{role}} : \mathcal{DR} \times \mathcal{R} \times \mathcal{R} \to \mathcal{DR}$.

Seien $dr = (OR, KA, SA, f^{\text{ext}})$ *ein Datenraum sowie* $r, r' \in \mathcal{R}$ *zwei Rollen, wobei* r *in* r' *umbenannt werden soll.*

$R \stackrel{\text{def}}{=} \{r \in \mathcal{R} \mid \exists O \subseteq O\colon (O, r) \in OR\}$ sei die Menge aller in dr genutzten Rollen. Dann sei $f^{\text{role}}(dr, r, r') = dr' \stackrel{\text{def}}{=} (OR', KA', SA', f^{\text{ext}\prime})$ definiert, falls $r \in R \wedge r' \notin R$. Hierbei ergeben sich OR', KA' und SA' aus OR, KA und SA, indem alle Vorkommen von r durch r' ersetzt werden. Insbesondere gilt das auch für die Aggregierungsfunktionen der summarischen Attribute und der ihnen zugeordneten Maßzahlen, bei denen r und r' gegeneinander ausgetauscht werden.

Die Extension des Datenraums bleibt unverändert.

Falls die Aggregierung über r und r' bisher in einer Maßzahl, die durch ein summarisches Attribut von *dr* beschrieben wird, nicht stets gleich erfolgte, wird diesem Attribut somit durch f^{role} eine neue Maßzahl (mit lediglich „umbenannten“ Rollen) zugeordnet. Auch der Maßzahl zugehörige rollenbezogene Metadaten (vgl. Abschnitt 4.4.2) sind ggf. anzupassen.

4.4 Weitere Metadaten zur näheren semantischen Differenzierung und Beschreibung der Konstrukte von *MADEIRA*

Auf allen Ebenen der Schemamodellierung in *MADEIRA* werden umfangreiche und differenzierte Mengen von Metadaten miteinbezogen, die über die grundlegende Beschreibung der Struktur multidimensionaler Daten deutlich hinausgehen. Neben den Komponenten des in den vorangegangenen Abschnitten definierten formalen Datenmodells, die eine *exakte* Definition der Datenverarbeitung ermöglichen, spielen noch weitere beschreibende Angaben zu den betrachteten Daten und Analyseverfahren eine wesentliche Rolle, um der Gestaltung eines benutzungsfreundlichen und auch effizient und erweiterbar implementierbaren Analysesystems Rechnung zu tragen. Auf diesbezügliche Metadaten wird in diesem Abschnitt etwas näher eingegangen, ohne sie jedoch in den Kernbereich des Entwurfs von *MADEIRA* zu stellen.

Die in diesem Abschnitt skizzierten Metadaten haben keinen Einfluß auf die Gestalt der grundlegenden Modellkonstrukte von *MADEIRA* bzw. deren Verarbeitung in *VIOLA* — sie erleichtern deren Handhabung jedoch entscheidend. Folgende zentrale Aspekte werden betrachtet:

- textuelle Angaben zur Information des Systembenutzers (Abschnitt 4.4.1),
- eine Klassifikation von Maßzahlen zur leichteren Implementierung und einfacheren Auswahl bzw. Beschreibung der Anwendbarkeitsüberprüfung (Abschnitt 4.4.2) sowie
- domänenspezifische Angaben zu Kategorien als Quelle von Hintergrundinformationen (Abschnitt 4.4.3).

Neben den rein führenden, textuellen Metadaten ergänzen die Betrachtungen zu Maßzahlen und Kategorien (den beiden Basiskomponenten zur Modellierung der summarischen bzw. kategoriellen Attribute eines Datenraums entsprechend) die bisher angestellten umfassenden Überlegungen zur Datenverarbeitung auf Ebene der *gesamten* Datenräume.

Die Palette sinnvoller Metadaten soll an dieser Stelle jedoch nicht abschließend festgelegt werden. Auswahl und Umfang von Metadaten sind flexibel zu implementieren, wofür die hier eingeführten Metadatenschemata genügend Freiraum lassen. In Abschnitt 4.4.4 wird eine zusammenfassende Einordnung der Metadaten in *MADEIRA* gegeben.

4.4.1 Angaben zur Benutzerinformation

In Datenanalysen ist es von zentraler Bedeutung, dem Datenanalysten möglichst viele Hintergrundinformationen zu den betrachteten Datensätzen zu liefern, um ihm die korrekte Interpretation der jeweils errechneten Maßzahlwerte zu ermöglichen. Diese Informationen sind oftmals nicht zu formalisieren und können somit lediglich in textueller Form angeboten werden. Entsprechend fließen sie auch nicht direkt in die Datenverarbeitung durch Operatoren von *MADEIRA* ein, sondern dienen allein zur Benutzerinformation.

Im folgenden werden einige wichtige Metadaten dieser Art kurz vorgestellt. Die Aufzählung kann sicher nicht vollständig sein, da in Abhängigkeit vom jeweiligen Anwendungsgebiet sehr unterschiedliche Informationen relevant sein können. Somit sollte eine Implementierung von *MADEIRA* gerade im Hinblick auf den Umfang textueller Metadaten leicht erweiterbar sein. Wir wollen davon ausgehen, daß die Aufgabe der Bereitstellung von Datenbasen und nutzbaren Maßzahlen sowie die Definition zugehöriger Metadaten einem Systemadministrator zufällt. Dessen Rolle soll im folgenden nicht näher interessieren. Vielmehr konzentrieren wir uns auf die Informationsbedürfnisse des Datenanalysten, der *VIOLA* zur Auswertung dieser Basisdaten einsetzt. Im Laufe einer Analyse greift er lesend auf die vordefinierten Metadaten zu. Weiterhin kann er Metadaten, die abgeleiteten Datenwürfeln automatisch zugewiesen wurden, sowohl lesen als auch modifizieren oder ergänzen, um z. B. eine individuelle Beschriftung von Analyseergebnissen vorzunehmen oder Analyseschritte und Zwischenergebnisse im Hinblick auf einen späteren Zugriff sowie im Sinne einer Analysedokumentation zu kommentieren.

Sei $\mathcal{S}$ eine Menge textueller Beschreibungen in Form beliebiger endlicher Zeichenketten (Strings) über einem vereinbarten Alphabet sowie $\mathcal{B} \subset \mathcal{S}$ eine Menge von Bezeichnern.

Die Abbildung $name: \mathcal{DR}^{\mathrm{ma}} \cup \mathcal{DR}^{\mathrm{mi}} \cup \mathcal{SA} \cup \mathcal{KA} \cup \ldots \to \mathcal{B}$ definiere zu jeder Entität des Datenmodells (vgl. Abb. 4.1) sowie auch zu beliebigen homogenen Mengen von Entitäten (Kategorienmengen, Eigenschaftsmengen etc.) einen (nicht notwendigerweise eindeutigen[67]) *Namen*. Dies ist besonders wichtig für die kurze Charakterisierung interaktiv berechneter Datenräume und ihrer Attribute, erleichtert aber auch die Handhabung der „statischen“ Typen, wie Maßzahlen, Dimensionen, Datentypen, Objektmengen etc. Vor allem auch ad–hoc „erzeugte“ Kategorien oder Aggregierungsebenen sowie im Rahmen von Analysen verwendete Parameter können so speziell identifiziert werden. Zur geeigneten Beschriftung von Auswahlmenüs sowie insbesondere für Ausgaben von Analayseergebnissen in Graphiken und Tabellen dürfte es in der Regel sinnvoll sein, verschiedene Varianten der *name*()–Funktion zu unterstützen, die unterschiedliche Kurz- und Langformen eines Namens bereitstellen.

Eine längere textuelle *Beschreibung* der betrachteten Entitäten spezifiziere zusätzlich die Abbildung $descr: \mathcal{DR}^{\mathrm{ma}} \cup \mathcal{DR}^{\mathrm{mi}} \cup \mathcal{SA} \cup \mathcal{KA} \cup \ldots \to \mathcal{S}$. Hierunter können für z. B. kategorielle Attribute Details zum Hintergrund der Auswahl bzw. der Gruppierung, für Operationsparameter Informationen zu Grund und Art ihrer Definition oder für Datenräume etwa auch Angaben zu Datenquellen oder zur Datenerhebung fallen.[68] Sehr hilfreich ist diese Funktion etwa auch für die Darstellung von Maßzahlen und Kategorien.

Darüber hinaus können weitere *Anmerkungen* oder „Fußnoten“ zu (Teilen von) Datenräumen über die Abbildung $note: \mathcal{DR}^{\mathrm{ma}} \cup \mathcal{DR}^{\mathrm{mi}} \cup \mathcal{SA} \cup \mathcal{KA} \cup \mathcal{A} \to \mathcal{S}$ angegeben werden bzw. für einzelne Zellen eines Datenraums mittels $note: \mathcal{SA} \times 2^{\mathcal{K}} \to \mathcal{S}$ auf auffällige Werte oder spezielle Umstände hinweisen. Schließlich vermerkt die Funktion $author: \mathcal{DR}^{\mathrm{ma}} \cup \mathcal{DR}^{\mathrm{mi}} \to \mathcal{S}$ den Erzeuger eines Datenraums.

In diesem Kontext seien auch Zeitstempel bzw. die Versionierung von Datenbeständen erwähnt. Eine solche kann bzw. muß in *MADEIRA* über unterschiedliche Objektmengeninstanzen eines Datenraums, ergänzt um Hinweise in den angeführten erläuternden Texten und/oder die Angabe der *Erstellungszeit* mittels $date: \mathcal{DR}^{\mathrm{ma}} \cup \mathcal{DR}^{\mathrm{mi}} \to TIME$, vorgenommen werden (*TIME* sei hierbei die Menge aller Zeitangaben). Eine genauere Modellierung, die etwaige Zeitstempel in die Datenverarbeitung einbezieht, ist — analog zu Zusammenführung und Abgleich von Datenbeständen — nicht Gegenstand von *MADEIRA*.

4.4.2 Klassen von Maßzahlen und Kategorien

Die Menge $\mathcal{M}$ aller Maßzahlen ist bisher noch weitgehend unstrukturiert. So ist z. B. ein Maß, das innerhalb einer Analyse in verschiedenen Skalierungen vorliegt, als eine Menge *verschiedener*, allerdings einfach ineinander umrechenbarer *MADEIRA*–Maßzahlen zu modellieren. Um die Implementierung von Maßzahlen, vor allem ihrer unterschiedlichen Berechnungsfunktionen, bzw. die Spezifikation für Berechnungen interessierender

[67] Je nach Verwendungszweck (nur zur Beschriftung oder etwa als Basis für Suchfunktionen) kann die Eindeutigkeit eines Namens natürlich oftmals hilfreich sein. Sie soll hier jedoch nicht vorausgesetzt werden.

[68] Dies ist nicht zu verwechseln mit dem Weg der Herleitung eines Datenraums aus Basisdaten, der durch *VIOLA* spezifiziert wird.

Maßzahlen zu erleichtern und damit auch die Anwendbarkeit von Analysefunktionen auf gegebene Maßzahlen genauer und kompakter zu charakterisieren, wird in diesem Abschnitt eine Klassifizierung von Maßzahlen anhand verschiedener Kriterien vorgenommen. Dies erlaubt zugleich, dem Nutzer von *VIOLA* umfassendere Informationen über die durchgeführten Berechnungen und erhaltenen Ergebnisse anzubieten.

Aus der Klassifikation von Maßzahlen ergibt sich (in Anlehnung an aus der KI bekannte framebasierte Modelle) eine Hierarchie allgemeinerer und speziellerer Maßzahlklassen, wie sie in Ansätzen etwa auch in [BMR94] vorgeschlagen wird. Weiterhin können Maßzahlklassen parametrisiert werden und somit gleichzeitig mehrere speziellere Klassen implementieren.

Nachfolgend werden einige Merkmale von Maßzahlen vorgestellt, anhand derer diese klassifiziert werden können. Merkmale beziehen sich auf

- den Datentyp einer Maßzahl,
- die Charakterisierung der Maßzahl selbst sowie
- Parameter zur Beschreibung von Berechnungsfunktionen oder Quellmaßzahlen.

Nicht alle Maßzahlen weisen alle aufgeführten Merkmale auf — speziellere Maßzahlen haben mehr, allgemeinere Maßzahlen weniger Merkmale. Weiterhin kann die Aufstellung sicher nicht für alle Anwendungsgebiete von *MADEIRA* vollständig sein, zumal bestimmte Merkmale nur für sehr spezielle Maßzahltypen relevant sind; je nach Zielsetzung sind weitere Merkmale zu ergänzen.

Merkmale seien jeweils dargestellt durch (partielle) Abbildungen auf der Menge $\mathcal{M}$ aller Maßzahlen:

Definition 4.41 (Merkmale auf Maßzahlen) *Die Menge aller* MERKMALE *(auch* FEATURES*) sei definiert als die Menge $\mathcal{F}^{\text{feat}}$ aller (partiellen) Abbildungen feature$: \mathcal{M} \to Y$ von der Menge aller Maßzahlen in eine beliebige Menge Y.*

$f^{\text{feat}} : \mathcal{M} \to 2^{\mathcal{F}^{\text{feat}}}$ spezifiziere die Menge aller Merkmale einer Maßzahl mz, d. h. $f^{\text{feat}}(mz) \stackrel{\text{def}}{=} \{ f \in \mathcal{F}^{\text{feat}} \mid f(mz)$ ist definiert $\}$.

Feiner–gröber–Beziehungen zwischen Elementen des Bildbereichs eines Merkmals seien — soweit es sie gibt — jeweils durch eine partielle Ordnungsrelation „$\preceq^{\text{feat}}$“ repräsentiert.

Maßzahlen werden quasi als eine Objektmenge (vgl. die Diskussion von Kennzahldimensionen in Abschnitt 4.2.6) betrachtet, deren Eigenschaften hier Merkmale genannt werden. Die Spezialisierungsbeziehungen auf Merkmalswerten entsprechen der in Abschnitt 4.2.1 eingeführten Subsumtion von Kategorien. Diese Betrachtungsweise soll hier jedoch nicht weiter vertieft werden.

Klassifikation von Datentypen

Der Datentyp T einer Maßzahl kann — wie allgemein in der Statistik üblich, vgl. etwa [HEK91], [Han92] oder [Mar97] — anhand seiner Skalierung *scale*() klassifiziert werden.[69] Im wesentlichen wird unterschieden zwischen

- *nominalen* (ungeordneten) Skalen,
- *ordinalen* (geordneten) Skalen, die
  - *nicht–metrisch* (ohne Abstandsfunktion) oder
  - *metrisch* sein können, wobei wiederum differenziert wird zwischen *Intervallskalen* und *Absolutskalen*, die einen Bezugspunkt (ein „neutrales Element“) aufweisen, zu dem ein Absolutbetrag als Abstand angegeben werden kann.

Metrische Skalen können stetig oder diskret sein (Merkmal *discretization*()) und eine Unter- und Obergrenze aufweisen (*lower_bound*() bzw. *upper_bound*()). Für Absolutskalen soll ferner angenommen werden, daß die Menge neutraler Element T_0 einelementig ist und so den Bezugspunkt der Skala spezifiziert.

[69] Nullwerte sollen von dieser Betrachtung ausgenommen sein.

Wie in Abschnitt 4.2.5 dargestellt, können auch Domänen als Datentypen interpretiert werden. Ein Flag *is_category_domain*() kennzeichne diese Datentypen. Die oben vorgestellten Skalen sollen sich in diesen Fällen stets auf die feinsten Kategorien einer Domäne *D*, also die Menge

$$\{k \in D \setminus (\mathcal{K}^{\delta} \cup \mathcal{K}^{\emptyset}) \mid \forall k \neq k' \in D\colon (k' \preceq k \Rightarrow k' \in \mathcal{K}^{\emptyset})\}$$

beziehen (etwa 1–Jahres–Altersgruppen in der Alters- oder Tagesangaben in der Zeitdomäne).

Typische Maßzahlklassen

In der Literatur zur Statistik (etwa [HEK91]) werden viele verschiedene Klassen von Maßzahlen unterschieden, etwa Lagemaße und Streuungsmaße, unterschiedliche Teststatistiken sowie (Schätzer für) Parameter vielfältiger Analysemodelle. Die Epidemiologie im speziellen kennt wiederum — über grundlegende Größen wie Raten und Risiken hinaus — Maße aus vielfältigen spezifischen Untersuchungsgebieten, wie etwa raumbezogene Clusterindizes (zur Betrachtung von Homogenität und/oder Autokorrelation zwischen Erkrankungshäufigkeiten), Maße zur Beschreibung von Überlebenswahrscheinlichkeiten etc. All diese Klassifikationen können oftmals auch nebeneinander angewendet werden. Im folgenden werden einige besonders wichtige Merkmale zur anwendungsbezogenen, adäquaten Beschreibung der jeweils interessierenden Maßzahlklassen herausgegriffen.

Ein grundlegender Aspekt ist (insbesondere für Anzahlen) zunächst einmal die Unterscheidung von Bewegungs- und Bestandsmaßen (Merkmal *stock_flow*()) [LS97]. *Bestandsmaße* repräsentieren zeitlich veränderliche Größen zu Betrachtungselementen, denen eine bestimmte Dauer der Existenz zugeordnet werden kann. Deren Bestand wurde zu einem bestimmten Zeitpunkt gemessen (z. B. Bevölkerungszahlen). *Bewegungsmaße* dagegen messen Ereignisse bzw. Veränderungen (eines Bestandsmaßes) in einem bestimmten Zeitraum (z. B. Neuerkrankungszahlen). Diese Unterscheidung bestimmt entscheidend etwa auch die zeitliche Aggregierbarkeit von Maßzahlen, die über Bestandsmaße nicht, über Bewegungsmaße dagegen problemlos möglich ist.

Das Flag *standardized*() gebe an, ob eine Maßzahl durch *Standardisierung* ermittelt wurde, also als eine gewichtete Summen von Werten (zu Teilen der betrachteten Population) zusammengesetzt ist, wobei die Gewichtung unabhängig von den Maßzahlwerten vorgegeben ist.

Ein Großteil typischerweise betrachteter Maßzahlen repräsentiert ein Verhältnis zwischen zwei anderen Maßen (z. B. Raten oder Risiken). Verschiedene Arten von *Verhältniszahlen* sind (über das Merkmal *proportional_measure*()) zu unterscheiden (nach [HEK91]):

- *Gliederungszahlen* beziehen eine Teilgröße auf eine übergeordnete Gesamtgröße (z. B. der Raucheranteil).
- *Beziehungszahlen* sind allgemein Quotienten sachlich sinnvoll in Verbindung stehender Maßzahlen. Hierbei beziehen
  - *Verursachungszahlen* ein Bewegungsmaß auf ein zugehöriges Bestandsmaß (etwa Geburten pro Einwohner) und
  - *Entsprechungszahlen* in diesem Sinne unabhängige Maße aufeinander (etwa Krankenhausbetten pro Einwohner).
- *Indexzahlen* vergleichen zwei Maßzahlen der gleichen Art miteinander. Hierbei beschreiben bei
  - *einfachen Indizes* (*Meßzahlen*) beide Maße einen realen Sachverhalt — es werden zwei (i. a. disjunkte) Teilmengen einer umfassenden Grundgesamtheit aufeinander bezogen (z. B. die Erkrankungsrate Niedersachsens auf die Erkrankungsrate des Saarlandes), während bei
  - *zusammengesetzten Indizes* eine der beiden Maße fiktiv (etwa standardisiert) ist (wie z. B. im Falle der standardisierten Mortalitätsratio *SMR*).

Schließlich spezifiziere noch ein Parameter *is_confidence_bound*(), ob es sich bei einer Maßzahl um die (obere oder untere) Grenze eines Konfidenzintervalls zum jeweiligen Maß handelt.

Parametrisierte Maßzahlen

Weitere Merkmale parametrisieren die Berechnungsvorschriften bzw. die Quellmaßzahlen, aus denen eine Maßzahl *mz* gemäß $mz.F$ berechenbar ist. Typische Parameter charakterisieren ggf.:

- die Eigenschaften, über die aggregiert wurde, etwa falls sich diese einfach durch die Angabe bestimmter Dimensionen *aggr_dimensions*() ergeben („Durchschnitt über die Zeit“, „Standardisierung über Alter und/oder Geschlecht“) oder spezielle Sachverhalte, wie etwa die Fallart *case_type*() (Inzidenz, Mortalität, Prävalenz) darstellen,
- die Einschränkung auf bestimmte Objektmengen *objects*(), etwa bei tumor- oder personenbezogenen Erkrankungszahlen, oder die spezielle Betrachtung bestimmter Rollen *roles*(), z. B. zur Unterscheidung zwischen der direkt standardisierten und der für die Studienpopulation erwarteten Rate[70],
- die der Maßzahlberechnung zugrundeliegende Basismaßzahl *src_measure*(), z. B. bei der durchschnittlichen Fallzahl die Fallzahl,
- unterschiedliche Einheiten *unit*() und Maßstäbe *scale_factor*(), wobei Implementierungen entsprechend parametrisierter Maßzahlklassen Umrechnungsfunktionen, also Berechnungsfunktionen von einer Maßzahl in eine semantisch äquivalente, aber anders skalierte Maßzahl kapseln könnten, so daß der Benutzer entsprechende Analyseschritte nicht explizit formulieren müßte,
- Spezifika bei der Berechnung, etwa eine Umrechnung der Quellmaßzahlwerte in Ränge, wie sie einige Clusterindizes vornehmen (Merkmal *range_based*()),
- explizite Berechnungsparameter, wie etwa die Altersgrenzen zur Bestimmung kumulativer Raten (*age_min*() und *age_max*()) oder für Konfidenzintervalle das zugrundeliegende Signifikanzniveau (*confidence_level*()).

Darüber hinaus können derartige Merkmale auch genutzt werden, um statistische Eigenschaften der jeweiligen Maßzahlen (statistische Verteilungen, Robustheit etc.) zu beschreiben.

Definition einer Maßzahlhierarchie

Auf der Basis obiger Überlegungen läßt sich nun leicht eine Klassen- und Typhierarchie auf Maßzahlen einführen.

Definition 4.42 (Hierarchie auf Maßzahlen) *Eine Maßzahl $mz \in \mathcal{M}$ heiße* SPEZIELLER *als eine Maßzahl $mz' \in \mathcal{M}$ (geschrieben „$mz \preceq^{\mathrm{M}} mz'$“), falls*

- $mz.F \subseteq mz'.F$ *(es gibt also weniger Wege zur Berechnung von mz),*
- $mz.T \subseteq mz'.T$, $mz.T_0 \subseteq mz'.T_0$ *sowie* $mz.T^{\mathrm{N}} \subseteq mz'.T^{\mathrm{N}}$,[71]
- $\forall (e, or) \in \mathcal{E} \times \mathcal{OR}: mz'$ *(disjunkt oder beliebig) aggregierbar über e und or* $\Rightarrow mz.f^{\mathrm{aggr}}_{e,or} = mz'.f^{\mathrm{aggr}}_{e,or}$ *(es gibt also mehr Aggregierungsmöglichkeiten für mz) sowie*
- *alle Merkmale von mz' auch für mz definiert sind, also* $f^{\mathrm{feat}}(mz') \subseteq f^{\mathrm{feat}}(mz)$, *und es gilt* $\forall feature() \in f^{\mathrm{feat}}(mz'): feature(mz) \preceq feature(mz')$.

Somit läßt sich nun z. B. modellieren, daß rohe Inzidenzraten eine spezielle Form von Beziehungszahlen sind oder daß Fallzahlen Inzidenz- oder Mortalitätsfallzahlen sein können. Allgemein ergibt sich ein azyklischer gerichteter Spezialisierungsgraph auf Maßzahlen (der jedoch in der Regel kein „einfacher“ Baum ist).

Obige Definition geht davon aus, daß $mz.F$ jeweils wirklich alle möglichen Berechnungswege beschreibt, d. h. insbesondere, daß

[70] Beide Maßzahlen werden mit derselben Formel berechnet. Im ersten Fall werden die Raten der Studienpopulation auf die Altersstruktur der Standardpopulation standardisiert, während im zweiten Fall die Rollen gerade vertauscht sind.

[71] Wie schon bei der Diskussion der Kennzahldimensionen in Abschnitt 4.2.6 werden auch hier Nullwerte gleicher Semantik in den beiden Datentypen miteinander identifiziert.

- mit $(\cdot,\cdot,\cdot,\mathit{false},\cdot) \in F$ auch $(\cdot,\cdot,\cdot,\mathit{true},\cdot) \in F$,
- mit $(\cdot,\cdot,\{\{\mathit{eor}_1,\mathit{eor}_2,\ldots\},\ldots\},\cdot,\cdot) \in F$ jeweils auch $(\cdot,\cdot,\{\{\mathit{eor}_1,\ldots\},\{\mathit{eor}_2,\ldots\},\ldots\},\cdot,\cdot) \in F$,
- mit $(\{\mathit{mz}_1,\mathit{mz}_2,\ldots\},\{\mathit{mz}_a,\ldots\},\cdot,\cdot,\cdot) \in F$ auch $(\{\mathit{mz}'_1,\mathit{mz}'_2,\ldots\},\{\mathit{mz}'_a,\ldots\},\cdot,\cdot,\cdot) \in F$, wobei $\mathit{mz}'_1 \preceq \mathit{mz}_1,\ldots$ und $\mathit{mz}_a \preceq \mathit{mz}'_a,\ldots$, sowie daß
- mit $(\cdot,MZ,\cdot,\cdot,(f_{mz})_{mz\in MZ}) \in F$ auch $(\cdot,MZ',\cdot,\cdot,(f_{mz})_{mz\in MZ'}) \in F$ für $\emptyset \neq MZ' \subset MZ$.

Andernfalls würden entsprechende Fälle nicht als Maßzahlspezialisierung erkannt werden. In einer Implementierung von *MADEIRA* läßt sich $mz.F$ natürlich durch Beschränkung auf die jeweils allgemeineren Berechnungswege vereinfachen, wobei darüber hinaus nun auch noch die Maßzahlhierarchie bei der Spezifikation der Quellmaßzahlen ausgenutzt werden kann. Zudem ist es nicht unbedingt nötig, die transitive Hülle von Berechenbarkeiten (in der Form „ist mz_1 aus mz_2 und mz_2 aus mz_3 berechenbar, dann auch mz_1 aus mz_3“) explizit aufzulösen.

Mit dieser Beschreibung einer Maßzahlhierarchie muß schließlich noch Def. 4.15 (Spezifikation von Maßzahlen) leicht korrigiert werden: Für $(\cdot,MZ,\cdot,\cdot,\cdot) \in mz.F$ gilt nicht notwendigerweise $mz \in MZ$, sondern es ist ausreichend, wenn $\exists mz' \in MZ : mz' \preceq mz$.

Die Definition einer Maßzahlhierarchie unterscheidet nicht zwischen Klassen und Instanzen von Maßzahlen und orientiert sich somit an einer framebasierten Sichtweise. Eine allgemeinere Maßzahl kann sowohl als Verallgemeinerung einer spezielleren als auch als disjunktive Vereinigung der Definitionen mehrerer spezieller Maßzahlen angesehen werden. Hierbei werden jeweils Datentypen und Berechnungsfunktionen vereinigt, Parameter auf gemeinsame Merkmale eingeschränkt (und deren Ausprägungen ggf. verallgemeinert) sowie die Menge der Aggregierungsfunktionen auf diejenigen Eigenschaften und Objektmengen eingeschränkt, über die alle zusammengeführten Maßzahlen (natürlich mit der gleichen Funktion) aggregierbar sind. In diesem Sinne müssen also „Oberklassen“ von Maßzahlen konservativer im Hinblick auf mögliche Aggregierungen sein, da sie aufgrund der unterschiedlichen Berechnungsmöglichkeiten auch weniger spezifisches Wissen nutzen können.

4.4.3 Nutzung von Dimensionsattributen für Zusatzinformationen über Kategorien

Als spezielle Datenquellen zur Bereitstellung von Kontextinformationen für bestimmte Berechnungen sollen Domänen dienen. Bereits im vorangegangenen Abschnitt wurden Domänen als *Klassen* von Kategorien charakterisiert, die zugleich Kategorientypen definieren. In diesem Sinne kann eine Domäne ihren Kategorien Maßzahlwerte gemäß einer einheitlichen Signatur zuordnen.

Definition 4.43 (Maßzahlwerte zu Kategorien) *Die Abbildung $f^{\mathrm{mass}} : \mathcal{DO} \to 2^{\mathcal{M}}$ ordne jeder Domäne eine Menge für ihre Kategorien definierter Maßzahlen zu.*

Die Zuordnung von Maßzahlausprägungen zu Kategorien beliebiger Domänen D beschreibe die Abbildung $f^{\mathrm{mval}} : \mathcal{K} \times \mathcal{M} \to \mathcal{W}$ mit $f^{\mathrm{mval}}(k,mz) \in mz.T$ für $mz \in f^{\mathrm{mass}}(D_k)$.

Über die Berechnungsmöglichkeiten $mz.F$ der betrachteten Maßzahlen mz sei hier nichts weiter ausgesagt: Weder wird die „Berechnung“ aus Kategorien als solche modelliert, noch werden andere Berechnungswege ausgeschlossen.

Um derartige Maßzahlwerte in *MADEIRA* als Ausprägungen summarischer Attribute modellieren zu können, müssen sie aus den Eigenschaften bestimmter Objekte abzuleiten sein. Hierzu werden einfach die Kategorien selbst als Objekte interpretiert.

Definition 4.44 (Interpretation von Domänen als Objektmengen) *Die Menge aller Kategorien $\mathcal{K}$ sei als Teilmenge in die Menge aller Objekte O eingebettet. Zu einer Domäne $D \in \mathcal{DO}$ bezeichne e_D eine Eigenschaft mit $O_{e_D} = D$ und (für $k \in D$) $O_{e_D,k} = \{k' \in D \mid k' \preceq k\}$. Ferner sei e_D die jeweils einzige Eigenschaft von „Kategorie–Objekten“, also $f^{\mathrm{eig}}(k) = \{e_D\}$, falls $k \in D$.*

Hiermit können wir nun direkt aus einer Kategorienmenge (etwa eines gegebenen kategoriellen Attributs) einen eindimensionalen Datenraum erzeugen.

Definition 4.45 (Erzeugung von Datenräumen aus Kategorienmengen) *Die Berechnung von Datenräumen aus Kategorienmengen sei definiert durch eine Abbildung* $f^{\text{cube}} : 2^{\mathcal{K}} \times \mathcal{M} \times \mathcal{R} \to \mathcal{DR}$.

Sei $D \in \mathcal{DO}$ *eine Domäne,* $K \subseteq D$ *eine endliche, nicht–leere Teilmenge von* D, $mz \in f^{\text{mass}}(D)$ *sowie* $r \in \mathcal{R}$. *Dann sei* $f^{\text{cube}}(K, mz, r) = dr \stackrel{\text{def}}{=} (OR, KA, SA, f^{\text{ext}})$ *definiert, wobei*

- $OR \stackrel{\text{def}}{=} \{(D, r)\}$ *mit* $\text{ext}(D, r) \stackrel{\text{def}}{=} K$,
- $KA \stackrel{\text{def}}{=} \{(D, K, \{(e_D, D, r\})\}$,
- $SA \stackrel{\text{def}}{=} \{(mz, \{(D, r)\}, f^{\text{sum}}, mz.f^{\text{aggr}})\}$ *mit* $f^{\text{sum}} : 2^D \to mz.T$ *und* $f^{\text{sum}}(K') \stackrel{\text{def}}{=} f^{\text{mval}}(\bigvee K')$ *für* $K' \subseteq D$ — *hieraus ergibt sich dann auch die Extension von dr*.

Implizit werden derartige Maßzahlwerte bereits bei der (kategorienbezogenen) Aggregierung auf Maßzahlen bzw. der Ableitung von Datenräumen (vgl. Def. 4.14 und 4.31) genutzt, da dort Kategorien mit ihren zugeordneten Maßzahlen jeweils mit in die Berechnung einbezogen werden (können). Explizite Anwendung finden kategorienbezogene Maßzahlen etwa bei der Berechnung kumulativer Raten, die über die Länge der betrachteten einzelnen Altersgruppen definiert sind, bei der Durchschnittsbildung über die Zeit unter Berücksichtigung der Länge der jeweiligen Zeitintervalle oder bei der Verknüpfung von Erkrankungsraten mit raumbezogenen Hintergrundinformationen. Außerdem wird ggf. auch bei der Konsolidierung von Mikrodaten (Def. 4.29) zur Berechnung von Maßzahlwerten aus Attributen bzw. Kategorien auf derartige Angaben zurückgegriffen.

Die Zuordnung von Maßzahlwerten zu Kategorien entspricht etwa der Diskussion von Dimensionsattributen oder Features in anderen Datenmodellen (siehe etwa [LAW98]). Diese Werte sollen jedoch hier nicht dazu genutzt werden, die betreffenden Objekte weiter zu klassifizieren — andernfalls hätten sie in die Kategorienhierarchien mit einfließen müssen —, sondern dienen allein zur Berechnung weiterer Maßzahlen. Auch werden den Kategorien auf unterschiedlichen Aggregierungsebenen einer Domäne nicht — einer mit der Subsumtion verträglichen Typhierarchie auf Kategorien entsprechend — unterschiedliche „Dimensionsattribute“ zugeordnet. Gegebenenfalls können statt dessen für bestimmte (gröbere) Kategorien Nullwerte (etwa t_{nexist}) zur Kennzeichnung nicht existenter Maßzahlausprägungen genutzt werden.

Würde man obige Interpretation von Kategorien als Objekte erweitern und ihnen wiederum Eigenschaften über anderen Domänen zuordnen, ließe sich die Generierung von Datenräumen aus Kategorienmengen — ähnlich der Konsolidierung aus Mikrodaten in Def. 4.29 — leicht auf beliebig–dimensionale Makrodatenräume ausdehnen (etwa für zeitvariante raumbezogene Hintergrunddaten). Auf eine spezielle Betrachtung verzichten wir hier jedoch; ggf. können auch mit den gegebenen Mitteln entsprechende Mikrodatenräume modelliert werden.

Viele der in den letzten Jahren vorgeschlagenen Datenmodelle (siehe Abschnitt 2.3.3) versuchen, kategorielle und summarische Attribute einheitlich zu modellieren bzw. ermöglichen die gegenseitige Umwandlung. Folgende Konzepte von *MADEIRA* verflechten diese beiden Attributarten:

- die Erzeugung von Maßzahlwerten bzw. Datenräumen aus Kategorienmengen durch die Funktionen f^{mval} und f^{cube},
- die flexible Nutzung von Mikrodatenattributen bei der Konsolidierung von Makrodaten (Def. 4.29) zur Definition summarischer oder kategorieller Attribute sowie
- die Definition von Kennzahldimensionen in homogenisierten Sichten auf Datenräume, die summarische Attribute als Kategorien interpretieren.

Weiterhin wird dem Nutzer von *VIOLA* über Operatoren zur Parameterdefinition (vgl. Abschnitt 5.2.4) die Möglichkeit eröffnet werden, Maßzahlwerte eindimensionaler Datenräume zu klassifizieren und dadurch eine Gruppierung der entsprechenden Kategorien zu definieren. Diese kann dann wiederum zur Abfrage und Berechnung weiterer Datenräume dienen.

Darüber hinaus wird in *MADEIRA* die Trennung zwischen summarischen und kategoriellen Attributen, die sich aus einer klaren multidimensionalen Modellierung bei der Definition und Generierung von Makrodaten auf der Basis zugrundeliegender Mikrodaten ergibt, aufrechterhalten.

4.4.4 Metadaten in *MADEIRA* — eine Einordnung

Im Sinne der in in Abschnitt 2.4 vorgestellten Klassifikationen bietet *MADEIRA* eine breite Palette von Metadaten, die den Datenanalyseprozeß umfassend unterstützen:

- Es sind sowohl führende Metadaten für den Anwender von *VIOLA* (Texte, Attributbeschreibungen) als auch kontrollierende Metadaten vorgesehen, die die automatisierte Datenverarbeitung im Rahmen einer Analyse unterstützen (insbesondere die korrekte Datenaggregierung über Kategorienhierarchien und Aggregierungsfunktionen).
- Als zentrale Aufgabengebereiche sind das Suchen, Browsen und vor allem Abfragen sowie Aspekte der Qualitätssicherung (der exakten Datendefinition) zu nennen.
- *MADEIRA* nutzt sowohl elementare als auch zusammengesetzte Metadaten auf verschiedenen Granularitätsebenen — von den Kategorien bis zum Datenraum.
- Alle gängigen Datentypen treten auf, Funktionen spielen eine besondere Rolle.
- In erster Linie werden die Metadaten intern genutzt, sie könnten jedoch auch als Basis für eine externe Schnittstelle von *VIOLA* zu anderen Analysesystemen dienen (vgl. Abschnitt 6.1.3).
- Metadaten werden explizit modelliert. Wenn auch die Verwaltung von Metadaten bisher nicht näher untersucht wurde, so erscheint doch eine Separierung von den Daten auf der Ebene von Datenbasen bzw. Datenräumen in einem integrierten Datenbanksystem, das der engen Verzahnung von Daten und Metadaten Rechnung trägt, sinnvoll.[72]

MADEIRA zielt vor allem auf die eigentliche Analysephase im Datenanalyseprozeß und bietet somit zwangsläufig auch Unterstützung bei der anschließenden Dateninterpretation. Zurückgegriffen wird auf Metadaten aus allen Phasen des in Abb. 2.21 dargestellten Prozesses, wobei Wissen über die Konzeption der Datenerhebung auf textuelle Beschreibungen beschränkt ist. Ziel von *VIOLA* muß es nun sein, das mit *MADEIRA* modellierte Wissen geeignet zu nutzen und dem Datenanalysten zur Verfügung zu stellen.

4.5 Zusammenfassung und Fazit

In den letzten Jahren wurden in der Literatur eine ganze Reihe von Modellen für multidimensionale Daten vorgeschlagen. Die meisten dieser Darstellungen orientieren sich jedoch — ausgerichtet auf den konkreten Datenbank–*Entwurf* bzw. die informative Beschreibung existierender Schemata — stark an einer relationalen Implementierung von Makrodaten in Form von *Sternschemata*. Sehr mächtige Operatoren, die auch die Behandlung von Mikrodaten einbeziehen, zielen insbesondere auf um statistische Funktionalität erweiterte Datenbankanfragen an sehr große betriebswirtschaftliche Datenbestände.

Demgegenüber liegt der Fokus des in diesem Kapitel vorgestellten Datenmodells *MADEIRA* darauf, eine formale Grundlage für eine intuitiv bedienbare, sehr nah am multidimensionalen Modell ausgerichtete Analyseumgebung zu schaffen, in der Funktionalität, die über einheitliche, navigierende Basisfunktionen hinausgeht, durch eigenständige Operatoren außerhalb des DBMS realisiert wird. Durch die flexible und semantisch umfassende Definition von Maßzahlen werden diese Funktionen in das Datenmodell und die Datenverarbeitung eingebunden. Ein weiterer Schwerpunkt von *MADEIRA* ist die Definition und Nutzung von Kategorien und

[72] Aufgrund der recht heterogenen Daten- und Metadatenarten wäre hier eine objektrelationale oder objektorientierte Umsetzung in Erwägung zu ziehen.

Kategorienhierarchien als unentbehrliche Grundlage einer strukturierten und semantisch eindeutigen Navigation durch die betrachteten Datenräume. Auch wird — wiederum metadatenbasiert — ein expliziter Bezug zu den jeweils modellierten Anwendungsobjekten hergestellt.

In [LAW98] stellt Lehner als wesentliches Ziel einer „Normalform“ multidimensionaler Daten, also in gewissem Sinne „wohlgeformter“ Datenwürfel, die Sicherstellung der korrekten Aggregierung von Makrodaten sowie die Vermeidung dünn besetzter Datenräume heraus. Der erstgenannte Aspekt wird in *MADEIRA* durch die klar definierte Kategoriensemantik in Zusammenhang mit (nach Eigenschaften und Objektmengen) differenzierten Aggregierungsfunktionen umfassend behandelt, wodurch per se ein „gutes“ Schema–Design erzwungen wird. Auch unterstützt die Beschreibung von Maßzahlen und Kategorien durch flexible Datentypen die sinnvolle und teilautomatisierte Anwendung von Analyseverfahren.

Wenn auch die Verarbeitung dünn besetzter Datenräume in *MADEIRA* keine zentrale Rolle spielt, ermöglichen doch zusammengesetzte kategorielle Attribute sowie auch die flexible Definition von Aggregierungsebenen (ähnlich der Nutzung von Features in [LAW98]) und die Kombination beliebig granularer Kategorien in *einem* kategoriellen Attribut die Behandlung entsprechender Anwendungssituationen. Speziell zur Modellierung betriebswirtschaftlicher Domänen mit heterogenen Produktpaletten schlägt Lehner die Einführung von Dimensionsattributen vor, die dort zur Vermeidung leerer Zellen von Datenräumen beitragen. Diese werden von *MADEIRA* — auch im Hinblick auf die relativ einfachen Kategorienhierarchien im Anwendungsgebiet der Epidemiologie — aufgrund der intuitiveren Handhabbarkeit eines einfacheren Klassifikationsmodells nicht unterstützt. Aus ähnlichen Gründen, insbesondere auch zugunsten einer klaren Trennung kategorieller und summarischer Attribute, wird auf Fakten in Form komplexer Records und über nicht–numerischen Datentypen (vgl. [BSHD98]) verzichtet.

Ähnlich wie in [MERS92] kann auch hier die Auswahl der Operatoren bzw. die Vollständigkeit der durch *MADEIRA* definierten Algebra auf Makrodaten durch Gegenüberstellung zu den Basisoperatoren der relationalen Algebra (und einer relationalen Repräsentation multidimensionaler Daten) motiviert werden:

- Die Restriktion in *MADEIRA* entspricht der Selektion (auf kategoriellen Attributen) im relationalen Modell. Eine entsprechende (wertebasierte) Einschränkung auf summarischen Attributen ist nicht vorgesehen, da das Ergebnis im allgemeinen Fall kein wohldefinierter, vollständiger multidimensionaler Datenraum mehr wäre.
- Die Selektion summarischer Attribute und die vollständige Aggregierung (zu einem Gesamtwert pro Dimension) bilden das Pendant zur Projektion (auf summarischen bzw. kategoriellen Attributen), wobei der Aggregierung mit ihrer Wertezusammenfassung eine gegenüber der relationalen Projektion andere (erweiterte) Datensemantik zugrunde liegt.
- Die beliebige (nicht vollständige) Aggregierung ist mit der Gruppierung in Implementierungen des relationalen Modells zu vergleichen.
- Die Vereinigung in *MADEIRA* kombiniert Vereinigung und Equi–Join (über gemeinsame Attribute) im relationalen Modell.
- Der beliebige Verbund ist in der Verarbeitung multidimensionaler Daten nicht klar definiert. Ebenso sind Differenz und Division nicht sinnvoll definierbar. Die Schnittbildung spielt in multidimensionalen Analysen kaum eine Rolle. Sie kann weiterhin auch durch geeignete Restriktionen realisiert werden und findet somit in *MADEIRA* keine direkte Entsprechung.
- Maßzahlberechnung und auch die Umbenennung von Rollen erweitern das relationale Modell wesentlich um Funktionalitäten zur Datenanalyse.
- Quasi als eine Brücke zum relationalen Modell behandeln Split und Merge zusammengesetzte kategorielle Attribute.
- Während alle anderen Funktionen allein auf Makrodaten operieren und auch (in einer abgeschlossenen Algebra) Makrodatenwürfel erzeugen, definieren Konsolidierung und Mikrodatenabfrage die Umwandlung von Mikro- in Makrodaten und umgekehrt.

Jede Ableitungsoperation auf einem Datenraum mit einem einzelnen summarischen Attribut kann für sich auch durch Anwendung des Operators zur expliziten Aggregierung und Vorgabe der „passenden“ Aggregierungsfunktion realisiert werden. Die Generizität eines Ableitungsoperators, der — ohne spezielle Parametrisierung — *beliebige* Datenräume vergröbert, wird bei diesem Vorgehen jedoch aufgegeben.

Weiterhin ist es möglich, die Selektion summarischer Attribute zu einer bestimmten Maßzahl *mz* durch die zellenweise Maßzahlberechnung zu simulieren, wenn *mz* gemäß $mz.F$ unter Anwendung der Identitätsfunktion aus sich selbst berechenbar ist. Diese Option ist jedoch auf Fälle beschränkt, in denen alle zu selektierenden Attribute die gleiche Maßzahl aufweisen und im konkret betrachteten Datenraum nur ein passendes Attribut enthalten ist. Außerdem berechnet die Analyseoperation unter Umständen auch dann die vorgegebene Maßzahl, wenn der Quelldatenraum gar kein entsprechendes summarisches Attribut enthält, aber eine Berechnung aus anderen Attributen gemäß $mz.F$ möglich ist. Wiederum definieren die beiden Operatoren zwei verschiedene Betrachtungsweisen, die nur in einigen Spezialfällen zum gleichen Ergebnis führen.

Darüber hinaus ist die vorgestellte Algebra — abgesehen von der zellenweisen Maßzahlberechnung, die auch durch die freie Maßzahlverknüpfung ausgedrückt werden kann — minimal.

Durch die Integration verschiedener beschreibender und klassifizierender Metadaten schlägt *MADEIRA* eine Brücke zwischen logischer und konzeptioneller Datenmodellierung. So wie im betriebswirtschaftlichen Umfeld die Bedeutung der Berücksichtigung von *Business Terms* hervorgehoben wird [MSR99, JT98], so werden auch hier enge Beziehungen zu den Objekten, Rollen und Eigenschaften des jeweils betrachteten Weltausschnitts hergestellt. Im Sinne von [Ber97] definiert *MADEIRA* ein Informationsmodell für multidimensionale Daten, das die zugrundeliegenden Datenbestände um semantische Informationen anreichert, die zu ihrer sinnvollen Nutzung nötig sind. Das Datenmodell zielt hierbei speziell auf die Unterstützung der explorativen Datenanalyse und bietet eine tiefgehende Integration der genutzten Metadaten in die Operatoren seiner Algebra. Hierdurch ist *MADEIRA* klar von den in Abschnitt 2.4 angeführten allgemeinen Ansätzen zur Modellierung von Metadaten (im Data Warehousing) abzugrenzen.

Brodie nennt in [Bro84] (vgl. auch [Heu97]) elf allgemeine Anforderungen an „gute“ Datenmodelle. Hierunter konzentriert sich *MADEIRA* mit der exakten und differenzierten Sprachdefinition und der Abstimmung auf die Nutzung durch High–Level–Werkzeuge (nämlich *VIOLA*) auf zwei Aspekte, die von vielen vergleichbaren Datenmodellen vernachlässigt werden. Auch in der Integration von Struktur und Verhalten geht *MADEIRA* mit der Einbindung von Aspekten einer operationalen Semantik über das Beschreibungsniveau vieler anderer Modelle hinaus. Ebenfalls eine wesentliche Rolle spielen die direkte Modellierbarkeit von Anwendungsobjekten (multidimensionalen Datenräumen), die inhärente „semantische Relativität“ des recht universellen Konstrukts des multidimensionalen Feldes, d. h. die Unterstützung unterschiedlicher Sichtweisen auf die Daten (etwa in Anlehnung an ROLAP bzw. MOLAP), und in gewissem Sinne (wie es auch Brodie versteht) die Nutzung von Konzepten der KI durch die metadatenbasierte Beschreibung von *Wissen* über Datenräume.

Kriterien wie Verständlichkeit, logische und physische Datenunabhängigkeit sowie Implementierbarkeit stehen in den Betrachtungen dieser Arbeit weniger im Vordergrund, werden aber auch nicht vernachlässigt: So dient *VIOLA* der verständlichen Präsentation von *MADEIRA*–Strukturen über eine geeignete Benutzungsschnittstelle zur Datenanalyse; gerade die umfassende Nutzung von Metadaten bietet eine anwendungsunabhängige Datenbeschreibung, und Strukturen und Operatoren sind auf der Basis von Abb. 4.1[73] direkt in einer hauptspeicherbasierten Implementierung für die Verarbeitung mittelgroßer Datenbestände umzusetzen. Schließlich seien noch die Aspekte des Abstraktionsvermögens und der Sprachmächtigkeit genannt, die bedingt durch die Fokussierung auf das Gebiet der multidimensionalen Datenanalyse für *MADEIRA* ausreichend, aber im Vergleich zu anderen Modellen sicher nur eingeschränkt berücksichtigt sind.

[73] Über die dargestellten Beziehungen hinaus sind auch noch die Spezialisierungen von Objekten durch Kategorien (im Rahmen der Einführung von Dimensionsattributen, Def. 4.44), von Objekten durch Maßzahlen (im Rahmen der Diskussion von Merkmalen auf Maßzahlen, Def. 4.41) sowie von Kategorien und Objekten durch summarische Attribute (im Rahmen der Betrachtung von Kennzahldimensionen, Def. 4.24) einzubeziehen.

Kapitel 5

VIOLA: Datenflußbasierte Modellierung multidimensionaler Datenanalysen

Wie bereits zum Abschluß von Kapitel 3 dargestellt, besteht das Ziel dieser Arbeit in der Konzeption einer visuellen Programmierumgebung, die Konzepte und Stärken von Datenflußsprachen, Systemen zur interaktiven Datenvisualisierung und OLAP–Tools kombiniert, um eine „intelligente“ Analyse multidimensionaler Daten zu ermöglichen. Zudem soll ein flexibles Interaktionsmodell unterstützt werden, das dem Benutzer große Freiheiten bei der Datenexploration gewährt. Als Akronym sowohl für die im folgenden entwickelte datenflußbasierte Analysesprache, die zunächst einmal unabhängig von ihrer visuellen Umsetzung entworfen wird[1], als auch für die entsprechende Programmierumgebung soll *VIOLA* (*VIsual On–Line data Analysis environment*) dienen. *VIOLA* führt den Entwurf von *MADEIRA* fort, indem die Operatoren der dort aufgestellten Algebra in Bausteine von Datenflußprogrammen überführt werden. Ergänzt werden diese durch Kontrollstrukturen, Methoden zur Parameterdefinition sowie Visualisierungsverfahren. Während *MADEIRA* also Gestalt und Bedeutung einzelner Datenräume definiert, beschreibt *VIOLA* in Gestalt von Datenflußprogrammen die Art und Weise ihrer Herleitung und bietet eine Umgebung zur Nutzung des Datenmodells. Hiermit entspricht *VIOLA* genau den Ausführungen zur Gestaltung „guter“ Datenanalysesprachen in [Hub94], wo die Unzertrennbarkeit von Datenanalyse–Ergebnissen und Pfad ihrer Herleitung sowie eine Datenanalyse–Sprache für Durchführung *und* Beschreibung von Auswertungen propagiert werden (vgl. Abschnitt 2.2.4). Ganz im Sinne *intelligenter* Datenanalyse (vgl. Kapitel 1 sowie Abschnitt 2.1.4) ergänzen sich *MADEIRA* und *VIOLA* in der Bereitstellung einer umfassenden „Kommunikations-, Informations- und Definitionsplattform“ zwischen Rechnersystem und Datenanalyst.

VIOLA zielt auf die explorative Analyse „mittelgroßer“ Datenbestände, d. h. Zwischen- und Endergebnisse von Analysen können durch multidimensionale Datenräume repräsentiert werden, die hauptspeicherresident gehalten werden können. Nur unter dieser Voraussetzung ist tatsächlich eine interaktive, schrittweise Auswertung und Visualisierung sinnvoll möglich (vgl. [Lee94]).[2] Datenräume können als ganzes im Hauptspeicher durch Analyseverfahren verarbeitet werden, wobei Zwischenergebnisse aus einzelnen Bausteinen eines Datenflußprogramms ggf. temporär auf Sekundärspeicher ausgelagert werden können. Als typisches Anwendungsgebiet kann das Auswertungsspektrum eines epidemiologischen Krebsregisters dienen: angefangen von Ad–hoc–Anfragen über Analysesequenzen im Rahmen eines Inzidenzmonitoring bis zur standardisierten Erstellung von Jahresberichten steht jeweils die flexible und großteils auch interaktive Konfigurierbarkeit von Datenanalysen im Vordergrund. Aber auch auf nahezu beliebige andere Anwendungsgebiete mit ähnlichen Datenvolumina ist das vorgeschlagene Analysemodell, das auf allgemein gehaltenen Elementaroperationen ba-

[1] Somit kann *VIOLA* auch als Grundlage einer Internrepräsentation von Datenanalysen bzw. einer verbalen Skriptsprache genutzt werden.

[2] In Abschnitt 6.2 wird noch auf einige Ansätze, durch Anfrage- und Verarbeitungsoptimierungen auch größere Datenbestände interaktiv verarbeitbar zu machen, eingegangen werden.

siert, problemlos übertragbar, solange Verarbeitungsoperationen ausschließlich auf Makrodaten durchzuführen sind.

5.1 Charakterisierung und Einordnung von *VIOLA*

Zur genaueren Einordnung von *VIOLA* soll eine Klassifikation der wichtigsten in Abschnitt 3.4 vorgestellten datenflußbasierten Analyseumgebungen anhand verschiedener zentraler Entwurfs- und Implementierungsaspekte dienen (siehe Tab. 5.1).

Neben der bereits in Abschnitt 3.4 angeführten Literatur konnte zur Einstufung von *AVS*, *IBM Data Explorer*, *KHOROS*/*Cantata*, *LabVIEW*, SIMULINK und *ViSta* auf jeweilige Evaluations- oder Vollversionen zugegriffen werden. *Clementine* konnte nur anhand der Darstellung in [WK97] eingeordnet werden, womit sich einige Fragezeichen in der tabellarischen Darstellung erklären. Zum *IRIS Data Explorer* waren die vorliegenden Informationen so unvollständig, daß auf eine Aufnahme in die Übersicht verzichtet wurde, zumal das System auch viele Parallelen zum *IBM Data Explorer* aufweist. Im Fall von *ViSta* bezieht sich die Darstellung auf die Workmaps.

Die Auswahl der Evaluationskriterien orientiert sich in wesentlichen Zügen an [WRH92, Hil93]. Im Hinblick auf Schwerpunkte bei der Konzeption von *VIOLA* wurden die dort vorgeschlagenen Kataloge noch um einzelne Aspekte ergänzt. Über Interna der Programmausführung (vgl. [WRH92]), wie Realisierung und Binden von Analysefunktionen, Übersetzung bzw. Interpretation, Interndarstellungen der Programme, Scheduler, Datenaustausch zwischen Programmbausteinen, externe Schnittstellen, Interoperabilität oder Parallelisierung der Ausführung (wie sie gerade in kommerziellen Systemen wie *AVS*, *KHOROS* und *IBM Data Explorer* eine große Rolle spielt) lieferte die vorliegende Literatur im Querschnitt der betrachteten Systeme nur recht wenige aussagekräftige Angaben. Die diesbezüglichen Kriterien müssen somit recht grob bleiben, stehen aber auch nicht im Vordergrund der Konzeption von *VIOLA*. Im einzelnen werden folgende Bereiche betrachtet (unter Angabe der jeweils verwendeten Einstufungen):

Anwendungsgebiet Auf welche Anwendungsdomäne zielt das System in seiner Grundanlage? Welche Art von Daten werden als zentrale Datenstrukturen verarbeitet? Wir unterscheiden

- Anfragen an (objekt-)relationale Datenbanken, wobei neben (um komplexe Objekte erweiterten) Relationen noch skalare Werte verarbeitet werden (D bzw. D^+),
- statistische Analysen (S) auf verschiedenen numerischen Datentypen und einfachen Records und Arrays,
- Analysen von Meß- oder simulierten Daten auf einheitlich numerischen Werten (M) oder einem differenzierten Typsystem (M^+),
- wissenschaftliche Visualisierung (V), basierend auf mehrdimensionalen Feldern (Gittern) über einfachen und komplexen Typen,
- Bild- und Signalverarbeitung (B) sowie
- Analyse multidimensionaler Datenwürfel im Sinne von OLAP (O).

Granularität Wie hoch ist die Granularität typischer Basisoperatoren? Entsprechen Sprachbausteine in Turing–mächtigen Umgebungen einzelnen Instruktionen verbaler Sprachen ($--$), sind es (sehr) einfache ($-$, $\circ$) oder überwiegende komplexere ($+$) Funktionen? Hiermit in engem Zusammenhang steht auch die Komplexität typischer Anwendungen (gemessen an der Anzahl nötiger Netzknoten).

Komposition Können Teilnetze visuell (durch einen Rahmen, mittels Darstellung in einem eigenen Fenster oder über „aufklappbare“ Icons) zu Prozeduren, die global oder innerhalb des jeweiligen Programmblockes als eigenständige Bausteine nutzbar sind, zusammengefaßt und somit Programme übersichtlich

Systeme	Anwendungsgebiet	Granularität	Komposition	Ein-/Ausgänge	Parametrisierung	Interaktive *Vis.*	Verzweigungen	Iteration	Globale Var.	Kommentierung	Sprechende *Icons*	Programmierung	Ausführung	Interne Verarb.	Typprüfung	Caching	Animation	Prg.–Generierung
AVS	V	–	++[a]	n/n	–	++	∘[c]	∘[c,e]	√	∘[g]	∘	ASK	Aa	V	++	∘	√	√
IBM Expl.	V	∘	+[a]	n/n	–	++	+[d]	∘[f]	–	+[g]	–	AS[h]	Aab	S	+	++	–	–
KHOROS	B	+	++[a]	n/n	++	+	+[d]	+[e]	√	∘[g]	–	–[h]	Da-dAc[i]	V	∘	∘	√	√
Tioga–2	D[+]	∘	++[a]	n/1	+	∘	+	–	–	–	–	V	Aa	G	++	∘[j]	–	–
DFQL	D	∘	–	n/1	+	–	–	–	–	–	+	–	Db	S	–	∘[j]	–	–
IDEA	D	∘	–	1/1	∘	–	–	–	–	–	∘	–	Db	S	–	+[j]	–	–
LabVIEW	M[+]	– –	+	n/n	–	–	+	++[e]	√	+	++	–	Db-d	C	++	–	√	–
SIMULINK	M	–	+	n/1	+	–	∘[d]	+[f]	√	+[g]	++	–	Dbd	V	–	–	–	–
ViSta	S	+	–	1/1	∘	∘	–	–	–	–[g]	∘	SK	Da	S	+	∘	–	–
Clementine	S	+	++	1/1	+	∘	–	–	–	–	+	–	Db	V	?	?	?	–[k]
VIOLA	O	∘	+[a]	1/1[b]	++	++	+	++	–	+[g]	+	A	DAa-c	G	++	++[j]	√	–

[a] Als weitere Möglichkeit zur Definition und Integration neuer benutzerdefinierter Funktionsbausteine werden geeignete Programmierschnittstellen oder sogar in angegliederten Programmierumgebungen Schablonen und/oder einfache verbale Programmiersprachen geboten. Unter diesem Aspekt soll nicht der Aufruf beliebiger externer Programme aus dem visuellen Programm über ein universelles Call–Interface verstanden werden, da hierdurch kein eigentlich neuer Baustein definiert wird.

[b] Knoten mit mehreren (jeweils gleich vielen) Ein- und Ausgängen ermöglichen in *VIOLA* die mehrfache Anwendung eines Verfahrens auf verschiedene Datenströme und deren gemeinsame Parametrisierung.

[c] Komplexe Kontrollstrukturen lassen sich in *AVS* außerhalb des Datenflußmodells über Bausteinparameter und durch geeignete Objektmethoden, die extern oder in der mächtigen verbalen Sprache *V* programmiert sind, realisieren.

[d] In diesen Systemen gibt es einen Baustein zur Selektion und Weiterleitung (Routing) eines oder mehrerer eingehender Datenströme.

[e] Weiterhin gibt es eine Möglichkeit, die Reihenfolge der Knotenausführung unabhängig vom Datenfluß zu steuern.

[f] Ein „Goto“–Operator erlaubt beliebige Sprünge, d. h. es erfolgt eine Datenpropagierung zu nicht visuell verbundenen Bausteinen.

[g] Zusätzlich leistet ein Hilfesystem baustein-, port- und parameterspezifische Hilfe.

[h] Der *IBM Data Explorer* bietet zusätzlich eine Programmierschnittstelle, über die die Nutzung der Funktionsbausteine ohne ihre datenflußbasierte Ausführung möglich ist. In *KHOROS* sind alle Funktionen ohnehin eigenständige Programme.

[i] Darüber hinaus werden verschiedene Ausführungsmodi von Ausgabe- (Visualisierungs-)knoten unterschieden, je nachdem, ob bei mehrfacher Ausführung bzw. Iterationen alte Ausgabefenster ersetzt, ihre Inhalte aktualisiert oder neue zusätzliche Fenster erzeugt werden.

[j] Aufgrund von Anfrageoptimierungen bzw. interner Zusammenfassungen von Operationen werden überhaupt nur ausgewählte Zwischenergebnisse berechnet.

[k] Gelernte Modelle, also Klassifikationen, Regeln etc., können als *C*–Code exportiert werden.

Tabelle 5.1: Gegenüberstellung datenflußbasierter Datenanalyse–Umgebungen

strukturiert werden (+)?[3] Wird die funktionale Erweiterung der Bausteinbibliothek zudem durch die Parametrisierung von Modulen mittels einfacher textueller Ausdrücke unterstützt (++)?

Ein-/Ausgänge Wieviele Ein- und Ausgänge, Funktionsargumenten und -ergebnissen entsprechend, haben typische Verarbeitungsbausteine (ein oder mehrere, die evtl. teilweise auch unbelegt bleiben und als Eingänge mit Defaultwerten versehen werden können)? Von der Betrachtung ausgenommen seien Kontrollstrukturen, Datenquellen und -senken. In allen betrachteten Systemen kann darüber hinaus eine Knotenausgabe an mehrere andere Knoten weitergeleitet werden; selten ist auch die Zusammenführung mehrerer Datenströme (etwa in einem Vektor) über einen Eingang möglich.

Parametrisierung Sind — gerade bei feingranularen Sprachen — „Parameter“ visueller Programme stets als eigene Netzbausteine (mit zugeordneten einzelnen Dialogkomponenten) realisiert, wobei ggf. für freie Knoteneingänge Default–Parameter angenommen werden (−), oder können Netzbausteine in einem glo-

[3] Bausteinparameter werden — falls vorgesehen — hierbei von den Teilkomponenten geerbt, können in der Regel aber nur direkt über diese manipuliert werden.

balen oder lokalen Parameterfenster über zusätzliche textuelle, nicht datenflußbasiert eingebundene Angaben interaktiv parametrisiert werden? Ist in diesem Fall die Parametrisierung nur initial möglich ($\circ$) oder auch nachträglich modifizierbar ($+$)? Sind auch konkrete Verfahren Parameter eines Netzknotens, der eine Verfahrensklasse repräsentiert ($++$)?

Interaktive Visualisierung Dienen Visualisierungen nur zur statischen Anzeige der Berechnungsergebnisse in Datensenken ($-$), sind hierauf lokale, also datenflußunabhängige Interaktionen (Rotation, Selektion, Zooming, Einfärbung, Einschränkung der Darstellung, bei *ViSta* auch Linking) möglich ($\circ$), kann auch eine (eingeschränkte) Propagierung von (ggf. parametrisierten) Selektionen aus Programmbausteinen an andere Netzknoten erfolgen ($+$), oder können sogar Parameter (oft Rotationswinkel) zu interaktiven Steuerung beliebiger anderer Visualisierungen propagiert werden ($++$)?

Verzweigungen Wie werden bedingte Verzweigungen (*If–then–else* oder *Case*) unterstützt: gar nicht ($-$), durch Aktivierung bzw. Deaktivierung von Knoten ($\circ$) oder durch entsprechende Bausteine ($+$)? (In letzterem Fall sind in der Regel auch Netzbausteine vorgesehen, die im Sinne eines *Merge* die Alternativzweige wieder zusammenführen.)

Iteration Wie erfolgt die Ablaufsteuerung, insbesondere in Schleifen: Sind überhaupt keine vom reinen Datenflußmodell abweichenden Kontrollstrukturen vorgesehen ($-$), gibt es Zähler (*For*–Schleifen) zum Durchlaufen einer Sequenz ganzer Zahlen oder von Listen- oder Feldelementen ($\circ$), oder sind beliebige Schleifen durch einen Zyklus im Datenflußprogramm ($+$) oder über gekapselte Teilnetze im Sinne einer strukturierten Programmierung ($++$) realisiert?[4]

Globale Variablen Werden globale Variablen unterstützt, also Datenspeicher, die nicht direkt nach Belegung, sondern beliebig später (an anderer Stelle im Datenflußprogramm) wieder abgefragt werden können ($\surd$ oder $-$)?

Kommentierung Sind verbale Kommentare gar nicht vorgesehen ($-$), „versteckt“ in eigenen Symbolen ($\circ$) oder (nahezu) beliebig im visuellen Programm plazierbar ($+$)?

„Sprechende“ Icons Ist die Funktionalität eines Bausteins nur an einer verbalen Beschriftung ($-$) erkennbar, haben einige wenige ($\circ$) oder mehrere ($+$) Klassen von Verfahren eigene Icons, oder ist ein Verfahren allein an seinem Symbol identifizierbar ($++$), wobei dieses oft auch vom Benutzer umdefiniert werden kann?

Programmierung Gibt es neben dem visuellen Editor andere Möglichkeiten der Programmspezifikation, etwa mittels einer Programmierschnittstelle (eines API) für externe Anwendungen (A), einer Skriptsprache (S), einer Kommandozeile (K), oder kann sogar über die Manipulation von Visualisierungen das unterliegende Datenflußprogramm generiert werden (V)?

Ausführung Erfolgt die Programmausführung daten- oder anforderungs-, also durch zu erstellende Visualisierungen, getrieben (D bzw. A)? Wie „lebendig“ (vgl. [Tan90]) sind die visuellen Sprachen: Werden Programmbausteine (bei vorliegender Verbindung mit Datenquellen oder -senken) sofort nach Erzeugung bzw. (Neu–)Parametrisierung (a), erst auf Benutzerbefehl zur Bearbeitung des gesamten Programms (b) oder zur (schrittweisen) Verarbeitung eines speziellen Bausteins und ggf. seiner Vorgänger (c) oder aber (in Animationen unter Erzeugung eines kontinuierlichen Datenstroms) ständig bis zum Programmabbruch (d) ausgeführt?

Interne Verarbeitung Auf welcher Interndarstellung werden Daten verarbeitet: Verwalten die visuellen Programm–Bausteine direkt die ausführbaren Prozeduren bzw. Referenzen darauf (V), wird das Datenflußprogramm in eine unterliegende verbale Skript- oder Anfragesprache übersetzt und interpretiert

[4] Rekursion ist in keinem der hier vorgestellten Systeme möglich — nicht zuletzt, weil eine visuelle Darstellung dieses Konzepts sehr schwierig ist.

(S), wird es zur Laufzeit compiliert (C) oder in einen Analysegraphen umgesetzt, der vor der Ausführung noch umgeformt und optimiert werden kann (G)?

Typprüfung Erfolgt eine Typprüfung auf Basis eines explizit modellierten, oft hierarchischen Typsystems (wobei häufig auch polymorphe Funktionen realisiert werden können)? Gar nicht ($-$), vor der Programmausführung ($\circ$), bereits beim Verbinden von Bausteinen ($+$), unterstützt durch typbezogene Kanteneinfärbungen oder spezifische Icons ($++$)?

Caching Wird ein Caching von den Netzbausteinen zugeordneten Zwischenergebnissen durchgeführt? Gar nicht ($-$), lokal in Objektzuständen des Bausteins oder temporären Datenbanktabellen bzw. Dateien ($\circ$), global, also auch für andere entfernte Bausteine nutzbar ($+$), oder global und vom Benutzer beeinflußbar ($++$)?

Animation Ist die Programmausführung animiert ($\surd$ oder $-$)?

Programm–Generierung Können eigenständige, unabhängig von der Programmierumgebung lauffähige Programme generiert werden ($\surd$ oder $-$)?

Auf die Art und Weise der Umsetzung der in Tab. 5.1 skizzierten Aspekte in *VIOLA* wird in den folgenden Abschnitten dieses Kapitels noch im Detail eingegangen werden. Als Alleinstellungsmerkmale bzw. herausragende Charakteristika von *VIOLA* sind das Anwendungsgebiet, die Unterstützung der auf multidimensionalen Datenräumen basierenden Datenanalyse, die formale Fundierung der implementierten Analysesprache und die nahtlose Integration interaktiver Graphiken in explorative Datenanalysen zu nennen.[5]

Anhand der Klassifikation visueller Programmiersprachen durch Burnett und Baker (siehe Tab. 2.1 auf Seite 40) läßt sich die Konzeption von *VIOLA* schwerpunktmäßig in folgende Rubriken einordnen:

- Vor allem der Entwurf einer datenflußbasierten Programmiersprache (VPL-II.A3), die visuell als Diagrammsprache (VPL-II.B1) mit Piktogrammen in den Bausteinen (VPL-II.B2) realisiert ist, aber auch ihre Umsetzung in einer visuellen Programmierumgebung (VPL-I.) werden behandelt. Auch Aspekte funktionaler Programmierung (die exakte Definition der Operatoren) sowie objektorientierter Sprachen (über die Betrachtung von Bausteinen als Objekte hinaus auch etwa die Typhierarchien auf Maßzahlen) spielen eine Rolle. Durch eine geeignete Abstraktion vom hinterliegenden formalen Modell in einer benutzungsfreundlichen Umgebung werden somit die Stärken dieser verschiedenen Paradigmen kombiniert (vgl. die Diskussion in Abschnitt 2.2.3).
- Unter den Spracheigenschaften sind Datentypen und -strukturen von besonderem Interesse (VPL-III.C).
- Von eher untergeordneter Bedeutung ist die Sprachimplementierung, hier wären vor allem Berechnungsmodelle (VPL-IV.A) zu erwähnen.
- Als Sprachzweck wäre eine neue Gruppe „Datenanalyse“ neben Datenbank- und Visualisierungssprachen (VPL-V.B bzw. D) einzuführen.
- Die Sprachtheorie spielt eine, wenn nicht sogar *die* zentrale Rolle, vor allem hinsichtlich formaler Sprachdefinition im allgemeinen (VPL-VI.A) sowie verschiedener Aspekte zum Sprachentwurf (VPL-VI.C5: Typprüfung, C6: Fragen der visuellen Darstellung).

Im weiteren Verlauf dieses Kapitels werden zunächst in Abschnitt 5.2 Module und ihre Parameter als Basiseinheiten von Datenflußprogrammen eingeführt. Abschnitt 5.3 zeigt, wie Module über Kanäle zu einem *VIOLA*–Programm verknüpft werden können und gibt — nach Einführung von Verbundmodulen zur Kapselung von Teilprogrammen — nochmals einen kurzen Überblick über alle Modulklassen von *VIOLA* und ihre Anwendbarkeit innerhalb eines Programms. In Abschnitt 5.4 wird nachfolgend die Semantik der Ausführung

[5] Auch *AVS* und *KHOROS* binden ähnliche Techniken ein. Diese haben jedoch recht einfache Gestalt und beziehen sich nicht auf einzelne Attribute der verarbeiteten Datenbestände, sondern auf Visualisierungen als Ganzes oder bestimmte Darstellungsparameter, womit das eigentliche Datenmanagement nicht unterstützt werden kann.

von *VIOLA*–Programmen definiert und auf deren Steuerung durch den Anwender über eine komfortable Benutzungsschnittstelle eingegangen. Abschließend faßt Abschnitt 5.5 die zentralen Merkmale von *VIOLA* nochmals zusammen und gibt eine kurze Bewertung.

5.2 Module: Bausteine von Datenflußprogrammen in *VIOLA*

VIOLA zielt auf die adäquate Unterstützung explorativer Datenanalyse–Sitzungen ab. Gegebene Datenbestände und Analyseverfahren sollen auch von Nicht–Programmierern flexibel selektiert, kombiniert und parametrisiert werden können. Hierzu bieten die Operatoren von *MADEIRA* eine geeignete Abstraktionsebene. Zusammen mit sinnvollen Hilfsstrukturen bilden sie die Basis für die Definition einer Datenflußsprache mit Bausteinen „mittlerer“ Granularität, die

- einerseits weitgehend intuitiv zu nutzen ist, von technischen Details abstrahiert, das schnelle Erstellen von Anwendungen typischer Größe ermöglicht und einen guten Überblick über diese gewährt und
- andererseits genügend Freiräume zur Konfiguration und Manipulation von Analyseabläufen in ihren jeweiligen anwendungsrelevanten Einzelschritten bietet, wobei die unterstützbare Anwendungsdomäne nicht zu stark eingegrenzt werden soll (vgl. auch entsprechende Diskussionen in [LAC+92, GR91]).

Insbesondere wird keine Turing–Mächtigkeit der Sprache angestrebt, komplexere Algorithmen und Abläufe werden in atomaren Programmbausteinen verborgen, wobei das Datenmodell *MADEIRA* die Mittel zur Metadaten–basierten Beschreibung der Bausteinsemantik bietet.

Im folgenden werden die Elemente von *VIOLA*, also die *Module* genannten Bausteine (Knoten) eines Datenflußprogramms, im Detail vorgestellt. Wie bereits in Abschnitt 2.1.2 dargestellt, werden neben der Navigation im Analyseverlauf grob Datenwahl, Statistik und Visualisierung als zentrale Bestandteile von Datenanalysen unterschieden. Die beiden erstgenannten Verfahrensklassen werden durch die Algebra von *MADEIRA* beschrieben, wobei im Rahmen der Datenwahl nochmals zwischen dem Zugriff auf Datenquellen und dem Datenmanagement differenziert wird (siehe Abschnitt 5.2.2). Auf die Ausgabe und Visualisierung von Analyseergebnissen wird in Abschnitt 5.2.3 näher eingegangen. Über diese grundlegenden Elemente hinaus spielt auch die Behandlung von *Parametern* der Verfahren, ihre Definition, Propagierung und Anwendung, eine wichtige Rolle.[6] Insbesondere wird — im Gegensatz zu vielen bestehenden datenflußbasierten Analyseumgebungen (vgl. Abschnitt 3.4) — eine enge Integration in das Datenflußkonzept angestrebt (siehe Abschnitt 5.2.4). Schließlich werden auch ausgewählte *Kontrollstrukturen* (in Abschnitt 5.2.5) eingeführt; die Möglichkeit der Zusammenfassung von Teilnetzen in *Verbunden* (quasi Subprozeduren) wird erst in Abschnitt 5.3.2 im Zusammenhang mit der Definition von *VIOLA*–Programmen (also der Verbindung von Modulen über Kanäle) diskutiert. Beide Modulklassen (Kontrollstrukturen und Verbunde) erleichtern dem Anwender die Programmierung bzw. Analyse, sollen jedoch nicht die Sprachmächtigkeit erhöhen. Zunächst aber werden in Abschnitt 5.2.1 einige grundlegende Definitionen vorgenommen sowie generelle Überlegungen zum Aufbau von Modulen und zur Gestalt ihrer Parameter angestellt.

5.2.1 Grundlegende Strukturen und Definitionen

Die Module von *VIOLA* realisieren alle ein einheitlichen Basisschema: sie verarbeiten eingehende Datenwürfel und erzeugen ihrerseits neue Datenwürfel, die zu nachfolgenden Modulen propagiert werden. Die Berechnungen eines Moduls können durch Parameter gesteuert werden, die im wesentlichen in Strukturen aus *MADEIRA* (also Komponenten von Datenräumen) bestehen und explizit durch den Benutzer festgelegt oder implizit aus vorliegenden Datenräumen extrahiert werden können. Dieser Abschnitt stellt verwendete Parametertypen und deren Handhabung sowie den grundlegenden Aufbau von Modulen vor.

[6] Angedeutet wird dieser Aspekt auch in [AC96a], wo neben der Transformation (der statistischen Analyse) und Dekomposition (im wesentlichen dem Datenmanagement entsprechend) auch die Reduktion von Daten (auf einzelne Maßzahlen, aber auch auf Größen, die für die Steuerung anderer Analyseschritte verwendet werden sollen) als dritte Hauptgruppe von Operationsbausteinen angeführt wird.

Die Definitionen im gesamten Kapitel 5 folgen den gleichen Notationskonventionen, wie sie bereits in Kapitel 4 eingeführt wurden; zusätzlich werden hier für dynamische Zuordnungen zur Ausführungszeit eines *VIOLA*–Programms (Belegungen von Modulen und Kanälen) analog zu dynamisch zuweisbaren Metadaten beliebige Funktionsbezeichner verwendet. $\mathcal{MOD}$ bezeichne die Menge aller Module, die im Laufe dieses Abschnitts noch genauer definiert werden sollen.

Parameter von Modulen

Wie aus den Definitionen der Operatoren von *MADEIRA* in Abschnitt 4.3 deutlich wird, sind Parameter zur näheren Spezifikation und Steuerung der Verarbeitung von Datenräumen unverzichtbar. Vor allem werden interessierende Teilkomponenten von Eingangs- oder Ergebnisdatenräumen näher beschrieben. Darüber hinaus benötigen vor allem Datenquellen und -senken interne Parameter zur Festlegung eher technischer Details des Datenzugriffs bzw. der Datenausgabe, die weitgehend unabhängig vom zugrundeliegenden Datenmodell sind.

Während letztere Ausprägungen von Parametern jeweils modulintern zu spezifizieren und zu nutzen sind, wird in *VIOLA* die oftmals modulübergreifende Nutzung der datenmodellbezogenen Parameter im Rahmen der datenflußbasierten Verarbeitung sichtbar gemacht. Die Darstellung entsprechender parameterbezogener Datenflüsse trägt wesentlich zum Verständnis der Gestalt und Semantik einer gerade durchgeführten Untersuchung bei und erleichtert die flexible Verknüpfung von Daten bzw. die Koordinierung von Analysepfaden in explorativen Datenanalysen. Interne Parameter dagegen bleiben auf der Datenflußebene vor dem Benutzer im Sinne einer adäquaten Abstraktion von Datenanalysen verborgen. Entsprechend dieser Überlegungen werden zwei Arten von Modulparametern, nämlich interne (Basis-) und externe (multidimensionale) Parameter unterschieden. Eine Sonderrolle spielen boolesche Parameter, die sowohl intern als auch im Zusammenhang mit Kontrollstrukturen extern genutzt werden können. Eine Parametrisierung von Modulen durch Ausdrücke oder Formeln (wie etwa in *KHOROS* oder *AVS*) ist in *VIOLA* (zunächst) nicht vorgesehen.[7]

Definition 5.1 ((Modul–)Parameter) *Ein* (MODUL–)PARAMETER $par = (b, dom, mod)$ *sei definiert durch*

- *einen Namen* $b \in \mathcal{B}$,
- *einen Wertebereich* $dom \subset \Omega$ *als Teilmenge eines (weiter unten noch genauer definierten) „Werteuniversums“ und*
- *ein Modul* $mod \in \mathcal{MOD}$, *dem par zugeordnet ist.*

$\mathcal{PAR}$ *sei die Menge aller Parameter. Hinsichtlich des Wertebereichs unterscheiden wir*

- MULTIDIMENSIONALE PARAMETER $\mathcal{PAR}^{\mathrm{m}} \subset \mathcal{PAR}$ *mit* $par \in \mathcal{PAR}^{\mathrm{m}} \Leftrightarrow par.dom \in$
  $\{\mathcal{E}, 2^{\mathcal{E}}, \bigcup_{D \in \mathcal{DO}} 2^{\mathcal{E}_D}, 2^{\mathcal{O}}, 2^{2^{\mathcal{O}}}, \mathcal{R}, 2^{\mathcal{R}}\} \cup \{\mathcal{OR}, 2^{\mathcal{OR}}, \mathcal{E} \times \mathcal{OR}, 2^{\mathcal{E} \times \mathcal{OR}}, \bigcup_{D \in \mathcal{DO}} 2^{\mathcal{E}_D \times \mathcal{OR}}\} \cup$
  $\{\mathcal{K}, \bigcup_{D \in \mathcal{DO}} 2^{D}, \mathcal{L}, \bigcup_{D \in \mathcal{DO}} 2^{\mathcal{L}_D}, \mathcal{DO}, 2^{\mathcal{DO}}\} \cup \{\mathcal{M}, 2^{\mathcal{M}}, \mathcal{DR}\}$,
- BASISPARAMETER $\mathcal{PAR}^{\mathrm{b}} \subset \mathcal{PAR} \setminus \mathcal{PAR}^{\mathrm{m}}$, *deren Wertebereiche gängige universelle Datentypen*[8], *wie cardinal* $\subset \mathbb{N}_0$, *integer* $\subset \mathbb{Z}$, *float* $\subset \mathbb{R}$, *string* $\subset \mathcal{S}$ *oder (in Form beliebiger Wertelisten) Aufzählungstypen repräsentieren, und*
- BOOLESCHE PARAMETER $\mathcal{PAR}^{\mathrm{bool}} = \mathcal{PAR} \setminus (\mathcal{PAR}^{\mathrm{b}} \cup \mathcal{PAR}^{\mathrm{m}})$ *mit* $par.dom = \mathbb{B}$.

Ω *bezeichne die Vereinigung aller dieser Domänen.*[9]

[7] Dies könnte etwa als Ad–hoc–Spezifikation neuer Maßzahlen angesehen werden und wäre unter Verwendung von Default–Datentypen und allgemeinen Berechenbarkeitsdefinitionen in das vorgestellte Modell einzubetten. Der Einfachheit und klareren Modellierung halber sollen derartige Erweiterungen zur Laufzeit jedoch nicht unterstützt werden.

[8] Die konkrete Auswahl hängt stark an Details der Implementierung von *VIOLA* und den konkret genutzten Maßzahlen und Verfahren. Deshalb wird an dieser Stelle von einer vollständigen Modellierung abgesehen.

[9] Bereits in Def. 4.13 wurde bei der Einführung von Datentypen als Basis für Maßzahlausprägungen ein universeller Wertevorrat $\mathcal{W}$ eingeführt, der auch die Menge aller Kategorien $\mathcal{K}$ umfaßt. Um deutlich zu machen, daß hier — etwa mit der Menge aller Maßzahlen selbst oder den Mengen von Eigenschaften, Objektmengen und Rollen — noch weitaus mehr Entitäten einbezogen sind, wird die Menge Ω definiert. Für sie kann $\mathcal{W} \subset \Omega$ angenommen werden.

Ferner sei $\mathcal{PAR}^{\text{ext}} = \mathcal{PAR}^{\text{m}} \cup \mathcal{PAR}^{\text{bool}}$ *die Menge der* EXTERNEN *und* $\mathcal{PAR}^{\text{int}} = \mathcal{PAR}^{\text{b}} \cup \mathcal{PAR}^{\text{bool}}$ *die Menge der* INTERNEN PARAMETER.

Eine (totale) Abbildung $val\colon \mathcal{PAR} \to \Omega$ *ordne jedem Parameter* $par \in \mathcal{PAR}$ *seine (jeweils aktuelle) Ausprägung* $val(par) \in par.dom$ *zu. Sofern der Kontext klar ist, soll im folgenden jeweils b stellvertretend für par selbst oder den zugeordneten Parameterwert verwendet werden — sinnvollerweise sollte b innerhalb der Parameter des Moduls mod eindeutig sein.*

Die Menge der Wertebereiche multidimensionaler Parameter deckt alle *sinnvoll* als Parameter von Modulen nutzbaren Entitätsmengen von *MADEIRA* in jeweils ein- und mengenwertiger Form ab — vgl. Abb. 4.1. Hierbei wird eine pragmatische Auswahl getroffen, die sich an den Erfordernissen der Operatoren von *VIOLA* und ihrem möglichst intuitiven Einsatz orientiert:

- Mikrodaten und deren Attribute spielen keine Rolle.
- Dimensionen sind mit Domänen identifizierbar und somit selbst nicht als gesonderter Parametertyp vorgesehen. Ebenenmengen werden (wie in Kategorienhierarchien) auf Ebenen der jeweils gleichen Domäne eingeschränkt. Ebenso sollen Kategorienmengen nur einer Domäne angehören.
- Demgegenüber können sich Eigenschaftsmengen auch auf mehrere Domänen beziehen, da in ihrer Verwendung (zur Beschreibung kategorieller Attribute) oftmals eine enge Affinität zu der von Domänenmengen besteht, während Kategorien- und Ebenenmengen eher der Wahl von Kategorien *eines* Attributs dienen.
- Kategorielle und summarische Attribute sind selbst zu spezifisch auf einen bestimmten Datenraum zugeschnitten. Ihre Nutzung als Parameter würde keinen Gewinn gegenüber der (deutlich flexibleren) Verwendung ihrer Komponenten bringen.
- Umgekehrt spielen Aggregierungsfunktionen und Datentypen keine Rolle als Parameter. Die Maßzahl ist die geeignetere Abstraktionsebene.

Schließlich sind auch vollständige Makrodatenräume als Parameter vorgesehen — zum einen, weil ein Datenwürfel(-schema) leicht als Repräsentant für verschiedene andere Parameter (Maße, Objektmengen etc.) angesehen werden kann, und zum anderen zur komfortableren Nutzung des Vereinigungsoperators (siehe Def. 5.10). Da in beiden Fällen der Einsatz von Datenraummengen keinen wesentlichen Vorteil bedeuten würde und zudem auch sonst nur einzelne Datenräume zwischen Modulen propagiert werden sollen, wird auf mengenwertige Datenraumparameter verzichtet.

Im folgenden bezeichnen $\widetilde{2^{\mathcal{K}}} \stackrel{\text{def}}{=} \bigcup_{D \in \mathcal{DO}} 2^{D}$, $\widetilde{2^{\mathcal{L}}} \stackrel{\text{def}}{=} \bigcup_{D \in \mathcal{DO}} 2^{\mathcal{L}_D}$, $\widetilde{2^{\mathcal{E}}} \stackrel{\text{def}}{=} \bigcup_{D \in \mathcal{DO}} 2^{\mathcal{E}_D}$ sowie $\widetilde{2^{\mathcal{E} \times \mathcal{OR}}} \stackrel{\text{def}}{=} \bigcup_{D \in \mathcal{DO}} 2^{\mathcal{E}_D \times \mathcal{OR}}$ jeweils eine Wertemenge über einer einzelnen Domäne.

Nicht nur Datenräume können stellvertretend für Parameter anderen Typs verwendet werden — ähnliche Zusammenhänge bestehen zwischen vielen Parameterarten und ermöglichen eine gewisse Generizität in der Parameternutzung. Tabelle 5.2 gibt einen Überblick über entsprechende implizite Konvertierungsmöglichkeiten, die sowohl kanonisch definierbar als auch intuitiv nutzbar sind.

Es ergibt sich hieraus folgende Definition zur Umwandlung von Ausprägungen zwischen Typen externer Parameter:

Definition 5.2 (Kompatibilität und Konvertierung externer Parameter) *Ein Parameter* $par_1 \in \mathcal{PAR}^{\text{ext}}$ *heiße zu einem Parameter* $par_2 \in \mathcal{PAR}^{\text{ext}}$ KOMPATIBEL, *falls*

- *gemäß Tab. 5.2 eine Konvertierungsmöglichkeit vom Wertebereich von* par_1 *in den Wertebereich von* par_2 *besteht,*
- *beide Parameter Maßzahlen oder Maßzahlmengen repräsentieren (Wertebereich* $\mathcal{M}$ *oder* $2^{\mathcal{M}}$*) — die (triviale) Konvertierung erfolgt hier analog zu den übrigen Wertebereichen — oder*
- *beide boolesche Wertebereiche besitzen.*

v.\nach	e	$E/\tilde{E}$[1]	O	$\mathbf{O}$[2]	r	R	(O,r)	OR	(e,O,r)	$EOR/\widetilde{EOR}$[1]
e	e	$\{e\}$	O_e	2^{O_e}	–	–	–	$2^{O_e}\times\mathcal{R}$	–	$\{e\}\times 2^{O_e}\times\mathcal{R}$
$E/\tilde{E}$[1]	$(-)$[4]	E	$\bigcup_{e\in E} O_e$	$\bigcup_{e\in E} 2^{O_e}$	–	–	–	$\bigcup_{e\in E} 2^{O_e}\times\mathcal{R}$	–	$\bigcup_{e\in E}\{e\}\times 2^{O_e}\times\mathcal{R}$
O	–	$f^{\text{eig}}(O)$[6]	O	$\{O\}$	–	–	–	$\{(O,\cdot)\}$	–	$\{(\cdot,O,\cdot)\}$
$\mathbf{O}^2$	–	$\bigcup_{O\in\mathbf{O}} f^{\text{eig}}(O)$	$(-)$	$\mathbf{O}$	–	–	–	$\{(O,\cdot)\mid O\in\mathbf{O}\}$	–	$\{(\cdot,O,\cdot)\mid O\in\mathbf{O}\}$
r	–	–	–	–	r	$\{r\}$	–	$\{(\cdot,r)\}$	–	$\{(\cdot,\cdot,r)\}$
R	–	–	–	–	$(-)$	R	–	$\{(\cdot,r)\mid r\in R\}$	–	$\{(\cdot,\cdot,r)\mid r\in R\}$
(O,r)	–	$f^{\text{eig}}(O)$[6]	O	$\{O\}$	r	$\{r\}$	(O,r)	$\{(O,r)\}$	–	$\{(\cdot,O,r)\}$
OR	–	$\bigcup_{(O,\cdot)\in OR} f^{\text{eig}}(O)$	$(-)$	$\{O\mid(O,\cdot)\in OR\}$	$(-)$	$\{r\mid(\cdot,r)\in OR\}$	$(-)$	OR	–	$\mathcal{E}\times OR$
(e,O,r)	e	$\{e\}$	O	$\{O\}$	r	$\{r\}$	(O,r)	$\{(O,r)\}$	(e,O,r)	$\{(e,O,r)\}$
$EOR/\widetilde{EOR}$[1]	$(-)$	$\{e\mid(e,\cdot,\cdot)\in EOR\}$	$(-)$	$\{O\mid(\cdot,O,\cdot)\in EOR\}$	$(-)$	$\{r\mid(\cdot,\cdot,r)\in EOR\}$	$(-)$	$\{(O,r)\mid(\cdot,O,r)\in EOR\}$	$(-)$	EOR
D	–	$\mathcal{E}_D$	–	–	–	–	–	–	–	$\mathcal{E}_D\times\mathcal{OR}$
$\mathbf{D}^2$	–	$\bigcup_{D\in\mathbf{D}}\mathcal{E}_D$	–	–	–	–	–	–	–	$\bigcup_{D\in\mathbf{D}}\mathcal{E}_D\times\mathcal{OR}$

v.\nach	k	K[3]	le	L[3]	D	$\mathbf{D}^2$
k	k	$\{k\}$	–	$\{le\mid k\in le.K\}$	$k.D$	$\{k.D\}$
K^3	$(-)$	K	–	$\{le\mid le.K\cap K\neq\emptyset\}$	$k.D$ $(k\in K)$	$\{k.D\mid k\in K\}$
le	$(-)$[5]	$le.K$	le	$\{le\}$	$le.D$	$\{le.D\}$
L^3	$(-)$	$\bigcup_{le\in L} le.K$	$(-)$	L	$le.D$ $(le\in L)$	$\{le.D\mid le\in L\}$
D	$d_D.k_0$	D	$d_D.le_0$	$\mathcal{L}_D$	D	$\{D\}$
$\mathbf{D}^2$	$(-)$	$(-)$	$(-)$	$(-)$	$(-)$	$\mathbf{D}$
$e/(e,\cdot,\cdot)$	–	–	–	–	D_e	$\{D_e\}$
E/EOR	–	–	–	–	$(-)$[5]	$\{D_e\mid e\in E\ldots\}$
$\tilde{E}/\widetilde{EOR}$[1]	–	–	–	–	$D_e(e\in\tilde{E}\ldots)$	$\{D_e\mid e\in\tilde{E}\ldots\}$

Konvertierung v. Datenräumen dr nach ...			
e	$(-)$[5]	$E/\tilde{E}$[1]	$\bigcup_{ka\in dr.KA}\{e\mid(e,\cdot,\cdot)\in ka.EOR\}$
O	$(-)$[5]	$\mathbf{O}^2$	$\{O\mid(O,\cdot)\in dr.OR\}$
r	$(-)$[5]	R	$\{r\mid(\cdot,r)\in dr.OR\}$
or	$(-)$[5]	OR	$dr.OR$
eor	$(-)$[5]	$EOR/\widetilde{EOR}$[1]	$\bigcup_{ka\in dr.KA} ka.EOR$
k	$(-)$[5]	K	$(-)$ (falls $dr.KA=\{ka\}$ einelementig: $ka.K$)
le	–	L	–
D	$(-)$[5]	$\mathbf{D}^2$	$\{ka.D\mid ka\in dr.KA\}$
mz	$(-)$[5]	MZ	$\{sa.mz\mid sa\in dr.SA\}$

[1] $\tilde{E}$ und $\widetilde{EOR}$ bezeichnen jeweils Eigenschaftsmengen zu einer einzelnen Domäne. Sie werden i. a. analog zu ihren jeweiligen Pendants E und EOR konvertiert, wobei Umwandlungen von letzteren — sowie von Domänenmengen und Datenräumen — in erstgenannte nur bei Eindeutigkeit der Domäne möglich sind. Eine Konvertierung von Rollen oder (rollenbehafteten) Objektmengen nach $\tilde{E}$ oder $\widetilde{EOR}$ ist nicht möglich.

[2] **D** und **O** bezeichnen eine *Menge* von Domänen bzw. Objektmengen.

[3] Kategorien- und Ebenenmengen entstammen stets einer eindeutigen Domäne.

[4] Der Eintrag „$(-)$“ besagt allgemein, daß lediglich für einelementige Mengen eine Konvertierung (analog zur Behandlung des einfachen Typs des Listenelements) möglich ist.

[5] Ebenen mit nur einer Kategorie, Eigenschaftsmengen über einer Domäne sowie Datenräume mit einelementigen Komponenten können analog zur Behandlung des jeweiligen mengenwertigen Ziel–Wertebereichs eindeutig in Einzelwerte konvertiert werden.

[6] Unter $f^{\text{eig}}(O)$ sollen die *gemeinsamen* Eigenschaften aller Objekte in O verstanden werden.

Tabelle 5.2: Konvertierung zwischen Domänen multidimensionaler Parameter

Hierbei seien die mit „$(-)$“ gefüllten Felder in Tab. 5.2 (inkl. der Konvertierung von Datenräumen in Kategorienmengen) sowie die Umwandlung von Domänenmengen, Datenräumen oder beliebigen Eigenschaftsmengen (E, EOR) in Eigenschaftsmengen zu einer eindeutigen Domäne ($\tilde{E}$, $\widetilde{EOR}$) eingeschlossen — entsprechende Parameterkombinationen werden als EINWERTIG KOMPATIBEL *bezeichnet.*

Sicherlich wären über die in Tab. 5.2 dargestellten Fälle hinaus noch weitere Konvertierungen denkbar — insbesondere als deren transitiver Abschluß, etwa von $2^{\mathcal{OR}}$ über $2^{\mathcal{E}}$ nach $2^{\mathcal{DO}}$. Derartige Fälle erscheinen aber nicht mehr intuitiv nutzbar, so daß von ihrer Unterstützung Abstand genommen wird.[10] Die genaue Abgrenzung „sinnvoller“ Umwandlungen ist sicher im Einzelfall diskutierbar und nach umfangreicheren Erfahrungen mit der Nutzung von *VIOLA* ggf. anzupassen.

Generell vereinfacht die Umsetzung der Kompatibilität von Parametertypen die Handhabung von *VIOLA*, da durch den Verzicht auf explizite Umwandlungen Programme schneller erstellt und übersichtlicher gehalten werden können. Auf mengenwertigen Parametern sollte in der Implementierung von *VIOLA* eine Ordnung definiert werden, um hierdurch insbesondere *sinnvolle* Iterationen definieren zu können — hierauf kommen wir in Abschnitt 5.2.5 und Abschnitt 5.3.2 nochmals zurück.

Aufbau eines Moduls

Module sind die Verarbeitungseinheiten bzw. Bausteine der Datenflußsprache *VIOLA*. In den folgenden Abschnitten werden verschiedene Klassen von Modulen eingeführt, die sich jeweils in Art und Menge ein- und ausgehender Daten sowie natürlich ihrer Verarbeitungsfunktion unterscheiden. Auch werden jeweils Mengen externer und interner Parameter vorgegeben, wobei nicht immer eine endgültige Festlegung der Parametermengen erfolgt, sondern gewisse Freiräume hinsichtlich der (anwendungsspezifischen) Spezialisierung von Modulklassen gelassen werden.

Definition 5.3 (Module) *Die Menge $\mathcal{MOD}$ aller* MODULE *(*DATENFLUSS–BAUSTEINE*) sei definiert als disjunkte Vereinigung der Mengen der Datenquellmodule, der Datenmanagementmodule, der Module zur Maßzahlberechnung, der Module zur Parameterdefinition, der Ausgabemodule, der Module zur Definition von Kontrollstrukturen und der Verbundmodule:* $\mathcal{MOD} = \mathcal{MOD}^{\text{src}} \cup \mathcal{MOD}^{\text{dm}} \cup \mathcal{MOD}^{\text{calc}} \cup \mathcal{MOD}^{\text{par}} \cup \mathcal{MOD}^{\text{sink}} \cup \mathcal{MOD}^{\text{ctrl}} \cup \mathcal{MOD}^{\text{sub}}$.

Ports bilden Anknüpfungspunkte von Modulen, über die diese Daten von anderen Modulen erhalten bzw. an andere Module weiterleiten. Es gibt Ports, die den Zugriff auf Mengen von ein- oder ausgehende Datenräumen gewähren, und solche, die externe Parameter repräsentieren. Die Datenports eines Moduls zerfallen in Eingabe- und Ausgabeports, die — entsprechend der Verarbeitungsfunktion eines Moduls — *unterschiedliche* Datenräume enthalten. Demgegenüber steht ein Parameterport stellvertretend für *einen* Modulparameter und vereint somit Import- und Exportfunktionalität. Es können über den Port sowohl Verbindungen zu anderen Modulen, über die eine Propagierung der Parameterwerte erfolgt, hergestellt als auch (nachträglich) Parameter interaktiv vom Benutzer modifiziert werden. Weiterhin kann ein Modul auch automatisch neue Parameterwerte aus anliegenden Datenräumen und anderen Parametern ermitteln.[11] Während der Import von Parameterwerten (zur Vermeidung von Mehrdeutigkeiten bei der Durchführung von Parameterberechnungen) auch ausgeschlossen werden kann, ist ihr Export stets vorgesehen. Dies dient zum einen zur flexiblen Mehrfachverwendung von Parameterwerten und zum anderen auch zur Information des Systembenutzers.

Definition 5.4 (Ports) *Ein* PARAMETERPORT $po^{\text{par}} = (b, par^{\text{ext}}, mod, in?)$ *mit einer Identifikation $b \in \mathcal{B}$ repräsentiere einen externen Parameter $par^{\text{ext}} \in \mathcal{PAR}^{\text{ext}}$ und sei einem Modul $mod \in \mathcal{MOD}$ zugeordnet.*

Es gelte $par^{\text{ext}}.mod = mod$. Das Flag $in? \in \mathbb{B}$ spezifiziere, ob po^{par} auch zum Import von Parameterwerten genutzt werden kann.

Sofern der Modulkontext mod eindeutig und die Identifikation b nicht relevant ist, wird im folgenden ein Port zu einem Parameter $par^{\text{ext}} = (b', dom, mod)$ auch kurz als „b': dom“ geschrieben. Hierbei werde als Default $in? = true$ angenommen.

$\mathcal{PO}^{\text{par}}$ sei die Menge aller Parameterports.

[10] Einzelne transitive Beziehungen, die noch als eingängig gelten können, wie etwa die Konvertierung von $\mathcal{E} \times \mathcal{OR}$ über $\mathcal{E}$ nach $\mathcal{DO}$, sind in Tab. 5.2 explizit angeführt.

[11] Diese alternative Definition von Parameterwerten über Datenpropagierung aus anderen Knoten des Datenflußnetzes oder interaktive Manipulation durch den Benutzer findet sich ähnlich auch bei *AVS* oder dem *IBM Data Explorer*.

Ein DATENPORT $po^{\mathrm{dat}} = (b, card, mod)$ *sei definiert durch den Bezeichner* $b \in \mathcal{B}$*, die maximale Kardinalität* $card \in \mathbb{N}_0 \cup \{*\}$ *der Menge von Datenräumen, die* po^{dat} *aufnehmen kann*[12]*, sowie ein zugeordnetes Modul* $mod \in \mathcal{MOD}$*.*

$\mathcal{PO}^{\mathrm{dat}}$ *sei die Menge aller Datenports.*

Eine (totale) Abbildung $data\colon \mathcal{PO}^{\mathrm{dat}} \to 2^{\mathcal{DR}}$ *ordne einem Datenport jeweils seine aktuelle Belegung durch eine Menge von Makrodatenräumen zu. Falls die maximale Kardinalität des Ports 1 ist, soll das Funktionsergebnis von* $data()$ *im folgenden bei Bedarf auch als Element von* $\mathcal{DR}$ *interpretiert bzw. als undefiniert angesehen werden, falls es sich um die leere Menge handelt.*

$\mathcal{PO} = \mathcal{PO}^{\mathrm{par}} \cup \mathcal{PO}^{\mathrm{dat}}$ *bezeichne die Menge aller Ports.*

Man beachte, daß Datenports auch „leer“ sein können (falls Daten noch nicht berechnet sind), während Parameterports stets einen definierten Parameterwert (ggf. also eine Default–Angabe) repräsentieren.

Nachfolgend wird die Basisstruktur, die allen Modulklassen von *VIOLA* zugrunde liegt, kurz vorgestellt, so daß in den nächsten Abschnitten hierauf Bezug genommen werden kann und nicht immer alle Einzelheiten erneut formalisiert werden müssen. Spezifische Modultypen konkretisieren diese Definition, schränken die Menge der Teilkomponenten ein oder fügen einzelne Details oder einschränkende Bedingungen im Kontext der jeweils realisierten Datenverarbeitung hinzu.

Definition 5.5 (Allgemeine Modulstruktur) *Ein Modul* $mod \in \mathcal{MOD}$ *eines VIOLA–Programms kann allgemein beschrieben werden als ein 7–Tupel* $(b, PO^{\mathrm{in}}, PO^{\mathrm{out}}, PO^{\mathrm{par}}, PAR^{\mathrm{int}}, f^{\mathrm{dat}}, f^{\mathrm{par}})$ *aus*

- *einem Bezeichner* $b \in \mathcal{B}$*,*
- *einer endlichen Menge* $PO^{\mathrm{in}} \subseteq \mathcal{PO}^{\mathrm{dat}}$ *von Datenports als Moduleingänge und einer endlichen Menge* $PO^{\mathrm{out}} \subseteq \mathcal{PO}^{\mathrm{dat}}$ *von Datenports als Modulausgänge, wobei* $PO^{\mathrm{out}} \cap PO^{\mathrm{in}} = \emptyset$ *— ferner gelte* $\forall po \in PO^{\mathrm{out}}$*:* $po.card = 1$ *sowie* $\forall po \in PO^{\mathrm{out}} \cup PO^{\mathrm{in}}\colon po.mod = mod$*,*
- *einer endlichen Menge* $PO^{\mathrm{par}} \subseteq \mathcal{PO}^{\mathrm{par}}$ *von Parameterports zum Zugriff auf externe Parameter (*$\forall po^{\mathrm{par}} \in PO^{\mathrm{par}}\colon po^{\mathrm{par}}.mod = mod$*),*
- *einer endlichen Menge* $PAR^{\mathrm{int}} \subseteq \mathcal{PAR}^{\mathrm{int}}$ *interner Parameter (*$\forall par \in PAR^{\mathrm{int}}\colon par.mod = mod$*),*
- *einer nicht immer eindeutigen (also u. U. nichtdeterministischen*[13]*) Vorschrift in Form einer Familie* $f^{\mathrm{dat}} = (f^{\mathrm{dat}}_{po})_{po \in PO^{\mathrm{out}}}$ *von partiellen Funktionen (bzw. Relationen)*

$$f^{\mathrm{dat}}_{po}\colon \bigtimes_{|PO^{\mathrm{in}}|} 2^{\mathcal{DR}} \times \bigtimes_{po \in PO^{\mathrm{par}}} po.par.dom \times \bigtimes_{par \in PAR^{\mathrm{int}}} par.dom \to \mathcal{DR}$$

zur Berechnung der Belegung der Datenausgangsports aus den Belegungen der Dateneingangsports unter Rückgriff auf interne und externe Parameter, und

- *einer Familie* $f^{\mathrm{par}} = (f^{\mathrm{par}}_{po})_{po \in PO^{\mathrm{par}}}$ *von (evtl. nichtdeterministischen und partiellen) Berechnungsfunktionen (-relationen) zur Definition von Werten externer Parameter:*

$$f^{\mathrm{par}}_{po}\colon \bigtimes_{|PO^{\mathrm{in}}|} 2^{\mathcal{DR}} \times \bigtimes_{po \in PO^{\mathrm{par}}} po.par.dom \times \bigtimes_{par \in PAR^{\mathrm{int}}} par.dom \to po.par.dom \quad .$$

Für importierte externe Parameter (d. h. $po.in? = true$*) wird* f^{par}_{po} *nicht benötigt und sei deshalb stets undefiniert.*

Die Mengen der internen und externen Parameter eines Moduls seien (im Hinblick auf boolesche Parameter) stets disjunkt. Wir schreiben $mod.b^{l}$ *zur Bezeichnung eines (internen oder externen) Parameters von mod, der die Identifikation* b^{l} *besitzt.*

[12] „$*$“ repräsentiere hierbei kein konkretes Maximum, sondern stehe für beliebig große endliche Mengen. Kardinalitäten von 0 werden nur zur einfacheren Modellierung von Verbundmodulen im Rahmen ihrer Konstruktion erlaubt — ansonsten wären derartige Fälle wenig sinnvoll, aber auch nicht störend.

[13] Dies entspricht der Definition der Operatoren von *MADEIRA* in Abschnitt 4.3. Auf die Problematik des Nichtdeterminismus kommen wir am Anfang von Abschnitt 5.2.2 noch zu sprechen.

In den folgenden Abschnitten wird im Rahmen der Definition spezifischer Modultypen jeweils nur noch auf solche Modulkomponenten eingegangen, die nicht die leere Menge (Ports oder Parameter) bzw. die Identitätsfunktion (f^{dat}) oder gänzlich undefiniert (f^{par}) sind.

Das in Abb. 5.1 gegebene *UML*–Klassendiagramm gibt (in der gleichen Notation wie im entsprechenden Diagramm für *MADEIRA* in Abb. 4.1) einen Überblick über den Zusammenhang zwischen den verschiedenen Arten von Modulparametern (links) und den Modulen sowie deren Ports (rechts).

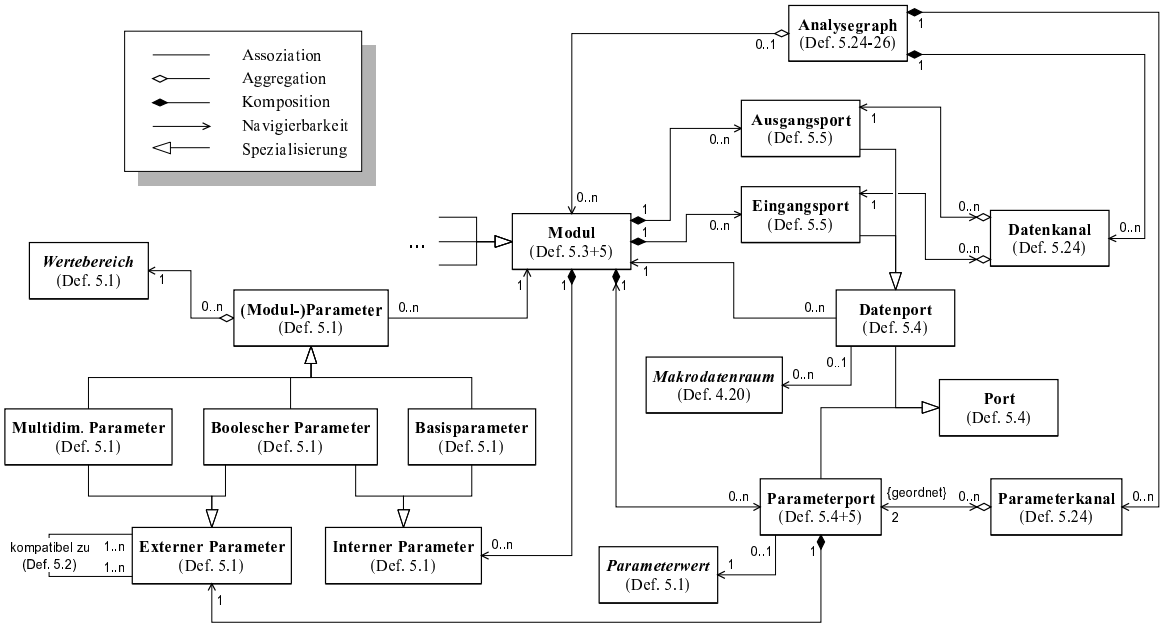

Abbildung 5.1: *UML*–Klassendiagramm der Basiskomponenten von *VIOLA*

Ferner ist bereits die Art der Verbindung von Modulen durch Kanäle innerhalb von Analysegraphen (Datenflußprogrammen) skizziert — hierauf werden wir später (in Abschnitt 5.3) noch genauer eingehen. Unterklassen von Modulen (Def. 5.13–5.15, 5.21–5.23 und 5.27) sind lediglich durch drei Punkte („. . .“) angedeutet.

Die Belegung von Daten- und Parameterports wird im *UML*–Diagramm so dargestellt, daß jeder Datenraum bzw. Parameterwert nur einem einzigen Port zugeordnet werden kann. Je nach Implementierung wären prinzipiell (etwa im Rahmen eines Caching von Datenräumen, vgl. Abschnitt 6.2.3) auch Mehrfachzuordnungen denkbar, was aber auf der konzeptionellen Ebene nicht sichtbar sein sollte.

Folgende Constraints bestehen bzgl. der Beziehungen zwischen den in Abb. 5.1 dargestellten Entitätsmengen:

- Mögliche Wertebereiche von Parametern ergeben sich gemäß der Spezialisierungen in verschiedene Parameterklassen.
- Kompatibilitäten zwischen Parametern begründen sich aus den jeweiligen Wertebereichen.
- Parameter, die ein Modul referenzieren, sind gleich denen, die dem Modul als interne Parameter oder über Parameterports zugeordnet sind.
- Analog besteht eine wechselseitige Beziehung zwischen Datenports und Modulen.
- Die Wertebelegung eines Parameterports stammt aus dem Wertebereich des jeweiligen Parameters.
- Die Mengen der internen und externen Parameter eines Moduls sind disjunkt.
- Alle Kanäle verlaufen stets zwischen Ports *verschiedener* Module.

- Die Parameter zweier durch einen Kanal verbundener Parameterports müssen kompatibel sein.
- Bestimmte Parameterports (mit $in? = false$) akzeptieren keine eingehenden Kanäle.
- Jeder Ausgangsport enthält nur einen Datenraum; die Anzahl der Datenkanäle zu einem Eingangsport ist durch eine portbezogene maximale Kardinalität *card* beschränkt und stets größer oder gleich der Zahl der enthaltenen Datenräume.
- Analysegraphen sind hinsichtlich enthaltener Kanäle in einem bestimmten, an dieser Stelle noch nicht näher spezifizierten Sinne zyklenfrei.

Es sei nochmals klar gesagt, daß der Schwerpunkt in der Funktionalität von Modulen eindeutig auf der Funktion f^{dat} liegt. Eine Parametermanipulation erfolgt in der Regel interaktiv durch den Nutzer, lediglich einige spezielle Module (siehe Abschnitt 5.2.4) dienen zur automatischen Parameterextraktion aus Datenräumen. Insgesamt gibt es kein Modul, in dem sowohl f^{dat} als auch f^{par} realisiert (d. h. weder die Identitätsfunktion noch undefiniert) sind, was zu einer klareren Programmstrukturierung beiträgt. Prinzipiell wäre aber auch in Erweiterungen von *VIOLA* eine Kombination beider Berechnungen denkbar. Ein enger Zusammenhang besteht zwischen den Berechnungsfunktionen f^{dat} und f^{par} einerseits und den Belegungsfunktionen $data()$ und $val()$ von Datenports bzw. Parametern andererseits: Die beiden erstgenannten definieren durch ihre Anwendung die Ausprägungen von letzteren neu.

Eine Besonderheit von *VIOLA* besteht darin, daß — bis auf wenige Ausnahmen (siehe Abb. 5.3) — die meisten Module darauf ausgelegt sind, die Daten all ihrer Eingänge jeweils *unabhängig* voneinander durch das *gleiche* Verfahren zu bearbeiten und zu einem korrespondierenden Ausgang weiterzuleiten. Ein- und Ausgangsports treten also jeweils in Paaren auf, und die Berechnungsvorschrift f^{dat} läßt sich vereinfachen zu einer Funktion, die jeweils die Datenräume *eines* Eingangsports unter Heranziehung verschiedener Parameter auf *einen* Ergebnisdatenwürfel abbildet. In den folgenden Abschnitten werden derartige Portmengen von Modulen stets als $PPO \subseteq \mathcal{PO}^{\text{dat}} \times \mathcal{PO}^{\text{dat}}$ spezifiziert. Abbildung 5.2 zeigt ein in diesem Sinne typisches Modul mit jeweils zwei Parameterports (entsprechend externen Parametern) und zwei internen Parametern. In den meisten Fällen wird pro Dateneingangsport lediglich ein einzelner Datenwürfel als Eingabe akzeptiert — ggf. sind gemeinsam zu verarbeitende Datenräume vorher zu vereinigen. Lediglich der Vereinigungsoperator selbst (siehe Abschnitt 5.2.2), das Routing von Datenräumen (Abschnitt 5.2.5) sowie Verbundmodule (Abschnitt 5.3.2) können pro Eingangsport Mengen von Datenwürfeln verarbeiten. Ausgangsports dagegen enthalten stets genau einen Datenwürfel, der an beliebig viele nachfolgende Module propagiert werden kann. Ähnlich sieht es im Falle der Parameterports aus; auf die Behandlung mehrerer in einen Parameterport eingehender Parameterwerte, z. B. durch Vereinigung oder boolesche Verknüpfung von Wertemengen aus mehreren Parameterkanälen, wird in Abschnitt 5.4 noch näher eingegangen.

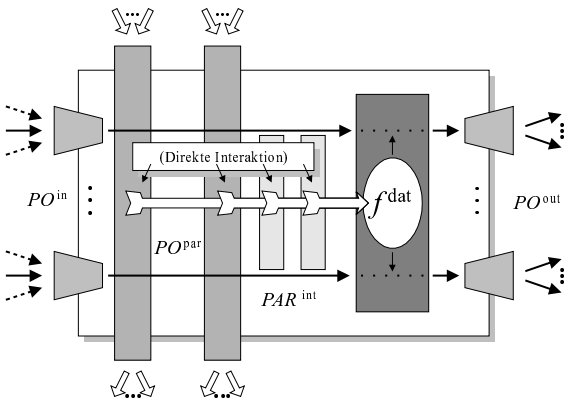

Abbildung 5.2: Grundlegender Aufbau von Modulen, den Bausteinen von *VIOLA*

Abgesehen von Datenquellen und bestimmten Datensenken (die keine Ein- bzw. Ausgangsports aufweisen) existieren in *VIOLA* noch zwei Arten gegenüber Abb. 5.2 anders aufgebauter Module, die in Abb. 5.3 dargestellt sind. Links sieht man ein Beispiel für die Parameterberechnung aus einem eingehenden Datenraum[14] und anderen Parametern (vgl. Abschnitt 5.2.4). Man beachte, daß der Wert des automatisch definierten Parameter(port)s nicht direkt aus anderen Modulen importiert, wohl aber nachträglich manuell modifiziert werden

[14] Hier wären Mengen von Datenräumen im Dateneingang kaum sinnvoll nutzbar.

kann. Prinzipiell können auch mehrere Parameter gleichzeitig in einem Modul berechnet werden, es soll jedoch in keinem Fall ein berechneter Parameter in die Definition eines anderen (im gleichen Modul) eingehen. In einem Modul zur Parameterberechnung ist — da pro Parameter nur *ein* Parameterport zur Aufnahme des Ergebnisses zur Verfügung steht — offenbar keine parallele Verarbeitung mehrerer unabhängiger Dateneingänge möglich, so daß hier maximal ein Datenportpaar vorgesehen ist. Die Funktion f^{dat} reicht den Eingangsdatenraum unverändert zum Ausgang weiter.[15] Bestimmte Module zur Parameterdefinition kommen auch ganz ohne Datenports aus.

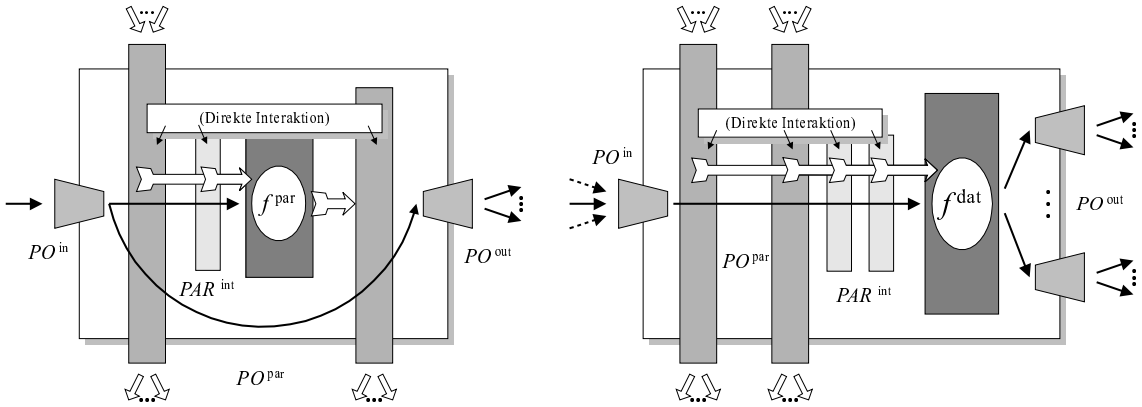

Abbildung 5.3: Typen speziell strukturierter *VIOLA*–Module

Auf der rechten Seite von Abb. 5.3 findet sich ein Beispiel für eine Abweichung von der „parallelen“ Verarbeitung von Eingängen zu korrespondierenden Ausgängen. In bestimmten Kontrollstrukturen (konkret bei der bedingten Verzweigung, vgl. Abschnitt 5.2.5) dient ein Eingang zur Definition *mehrerer* Ausgänge. Während diese Besonderheit unverzichtbar für die Basisfunktionalität von *VIOLA* ist, gibt es jedoch kein Modul, in dem *mehrere* Dateneingangsports in die Berechnung *eines* Ausgangsports einfließen, so daß eine Verwässerung der Grundidee der unabhängigen Verarbeitung von Eingängen vermieden wird.

Die Grundidee, die hinter einer derartigen Konzeption der Module von *VIOLA* steht, besteht in einer verbesserten Unterstützung explorativer, vergleichender Analysepfade, indem mit einer einzigen Parameteränderung in einem Modul auf einfache Weise mehrere zusammengehörende Auswertungen auf verschiedenen (Teil–)Datenräumen gleichermaßen manipuliert werden können. Die Vergleichbarkeit von Analyseergebnissen bleibt automatisch erhalten. Die konkrete Anzahl von Datenportpaaren derartiger Module ist stets beliebig, sie kann sogar problemlos zur Laufzeit erhöht (oder eingeschränkt) werden.

Ein Nachteil dieses Ansatzes ist sicherlich darin zu sehen, daß mehrstellige Funktionen nicht direkt über klare Zuordnungen der Funktionsargumente zu verschiedenen Eingangsports unterstützt werden können. Diese Schwäche wiegt aber nicht so schwer, da aufgrund der differenzierten Metadatenbeschreibung — insbesondere im Hinblick auf Rollen von Objektmengen sowie Arten von Maßzahlen — eine automatische Zuordnung verschiedener summarischer Attribute *eines* Datenraums zu den jeweiligen Funktionsargumenten leicht möglich ist. Zumal alle verarbeiteten Daten multidimensionale Datenräume sind, die sich einheitlich beschreiben und leicht vereinigen lassen, lassen sich Programme in *VIOLA* vielmehr mit dem hier verfolgten Ansatz deutlich einfacher erzeugen und übersichtlicher gestalten.

Als eine mögliche Designalternative sei noch die visuelle Replikation von Modulen eines Programms erwähnt, wie sie in [KMK95] verfolgt wird: Ein einzelnes Modul hat mehrere Repräsentationen in der visuellen Darstellung eines Datenflußprogramms in einem entsprechenden Editor. Somit kann auch hier eine lokale Änderung auf mehrere Datenflüsse synchron angewendet werden. Jedoch sind entsprechende versteckte Zusammenhänge nur schwer darzustellen bzw. zu erfassen, so daß der *VIOLA*–Ansatz eine höhere Klarheit der

[15] Im Sinne einer flexiblen Mehrfachverwendung von Daten wird auch hier nicht auf den Ausgangsport verzichtet.

Programmdarstellung verspricht. In die gleiche Richtung zielt auch die Beschränkung auf *einzelne* Datenräume als Eingangsdaten eines Moduls bzw. die explizite Zusammenfassung von Datenraummengen in Vereinigungsoperationen.

5.2.2 Umsetzung der Operatoren von *MADEIRA*

Die wesentliche Verarbeitungsfunktionalität erhält *VIOLA* aus der Umsetzung der mit *MADEIRA* definierten Algebra auf multidimensionalen Daten. Durch eine relativ kleine Menge parametrisierbarer Operatortypen läßt sich ein sehr breites Spektrum an Datenmanagement- und Analyseverfahren abdecken.

Bevor jedoch eine Definition der entsprechenden *VIOLA*–Module erfolgen kann, sind zunächst einige leichte Anpassungen an den durch *MADEIRA* definierten Funktionen vorzunehmen. Während nämlich in Abschnitt 4.3 die grundlegende Zielsetzung und Funktionalität der Operatoren sowie deren Definition in einer möglichst einfachen, elementaren und allgemeinen Form im Vordergrund stand sowie eine eher *prozedurale* Betrachtungsweise verfolgt wurde, ist nun in *VIOLA* eine Abstimmung auf die datenflußbasierte Sicht erforderlich.[16] Insbesondere soll *VIOLA* — soweit das in Umsetzung des Datenflußmodells möglich ist — eher einen *deklarativen* Charakter haben, d. h. durch Module und ihre Parameter werden Charakteristika und Komponenten spezifiziert, die die jeweiligen Ergebnisdatenräume aufweisen sollen. Somit kann die Gestaltung eines *VIOLA*–Programms weitgehend unabhängig von den konkret verarbeiteten Daten erfolgen, und verschiedene Datensätze sind flexibel durch die gleichen Verfahrensfolgen bearbeitbar. Ein Ziel ist es auch, möglichst große Freiheiten bei der Umstrukturierung und Modifikation eines vorliegenden Analyseablaufs zu gewähren und eine intuitive Spezifikation von Operationsparametern ebenso wie von typischen Kombinationen elementarer Operationen zu ermöglichen.

Gemäß dieser Ansprüche sind vor allem kategorielle und summarische Attribute, die meist spezifisch für einen konkreten Datenraum festgelegt sind, als Parameter durch Angaben zu ihren Komponenten zu ersetzen. Hierbei können auch Abstriche in der Mächtigkeit von Operatoren zugunsten einfacherer Nutzbarkeit sinnvoll sein. Elementare Verarbeitungsschritte werden — je nach verarbeitetem Datenraum — zu gemeinsam auszuführenden Gruppen kombiniert, die eine bestimmte Gestalt des Zieldatenraums generieren. In diesem Kontext bildet folgende Definition die Grundlage zur Auswahl von Mengen für bestimmte Operationen interessierender summarischer bzw. kategorieller Attribute anhand ihrer Komponenten:

Definition 5.6 (Auswahl kategorieller und summarischer Attributmengen in *VIOLA*) *Sei*

$$\mathcal{KA}_{EOR,K} \stackrel{\text{def}}{=} \{ka \in \mathcal{KA} \mid ka.EOR \cap EOR \neq \emptyset \wedge ka.K \cap K \neq \emptyset\}$$

die Menge durch eine Menge $EOR \subseteq \mathcal{E} \times \mathcal{OR}$ von Eigenschaften zu rollenbehafteten Objektmengen und eine Menge $K \subseteq \mathcal{K}$ von Kategorien spezifizierter kategorieller Attribute sowie

$$\mathcal{SA}_{MZ,OR} \stackrel{\text{def}}{=} \{sa \in \mathcal{SA} \mid (\exists mz \in MZ : sa.mz \preceq mz) \wedge \forall r \in \mathcal{R} : ((\exists O' \subseteq O : (O', r) \in OR \wedge \exists O \subseteq O : (O, r) \in sa.OR) \Rightarrow \exists \widetilde{O} \subseteq O : (\widetilde{O}, r) \in OR \cap sa.OR)\}$$

die Menge durch eine Menge $MZ \subseteq \mathcal{M}$ von Maßzahlen sowie eine Menge $OR \subseteq \mathcal{OR}$ rollenbehafteter Objektmengen spezifizierter summarischer Attribute.

In der durch diese Definition vorgegebenen Form werden verschiedene *VIOLA*–Operatoren jeweils Attribute bzw. hierüber (Teil–)Datenräume aus einem oder mehreren Operationsargumenten identifizieren, die speziell bearbeitet werden sollen. Sicherlich wären prinzipiell auch alternative Formen der Spezifikation von Attributmengen (Disjunktion der Teilbedingungen, Teilmengenbeziehungen anstelle von Durchschnittsbetrachtungen

[16] Natürlich wäre es auch möglich gewesen, *MADEIRA* gleich *exakt* passend für *VIOLA* zu definieren. In der hier gewählten Form bietet sich jedoch aufgrund der allgemeineren Definition eine breitere Nutzbarkeit von *MADEIRA* auch außerhalb von *VIOLA*. Zudem ermöglicht der zweistufige Definitionsansatz einen einfacheren, schrittweisen Einstieg in die Konzeption von *VIOLA*.

etc.) und mächtigere Kombinationen von Selektionsprädikaten möglich. Aber da Def. 5.6 nicht als alleinige Basis einer freien Anfragesprache für Datenräume zu sehen ist — es werden nur Quelldaten eingegrenzt, nicht aber direkt Berechnungsergebnisse beschrieben — ist der gewählte Weg für unsere Zwecke völlig ausreichend. Konkrete Beispiele zur sinnvollen Nutzung obiger Definition werden die Operatordefinitionen in diesem Abschnitt bringen.

Die Spezifikation kategorieller Attribute ist recht intuitiv: Alle Attribute, die eine der gegebenen Eigenschaften beschreiben und mindestens eine der vorgegebenen Kategorien enthalten, werden ausgewählt.[17] Dagegen soll die Vorgabe einer Menge $OR \subseteq \mathcal{OR}$ zur Wahl summarischer Attribute insofern *rollenbezogen* interpretiert werden, als ein summarisches Attribut *sa* des Quelldatenraums — um selektiert zu werden — zu *jeder* in *OR* auftretenden Rolle *r*, die auch für *sa* von Bedeutung ist, eine der in *OR* zusammen mit *r* angeführten Objektmengen beschreiben muß. Eine zu den kategoriellen Attributen analoge Selektionsdefinition würde dagegen nicht in jedem Fall alle vorgegebenen Rollen berücksichtigen. Mit anderen Worten: *OR* gibt für bestimmte Rollen Objektmengen als Kandidaten vor, z. B. können mittels $OR = \{(Personen, Standard), (Personen, Studie)\}$ standardisierte Raten, deren Standard- *und* Studienpopulation sich auf Personen (und nicht etwa Tumoren) beziehen, ausgewählt werden. Diese Unterscheidung in der Behandlung summarischer und kategorieller Attribute beruht auf der unterschiedlichen Bedeutung, die Rollen jeweils als deren Komponenten aufweisen: Während für kategorielle Attribute Eigenschaften, Rollen und Objektmengen als gekapselte Einheit anzusehen sind und beliebig viele derartige Tripel zur Attributbeschreibung dienen können, enthält die Definition eines summarischen Attributs je nach Maßzahl stets eine festgelegte Menge bestimmter Rollen, die quasi durch Objektmengen belegt werden.

Mehrdeutigkeiten, wie sie bereits einige der in Abschnitt 4.3 eingeführten Operatoren zulassen, werden in *VIOLA* in der Regel nicht aufgelöst. Eher im Gegenteil: Mit dem Streben nach allgemein anwendbaren Operatoren werden jeweils quasi Mengen möglicherer Zieldatenräume spezifiziert, so daß u. U. auch bei einem konkret zu betrachtenden Eingangsdatenraum mehrere Funktionsergebnisse zur Auswahl stehen. Dies soll jedoch nicht weiter stören, da derartige Situationen in typischen Datenanalysen die große Ausnahme bilden und oft auch ein Zeichen von nicht sorgfältig spezifizierten Analyseprogrammen bzw. nicht sinnvoll zusammengestellten (Teil–)Datenräumen darstellen. Der Anwender kann ggf. auf Mehrdeutigkeiten hingewiesen werden und diese selbst auflösen. Ohne Eingriff des Nutzers kann eine beliebige Variante automatisch gewählt werden, was auch in einigen Fällen völlig ausreichend sein kann. Insgesamt erscheinen die Nachteile mehrdeutiger Operationen vernachlässigbar gegenüber der hohen Flexibilität, die sie bei der Analyse gewähren.

Natürlich ist die konkrete Umsetzung der Parametrisierung der Operatoren von *MADEIRA* im Einzelfall diskussionswürdig und bedarf einer umfassenden Evaluation. Zentrales Ziel ist stets der höchstmögliche Benutzungskomfort für die Fachanwender.

Anpassung der Operatoren an die interaktive Nutzung in *VIOLA*

Im folgenden wird nun ein Großteil der in *MADEIRA* definierten Verarbeitungsfunktionen durch jeweils gleichnamige (quasi polymorphe) Funktionen mit an die Nutzung in *VIOLA* angepaßten Parametersätzen und gleicher oder zumindest sehr ähnlicher Semantik ersetzt. Die Definitionen der neuen Operatoren sind jeweils unter explizitem Rückgriff auf ihre *MADEIRA*–Pendants formuliert. Anhand der Funktionssignaturen ist jeweils eine eindeutige Unterscheidung zwischen *MADEIRA*– und *VIOLA*–Operator möglich.

Die Umbenennung von Rollen mittels f^{role} sowie die Berechnung neuer Maßzahlen durch zellenweise Verarbeitung oder Maßzahlverknüpfung mittels f^{cell} und f^{join} brauchen nicht verändert zu werden. Auf die Behandlung von Mikrodaten im Rahmen ihrer Konsolidierung zu Makrodaten sowie eines Drill–through wird weiter unten bei der Diskussion von Datenquellen sowie im Kontext der Datenvisualisierung (Abschnitt 5.2.3) gesondert eingegangen.

[17] Zum Beispiel wäre eine typische Selektion die aller Attribute zum Neuerkrankungsalter von Personen der Studienpopulation, die eine Altersgruppe zwischen 10 und 40 beinhalten.

Ableitung Die Ableitung bildet die zentrale Operation im Umgang mit multidimensionalen Daten (vgl. Abschnitt 4.3). Aus diesem Grund wird sie hier besonders sorgfältig behandelt.

Wie bereits in Def. 4.31 in Abschnitt 4.3.2 mit der Unterscheidung von Restriktion und impliziter Aggregierung angedeutet, kann die Ableitung eines Datenraums in unterschiedlichen Varianten mit jeweils anderer Zielsetzung erfolgen. Der zentrale Ableitungsoperator, wie er in Kapitel 4 definiert wurde, bietet die Basis für eine einfache und flexibel erweiterbare Implementierung des Datenmanagements, wobei spezifische Varianten über die Art der Festlegung der Zielkategorien bzw. des kategoriellen Zielattributs ka' modelliert werden können. Folgende Definition führt hierzu sechs verschiedene Möglichkeiten ein:

Definition 5.7 (Varianten der Ableitung) *Sei $dr \in \mathcal{DR}$ ein zu betrachtender Datenraum sowie $ka \in dr.KA$ ein kategorielles Attribut über der Domäne $D = ka.D$, über das ein Ableitungsschritt zu einem Zielattribut $ka' \in \mathcal{KA}$ (das die gleichen Eigenschaften wie ka beschreibt) erfolgen soll.*

Die Spezifikation der Kategorienmenge von ka' erfolge stets über eine der folgenden sechs Varianten, wobei (außer in Variante 1) eine nicht–leere, endliche Menge $\widetilde{K} \subseteq D$ von Kategorien vorzugeben ist. Es gelte jeweils $ka'.K \stackrel{\text{def}}{=}$

1. *$\bigvee ka.K$ (vollständige (implizite) Aggregierung zu einem Gesamtwert).*
2. *$\widetilde{K}$ (die Ableitung, so wie sie in Def. 4.31 eingeführt wurde).*
3. *$\{k' \in \widetilde{K} \mid k' \text{ ableitbar aus } ka.K\}$ (Ableitung, wobei nur Kategorien aus $\widetilde{K}$ gefordert werden, die auch tatsächlich (disjunkt)[18] ableitbar sind — häufig wird man hier etwa als $\widetilde{K}$ alle Kategorien einer Aggregierungsebene vorgeben).*
4. *$ka.K \cap \widetilde{K}$ (nur Restriktion, keine Aggregierungen).*
5. *$ka.K \cap \{k \in D \mid k \preceq \bigvee \widetilde{K}\}$ (Restriktion auf Kategorien, die von $\widetilde{K}$ subsumiert werden).*
6. *$\{\bigvee \{k \in ka.K \mid k \preceq k'\} \mid k' \in \widetilde{K}\}$ (im Gegensatz zu (2) unvollständige Gruppierung von Kategorien — quasi eine Verallgemeinerung der Hintereinanderausführung von (5) und (1) auf* mehrere *Gruppen, die durch $\widetilde{K}$ beschrieben werden).*

Abbildung 5.4 faßt die vorgestellten Ableitungstypen anhand eines Beispiels im Rahmen einer datenflußbasierten visuellen Anfrage zusammen. Hierbei repräsentiere der Sozialstatus keine eigenständige Dimension, sondern eine Aggregierungsebene auf der Gebietsdomäne, die verschieden „reiche“ Regionen (Gebietsgruppen) vorgibt.

Wie man sieht, beeinflußt bei typischen Anwendungen der Ableitung auch der jeweils betrachtete Datenraum *dr* bzw. dessen für die Ableitung relevantes Attribut *ka* die Durchführung dieser Operation: Es werden nicht einfach (nur) *Ziel*kategorien vorgegeben, zu denen Werte berechnet werden sollen, sondern oftmals wird vielmehr ein Vorgehen spezifiziert, wie die in *ka* vorhandenen *Quell*kategorien ausgewählt und zusammengefaßt werden sollen.

Weitere Spezialfälle der aufgeführten grundlegenden Ableitungstypen sind sicherlich denkbar, z. B. könnten in den Varianten 2 und 3 aus Def. 5.7 auch in *dr* bzw. *ka* vorhandene Kategorien, die ihrerseits aus den Kategorien in $\widetilde{K}$ ableitbar sind, bei der Ableitung im Zieldatenraum erhalten bleiben (im Sinne „Aggregiere zu diesen Kategorien oder gröber ...“). Die Nutzung derartiger Spezialfälle dürfte jedoch oftmals auch anwendungsspezifisch erfolgen. Um den Benutzer von *VIOLA* nicht mit vielen Sonderfällen zu verwirren, werden diese Fälle Gegenstand möglicher Systemerweiterungen bzw. Benutzerkonfigurationen bleiben.

Die Spezifikation eines Ableitungsschrittes erfolgt nun auf der Basis obiger Definition von Ableitungsvarianten über Eigenschaften und Kategorienmengen, indem über eine Menge $EOR \subseteq \mathcal{E} \times \mathcal{OR}$ — wie in Def. 5.6 dargestellt — interessierende kategorielle Attribute eines Quelldatenraums (zu mindestens einer dieser Eigenschaften) identifiziert und über eine Menge von Kategorien — im Zusammenhang mit der Auswahl einer Ableitungsvariante — die gewünschten Zielkategorien festgelegt werden. Durch diese Art der Angabe eines Zielattributs können natürlich u. U. jeweils gleich mehrere „passende“ Quellattribute zur Durchführung der

[18] Disjunktheit ist gefordert, falls mindestens ein summarisches Attribut *sa* aus *dr* nur disjunkt bzgl. *e* und *O* aggregierbar ist.

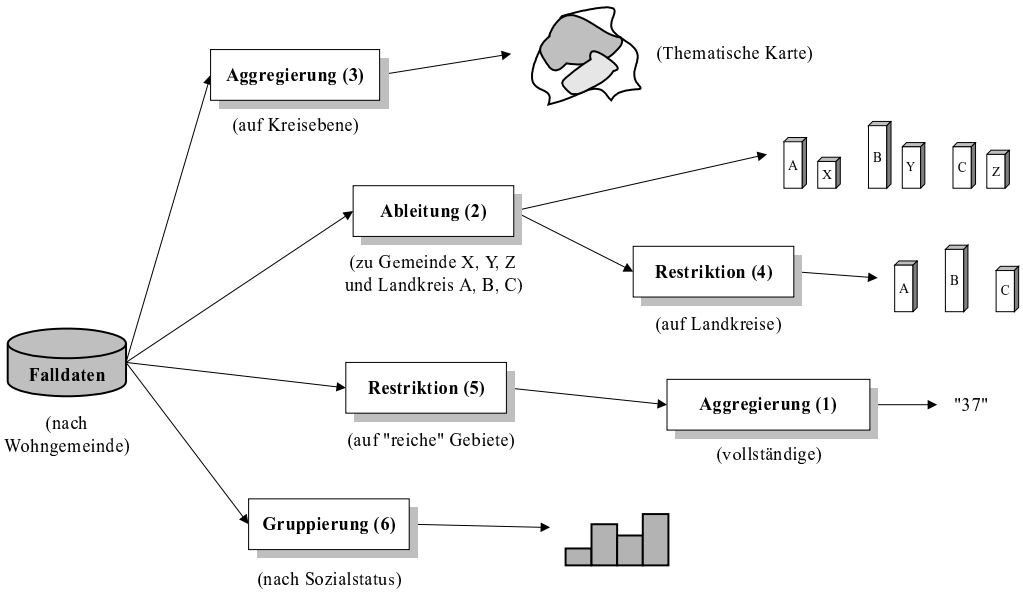

Abbildung 5.4: Beispiele für Ableitungsschritte in visuellen Anfragen

Ableitung selektiert werden. Dies berücksichtigt die folgende Umsetzung der Ableitung in *VIOLA*, wobei die Variante 1 aus Def. 5.7 aufgrund ihrer etwas abweichenden Parametrisierung gesondert behandelt wird:

Definition 5.8 (Ableitung in *VIOLA*) *Die* ABLEITUNG *von Datenräumen sei in VIOLA umgesetzt durch*

1. *eine (partielle) Abbildung* $f^{\text{abl}} : \mathcal{DR} \times \widetilde{2^{\mathcal{K}}} \times \widetilde{2^{\mathcal{E}\times\mathcal{OR}}} \times \mathbb{N} \to \mathcal{DR}$*, die mehrere Ableitungsschritte (gemäß Def. 4.31) über verschiedene kategorielle Attribute zu einer Domäne auf der Grundlage einer Ableitungsvariante nach Def. 5.7 zusammenfaßt.*

   Es seien $dr \in \mathcal{DR}, \emptyset \neq \widetilde{K} \subseteq \mathcal{K}, \emptyset \neq EOR \subseteq \mathcal{E} \times \mathcal{OR}$ *sowie* $D \in \mathcal{DO}$*, so daß* $\forall k \in \widetilde{K} : D_k = D$ *und* $\forall (e, O, \cdot) \in EOR : D_e = D \wedge O \subseteq O_e$*.* $\widetilde{K}$ *sei endlich. Ferner diene* $i \in \{2, \ldots, 6\}$ *zur Spezifikation einer Ableitungsvariante.*

   Sei $KA \stackrel{\text{def}}{=} dr.KA \cap \mathcal{KA}_{EOR,\mathcal{K}} = \{ka_1, \ldots, ka_m\}$ *die Menge der von der Ableitung betroffenen Attribute aus* dr *(mit beliebigen Kategorien aus* $\mathcal{K}$*) sowie — zu einem Attribut* $ka_j \in KA$ *—* $ka'_j \stackrel{\text{def}}{=} (D, K'_j, ka_j.EOR)$ *jeweils das gemäß Def. 5.7 über* $\widetilde{K}$ *spezifizierte Zielattribut.*

   Falls $KA \neq \emptyset$ *und alle* $K'_j \neq \emptyset$*, so sei*

   $$f^{\text{abl}}(dr, K, EOR, i) \stackrel{\text{def}}{=} f^{\text{abl}}(\ldots(f^{\text{abl}}(f^{\text{abl}}(dr, ka_1, ka'_1), ka_2, ka'_2), \ldots,), ka_m, ka'_m) \quad ,$$

   andernfalls sei das Ergebnis der Ableitung undefiniert.

2. *eine (partielle) Abbildung* $f^{\text{abl}} : \mathcal{DR} \times 2^{\mathcal{E}\times\mathcal{OR}} \to \mathcal{DR}$*, die über eine gewählte Menge von Eigenschaften (über beliebigen Domänen) jeweils vollständig aggregiert — dies entspricht also einer wiederholten Anwendung der Variante 1 aus Def. 5.7.*

   Seien wiederum $dr \in \mathcal{DR}, \emptyset \neq EOR \subseteq \mathcal{E} \times \mathcal{OR}$ *sowie KA wie oben und jeweils* $ka'_j = (ka_j, \bigvee ka_j.K, ka_j.EOR)$ *das Zielattribut. Dann sei*

   $$f^{\text{abl}}(dr, EOR) \stackrel{\text{def}}{=} \begin{cases} f^{\text{abl}}(\ldots(f^{\text{abl}}(dr, ka_1, ka'_1), \ldots,), ka_m, ka'_m) & \text{falls } KA \neq \emptyset, \\ dr & \text{sonst.} \end{cases}$$

Letztere Variante dient quasi der „Elimination“ von Dimensionen eines Datenraums („Betrachte Fallzahlen ohne Differenzierung nach Alter und Geschlecht.“). Da eine derartige Verarbeitung oftmals mehrere Domänen gleichzeitig betrifft, wird mit obiger Definition eine entsprechende gemeinsame Verarbeitung in *einem* Schritt unterstützt.[19] Vor dem Hintergrund dieser Betrachtungsweise wird es auch verständlich, daß bei der vollständigen Aggregierung — im Gegensatz zur ersten Variante, wo die Erzeugung bestimmter Zielkategorien im Vordergrund steht — Datenräume, die keine interessierenden kategoriellen Attribute aufweisen, unverändert bleiben: Die betroffenen Dimensionen sind quasi bereits eliminiert.

Sind gleichzeitig mehrere kategorielle Attribute von einem Ableitungsschritt betroffen, werden in beiden Varianten aus obiger Definition die Zellwerte des Ausgangsdatenraums sukzessive über die jeweils gruppierten Kategorien dieser Attribute aggregiert. Hierbei sieht man sehr schön, daß alle Aggregierungsfunktionen eines summarischen Attributs generell untereinander kommutativ sein müssen, da die Reihenfolge der Aggregierung über unterschiedliche Attribute offenbar keine Rolle spielen darf.

Ein weiterer Aspekt, der einer expliziten Erwähnung bedarf, betrifft den Zusammenhang zwischen dem Parameter *EOR* der Ableitung und den summarischen Attributen des verarbeiteten Datenraums im Falle einer (impliziten) Aggregierung. Dieser Parameter dient allein zur Selektion interessierender kategorieller Attribute *KA*. Stets werden Werte *aller* summarischen Attribute des Datenraums aggregiert — auch derer, die andere Objektmengen beschreiben, als durch *EOR* vorgegeben ist. Ein derartiges Vorgehen ist durchaus sinnvoll, da man durch die Ableitungsoperation nicht die Gestalt des Datenraums, also Anzahl von Dimensionen und Menge summarischer Attribute, bzw. die Entsprechungen zwischen verschiedenen Maßzahlwerten einer Zelle verändern möchte. So beschreibt etwa in einem Datenraum mit Fall- und Bevölkerungszahlen das Zeitattribut sowohl die Erkrankungszeitpunkte zur Objektmenge *Tumor* als auch den Erhebungszeitpunkt zur Objektmenge *Person*. Bei einer Aggregierung der Falldaten über Erkrankungszeitpunkte möchte man sicherlich automatisch auch die Bevölkerungsdaten gleichermaßen zusammenfassen. Eine entsprechend korrekte Verarbeitung garantiert die auf kategorielle Attribute bezogene Festlegung der Aggregierungsfunktionen summarischer Attribute in einem Datenraum gemäß Def. 4.23.

Selektion summarischer Attribute Analog zu den kategoriellen Attributen im Fall des Ableitungsoperators erfolgt auch die Auswahl summarischer Attribute nicht direkt, sondern über geeignete Attributkomponenten, nämlich — entsprechend Def. 5.6 — über Maßzahlen und rollenbehaftete Objektmengen.

Definition 5.9 (Selektion summarischer Attribute in *VIOLA*) *Die* SELEKTION *summarischer Attribute realisiere eine Abbildung* $f^{\text{sel}}\colon \mathcal{DR} \times 2^{\mathcal{M}} \times 2^{\mathcal{OR}} \to \mathcal{DR}$.

Zu $dr \in \mathcal{DR}$, $\emptyset \neq MZ \subseteq \mathcal{M}$ *und* $OR \subseteq \mathcal{OR}$ *sei* $f^{\text{sel}}(dr, MZ, OR) \stackrel{\text{def}}{=} f^{\text{sel}}(dr, \mathcal{SA}_{MZ,OR})$ *(vgl. Def. 4.32 in Abschnitt 4.3.3).*

Während eine leere Maßzahlmenge offensichtlich keine sinnvolle Parametrisierung darstellt, selektieren leere Mengen $\mathcal{OR}$ alle vorhandenen Attribute (zur jeweiligen Maßzahl). Umgekehrt sind sinnvolle Anwendungen für die Auswahl beliebiger Maßzahlen zu bestimmten Objektmengen und Rollen nur schwer vorstellbar, weshalb auf eine entsprechende Erweiterung der Definition verzichtet wird.

Vereinigung Die Vereinigung von Datenräumen basiert gemäß Abschnitt 4.3.4 auf einer jeweils dynamisch definierten Klassifizierung der kategoriellen Attribute der betrachteten Datenräume, die vorgibt, welche Attribute miteinander zu identifizieren sind. Wiederum wäre ein derartiger Parameter in *VIOLA* zu spezifisch auf konkrete Datenräume ausgelegt und andererseits auch zu aufwendig zu spezifizieren. Aus diesem Grund erfolgt eine Einschränkung der Mächtigkeit des Vereinigungsoperators auf eine einfachere, für typische Analysen völlig ausreichende[20] Variante: Es werden Domänen vorgegeben, über die *alle* Attribute verschiedener

[19] Hier ist also die gleichzeitige Aggregierung über mehrere kategorielle Attribute durchaus die Regel, während dies in anderen Varianten der Ableitung eher eine Ausnahme darstellt.

[20] Gegebenenfalls sind mehrere *VIOLA*–Vereinigungsoperationen in Folge auszuführen, um auch komplexere Spezialfälle umsetzen zu können.

Datenräume einander zuzuordnen sind. Da sich in sehr vielen Anwendungsfällen jedes kategorielle Attribut eines Datenraums auf eine andere Domäne bezieht, erfolgt ohnehin eine Vereinigung von Datenräumen sehr oft unter Verschmelzung *aller* kategoriellen Attribute zur jeweils gleichen Domäne. Zum Beispiel finden sich in der Epidemiologie alle Altersangaben (egal ob zum Erkrankungsalter, zum Sterbealter oder zur Beschreibung von Bevölkerungszahlen; zur Studien- genau wie zur Standardpopulation) meist gemeinsam in einem eindeutigen Altersattribut eines Datenraums.

Anders ist die Sachlage z. B. in der Gebiets- oder der Zeitdimension. Hier sind Beschreibungen von Studien- und Standardpopulation oftmals voneinander zu trennen und in jeweils eigenen kategoriellen Attributen zu spezifizieren. Deshalb sollen — in Ergänzung zur Verwendung von Domänen als Vereinigungsparameter — durch die Angabe von Eigenschaften zu rollenbehafteten Objektmengen einzelne Attribute auch von der Verschmelzung ausgeschlossen werden können (etwa: „Identifiziere alle Gebietsangaben mit Ausnahme der Angaben zu einer Standardpopulation.“). Unter Berücksichtigung der Vereinigungsregeln aus Def. 4.33 ergibt sich folgendes Verfahren:

Definition 5.10 (Datenraumvereinigung in *VIOLA*) *Die* VEREINIGUNG *von Datenräumen werde implementiert durch eine Abbildung* $f^{\cup} : 2^{\mathcal{DR}} \times 2^{\mathcal{DO}} \times 2^{\mathcal{E} \times \mathcal{OR}} \to \mathcal{DR}$. *Seien* $DR \subseteq \mathcal{DR}$, $\mathbf{D} \subseteq \mathcal{DO}$ *und* $EOR \subseteq \mathcal{E} \times \mathcal{OR}$.

Eine Partition $\mathbf{KA} \subseteq 2^{\mathcal{KA}}$ *auf den kategoriellen Attributen* $KA \stackrel{\text{def}}{=} \bigcup_{dr \in DR} dr.KA$ *gemäß Def. 4.33 werde definiert durch Anwendung des Algorithmus 5.1 auf KA,* $\mathbf{D}$ *und EOR.*

Liefert der Partitionierungsalgorithmus $\mathbf{KA} \neq \emptyset$, *so sei* $f^{\cup}(DR, \mathbf{D}, EOR) \stackrel{\text{def}}{=} f^{\cup}(DR, \mathbf{KA})$, *andernfalls ist die Vereinigung nicht möglich.*

procedure Partitioniere(KA: $2^{\mathcal{KA}}$, $\mathbf{D}$: $2^{\mathcal{DO}}$, EOR: $2^{\mathcal{E} \times \mathcal{OR}}$) **returns** $2^{2^{\mathcal{KA}}}$:
  {Initialisierung der Ergebnismenge **KA**:}
  $\mathbf{KA} \stackrel{\text{def}}{=} \{\{ka\} \mid ka \in KA\}$;
  {Unbedingt nötige Verschmelzungen von Attributen zu gleichen Eigenschaften gemäß Def. 4.33:}
  while $\exists KA_1 \neq KA_2 \in \mathbf{KA}\ \exists ka_1 \in KA_1, ka_2 \in KA_2 : ka_1.EOR \cap ka_2.EOR \neq \emptyset$ **do**
    $\widetilde{KA} \stackrel{\text{def}}{=} KA_1 \cup KA_2$;
    {Kategorielle Attribute desselben Datenraums können nicht zusammengefaßt werden:}
    if $\exists ka_1 \neq ka_2 \in \widetilde{KA}$: ka_1 und ka_2 stammen aus demselben Datenraum **then**
      return $\emptyset$; {Die Partitionierung kann nicht definiert werden.}
    end if
    $\mathbf{KA} \stackrel{\text{def}}{=} \mathbf{KA} \setminus \{KA_1, KA_2\} \cup \{\widetilde{KA}\}$;
  end while
  {Verschmelzungen gemäß der gegebenen Parameter:}
  while $\exists KA_1 \neq KA_2 \in \{\widetilde{KA} \in \mathbf{KA} \mid \forall ka \in \widetilde{KA} : ka.D \in \mathbf{D} \wedge ka.EOR \cap EOR = \emptyset\}$:
      $\forall ka_1 \neq ka_2 \in KA_1 \cup KA_2 : ka_1$ und ka_2 stammen aus verschiedenen Datenräumen $\wedge$
      $ka_1.D = ka_2.D$ **do**
    $\widetilde{KA} \stackrel{\text{def}}{=} KA_1 \cup KA_2$;
    $\mathbf{KA} \stackrel{\text{def}}{=} \mathbf{KA} \setminus \{KA_1, KA_2\} \cup \{\widetilde{KA}\}$;
  end while
  return KA; {Partitionierung erfolgreich.}
end procedure

Algorithmus 5.1: Partitionierung kategorieller Attribute zur Datenraumvereinigung

Die Konformität dieses Vorgehens mit der Definition aus Abschnitt 4.3 ist leicht zu verifizieren. Unter Umständen können in der zweiten While–Schleife des Partitionierungsalgorithmus Mehrdeutigkeiten auftreten, die zu unterschiedlichen Vereinigungsergebnissen führen, etwa wenn ein Datenraum zwei Attribute über

derselben Domäne aufweist, über die eine Verschmelzung mit einem Attribut eines weiteren Datenraums stattfinden soll.

Merge und Split Eine zu den vorangegangenen Definitionen analoge Umsetzung der Split– und Merge–Operatoren aus *MADEIRA* durch „Umwandlung“ der auf kategorielle Attribute bezogenen Parameter in Eigenschaftsspezifikationen (genauer gesagt: Mengen von Eigenschaftspaaren) würde eine Reihe Probleme aufwerfen bzw. zu einem „unschönen“ Design führen: Ähnlich wie im Fall der Vereinigung paßt die Gestalt der Eigenschafts*paare* nicht recht zu dem *VIOLA* zugrundegelegten Parameterkatalog und wäre auch eher aufwendig interaktiv zu definieren bzw. aus Datenräumen zu extrahieren. Zudem wäre die Wahrscheinlichkeit für das Auftreten kaum aufzulösender Mehrdeutigkeiten bei der Operatoranwendung relativ groß. Aus diesen Gründen und zumal diese eher seltenen Operationen im Fall ihrer Nutzung meist sehr spezifisch auf die verarbeiteten Daten abgestimmt sind, soll eine recht elementare („eher prozedurale“) Definition ausreichend sein:

Definition 5.11 (Merge und Split in *VIOLA*) *Die Behandlung zusammengesetzter kategorieller Attribute mittels* MERGE *und* SPLIT *aus Def. 4.38 und 4.39 in Abschnitt 4.3.7 werde durch zwei Funktionen* $f^{\text{merge}}: \mathcal{DR} \times \mathcal{E} \times \mathcal{OR} \times \mathcal{E} \to \mathcal{DR}$ *bzw.* $f^{\text{split}}: \mathcal{DR} \times \mathcal{E} \times \mathcal{OR} \times \mathcal{E} \to \mathcal{DR}$ *umgesetzt.*

Seien $dr \in \mathcal{DR}$, $(e,O,r) \in \mathcal{E} \times \mathcal{OR}$ *und* $e' \in \mathcal{E}$. *Damit ergebe sich*

$$f^{\text{merge}}(dr,(e,O,r),e') \stackrel{\text{def}}{=} \begin{cases} f^{\text{merge}}(dr,ka,ka') & \text{falls } \exists ka \neq ka' \in dr.KA: \\ & (e,O,r) \in ka.EOR \wedge (e',O,r) \in ka'.EOR, \\ dr & \text{sonst} \end{cases}$$

sowie

$$f^{\text{split}}(dr,(e,O,r),e') \stackrel{\text{def}}{=} \begin{cases} f^{\text{split}}(dr,ka) & \text{falls } \exists ka \in dr.KA: (e \circ e',O,r) \in ka.EOR, \\ dr & \text{sonst.} \end{cases}$$

Es werden also nicht — wie etwa im Fall der Ableitung — u. U. gleich mehrere Verarbeitungsschritte in einer Operation zusammengefaßt. Gegebenenfalls sind statt dessen durch den Anwender mehrere entsprechende *VIOLA*–Bausteine hintereinanderzuschalten. Vor dem Hintergrund dieser Option wird auch verständlich, warum Datenräume mit „unpassenden“ Attributen durch Split und Merge unverändert als Funktionsergebnis zurückgeliefert werden. Hierdurch wird ein Split auf bereits im Vorfeld schon getrennten (oder gar nicht erst zusammengeführten) Attributen eines Datenraums einfach ignoriert. In ähnlicher Weise kann ein Merge auch ohne Wirkung angewendet werden, wenn das gewünschte zusammengesetzte Attribut bereits vorliegt.[21]

Man beachte, daß die beiden kategoriellen Attribute, über die die Merge–Operation ggf. erfolgt, aufgrund der Eindeutigkeit der Zuordnung von Eigenschaften zu kategoriellen Attributen eines Datenraums (Def. 4.20) auch wiederum eindeutig sind (falls sie existieren).

Explizite Aggregierung Schließlich wird noch analog zur Implementierung der Ableitung die explizite Aggregierung betrachtet. Da hier — im Gegensatz zur impliziten Aggregierung — Verarbeitungsschritte über verschiedene Dimensionen jeweils auch unterschiedlich zu interpretieren sind, wird die oben vorgestellte zweite Variante der Ableitung (gleichzeitige Bearbeitung mehrerer Domänen) nicht umgesetzt — bei Bedarf muß der Anwender mehrere Aggregierungsoperatoren hintereinander schalten.

Definition 5.12 (Explizite Aggregierung in *VIOLA*) *Die* EXPLIZITE AGGREGIERUNG *von Datenräumen werde durch eine (partielle) Funktion* $f^{\text{eaggr}}: \mathcal{DR} \times \widetilde{2^{\mathcal{K}}} \times \widetilde{2^{\mathcal{E} \times \mathcal{OR}}} \times \mathbb{N} \times \mathcal{M} \to \mathcal{DR}$ *implementiert, die mehrere Aggregierungsschritte (gemäß Def. 4.35 aus Abschnitt 4.3.5) über verschiedene kategorielle Attribute zu einer Domäne auf der Grundlage einer Ableitungsvariante nach Def. 5.7 zusammenfaßt.*

[21] Wiederum wird — wie schon bei der zweiten Ableitungsvariante — nicht eine bestimmte Gestalt der Zieldatenräume garantiert, sondern vielmehr eine bestimmte Gestalt ausgeschlossen.

Seien dr, K', EOR sowie D wie in Def. 5.8 gegeben. Ferner sei $mz \in \mathcal{M}$ sowie $i \in \{2,3,6\}$ eine Ableitungsvariante[22].

Es seien KA sowie zu den Quellattributen ka_j die jeweiligen Zielattribute $ka'_j = (D, K'_j, ka_j.EOR)$ wie in Def. 5.8 definiert. Falls $KA \neq \emptyset$ und alle $K'_j \neq \emptyset$, so sei

$$f^{\text{eaggr}}(dr, K, EOR, i, mz) \stackrel{\text{def}}{=} f^{\text{eaggr}}(\ldots(f^{\text{eaggr}}(dr, ka_1, ka'_1, mz), \ldots,), ka_m, ka'_m, mz) \quad ,$$

andernfalls sei das Ergebnis der Aggregierung undefiniert.

Anders als bei der Ableitungsdefinition kann die nicht–eindeutige Reihenfolge der Anwendung der verschiedenen expliziten Aggregierungsschritte (insbesondere für nicht (semi-)additive Aggregierungsfunktionen) einen Einfluß auf das Operationsergebnis haben. So könnten etwa über einen Datensatz mit regionenbezogenen standardisierten Erkrankungsraten für jeweils eine Reihe von Standardpopulationen (die ebenfalls über der Gebietsdomäne codiert sind), nacheinander Durchschnitte über beide raumbezogenen Attribute unter Gruppierung auf einer höheren Aggregierungsebene gebildet werden. Derartige Mehrdeutigkeiten sollten vom jeweiligen Datenanalysten von vornherein durch geeignete Datenwahl oder Parametrisierung vermieden werden. Insgesamt gesehen dürften die meisten expliziten Aggregierungen jeweils *vollständig* über gesamte Dimensionen aggregieren, wofür der — gerade im Hinblick auf die Kombination kategorieller Attribute über verschiedenen Domänen — flexiblere allgemeine Berechnungsoperator f^{join} angewendet werden kann.

Unter Verwendung der in diesem Abschnitt gegebenen Definitionen können im folgenden nun die jeweiligen *VIOLA*–Module eingeführt werden.

Datenquellen

Datenquellmodule generieren anhand von Parameterangaben, aber ohne einfließende multidimensionale Daten im jeweiligen Kontext „neue“ Datenräume. Sie haben dementsprechend keine Dateneingänge und jeweils genau einen Datenausgang. Es werden vier Fälle unterschieden:

1. die Bereitstellung persistenter Makrodaten aus einer Datenbank (einer *physischen*, durch ein DBMS verwalteten Datenquelle) oder die dynamische Simulation[23] entsprechender Daten,
2. die Generierung multidimensionaler Datenräume aus persistenten Mikrodatenräumen (entsprechend dem Konsolidierungsoperator von *MADEIRA*, siehe Def. 4.29 in Abschnitt 4.3.1),
3. die Realisierung von Anknüpfungspunkten innerhalb der von Verbundmodulen gekapselten Teilgraphen, die den lesenden Bezug zur externen Umgebung des Verbundes (also zu den an dessen Eingangsports „als Argumente übergebenen“ Datenräumen) herstellen (vgl. Abschnitt 5.3.2), und
4. die Erzeugung von Datenräumen durch metadatenbasierte Berechnung von Maßzahlwerten aus Kategorien (vgl. Def. 4.45 in Abschnitt 4.4.3).

Die beiden erstgenannten Arten von Datenquellen benötigen in den meisten Fällen keine externen Parameter. Eingangsmodule von Verbunden verwenden gemäß Def. 5.6 eine Reihe von Parametern zur Auswahl des jeweils „richtigen“ Datenraums aus der Menge an einem Port des Verbundmoduls anliegender Räume. Außerdem können Eingänge als optional gekennzeichnet werden, wenn der Verbund seine Berechnungen auch ohne die entsprechenden Daten sinnvoll durchführen kann. Für die vierte Art Datenquellmodul schließlich sind natürlich Maßzahl und Kategorien als Parameter anzugeben.

[22] Reine Restriktionen sind hier im Gegensatz zur allgemeinen Ableitungsdefinition sinnlos; und die vollständige Aggregierung erfolgt besser mit dem allgemeinen Maßzahlberechnungsoperator f^{join}.

[23] Ein Anwendungsbeispiel für die Simulation von Makrodaten als Grundlage der Abschätzung der statistischen Signifikanz von beobachteten Auffälligkeiten im „echten“ Datenbestand wird in Abschnitt 6.3 gegeben werden.

Definition 5.13 (Datenquellmodule) *$\mathcal{MOD}^{\text{mak}}$ sei die Menge aller* MAKRODATENQUELLEN *und $\mathcal{MOD}^{\text{mik}}$ die aller* MIKRODATENQUELLEN. *Ein Datenquellmodul $mod \in \mathcal{MOD}^{\text{mak}} \cup \mathcal{MOD}^{\text{mik}}$ sei definiert als ein Quadrupel $(b, po^{\text{out}}, PO^{\text{par}}, PAR^{\text{int}}, f^{\text{dat}})$, wobei f^{dat} allein über die internen und externen Parameter die Belegung von po^{out} definiert.*

$\mathcal{MOD}^{\text{in}}$ sei die Menge aller EINGANGSMODULE *von Verbundmodulen. Ein Eingang $mod^{\text{in}} \in \mathcal{MOD}^{\text{in}}$ sei definiert als ein Quadrupel $(b, po^{\text{out}}, PO^{\text{par}}, par^{\text{int}}, mod^{\text{sub}})$, wobei*

- *$PO^{\text{par}} = \{$eigenschaften: $2^{\mathcal{E}\times\mathcal{OR}}$, kategorien: $\mathcal{K}$, objektrollen: $2^{\mathcal{OR}}$, masszahlen: $2^{\mathcal{M}}\}$,*
- *$par^{\text{int}} = (optional?, \mathbb{B}, mod^{\text{in}})$ ein boolesches Flag zur Kennzeichnung optionaler Eingänge und*
- *$mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}}$ das zugehörige Verbundmodul.*

$\mathcal{MOD}^{\text{attr}}$ sei die Menge aller KATEGORIELLEN DATENQUELLEN. *Ein (kategorielles) Datenquellmodul $mod \in \mathcal{MOD}^{\text{attr}}$ sei definiert als ein Quadrupel $(b, po^{\text{out}}, PO^{\text{par}}, f^{\text{dat}})$, wobei*

- *$PO^{\text{par}} = \{$kategorien: $\widetilde{2^{\mathcal{K}}}$, rolle: $\mathcal{R}$, mass: $\mathcal{M}\}$ und*
- *$f^{\text{cube}}(kategorien, rolle, mass)$ Funktionswert von f^{dat}.*

Die Menge aller Datenquellmodule sei $\mathcal{MOD}^{\text{src}} = \mathcal{MOD}^{\text{mak}} \cup \mathcal{MOD}^{\text{mik}} \cup \mathcal{MOD}^{\text{in}} \cup \mathcal{MOD}^{\text{attr}}$.

Die Menge der externen Parameter(ports) von Mikro- und Makrodatenquellen ist in vielen Fällen leer, d. h. die Auswahl einer (vollständigen) Datenbasis erfolgt allein über interne Parameter. Auch wenn externe Parameter vorgesehen werden, sollen diese nicht dazu dienen, eine in der Datenbank vorliegende Datenbasis einzuschränken — hierzu sind anschließende Datenmanagementoperationen einzusetzen. Vielmehr werden auch Maßzahlen, Domänen, Kategorien etc. nur dazu genutzt, eine bestimmte, „passende“ Datenbasis zu selektieren oder aber den zu füllenden Datenraum für eine dynamische Simulation von Makrodaten zu spezifizieren.

Das Angebot interner Parameter von Datenquellmodulen zum Zugriff auf Datenbanken mit Mikro- und Makrodaten ist stark abhängig von der internen Realisierung des Zugriffs. Datenräume können etwa über Listen (Aufzählungstypen) ausgewählt werden, wobei zusätzlich Benutzeridentifikationen, Paßwörter, Spezifikationen der Datenbank–Instanzen und weitere Verbindungsparameter anzugeben sind. Es wird an dieser Stelle davon ausgegangen, daß die Spezifikation eines Mikrodatenraums stets auch bereits die nötigen Konsolidierungsparameter (quasi in Form einer Makrodatensicht) umfaßt.[24] Eine größere Flexibilität des Zugriffs unter Einbeziehung externer Parameter wäre denkbar, aber nicht unbedingt erforderlich, da die Art der Nutzung eines Datenbestandes, also die Wahl von summarischen und kategoriellen Attributen in der Regel bereits im Vorfeld auf wenige Varianten beschränkt ist.

Eher „technische“ Details der Realisierung der Datenbereitstellung in den Datenquellmodulen können in der Implementierung von *VIOLA* durch die Definition verschiedener Unterklassen zum Zugriff auf spezifische Datenbanksysteme unter Verwendung jeweils einheitlicher Parametersätze gekapselt werden.

Eingangsmodule von Verbundmodulen benötigen externe Parameter zur Auswahl eines am Verbundmodul anliegenden Datenraums anhand seiner Attribute gemäß Def. 5.6, falls die Eingangsports des Verbundmoduls *Mengen* von Datenräumen akzeptieren. Auf die Details der Datenübernahme, die hier nicht als Funktion f^{dat} spezifiziert werden soll, da sie erst durch den globalen Verarbeitungsalgorithmus von *VIOLA* und hierin lediglich als einfaches „Kopieren“ eines Datenraums definiert ist, kommen wir in Abschnitt 5.3.2 sowie in Abschnitt 5.4.1 noch im Detail zurück. Ein einfaches Beispiel wäre ein Verbundmodul mit zwei Eingangsmodulen zur Berechnung verschiedener standardisierter Erkrankungsraten und -risiken, von denen das eine anliegende Daten zur Studien- und das andere Daten zur Standardbevölkerung innerhalb des Verbundes bereitstellt. Über das Flag *optional*? kann spezifiziert werden, ob das jeweilige Eingangsmodul nicht notwendigerweise gültige Daten erhalten muß, um die durch das Verbundmodul gekapselte Berechnung ordnungsgemäß durchführen, d. h. die jeweiligen Datensenken und Ausgänge des Verbundes bedienen zu können.

[24] Gegebenenfalls kann ein Mikrodatenraum unter verschiedenen Namen mit unterschiedlichen Parametern also auch mehrfach zur Auswahl angeboten werden.

Auf einen Spezialfall der Erzeugung von Makrodaten aus Kategorien soll abschließend noch kurz eingegangen werden. In Kapitel 4 wurde begründet, daß sich Maßzahlwerte in einer Zelle eines Datenraums stets nur auf die jeweils repräsentierten Teilgruppen der betrachteten Objektmengen beziehen können, so daß etwa die Definition von Rängen in diesem Kontext nicht möglich ist. Betrachten wir ein einfaches Beispiel: Ein eindimensionaler Datenraum enthalte Erkrankungsraten zu verschiedenen Gebieten — die betrachtete Objektmenge sind Personen bzw. Tumoren. Zur Erstellung einer thematischen Karte, die alle Gebiete gemäß ihrer Raten in z. B. fünf Gruppen einteilt und entsprechend unterschiedlich einfärbt, müßte jedem Gebiet eine Gruppennummer bzw. ein Rang (der jeweiligen Gruppe) zugeordnet werden. Diese ließe sich jedoch nicht „lokal“ als Maßzahl auf der entsprechenden Personen- oder Tumormenge definieren. Statt dessen kann man in *VIOLA* aus dem Quelldatenraum durch ein Modul zur Parameterextraktion eine geordnete Menge aggregierter Gebietskategorien extrahieren, die die jeweiligen Gruppen repräsentieren (vgl. Abschnitt 5.2.4). Über ein Datenquellmodul kann nun auf der Objektmenge *Gebiete* dynamisch eine Maßzahl *Gruppe* definiert und ein neuer, eindimensionaler Datenraum über den als Parameter übergebenen fünf Gebietsgruppen erzeugt werden, der anschließend visualisiert werden kann.[25] Diese Maßzahl *Gruppe* ist nicht aus anderen Maßzahlen berechenbar, sondern direkt über die „Eigenschaft“ von Gebieten, eine bestimmte Erkrankungsrate zu haben, definiert. Im Gegensatz zu anderen ist diese Eigenschaft *nicht* statisch durch die betrachtete Objektmenge gegeben, sondern wird jeweils mit der Gebietsklassifikation dynamisch definiert, was jedoch für *MADEIRA* kein konzeptionelles Problem darstellt.

Datenmanagement

Unter Datenmanagementmodulen werden in *VIOLA* solche Bausteine verstanden, die im Rahmen der Verarbeitung von Datenräumen keine neuen Maßzahlen berechnen, sondern Makrodaten lediglich umstrukturieren bzw. selektieren. Hierunter fallen Ableitung, Selektion summarischer Attribute, Vereinigung, Split, Merge und die Umbenennung von Rollen.

Alle Datenmanagementmodule weisen gleich viele Datenein- und -ausgänge auf — anliegende Daten werden jeweils parallel und unabhängig verarbeitet. Abgesehen vom Ableitungsoperator werden keine internen Parameter benötigt; die automatische Berechnung von Parameterwerten erfolgt in keinem Fall.

Definition 5.14 (Datenmanagementmodule) *Ein* DATENMANAGEMENTMODUL *sei definiert als* $mod = (b, PPO, PO^{\text{par}}, PAR^{\text{int}}, f^{\text{dat}})$ *durch*

- *einen Bezeichner* $b \in \mathcal{B}$,
- *eine Menge* $PPO \subseteq \mathcal{PO}^{\text{dat}} \times \mathcal{PO}^{\text{dat}}$ *von Paaren aus Eingangs- und Ausgangsport,*
- *eine Menge von Parameterports* $PO^{\text{par}} \subseteq \mathcal{PO}^{\text{par}}$,
- *eine Menge interner Parameter* $PAR^{\text{int}} \subseteq \mathcal{PAR}^{\text{int}}$ *und*
- *eine Verarbeitungsfunktion* f^{dat}.

Parameter und Verarbeitungsfunktion sind je nach Art des Moduls unterschiedlich durch die Operatoren von MADEIRA definiert. Sei $(po^{\text{in}}, po^{\text{out}}) \in PPO$, *so gelte für*

- *die Menge aller (allgemeinen) Ableitungsmodule* $\mathcal{MOD}^{\text{abl}}$*:*
  $PAR^{\text{int}} = \{(variante, \mathbb{N}, mod)\}$, $PO^{\text{par}} = \{zielkategorien: \widetilde{2^{\mathcal{K}}}, eigenschaften: \widetilde{2^{\mathcal{E}\times\mathcal{OR}}}\}$, *und* f^{dat} *sei definiert durch* $f^{\text{abl}}(data(po^{\text{in}}), zielkategorien, eigenschaften, variante)$.
- *die Menge aller Ableitungsmodule zur vollständigen Ableitung (*$\mathcal{MOD}^{\text{ablv}}$*):*
  $PO^{\text{par}} = \{eigenschaften: 2^{\mathcal{E}\times\mathcal{OR}}\}$, *und* f^{dat} *sei definiert durch* $f^{\text{abl}}(data(po^{\text{in}}), eigenschaften)$.
- *die Selektion summarischer Attribute (*$\mathcal{MOD}^{\text{sel}}$*):* $PO^{\text{par}} = \{mass: 2^{\mathcal{M}}, objektrollen: 2^{\mathcal{OR}}\}$, *und* f^{dat} *sei definiert durch* $f^{\text{sel}}(data(po^{\text{in}}), mass, objektrollen)$.

[25] Dieses Vorgehen mag ein wenig umständlich erscheinen, dient aber zur strengen Einhaltung der in *MADEIRA* definierten Datensemantik. Und in der Tat sind thematische Karten mit ihrer impliziten Klassifikation von Kategorien „anders“ als andere Visualisierungsverfahren.

- *die Vereinigung ($\mathcal{MOD}^{\cup}$): $PO^{\text{par}} = \{$domaenen: $2^{\mathcal{DO}}$, ausschluss: $2^{\mathcal{E}\times\mathcal{OR}}$, datenraum: $\mathcal{DR}\}$, und f^{dat} sei definiert durch $f^{\cup}(data(po^{\text{in}}) \cup \{datenraum\}, domaenen, ausschluss)$.*
- *Split ($\mathcal{MOD}^{\text{split}}$): $PO^{\text{par}} = \{$objekteigenschaft: $\mathcal{E}\times\mathcal{OR}$, eigenschaft: $\mathcal{E}\}$, und f^{dat} sei definiert durch $f^{\text{split}}(data(po^{\text{in}}), objekteigenschaft, eigenschaft)$.*
- *Merge ($\mathcal{MOD}^{\text{merge}}$): $PO^{\text{par}} = \{$objekteigenschaft: $\mathcal{E}\times\mathcal{OR}$, eigenschaft: $\mathcal{E}\}$, und f^{dat} sei definiert durch $f^{\text{merge}}(data(po^{\text{in}}), objekteigenschaft, eigenschaft)$.*
- *Rollenumbenennung ($\mathcal{MOD}^{\text{role}}$): $PO^{\text{par}} = \{$quelle: $\mathcal{R}$, ziel: $\mathcal{R}\}$, und f^{dat} sei definiert durch $f^{\text{role}}(data(po^{\text{in}}), quelle, ziel)$.*

Für alle Modularten außer der allgemeinen Ableitung sei $PAR^{\text{int}} = \emptyset$. Für die Vereinigung gelte ferner: $\forall(po^{\text{in}}, \cdot) \in PPO: po^{\text{in}}.card = *$; bei allen anderen Operatoren sei die Kardinalität der Eingangsports jeweils auf einen Datenraum beschränkt.

$\mathcal{MOD}^{\text{dm}} = \mathcal{MOD}^{\text{abl}} \cup \mathcal{MOD}^{\text{ablv}} \cup \mathcal{MOD}^{\text{sel}} \cup \mathcal{MOD}^{\cup} \cup \mathcal{MOD}^{\text{split}} \cup \mathcal{MOD}^{\text{merge}} \cup \mathcal{MOD}^{\text{role}}$ sei die Menge aller Datenmanagementmodule.

Diese Definitionen ergeben sich somit direkt aus dem Datenmodell *MADEIRA* bzw. den im vorigen Abschnitt angestellten Überlegungen zur Umsetzung der *MADEIRA*–Operatoren. Das Vereinigungsmodul ergänzt die entsprechende Definition aus *MADEIRA* noch um einen Parameter, der einen Datenraum bereitstellt, welcher jeweils mit den Datenraummengen in allen Eingangsports gleichermaßen vereinigt wird. Dieser Parameterport kann nur einen Datenraum aufnehmen, ggf. sind Mengen von Parameterdatenräumen durch einen weiteren vorgeschalteten Vereinigungsoperator zusammenzufassen.

Maßzahlberechnung

Die Umsetzung der Maßzahlberechnung erfolgt — ähnlich zu den Datenmanagementmodulen — recht kanonisch aus den *MADEIRA*–Operatoren bzw. der obigen Definition der expliziten Aggregierung. Als einzige Ergänzung wollen wir vorsehen, daß zur näheren Spezialisierung einer allgemeineren Maßzahl noch weitere externe Parameter übergeben werden können (in Def. 5.15 jeweils durch „. . .“ bei der Spezifikation von PO^{par} angedeutet). Anhand von den Maßzahlen zugeordneten Metadaten gemäß Abschnitt 4.4.2 kann so eine Konkretisierung des zu berechnenden Maßes erfolgen. Zum Beispiel können so zu einem Standardisierungsverfahren die Domänen angegeben werden, über die standardisiert werden soll, oder die Berechnung einer Rate kann auf die Berücksichtigung von Daten zur Standard- oder Studienpopulation eingegrenzt werden.

Definition 5.15 (Module zur Maßzahlberechnung) *Ein* MODUL ZUR MASSZAHLBERECHNUNG *$mod = (b, PPO, PO^{\text{par}}, PAR^{\text{int}}, f^{\text{dat}})$ werde wie ein Datenmanagementmodul definiert; alle Eingangsports können maximal einen Datenraum aufnehmen. Sei $(po^{\text{in}}, po^{\text{out}}) \in PPO$, so gelte für*

- *alle Module zur zellenweisen Maßzahlberechnung $\mathcal{MOD}^{\text{cell}}$:*
  $PO^{\text{par}} = \{$mass: $\mathcal{M}, \ldots\}$, $PAR^{\text{int}} = \emptyset$, und f^{dat} sei definiert durch $f^{\text{cell}}(data(po^{\text{in}}), mass)$.
- *die explizite Aggregierung ($\mathcal{MOD}^{\text{aggr}}$):*
  $PO^{\text{par}} = \{$kategorien: $\widetilde{2^{\mathcal{K}}}$, eigenschaften: $\widetilde{2^{\mathcal{E}\times\mathcal{OR}}}$, mass: $\mathcal{M}, \ldots\}$, $PAR^{\text{int}} = \{(variante, \mathbb{N}, mod)\}$,
  und f^{dat} sei definiert durch $f^{\text{eaggr}}(data(po^{\text{in}}), kategorien, eigenschaften, variante, mass)$.
- *die freie Verknüpfung von Maßzahlen ($\mathcal{MOD}^{\text{join}}$):*
  $PO^{\text{par}} = \{$mass: $\mathcal{M}, \ldots\}$, $PAR^{\text{int}} = \emptyset$, und f^{dat} sei definiert durch $f^{\text{join}}(data(po^{\text{in}}), mass)$.

Die Menge aller Module zur Maßzahlberechnung sei $\mathcal{MOD}^{\text{calc}} = \mathcal{MOD}^{\text{cell}} \cup \mathcal{MOD}^{\text{aggr}} \cup \mathcal{MOD}^{\text{join}}$.

Wie man sieht, kann die zellenweise Maßzahlberechnung (den Definitionen von f^{cell} und f^{join} entsprechend) als Spezialisierung der freien Maßzahlverknüpfung betrachtet werden. Dementsprechend läßt sich eine gegebene Maßzahl, die zellenweise berechenbar ist, immer auch durch ein Modul zur allgemeinen Maßzahlverknüpfung ermitteln. Trotzdem bietet *VIOLA* hierzu zwei verschiedene Modulklassen, um (unter Verwendung verschiedener Modulicons, siehe Abschnitt 5.4.6) eine klarere Programmstrukturierung zu ermöglichen.

Bemerkenswert ist auch, daß alle Berechnungsfunktionen u. U. mehrdeutig bzgl. der Nutzung von summarischen Attributen des Quelldatenraums sein können. Hier erscheint es sinnvoll, jeweils auf den Berechnungsvorschriften einer Maßzahl eine global vorgegebene Ordnung zu definieren, nach der beim Vorliegen von Alternativen z. B. einfachere, weniger rechenintensive Verfahren oder solche, die keine weiteren Maßzahlen als Zusatzgrößen liefern, präferiert werden, sofern der Nutzer keine explizite Wahl trifft. Auch sollten vorzugsweise bzgl. der Maßzahlhierarchie speziellere, „aussagekräftigere“ Zielmaße (z. B. *Inzidenzraten* anstelle von *Quotienten*) bzw. — sofern eine Teilmengenbeziehung gegeben ist — solche mit einer umfangreicheren Menge **EOR** zu aggregierender Eigenschaften gewählt werden (etwa *Alters- und Geschlechtsstandardisierung* anstelle von *Altersstandardisierung*, falls möglich). Bei der expliziten Aggregierung kommt zu diesen Mehrdeutigkeiten noch hinzu, daß evtl. mehrere Wege zur Ableitung von Ziel- aus Quellkategorien bestehen können.

Analog zu den Datenquellmodulen bietet es sich auch für die Maßzahlberechnung an, entsprechende Module in verschiedenen Unterklassen zu implementieren, die dem Anwender jeweils bzgl. der gegebenen Maßzahlhierarchie (vgl. Abschnitt 4.4.2) „ähnliche“ Gruppen (Unterklassen) von Maßzahlen (evtl. unter Nutzung zusätzlicher externer Parameter) zur Auswahl bieten. Eine derartige Klassifikation bildet auch die Basis für einen weitgehend problemlosen (d. h. Anwendbarkeitsbedingungen nicht verletzenden) modulbezogenen Wechsel zwischen verschiedenen Maßen im Rahmen explorativer Datenanalysen sowie eine Verfeinerung von allgemeinen Maßzahlvorgaben im Sinne einer Konkretisierung von Analysestrategien: Da die Maße, zwischen denen innerhalb eines Moduls gewechselt wird, relativ „ähnlich“ sind, ist die Wahrscheinlichkeit groß, daß in nachfolgenden Modulen ausgeführte Operationen auch nach einer Änderung der Maßzahlwahl anwendbar bleiben. Weiterhin muß in diesen Analysemodulen die interaktive Spezifikation der gewünschten Maßzahl nicht notwendig über eine Auswahlliste oder -hierarchie erfolgen, sondern kann auch anhand der den jeweiligen Maßzahltypen zugeordneten Metadaten (vgl. Abschnitt 4.4.2) geschehen.

5.2.3 Datenausgabe und Visualisierung

Die Visualisierung von Daten spielt sicherlich im Kontext der explorativen Datenanalyse eine wichtige, wenn nicht *die* zentrale Rolle. In diesem Abschnitt wird dargestellt, wie sich die Datenvisualisierung in die visuelle Programmiersprache *VIOLA* homogen einpaßt, ohne auf alle Details der Graphikkonstruktion selbst einzugehen. Graphiken können in *VIOLA* sowohl — neben Modulen zum Abspeichern von Analyseergebnissen — als Datensenken als auch — auf der Basis ihrer interaktiven Manipulation durch den Benutzer — als Quellen neuer Informationen und Analysen angesehen werden.

Zunächst wird im folgenden ein einfaches Modell für diese Arbeit interessierender Graphiken eingeführt, auf dessen Grundlage dann im Anschluß *VIOLA*–Module zur Realisierung von Datensenken, insbesondere Visualisierungsmodule, definiert werden. Es wird kurz dargestellt, wie die Inhalte eines Datenraums auf Komponenten einer Graphik abgebildet und graphische Darstellungen interaktiv und dynamisch manipuliert werden können.

Ein einfaches Graphikmodell

Im Rahmen von *VIOLA* interessieren lediglich Graphiken, die aggregierte Daten zu multidimensionalen Kategorisierungen direkt visualisieren. Typische Beispiele bilden etwa Balken- und Kurvendiagramme, aber auch Tabellen können in analoger Weise erstellt werden. Weiterhin sollen auch thematische Karten generiert werden können, die aufgrund von maßzahlbasierten Klassifikationen und Ordnungen von Kategorien (Gebieten) definiert sind: Hier wird strenggenommen nicht der jeweilige Maßzahlwert selbst, sondern eine Gruppenzugehörigkeit des jeweiligen Gebiets in Form einer entsprechenden Farbgebung visualisiert. Ein Ansatz zur Behandlung dieser Besonderheit wurde bereits im Anschluß an die Definition kategorieller Datenquellen auf Seite 186 diskutiert.

Mikrodatenbasierte Darstellungen — wie etwa Scatterplots oder Histogramme — werden nicht betrachtet. Für letztere wäre die Einführung spezieller Visualisierungsverfahren denkbar, die die benötigten Informationen auch aus den jeweils vorliegenden Makrodaten extrahieren. Da sowohl Erzeugung, Parametrisierung als auch

interaktive Nutzung derartiger Graphiken aus dem Rahmen der oben angesprochenen Standardverfahren fallen würden, wird hierauf jedoch verzichtet.

Das im folgenden vorgestellte Modell beschreibt einerseits Klassen von Graphiken, die durch *Visualisierungsverfahren* spezifiziert werden. Diese legen über *Symboldefinitionen* die zu verwendenden Mengen von *Darstellungssymbolen* fest, die die jeweiligen Datenwerte, also die Zellen eines multidimensionalen Datenraums, mit Hilfe *graphischer Attribute* repräsentieren. Ein Symbol weist jeweils mehrere solcher Attribute bzw. Attributausprägungen auf, was die Mehrdimensionalität der dargestellten Daten widerspiegelt. Andererseits werden auf dieser Basis auch konkrete *Graphiken* modelliert, die Symbole gleicher Gestalt in einer oder mehreren *Datenreihen* zusammenfassen und visualisieren. Abbildung 5.5 gibt in einem *UML*–Klassendiagramm einen Überblick über diese Zusammenhänge sowie die Kardinalitäten, mit denen Entitäten in die Definition anderer Komponenten eingehen.

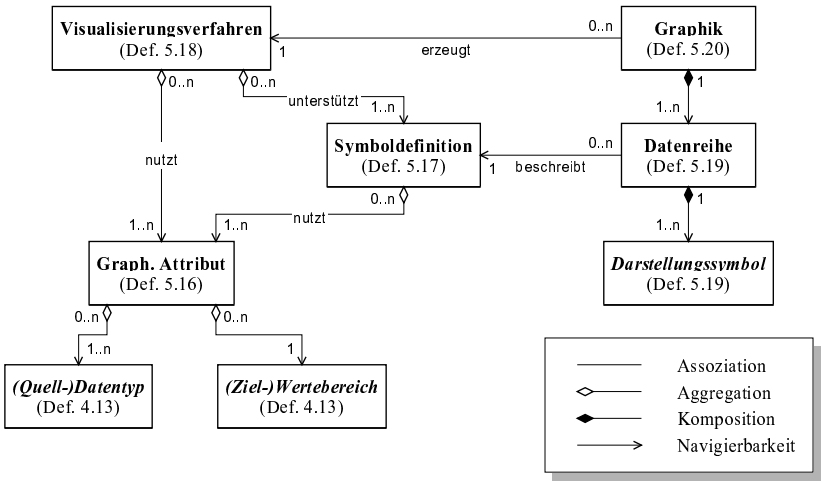

Abbildung 5.5: *UML*–Spezifikation von Visualisierungsverfahren (links) und Komponenten von Graphiken (rechts)

An Constraints auf den Beziehungen sind zu nennen:

- Die einem Visualisierungsverfahren zur Verfügung stehenden graphischen Attribute ergeben sich als Vereinigung der Attribute zu allen Symboldefinitionen des Verfahrens.
- Die Anzahl aller Darstellungssymbole pro Datenreihe ist ebenso wie die Anzahl der Datenreihen einer Graphik pro Symboldefinition durch Merkmale der Symboldefinition begrenzt.
- Die Symboldefinition, die eine Datenreihe beschreibt, ist auch dem Visualisierungsverfahren zur Erzeugung der jeweiligen Graphik zugeordnet.

Wir verzichten hier auf die Formalisierung aller Details. Skalen und Beschriftungen sowie verschiedene datenunabhängige Layoutparameter fehlen in unserem Modell — wir konzentrieren uns lediglich auf diejenigen Aspekte, die zur Beschreibung der visuellen Präsentation der Zellen eines Datenraums unbedingt notwendig sind. Eine allgemeine und ausführliche Diskussion von Gestalt und Konstruktion von Graphiken, die viele Varianten und Spezialfälle berücksichtigen müßte, ginge über Thema und Rahmen dieser Arbeit hinaus; einige entsprechende Ansätze finden sich etwa in [Sch99].

Die Grundlage der Definition von Visualisierungsverfahren bilden graphische Attribute, die die darzustellenden Größen auf visuelle Merkmale von Darstellungssymbolen, wie x– und y–Position, Größe, Form, Farbe etc. abbilden. Auf diese Weise zu visualisierende Größen sind entweder Inhalt und Koordinaten einer Datenraumzelle, also Maßzahlwerte bzw. Kategorien selbst, oder aber aus Kategorien in Form von Metadaten abgeleitete Maßzahlwerte (vgl. Abschnitt 4.4.3). Weiterhin kann auch eine beliebige Ordnung (anhand von

Namen, Beschreibung, aber auch Datenraum–Inhalten) auf Kategorien zur Parametrisierung graphischer Symbole verwendet werden, etwa zur Wahl der Reihenfolge der Balken in einem Balkendiagramm über Alter, Zeit oder Geschlecht. In jedem Fall ist somit ein Datentyp gegeben, dessen Ausprägungen durch das graphische Attribut geeignet umzusetzen sind.[26] Natürlich kann nicht jedes Visualisierungsverfahren jede Art von Daten visualisieren, z. B. können nicht–raumbezogene Daten nicht in einer thematischen Karte dargestellt werden. Oftmals wird auch zwischen nominalen und ordinalen Typen oder anhand ähnlicher Metadaten differenziert (vgl. Abschnitt 4.4.2). Somit definiert ein graphisches Attribut auch, welche Datentypen es überhaupt verarbeiten kann, und natürlich, wie dies im Einzelfall geschieht, z. B. durch Multiplikation mit einem bestimmten Faktor bei der Umrechnung numerischer Maße in Bildschirmkoordinaten.

Definition 5.16 (Graphische Attribute) *Ein* GRAPHISCHES ATTRIBUT $ga = (W, \mathbf{T}, f^{\text{vis}})$ *sei definiert durch*

- *einen nicht–leeren Wertebereich* $W \subseteq \mathcal{W}$ *eines visuellen Merkmals,*
- *eine endliche, nicht–leere Menge* $\mathbf{T} \subseteq 2^{\mathcal{W}}$ *von Datentypen, auf denen Ausprägungen dieses Merkmals definiert werden können, und*
- *eine Abbildung* $f^{\text{vis}} : \bigcup_{T \in \mathbf{T}} T \to W$*, die einer Ausprägung eines Datentyps einen Wert aus* W *zuordnet.*

$\mathcal{GA}$ *sei die Menge aller graphischen Attribute.*

Wie bereits erwähnt, dienen graphische Attribute zur Parametrisierung von Darstellungssymbolen wie etwa Balken, Punkten, Linien oder Gebieten (in einer thematischen Karte). Eine Graphik weist eine oder mehrere[27] Klassen von Symbolen auf, die alle einem einheitlichen Schema folgen, denen also die gleichen graphischen Attribute zugeordnet sind, z. B. einem Balken Höhe und x–Koordinate oder einem Gebiet Form und Färbung. Diese Zuordnung leistet eine Symboldefinition. Sie gibt außerdem maximale Kardinalitäten für die Nutzung von Symbolen vor: Eine Art von Symbol kann nur von einer bestimmten Anzahl von Datenreihen in einer Graphik verwendet werden (ein Kurvendiagramm darf z. B. nicht beliebig viele Kurven haben), und eine Datenreihe darf nur aus einer bestimmten Anzahl von Symbolen bestehen (ein Kurvendiagramm hat nur begrenzt viele Meßpunkte).

Definition 5.17 (Symboldefinitionen) *Eine* SYMBOLDEFINITION $sd = (GA, maxdat, maxsym)$ *sei definiert durch*

- *eine endliche, nicht–leere Menge* $GA \subseteq \mathcal{GA}$ *von graphischen Attributen, die einer Menge von Symbolen zuzuordnen sind,*
- *eine maximale Anzahl* $maxdat \in \mathbb{N}$ *von Datenreihen, die diese Symboldefinition innerhalb einer Graphik gleichzeitig nutzen können, sowie*
- *eine maximale Anzahl* $maxsym \in \mathbb{N}$ *von Elementen (Symbolen) pro Datenreihe.*

$\mathcal{SD}$ *sei die Menge aller Symboldefinitionen.*

Ein Visualisierungsverfahren schließlich vereint mehrere Symboldefinitionen und deren graphische Attribute, wobei auch graphische Attribute von mehreren Symboldefinitionen verwendet werden können. Etwa spielt in einer Kombination von Balken- und Kurvendiagramm die x–Koordinate für beide Symboldefinitionen die gleiche Rolle.

Definition 5.18 (Visualisierungsverfahren) *Ein* VISUALISIERUNGSVERFAHREN $vv = (GA, SD)$ *sei definiert durch eine endliche, nicht–leere Menge* $GA \subseteq \mathcal{GA}$ *graphischer Attribute und eine endliche, nicht–leere Menge* $SD \subseteq \mathcal{SD}$ *von Symboldefinitionen, die jeweils einen Teil der graphischen Attribute nutzen, also* $\bigcup_{sd \in SD} SD.GA = GA$*.* $\mathcal{VV}$ *sei die Menge aller Visualisierungsverfahren.*

[26] Zur Beschreibung von Rängen dienen die natürlichen Zahlen, und Domänen bilden gemäß Def. 4.13 ebenfalls Datentypen.

[27] So können etwa Balken- und Kurvendiagramm kombiniert werden.

Nun kann die Zusammensetzung konkreter Graphiken aus Symbolen und Datenreihen konkretisiert werden. Wie bereits angedeutet, sind Datenreihen im wesentlichen in ihrer Kardinalität begrenzte Mengen von Darstellungssymbolen zur gleichen Symboldefinition. Die Symbole wiederum weisen jeweils Ausprägungen zu den ihnen gemäß der Symboldefinition zugeordneten graphischen Attributen auf. Symbole werden als abstrakte Entitäten modelliert.

Definition 5.19 (Datenreihen über graphischen Symbolen) *Eine* DATENREIHE *sei definiert als* $dat = (SYM, sd, graph)$ *durch*

- *eine nicht–leere Menge SYM von maximal* $sd.maxsym$ (DARSTELLUNGS–)SYMBOLEN,
- *eine allen Symbolen gemeinsame Symboldefinition* $sd \in \mathcal{SD}$ *sowie*
- *eine Familie* $graph = (graph_{ga})_{ga \in sd.ga}$ *von Abbildungen* $graph_{ga} : SYM \to ga.W$*, die einem Symbol jeweils eine Ausprägung jedes graphischen Attributs zuordnen.*

$\mathcal{DAT}$ *sei die Menge aller Datenreihen und* $\mathcal{SYM}$ *die Menge aller Darstellungssymbole.*

Eine Graphik enthält nunmehr einfach eine Menge von (in ihrer Anzahl beschränkten) Datenreihen zu den Symboldefinitionen eines Visualisierungsverfahrens. Nicht jede Symboldefinition muß letztendlich durch eine Datenreihe präsent sein. So können wiederum Balken- und Liniendiagramme sowie auch ihre Kombination durch das gleiche Visualisierungsverfahren modelliert werden.

Definition 5.20 (Graphiken) *Eine* GRAPHIK $g = (vv, DAT)$ *sei definiert durch ein Visualisierungsverfahren* $vv \in \mathcal{VV}$ *sowie eine nicht–leere Menge* $DAT \subseteq \mathcal{DAT}$ *von Datenreihen, wobei* $\forall dat \in DAT : dat.sd \in vv.SD$ *und* $\forall sd \in vv.SD : |\{dat \in DAT \mid dat.sd = sd\}| \leq sd.maxdat$*.*

$\mathcal{G}$ *sei die Menge aller Graphiken.*

Abbildung 5.6 gibt abschließend ein einfaches Beispiel einer Graphik als kombiniertes Balken- und Kurvendiagramm, das im allgemeinen etwa maximal zwei Balken– und fünf Linien–Datenreihen aus jeweils bis zu einhundert Punkten enthalten könnte.

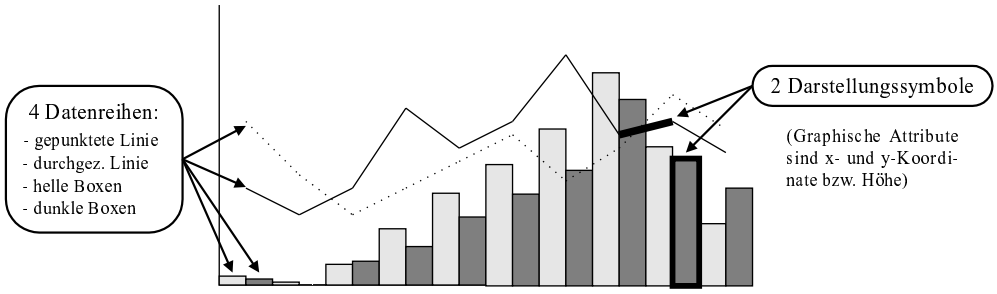

Abbildung 5.6: Beispiel einer Graphik

Datensenken in *VIOLA*

Drei verschiedene Modularten fungieren in *VIOLA* in dem Sinne als Datensenken, daß sie als Nebeneffekt ein Analyseergebnis herausgreifen, präsentieren oder persistent machen:

- analog zu den Eingangsmodulen haben Verbundmodule auch Datenausgänge,
- Speicherbausteine legen Beschreibungen von Datenräumen auf Sekundärspeicher ab oder drucken diese aus, und

- Visualisierungsbausteine erzeugen anhand verschiedener Parameter Graphiken bestimmten Typs zu anliegenden Datenräumen, speichern diese u. U. ebenfalls auf Sekundärspeicher ab und erlauben die interaktive Extraktion von zusätzlichen Informationen und multidimensionalen Parametern im Sinne der Nutzung interaktiver Graphiken.

Im Detail ergeben sich folgende Definitionen:

Definition 5.21 (Datensenken) *Ein* AUSGANGSMODUL *für Verbundmodule sei definiert als* $mod^{\text{out}} = (b, po^{\text{in}}, mod^{\text{sub}})$, *wobei* $po^{\text{in}}.card = 1$ *und* $mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}}$ *ein Verbundmodul.* $\mathcal{MOD}^{\text{out}}$ *sei die Menge aller Ausgangsmodule.*

Ein SPEICHERMODUL *sei definiert als* $mod = (b, PPO, PAR^{\text{int}})$, *wobei die internen Parameter* PAR^{int} *technische Details der Datenablage spezifizieren.* $\mathcal{MOD}^{\text{save}}$ *sei die Menge aller Speichermodule.*

Ein VISUALISIERUNGSMODUL *sei definiert als* $mod = (b, po^{\text{in}}, po^{\text{out}}, PO^{\text{par}}, PAR^{\text{int}}, vv)$, *wobei* $po^{\text{in}}.card = 1$ *und* $vv \in \mathcal{VV}$ *ein Visualisierungsverfahren. Die internen Parameter* PAR^{int} *spezifizieren die konkrete Abbildung der Zellen des anliegenden Datenraums auf die Ausprägungen der graphischen Attribute der zu erzeugenden Graphik (im Zusammenwirken mit den Visualisierungsfunktionen* f^{vis} *dieser Attribute). Ferner können durch diese Parameter — oftmals ebenfalls unter Rückgriff auf Datenraumattribute und –Metadaten — Layoutdetails und Beschriftungen vorgegeben werden. Für alle Parameterports* $po^{\text{par}} \in PO^{\text{par}}$ *gelte* $po^{\text{par}}.in? = false$. $\mathcal{MOD}^{\text{vis}}$ *sei die Menge aller Visualisierungsmodule.*

$\mathcal{MOD}^{\text{sink}} = \mathcal{MOD}^{\text{out}} \cup \mathcal{MOD}^{\text{save}} \cup \mathcal{MOD}^{\text{vis}}$ *sei die Menge aller* DATENSENKEN(–MODULE).

Speicher- und Visualisierungsmodule leiten eingehende Daten unverändert an die jeweiligen Ausgangsports weiter. Datenablage und Visualisierung erfolgen somit quasi als Nebeneffekte.

Im Gegensatz zu den meisten anderen Modularten müssen Visualisierungsmodule relativ häufig konkret auf die jeweils betrachteten, am Eingangsport anliegenden Datenräume zugeschnitten, d. h. entsprechend parametrisiert werden. Nur in bestimmten Fällen — gerade für sehr einfache Visualisierungsverfahren, die z. B. nur eine Datenreihe mit Hilfe eines einzigen graphischen Attributs darstellen — kann ein Datenraum eindeutig und automatisch (oder gar nicht) visualisiert werden. Oftmals bestehen jedoch viele Möglichkeiten zur Wahl der betreffenden Abbildungen; ein automatisierter Default kann lediglich einen Vorschlag liefern und ist oft noch zu modifizieren. Weiterhin beziehen sich auch die durch Benutzerinteraktionen über die jeweilige Graphik definierten externen, multidimensionalen Parameter stets auf genau einen visualisierten Datenraum. Aus diesen beiden Gründen kann ein Visualisierungsmodul — im Gegensatz zu den meisten anderen *VIOLA*–Modulen — stets nur einen einzelnen Datenraum aus einem einzigen Eingangsport verarbeiten.

Spezifikation einer Graphik

Mit Hilfe der internen Parameter eines Visualisierungsmoduls wird die konkrete Umsetzung eines Datenraums in eine graphische Darstellung spezifiziert. Im folgenden werden die hierfür nötigen eher technischen Angaben und Vorgehensweisen etwas näher betrachtet, ohne jedoch alle Details sowie die konkreten Parameterdefinitionen festzulegen, da diese i. a. sehr spezifisch vom jeweiligen Visualisierungsverfahren abhängen. In ähnlicher Weise werden auch lediglich allgemeine Kriterien der Anwendbarkeit eines Verfahrens auf gegebene Daten diskutiert.

Graphiken unterscheiden oftmals nicht zwischen summarischen und kategoriellen Attributen. Zum Beispiel können in einem Kurvendiagramm unterschiedliche Kurven verschiedene Maßzahlen *oder* verschiedene Ausprägungen eines kategoriellen Attributs (etwa unterschiedliche Zeiträume) repräsentieren. Somit läßt sich die Visualisierung eines Datenraums $dr = (OR, KA, SA, f^{\text{ext}})$ am einfachsten anhand seiner homogenisierten Sicht $f^{\text{hom}}(dr)$ mit kategoriellen Attributen $\widetilde{KA}$ spezifizieren, da auch dort Kategorien und Maßzahlen gleich behandelt werden. Sei $vv = (GA, SD) \in \mathcal{VV}$ ein zu verwendendes Visualisierungsverfahren.

Zunächst sind die klassifizierenden Attribute aus $\widetilde{KA}$ in zwei Gruppen $KA^{\text{dat}} \stackrel{\text{def}}{=} (ka_1, \ldots, ka_n)$ und $KA^{\text{val}} \stackrel{\text{def}}{=} (ka_{n+1}, \ldots, ka_{n+v})$ vollständig und disjunkt zu unterteilen. Hierbei kann eine der Attributmengen durchaus auch

leer bleiben.[28] Für jede Kombination der Ausprägungen der Attribute in KA^{dat} wird eine Datenreihe erzeugt werden, die jeweils auf Einzelwerten basiert, die durch Tupel von Ausprägungen der Attribute in KA^{val} *und*[29] KA^{dat} sowie durch die entsprechenden Zellinhalte spezifiziert werden. In der Regel wird das Kennzahlattribut in die Menge KA^{dat} fallen, so daß eine Datenreihe ein einheitliches summarisches Attribut darstellt.

Sei $\widetilde{SD} \subseteq SD$ die Menge zu verwendender Symboldefinitionen von *vv* — dies müssen nicht alle möglichen sein. Gibt ein Visualisierungsverfahren etwa Symbole für ein Kurven- und ein Balkendiagramm vor, so kann man sich auf eine Variante beschränken oder beide kombinieren. Für jede Symboldefinition $sd \in \widetilde{SD}$ muß gelten: $|ka_{n+1}.K| \cdot \ldots \cdot |ka_{n+v}.K| \leq sd.maxsym$, d. h. die Datenreihen dürfen nicht „zu lang“ werden.

Pro Symboldefinition *sd* soll die Belegung der jeweiligen graphischen Attribute gemäß der Koordinaten und Inhalte der Datenraumzellen natürlich einheitlich erfolgen. Das Symbol „#“ symbolisiere im folgenden jeweils den Zellinhalt (gewissermaßen das einzelne summarische Attribut von $f^{\mathrm{hom}}(dr)$). So werde nun jeweils eine partielle, nicht notwendigerweise injektive[30] oder surjektive[31] Abbildung $f^{\mathrm{ka}}_{sd} : sd.GA \to KA^{\mathrm{val}} \cup \{\#\} \cup KA^{\mathrm{dat}}$ definiert, die einem graphischen Attribut ein kategorielles Attribut (oder den „Zelleninhalt“) zur Darstellung zuordnet.

Ferner ist noch festzulegen, ob das jeweilige kategorielle Attribut selbst, ein abgeleitetes Metadatum (zu einer Alterskategorie z. B. Ober- oder Untergrenze des Altersintervalls) oder eine Ordnungsnummer (in der Regel etwa für das Kennzahlattribut) visualisiert werden soll. Auf dieser Entscheidung basiert die Wahl eines Datentyps $T_{ga,sd} \in ga.\mathbf{T}$ für jedes genutzte graphische Attribut $ga \in sd.GA$, das nicht auf # abgebildet wird, sowie einer Abbildung $f^{\mathrm{T}}_{ga,sd} : f^{\mathrm{ka}}_{sd}(ga).K \to T_{ga,sd}$, die die konkreten zu visualisierenden Werte zu den Kategorien spezifiziert.[32] Werden die Kategorien direkt visualisiert, ist f^{T} also die Identität, andernfalls liefert die Abbildung einen abgeleiteten Wert zu einer Kategorie.

Eine spezielle Behandlung erfahren die Zelleninhalte (#) von $f^{\mathrm{hom}}(dr)$. Sie werden — als numerische Werte — stets direkt visualisiert, es erfolgt keine Umrechnung in einen anderen Datentyp. Falls $ka^{\mathrm{kz}} \in KA^{\mathrm{val}}$, muß somit $T^{\mathrm{kz}} \in ga.\mathbf{T}$ für alle graphischen Attribute *ga* mit $f^{\mathrm{ka}}_{sd}(ga) = \#$ gelten, da pro Datenreihe Maßzahlwerte verschiedenen Typs darzustellen sind — eine Kompatibilität zum „universellen“ Kennzahltyp T^{kz} liefert hier die nötige Typsicherheit. Andernfalls stellt jede Datenreihe ein einheitliches summarisches Attribut dar, so daß die Typkompatibilität je nach der einer Datenreihe zugeordneten Symboldefinition zu untersuchen ist (siehe weiter unten).

Die Datenreihen *DAT* der erzeugten Graphik $g = (vv, DAT)$ zum Visualisierungsverfahren *vv* ergeben sich jetzt als eine Menge $\{dat_{k_1,\ldots,k_n} \mid k_i \in ka_i.K, i = 1,\ldots,n\}$. Jedem solchen $dat_{k_1,\ldots,k_n}$ werde eine Symboldefinition *sd* zugeordnet, so daß für jede Symboldefinition $sd \in SD$ die Anzahl zugeordneter Datenreihen maximal $sd.maxdat$ beträgt. Falls $ka^{\mathrm{kz}} \in KA^{\mathrm{dat}}$ — es gibt also für jede Datenreihe ein eindeutiges $k_j \in \mathcal{SA}, j \in \{1,\ldots,n\}$ —, muß außerdem der Typ dieses summarischen Attributs mit den graphischen Attributen, die den Zellinhalten zugeordnet sind, kompatibel sein, also $k_j.T \in ga.\mathbf{T}$ für $f^{\mathrm{ka}}_{sd}(ga) = \#$.

Die Symbolmenge *SYM* zu $dat_{k_1,\ldots,k_n}$ erhalte man jeweils, indem jeder Zelle $(k_1,\ldots,k_{n+v})$, $k_i \in ka_i.K (i = n+1,\ldots,n+v)$ von $f^{\mathrm{hom}}(dr)$ ein Symbol *sym* zugeordnet wird. Die Ausprägung $graph_{ga}(sym)$ eines graphischen Attributs von *sym* sei dann jeweils — unter Nutzung der Visualisierungsfunktion f^{vis} des graphischen Attributs — definiert als $f^{\mathrm{vis}}(f^{\mathrm{T}}_{ka_i,sd}(k_i))$, wenn $f^{\mathrm{ka}}_{sd}(ga) = ka_i$. Und die Ausprägung $graph_{ga}(sym)$ für $f^{\mathrm{ka}}_{sd}(ga) = \#$

[28] Hat der Datenraum keine echt klassifizierenden Attribute, so ist eine graphische Darstellung in der Regel wenig sinnvoll, aber prinzipiell auch möglich.

[29] Auch verschiedene Datenreihen können eine unterschiedliche graphische Erscheinungsform aufweisen.

[30] Ein kategorielles Attribut kann durch mehrere graphische Attribute visualisiert werden.

[31] Attribute aus KA^{dat} können u. U. auch ohne Zuordnung bleiben, wenn verschiedene Datenreihen unterschiedliche Symboldefinitionen nutzen oder etwa nicht durch ihr Format, sondern nur durch eine Beschriftung identifiziert werden. Demgegenüber sollten Attribute aus KA^{val} sowie # in der Regel im Bildbereich enthalten sein, womit sich eine Anforderung an die Mindestanzahl graphischer Attribute von *sd* ergibt.

[32] Aufgrund dieser Typkompatibilitäten zwischen einem aus dem kategoriellen Attribut abgeleiteten Datentyp und den Darstellungsmöglichkeiten des graphischen Attributs läßt sich in gewissem Rahmen die Anwendbarkeit eines Visualisierungsverfahrens auf einen gegebenen Datenraum entscheiden bzw. eine automatische Abbildung zwischen kategoriellen und graphischen Attributen vornehmen. Ein entsprechender Algorithmus wird jedoch mit steigender Anzahl von Attributen sowie Möglichkeiten zur Nutzung von Metadaten schnell recht komplex, rechenintensiv und mehrdeutig.

ergebe sich als $f^{\text{vis}}(\widetilde{dr}.f^{\text{ext}}_{sa^{\text{kz}}}(k_1,\ldots,k_{n+v}))$. Da bei dieser Vorgehensweise — gerade bei geringer Dimensionalität des verarbeiteten Datenraums — nicht alle verfügbaren graphischen Attribute mit Werten belegt werden müssen, sind für diese ggf. Default–Vorgaben vorzusehen.

Ein einfaches Beispiel: Die Graphik in Abb. 5.6 könnte aus einem zweidimensionalen Datenraum erzeugt worden sein, der Bevölkerungszahlen und Erkrankungsraten nach Alter und Geschlecht enthält. Hierbei besteht KA^{dat} aus dem Geschlecht und dem Kennzahlattribut, für KA^{val} verbleibt das Alter. Es gibt zwei Symboldefinitionen (Linien und Balken). Das Alter ist jeweils über die Untergrenze eines Altersintervalls der x–Koordinate und der Zelleninhalt (#) über einen bestimmten Skalierungsfaktor jeweils der y–Position zugeordnet. Das Geschlecht wird — in beiden Fällen über eine Ordnung gemäß „weiblich = 1, männlich = 2“ — im Liniendiagramm durch einen Linienstyle und im Balkendiagramm durch einen x–Offset repräsentiert. Alle Datenreihen zur Bevölkerung verwenden die Linien–Symboldefinition und alle Datenreihen zur Fallzahl die Balken–Symboldefinition.

Im allgemeinen können verschiedene Symboldefinitionen einer Graphik unterschiedlich viele graphische Attribute vorhalten. Somit sind prinzipiell auch Daten verschiedener Dimensionalität in einer Graphik visualisierbar (z. B. kleine Tortendiagramme zu allen Gebieten einer thematischen Karte). Da die Abbildung f^{ka} oben bereits als nicht surjektiv modelliert wurde, stellt die Realisierung derartiger Graphiken kein Problem dar. Eine Implementierung sollte natürlich die Menge der Symbole einer Datenreihe auf das Kreuzprodukt der jeweils *relevanten* kategoriellen Attribute einschränken.

Nicht näher eingegangen wird an dieser Stelle auf Beschriftungen, Titel, Skalen sowie weitere Darstellungskomponenten und Layout–Parameter von Graphiken, die oftmals für ein Visualisierungsverfahren sehr spezifisch zu definieren sind, teilweise nur optionalen Charakter haben und eine große Palette von Zuordnungsmöglichkeiten bieten (vgl. die in Abschnitt 2.1.3 angegebenen Referenzen zur automatisierten Graphikerstellung). Im folgenden seien einige grundlegende Aspekte kurz angerissen:

- Jede Datenreihe kann einen Text erhalten (gebildet i. a. aus den jeweiligen Ausprägungen der Attribute aus KA^{dat} und evtl. zugeordneten Metadaten, z. B. Einheiten). Die Beschriftung erfolgt direkt an der Datenreihe oder in einer separaten Legende.

- Auch jedes einzelne Symbol kann mit den entsprechenden Koordinaten und Ausprägungen oder zusätzlichen Metadaten, wie etwa Fußnoten, beschriftet werden.

- Zu jeder Kombination von graphischem Attribut, zugeordnetem kategoriellen Attribut und zur Visualisierung herangezogener Abbildung $f^{\text{T}}_{ka,sd}$ wird in der Regel eine Skala oder eine Legende definiert, die wiederum mit Titel und Instanzen in der Form beschriftet ist, daß die Zuordnung zwischen Kategorien und den Ausprägungen des genutzten Datentyps T bzw. des graphischen Attributs deutlich wird. Entsprechende Skalen sind u. U. an mehreren Stellen der Graphik zu plazieren, falls ein graphisches Attribut einen Offset auf die durch ein anderes Attribut gegebene Koordinate definiert (etwa in geschachtelten Tabellen oder im Beispiel der Unterdiagramme zu Gebieten einer thematischen Karte).

- Zur Darstellung der Zuordnung der Zelleninhalte (#) kann auch jedes summarische Attribut des Quelldatenraums pro in unterschiedlicher Weise genutzter Symboldefinition eine Skala spezifizieren. Summarische Attribute über dieselbe Maßzahl bzw. denselben Datentyp (falls Skalierungsfaktor und Einheit identisch sind) können auch Skalen gemeinsam nutzen.

- Weitere Layout- und Skalierungsparameter können teilweise automatisch aus dem jeweiligen Datenraum ermittelt, teilweise aber auch interaktiv über interne Parameter vorgegeben werden.

Unter Umständen können auch weitere, dem visualisierten Datenraum oder seinen Komponenten zugeordnete Metadaten die Gestaltung der Graphik oder auch die Beurteilung der Anwendbarkeit eines Visualisierungsverfahrens beeinflussen.

Interaktive Nutzung von Graphiken

In [Lee94] werden unterschiedliche Ausprägungen der Interaktion zwischen Analysesystem und Anwender diskutiert:

1. Modifikation des Analyse- und Visualisierungsprozesses,
2. Modifikation der Transformationsparameter,
3. Interaktion mit der Ausgabe zur Veränderung des Prozesses und
4. Interaktion mit den Daten.

Während die beiden erstgenannten Aspekte bereits durch die Grundkonzepte von *VIOLA* (die datenflußbasierte Umgebung bzw. die Parametrisierung von Modulen) behandelt werden, sind die beiden anderen Punkte im Kontext von Visualisierungsmodulen zu sehen. So sollen in *VIOLA* Möglichkeiten zum Linking von Graphiken einerseits sowie zur Manipulation von Darstellungsparametern und zur Abfrage von Informationen anhand von Graphikkomponenten andererseits geboten werden, so daß dem Benutzer insgesamt eine umfassende, sich wechselseitig ergänzende Palette von Interaktionsalternativen zur Verfügung steht (vgl. [RCK+97]).

Die Parametrisierung von Graphiklayout und -gestaltung erfolgt über interne Parameter der jeweiligen Visualisierungsmodule. Werden mehrere Visualisierungsmodule in einem Verbundmodul zusammengefaßt, besteht die Möglichkeit, über Parameter des Verbundmoduls und dessen Zuordnungsfunktion f^{pp} (siehe Abschnitt 5.3.2) gleichzeitig die Darstellung mehrerer Graphiken zu steuern und so eine bzgl. dieser Parameter einheitliche Ansicht zu gewähren. Ferner ist es — je nach Visualisierungsverfahren — denkbar, die Einstellung bestimmter Parameter mittels direkter Manipulation von Graphikkomponenten zu ermöglichen. So könnte ein Zooming über direktes Ziehen oder Stauchen von Skalen oder die Einstellung von Farben oder Formaten über Pop–up–Menüs von Darstellungssymbolen erfolgen.

Darstellungssymbole sowie Skalen und Legenden bieten vielfältige Möglichkeiten zur Abfrage von Informationen. Ein Symbol kann die jeweils zugrundeliegenden Attributwerte anzeigen oder — unter Angabe von Objektmenge, Rolle und interessierenden Eigenschaften — ein Drill–through in die unterliegenden Mikrodaten realisieren (vgl. Abschnitt 4.3.6). Über Skalen und Legenden sind Metadaten zu Kategorien und Maßzahlen, insbesondere textuelle Erläuterungen, abrufbar.

Einen Schwerpunkt in der Konzeption von *VIOLA* bildet die Unterstützung des Linking von Graphiken (vgl. Abschnitt 2.1.3): Jede Graphik bietet die Möglichkeit, über interaktive Selektion ihrer Komponenten multidimensionale Parameter zu definieren, die dann über einen Parameterport an andere Module, insbesondere Datenmanagementmodule weitergeleitet werden können. Folgen auf diese Bausteine wiederum andere Visualisierungen, erhält man ein dynamisches Linking. Natürlich muß hierfür eine Propagierung von Selektionsergebnissen „in Echtzeit“ gewährleistet sein (vgl. Abschnitt 5.4). Beschränkt man diese Echtzeit–Verarbeitung auf den Pfad zwischen Datenmanagementmodul (mit einem geeigneten Widget zur Parametereingabe, etwa einem Schieberegler) und Visualisierungsmodul, ergibt sich auf Basis des gleichen Grundkonzepts eine Implementierung von Dynamic Queries.

Zur Definition multidimensionaler Parameter anhand von interaktiven Selektionen gibt es eine Vielzahl von Möglichkeiten, die je nach Visualisierungsverfahren unterschiedlich angeboten werden können. Typischerweise können über Darstellungssymbole oder Gruppen von Symbolen, aber auch über Skalen oder Legenden Kategorienmengen, Eigenschaftsmengen bzw. Domänen und/oder Maßzahlmengen (als Kategorien des Kennzahlattributs) ausgewählt werden. Aber auch fast alle anderen Arten multidimensionaler Parameter sind prinzipiell sinnvoll zu belegen und über entsprechende Ports zu nutzen.[33] Als Selektionswerkzeuge können z. B. Zeiger (für einwertige Parametertypen) oder Selektionsboxen (für mengenwertige Typen) zum Einsatz kommen.

Implementierungen des Linking in bestehenden Systemen (vgl. Kapitel 3, insb. Abschnitt 3.2.1) definieren in der Regel implizite Verbindungen zwischen allen „zueinander passenden“ Graphiken. Demgegenüber

[33] Von allen Möglichkeiten zur Parameterdefinition muß bzw. sollte natürlich nicht gleichzeitig Gebrauch gemacht werden, um die Übersichtlichkeit der Nutzung des jeweiligen Visualisierungsmoduls bzw. des *VIOLA*–Programms zu wahren.

wird in *VIOLA* ein Linkingpfad über entsprechende Kanäle explizit sichtbar und wählbar gemacht. Dies bietet sicherlich Vorteile im Hinblick auf eine flexible Nutzung dieses Konzepts, trägt aber auch den Nachteil in sich, daß Kombinationen mehrerer zu linkender Graphiken (und auch die Bidirektionalität des Linking zwischen zwei Graphiken) explizit durch eigene Kanäle zu definieren sind, was einen gewissen Aufwand und u. U. Unübersichtlichkeit nach sich zieht. Darüber hinaus sind jedoch noch eine ganze Reihe von Vorzügen der Linking–Implementierung in *VIOLA* anzuführen:

- In Verbindung mit beliebigen Managementmodulen oder anderen parametrisierbaren Bausteinen ist nicht nur — wie in anderen Systemen oftmals als einzige Variante vorgesehen — eine interaktive Restriktion, sondern unter anderem auch eine Gruppierung von Kategorien, eine Verknüpfung unterschiedlich granularer Angaben (z. B. eine Selektion von Gemeinden über übergeordnete Landkreise) oder auch eine Auswahl an anderer Stelle zu selektierender oder zu berechnender Maßzahlen möglich.

- Je nach Weiterleitung der interaktiv spezifizierten Parameter kann nicht nur ein Highlighting von Teilen anderer Graphiken, sondern auch ein Ersetzen oder Hinzufügen von Inhalten erfolgen.

- Zwischen den „gelinkten“ Darstellungen können beliebige Berechnungen in anderen Modulen erfolgen (ähnlich dem Ansatz des algebraischen Linking).

- Ein Linking kann — im Gegensatz zur Beschränkung auf einen gemeinsamen Basisdatenraum in vielen bestehenden Analyse- und Visualisierungssystemen — zwischen Darstellungen über beliebigen verschiedenen Datenräumen erfolgen, die nicht aus den gleichen Quelldaten berechnet sein müssen. Die beteiligten Datenräume müssen lediglich über einzelne Parameter miteinander in Beziehung stehen, z. B. ein kategorielles Attribut über der gleichen Domäne aufweisen.

- Selektionen aus verschiedenen Graphiken können über mehrere Managementmodule logisch miteinander verknüpft werden (vgl. die Diskussion von Selektionssequenzen in [The96b]), z. B. können zwei hintereinandergeschaltete Restriktionsmodule Angaben aus zwei verschiedenen Graphiken im Sinne eines Und–Operators kombinieren. Um den Laufzeitaufwand bei der Berechnung intermediärer Datenräume einzusparen, wird in Abschnitt 5.4.2 ein Konzept zur Verknüpfung von Parametermengen aus mehrfachen Eingängen in Parameterports (mittels Vereinigung, Durchschnitt etc.) eingeführt werden.

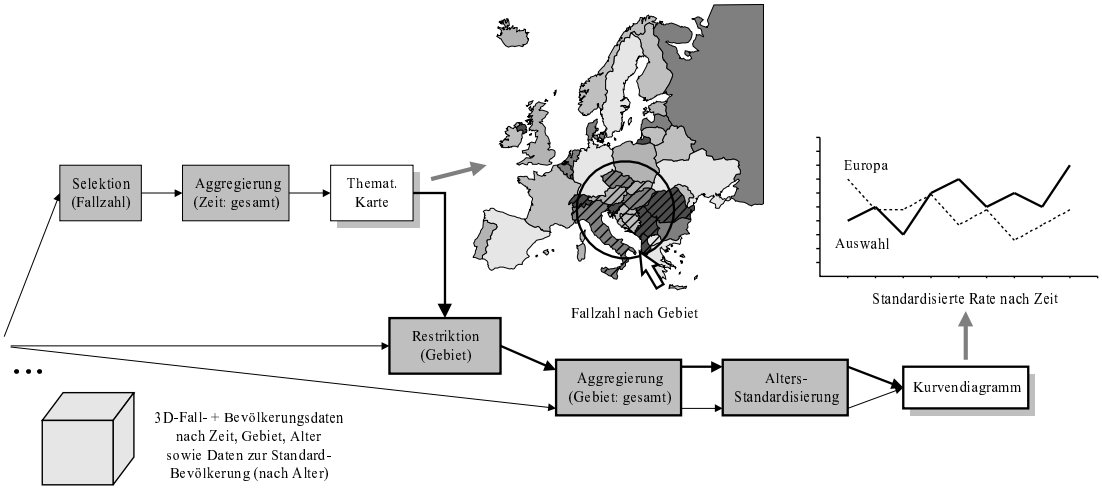

Abbildung 5.7: Schematische Darstellung des Linking in *VIOLA*

Abbildung 5.7 zeigt ein einfaches Beispiel für unidirektionales Linking in *VIOLA*: Über die Wahl eines Teilgebiets von Europa wird eine Kurve eines Kurvendiagramms parametrisiert. Module und Kanäle, die ihre

Daten in Echtzeit propagieren müssen, sind fett gezeichnet.[34]

Einige alternative Ansätze zur Implementierung des Linking in einer datenflußbasierten Analyseumgebung werden in [Bra98b] vorgestellt und diskutiert, etwa auch das implizite, multidirektionale Linking aller Visualisierungsbausteine in Teilbereichen eines Analysegraphen auf der Basis eines gemeinsamen Vorgängermoduls. Bei der Angabe expliziter Linkingpfade werden Visualisierungsbausteine stets direkt miteinander verbunden, wobei das gewünschte Restriktionsverfahren in den Zielbaustein quasi integriert ist. Ein derartiges Vorgehen begründet sich teilweise auch darin, daß in [Bra98b] ausschließlich multidimensionale Datenräume auf Kanälen eines Analysegraphen propagiert werden. Die Erweiterung um die explizite Propagierung multidimensionaler Parameter sowie die separate Spezifikation von Datenmanagementmodulen in *VIOLA* trägt demgegenüber wesentlich zur Klarheit und flexiblen Nutzung des Linking bei.

5.2.4 Definition und Nutzung multidimensionaler Parameter

Bereits zu Beginn von Abschnitt 5.2 wurde die große Bedeutung multidimensionaler Parameter für die Datenverarbeitung in *VIOLA* hervorgehoben. Auch deren Definition und Verwendung soll in einer einheitlichen datenflußbasierten Sichtweise visualisiert werden. Während die Semantik und Verarbeitung multidimensionaler Datenräume jedoch exakt durch das Datenmodell *MADEIRA* beschrieben wird, erscheint eine entsprechend genaue Formalisierung der Parameterdefinition (etwa auch im Rahmen von *MADEIRA*) nicht nur schwierig, sondern auch nicht angemessen. Hier wird nämlich nicht das „Was“ und „Wie“, sondern eher das „Warum“ der Datendefinition beschrieben, z. B. bei der Auswahl zu untersuchender Gebiete um eine Emissionsquelle oder bei der Selektion von Zeiträumen mit auffallend hohen Erkrankungsraten zur näheren Betrachtung. Derartige Zusammenhänge sind in ihrer Vielfalt und Komplexität kaum exakt spezifizierbar; auch sind die Grenzen zwischen willkürlich bzw. intuitiv vorgenommenen Parameterangaben und nach konkreten Algorithmen ermittelten Ausprägungen nicht immer klar zu ziehen. Umso mehr Bedeutung gewinnt in diesem Kontext die Angabe von (textuellen) Metadaten als Hintergrundinformation zu selektierten Parametern und hierüber erzeugten Datenräumen.

Vor diesem Hintergrund ergibt sich eine recht einfache Definition von Modulen zur Definition von (externen) Parameterwerten. Diese ergänzen die Möglichkeiten zur direkten Verwendung von Datenräumen als Repräsentanten für über deren Komponenten definierte Parameter (vgl. Tab. 5.2) sowie die direkte interaktive Manipulation. Außerdem dienen spezielle Parametermodule — ähnlich dem Im- und Export von Datenräumen in Verbundmodulen durch Module aus $\mathcal{MOD}^{\text{in}}$ und $\mathcal{MOD}^{\text{out}}$ — als Schnittstellen zur externen Umgebung von Verbundmodulen. Man beachte, daß durch diese Schnittstellenmodule sowohl Parameterwerte in das Verbundmodul hinein- als auch wieder herausleiten können (vgl. die Spezifikation von Verbundmodulen in Abschnitt 5.3.2).

Definition 5.22 (Parametermodule) *Ein* PARAMETERDEFINITIONSMODUL *zur Bestimmung eines Parameters* $par \in \mathcal{PAR}^{\text{ext}}$ *sei definiert als* $mod = (b, po^{\text{in}}, po^{\text{out}}, PO^{\text{par}}, po^{\text{par}}, PAR^{\text{int}}, f^{\text{par}})$*, wobei* $po^{\text{in}}.card = 1$ *und* $po^{\text{par}} = (b', par, mod, false)$ *— die Werte von par können also nicht importiert werden. Unter Umständen können Datenein- und -ausgang auch entfallen.* $\mathcal{MOD}^{\text{defpar}}$ *sei die Menge aller Parameterdefinitionsmodule.*

Ein PARAMETERSCHNITTSTELLENMODUL *sei definiert als ein Tripel* $mod^{\text{iopar}} = (b, po^{\text{par}}, po^{\text{sub}})$*, wobei* $po^{\text{sub}}.mod = mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}}$*, d. h. das Parametermodul referenziert einen Parameterport eines Verbundmoduls.* $\mathcal{MOD}^{\text{iopar}}$ *sei die Menge aller Parameterschnittstellenmodule.*

$\mathcal{MOD}^{\text{par}} = \mathcal{MOD}^{\text{defpar}} \cup \mathcal{MOD}^{\text{iopar}}$ *sei die Menge aller* PARAMETERMODULE.

Die automatische Parameterdefinition erfolgt also in der Regel anhand eines einzelnen Datenraums, der (gemäß standardmäßig als Identität definierter Berechnungsfunktion f^{dat}) selbst unverändert durch das Modul weitergereicht wird.[35] Der Datenausgang eines Parametermoduls ist im Prinzip natürlich überflüssig, vereinfacht aber — ebenso wie der konsequente Export multidimensionaler Parameter — die Programmerstellung im

[34] Von den bereits angesprochenen Problemen bei der Erstellung thematischer Karten in *VIOLA* wird hier abstrahiert.

[35] Ein ähnliches Konzept von „Extraktoren“ findet sich auch in *KHOROS*/*Cantata*. Jedoch werden dort im Gegensatz zu *VIOLA* Parameterwerte in globalen Variablen abgelegt.

Hinblick auf Layout und Datenkombination. Entfallen Datenein- und -ausgang, bietet das Parametermodul die Möglichkeit zur Parameterberechnung aus anderen Parametern oder zur interaktiven Parameterwahl, die klar von Berechnungsvorgängen in anderen Modulen separiert ist, und so zur übersichtlicheren Programmgestaltung beiträgt.

Die konkrete Definition der Parameterberechnung in einem Parametermodul sollte, wie schon die Datenraumverarbeitung in Modulen zum Datenmanagement und zur Maßzahlberechnung, möglichst generisch und allgemein auf unterschiedliche Datenräume anwendbar gehalten werden. So könnten etwa durch Vorgabe einer Eigenschaft alle Kategorien aus einem entsprechenden kategoriellen Attribut extrahiert werden. Auch hier können Mehrdeutigkeiten auftreten, etwa wenn man anstelle einer Eigenschaft eine Domäne (und somit Eigenschaftsmengen) als Parameter verwenden würde.

Ein weiteres Beispiel wäre die Selektion der *n* Kategorien eines eindimensionalen Datenraums zu Teilpopulationen mit den größten (kleinsten) Ausprägungen einer Maßzahl[36] oder weitergehend die Klassifikation von Kategorien nach Maßzahlwerten (vgl. hierzu auch das im Zusammenhang mit kategoriellen Datenquellen im Anschluß an Def. 5.13 angeführte Beispiel). Schließlich können auch als Grundlage verschiedener Kontrollstrukturen (siehe Abschnitt 5.2.5) boolesche Werte in Parametermodulen definiert werden.

Wie man an obigen Beispielen sieht, können Parameter sowohl allein über das Schema eines Datenraums, also die (summarischen und kategoriellen) Attribute sowie die durch den Datenraum beschriebenen Objektmengen, als auch über die Extension, also die konkreten Maßzahlwerte, definiert sein. Diese Unterscheidung wird noch eine Rolle für die Verarbeitung von *VIOLA*–Programmen spielen (siehe Abschnitt 5.4).

5.2.5 Einige einfache Kontrollstrukturen

VIOLA soll und kann keine berechnungsvollständige Sprache für allgemeine Problemstellungen liefern. Die Programmierumgebung beschränkt sich zum einen auf die Unterstützung typischer Abläufe explorativer Datenanalysen und zum anderen auf die Verknüpfung vordefinierter, teilweise funktional mächtiger Bausteine, in denen komplexere Verarbeitungsalgorithmen codiert und gekapselt sind. Einfache Kontrollstrukturen sollen in *VIOLA* in erster Linie dazu dienen, dem Anwender Hilfestellungen bei der Programmausführung zu bieten, indem standardisierte Interaktionsfolgen zusammengefaßt oder abgekürzt werden können. Wichtig bleibt bei alledem die Gewährleistung übersichtlicher Programmstrukturen. Im einzelnen sind Kontrollmodule für folgende Aufgaben vorgesehen:

- die Auswahl eines weiterzuleitenden Datenraums aus einer Datenraummenge mittels eines Routing–Operators,
- die bedingte Verzweigung zwischen zwei Alternativpfaden,
- das Durchlaufen einfacher Schleifen über eine iterative Variation von Modulparametern sowie
- der vorzeitige Abbruch von Schleifen.

Alle genannten Operatoren sind auf typische Anwendungsfälle im Rahmen von Datenanalysen zugeschnitten:

- Das *Routing* faßt die Nacheinanderausführung von Vereinigung und Selektion zusammen. Ein Routingmodul kann sinnvoll angewendet werden, wenn aus einer Menge von Datenräumen ähnlicher Gestalt mit bereits getrennt berechneten Maßzahlen auf einfache Weise (eben über ein Modul, das quasi als „Schalter“ fungiert) ein Datenraum mit *einer* näher interessierenden Maßzahl zur weiteren Verarbeitung dynamisch, im flexiblen Wechsel ausgewählt werden soll.

- Die bedingte *Verzweigung* dient zur Wahl zwischen zwei alternativen Analysepfaden anhand eines beliebig definierten booleschen Flags, das sich in der Regel aus dem jeweils betrachteten Datenraum ermittelt. Hierdurch können einerseits Analysen an bestimmten Stellen abgebrochen und andererseits Analyseschritte flexibel von Dateninhalten abhängig gemacht werden. Vor allem aus zwei Gründen wird darauf

[36] Dies schließt die entsprechende Lücke in *MADEIRA* (vgl. Abschnitt 4.3.2), die aufgrund fehlender Formalisierbarkeit wertebasierter Restriktionen gelassen wurde.

verzichtet, auch mehrfache Verzweigungen (im Sinne einer Case–Anweisung) zu realisieren: Zum einen bestünde die Gefahr, daß entsprechende *VIOLA*–Programme recht unübersichtlich werden, und zum anderen gäbe es für ein „Case“ auch nur wenige sinnvolle Anwendungsszenarien. Eine diesbezügliche Erweiterung von *VIOLA* wäre konzeptionell jedoch relativ problemlos möglich.

Das Verzweigungsmodul ist das einzige Modul, in dem ein Dateneingang nicht zur Bestimmung des Inhalts eines *einzelnen* Datenausgangs dient. Aus diesem Grunde können durch dieses Modul auch nicht unabhängig voneinander mehrere Eingänge bearbeitet werden.

- *Schleifen* in *VIOLA* sollen ausdrücklich nicht dazu dienen, Algorithmen zur Datenanalyse zu implementieren, sondern lediglich die wiederholte Durchführung von Auswertungssequenzen mit unterschiedlichen Parametrisierungen in Form eines „*Forall ... do ...* “ unterstützen, wie sie etwa bei der Erstellung von Berichten in Krebsregistern oder der Durchführung eines standardisierten Inzidenzmonitoring anfallen. In der Terminologie von Ambler und Burnett in [AB90] handelt es sich hierbei stets um eine horizontal parallele Iteration, d. h. das Ergebnis eines Zyklus beeinflußt nicht den nächsten Durchlauf.[37] In dieser Form weisen Schleifen eine einfach zu definierende Semantik auf und sind leicht verständlich. Zudem erfolgt in *VIOLA* eine Kapselung von Schleifen in Verbundknoten (siehe Abschnitt 5.3.2) im Sinne strukturierter Programmierung ähnlich wie in *LabVIEW*. Sieht man einmal — wie bei allen anderen Modulen auch — von der internen bzw. benutzerdefinierten Parametrisierung sowie vom Zwischenspeichern von Ergebnissen zur Effizienzsteigerung ab, können somit auch Schleifen als (von außen betrachtet) zustandslose Teilmodule angesehen werden.

  Die eigentliche Schleife in *VIOLA* wird als ein „Unterprogramm“ durch einen Verbundknoten realisiert. Dieser Verbund enthält ein oder mehrere Schleifenmodule, die ihrerseits jeweils lediglich in jedem Durchlauf die nächste Ausprägung eines (geordneten) mengenwertigen multidimensionalen Parameters des Verbundmoduls bereitstellen.[38] So können zur Erstellung eines Krebsregister–Jahresberichts Auswertungsmengen nacheinander für verschiedene Krebsdiagnosen durchgeführt und Ergebnisse in entsprechenden Datensenken abgespeichert werden. Ein anderes Beispiel bilden Folgen von Aggregierungsebenen einer Domäne, auf denen sukzessive verfeinernd oder vergröbernd nach Auffälligkeiten etwa der Erkrankungshäufigkeiten gesucht wird. Als Ergänzung der Schleifenmodule können *Abbruchmodule* eingesetzt werden, die bei Erfüllung beliebig (über Parameterdefinitionsmodule) spezifizierter boolescher Bedingungen und die entsprechende Belegung eines Abbruchparameters die Schleife vorzeitig beenden können. An den Ausgangsports des Verbundmoduls, das die Schleife realisiert, liegt nach deren Beendigung das Ergebnis des letzten Durchlaufs. Im Zuge der Formalisierung der Verarbeitung von *VIOLA*–Programmen in Abschnitt 5.4 werden wir diese Zusammenhänge noch verdeutlichen.

Es folgt die formale Definition der zur Realisierung von Kontrollstrukturen in *VIOLA* benötigten Module:

Definition 5.23 (Kontrollstrukturen in *VIOLA*) *Ein* ROUTINGMODUL *sei definiert als* $mod = (b, PPO, PO^{\text{par}}, f^{\text{dat}})$, *wobei* $PPO \subseteq \mathcal{PO}^{\text{dat}} \times \mathcal{PO}^{\text{dat}}$, $PO^{\text{par}} = \{mass: \mathcal{M},\ rollenobj: 2^{\mathcal{OR}}\} \subseteq \mathcal{PO}^{\text{par}}$, *und für alle* $(po^{\text{in}}, po^{\text{out}}) \in PPO$ *gelte* $po^{\text{in}}.card = *$ *sowie*

$$f^{\text{dat}}(data(po^{\text{in}}), mass, rollenobj) \stackrel{\text{def}}{=} \begin{cases} dr & \text{falls } dr \in data(po^{\text{in}}) \wedge dr.SA \cap \mathcal{SA}_{\{mass\},rollenobj} \neq \emptyset, \\ undef. & \text{sonst.} \end{cases}$$

$\mathcal{MOD}^{\text{route}}$ *sei die Menge aller Routingmodule.*

Ein VERZWEIGUNGSMODUL *sei definiert als* $mod = (b, po^{\text{in}}, po_1^{\text{out}}, po_2^{\text{out}}, po^{\text{par}}, f^{\text{dat}})$, *wobei* $po^{\text{in}}.card = 1$

[37] Vgl. auch komplexere Verarbeitungsmodelle in (großteils feinergranularen) datenflußbasierten Sprachen in [Bar92, AB90, Sha92a, HYT+92, AB92].

[38] Mehrere Schleifenmodule in einem Verbundmodul durchlaufen also stets *synchron* unterschiedliche Parametermengen, bis die erste dieser Mengen vollständig abgearbeitet ist.

und $po^{\text{par}} = flag: \mathbb{B}$ *und*

$$f^{\text{dat}}(data(po^{\text{in}}), flag) \stackrel{\text{def}}{=} \begin{cases} (data(po^{\text{in}}), undef.) & falls\ flag = true, \\ (undef., data(po^{\text{in}})) & sonst, \end{cases}$$

d. h. je nach dem Wert des Parameterflags wird einmal der eine und einmal der andere Ausgang mit dem Eingangsdatenraum belegt. $\mathcal{MOD}^{\text{if}}$ *sei die Menge aller Verzweigungsmodule.*

Ein SCHLEIFENMODUL *sei definiert als* $mod^{\text{loop}} = (b, PO^{\text{par}}, po^{\text{sub}})$*, wobei* $PO^{\text{par}} = \{po^{\text{par}}, po^{\text{par}\prime}\}$ *mit* $po^{\text{par}}, po^{\text{par}\prime}, po^{\text{sub}} \in \mathcal{PO}^{\text{par}}$ *und* $po^{\text{sub}}.mod \in \mathcal{MOD}^{\text{sub}}$*. Es seien* $po^{\text{par}}.in? = po^{\text{par}\prime}.in? = false$*,* $po^{\text{par}}.par = (current, dom, mod^{\text{loop}})$ *und* $po^{\text{par}\prime}.par\prime = (next, dom, mod^{\text{loop}})$ *für einen beliebigen Wertebereich* $dom \subseteq \Omega$*, wobei jedoch* $po^{\text{sub}}.par.dom = 2^{dom}$*.* $\mathcal{MOD}^{\text{loop}}$ *sei die Menge aller Schleifenmodule.*

Ein ABBRUCHMODUL *sei definiert als* $mod^{\text{stop}} = (b, po^{\text{par}}, mod^{\text{sub}})$*, wobei* $mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}}$ *ein Verbundmodul und* $po^{\text{par}} = stop? : \mathbb{B}$ *ein boolescher Parameter, der angibt, ob die jeweilige Schleife vorzeitig abzubrechen ist.* $\mathcal{MOD}^{\text{stop}}$ *sei die Menge aller Abbruchmodule.*

$\mathcal{MOD}^{\text{ctrl}} = \mathcal{MOD}^{\text{route}} \cup \mathcal{MOD}^{\text{if}} \cup \mathcal{MOD}^{\text{loop}} \cup \mathcal{MOD}^{\text{stop}}$ *sei die Menge aller* KONTROLLMODULE.

Man beachte, daß die Definition des Routing wiederum nichtdeterministisch ist, falls mehrere vorliegende Datenräume die gesuchte Maßzahl enthalten. Die Definition der Datenraumauswahl orientiert sich eng an der Umsetzung der Selektionsoperation (siehe Def. 5.9). Im Gegensatz zur Auswahl von Dateneingängen durch Eingangsmodule von Verbundmodulen (siehe Def. 5.13) wird hier mangels sinnvoller Anwendungsbeispiele darauf verzichtet, auch die Beschreibung kategorieller Attribute in die Datenraumauswahl einfließen zu lassen.

Eine notwendige und hinreichende Anwendbarkeitsbedingung für Schleifenmodule besteht darin, daß der zugeordnete Parameter des Verbundmoduls eine nicht–leere Liste von Werten enthält. Um auch Berechnungen zu ermöglichen, die beim Eintreten einer Abbruchbedingung die Ergebnisse des *vorangegangenen* Schleifendurchlaufs zur weiteren Verarbeitung (an den Ausgängen des Verbundmoduls) zur Verfügung stellen, erhalten Schleifenmodule einen zweiten Parameter *next*. Dieser enthält jeweils das auf den aktuellen, durch den Parameter *current* repräsentierten Wert folgende Listenelement. Gibt es kein nächstes Element, sind beide Parameterwerte gleich. Somit können gleichermaßen Repeat– und While–Schleifen implementiert werden.

Eine Erweiterung von *VIOLA* um zusätzliche Kontrollstrukturen oder durch Bereitstellung mächtigerer Varianten der vorgestellten Module ist prinzipiell denkbar, sollte aber die Sprache nicht für einzelne Spezialfälle unnötig kompliziert machen, sondern an den wesentlichen Bedürfnissen der explorativen Datenanalyse als zentralem Anwendungsgebiet orientiert sein und sich homogen in die bisherige eingängige Handhabung von Datenräumen und multidimensionalen Parametern sowie das zugrundeliegende Datenflußkonzept einpassen (vgl. [Hil93]).

5.3 Struktur von *VIOLA*–Programmen

Nachdem in den vorangegangenen Abschnitten die Basismodule von *VIOLA* im Detail vorgestellt wurden, soll uns nun interessieren, wie diese Module zur Beschreibung von Datenanalyseabläufen gruppiert und miteinander verbunden werden können. Als Beschreibungsformalismus hierzu bieten sich gerichtete Graphen an, die unmittelbar die Propagierung von Daten in einem Datenflußprogramm widerspiegeln. Da Module innerhalb eines Datenflußprogramms nahezu beliebig miteinander verbunden werden können, verzichten wir auf die Angabe einer erzeugenden (Graph–)Grammatik. Ein gerichteter Graph ist als deutlich einfachere Notation völlig ausreichend und bietet zudem eine solide Grundlage zur operationalen Spezifikation der Sprachsemantik.

Im Anschluß an die Definition von Analysegraphen zur Repräsentation von Netzwerken aus *VIOLA*–Modulen in Abschnitt 5.3.1 werden in Abschnitt 5.3.2 Verbundmodule eingeführt, die ihrerseits Datenanalysen in Form von Unterprogrammen kapseln. Abschließend faßt Abschnitt 5.3.3 nochmals alle *VIOLA*–Module in einer Übersicht zusammen.

5.3.1 Datenfluß in gerichteten Graphen

Ein Analysegraph dient zur Modellierung von Netzen aus *VIOLA*–Modulen als gerichtete Graphen, wobei Kanten zwischen Modulen jeweils zwischen Daten- oder Parameterports verlaufen. Unter den Parameterports können nur solche mit *kompatiblen* Parametertypen (vgl. Def. 5.2) miteinander verbunden werden.

Definition 5.24 (Analysegraphen) *Ein* ANALYSEGRAPH $ag = (MOD, REL^{\text{dat}}, REL^{\text{par}})$ *sei definiert durch eine endliche Menge* $MOD \subseteq \mathcal{MOD}$ *von Modulen sowie zwei* VERBINDUNGSRELATIONEN*, die die Kanten zwischen Daten- bzw. Parameterports von Modulen aus MOD spezifizieren.*

Es seien $PO^{\text{in}} \stackrel{\text{def}}{=} \{po \in \mathcal{PO}^{\text{dat}} \mid po.mod \in MOD \wedge po \in po.mod.PO^{\text{in}}\}$ *und analog* PO^{out} *und* PO^{par} *die Mengen aller Eingangs-, Ausgangs- und Parameterports von Modulen aus mod. Hiermit seien* $REL^{\text{dat}} \subseteq PO^{\text{in}} \times PO^{\text{out}} \cup PO^{\text{out}} \times PO^{\text{in}}$ *und* $REL^{\text{par}} \subseteq PO^{\text{par}} \times \{po \in PO^{\text{par}} \mid po.in? = true\}$.

Paare $(po_1, po_2) \in REL^{\text{dat}} \cup REL^{\text{par}}$ *sollen als* DATEN- *bzw.* PARAMETERKANÄLE *bezeichnet werden; für sie gelte stets* $po_1.mod \neq po_2.mod$*, d. h. Kanäle verlaufen nur zwischen verschiedenen Modulen. Schließlich gelte für Parameterkanäle stets, daß* $po_1.par$ *kompatibel zu* $po_2.par$ *ist.*

PO bezeichne die Vereinigung $PO^{\text{in}} \cup PO^{\text{out}} \cup PO^{\text{par}}$*. Dann heiße* $f^{\text{in(e)}} : PO \to 2^{PO}$ *mit* $f^{\text{in(e)}}(po) \stackrel{\text{def}}{=} \{po' \in \mathcal{PO} \mid (po', po) \in REL^{\text{dat}} \cup REL^{\text{par}}\}$ EXTERNER FAN–IN *eines Ports* $po \in PO$*. Der* EXTERNE FAN–OUT $f^{\text{out(e)}}$ *eines Ports sei analog definiert.*

Für alle $po \in PO^{\text{in}}$ *gelte* $| f^{\text{in(e)}}(po) | \leq po.card$*. Externer Fan–in von Parameterports sowie externer Fan–out aller Ports seien beliebig groß.*

$\mathcal{AG}$ *sei die Menge aller Analysegraphen.*

Von besonderem Interesse, gerade im Hinblick auf die Verarbeitung von Analyseprogrammen, ist sicherlich die Zyklenfreiheit von Analysegraphen. Prinzipiell sind Datenanalysen zyklenfrei zu modellieren: Ein Datenanalyseschritt folgt auf den anderen, ohne daß im gleichen Berechnungsablauf einmal erzeugte Ergebnisse wieder modifiziert werden — dies geschieht erst durch interaktive Parametermanipulation oder nachträgliche Erweiterung des Analyseprogramms und anschließende Neuberechnung. Aufgrund der Beschränkung ihrer Mächtigkeit ändern hieran auch die in *VIOLA* vorgesehenen Schleifen konzeptionell nichts: Sie ließen sich mit äquivalenter Semantik[39] auch als „parallele“, unabhängige Gruppen von Auswertungen gleicher Gestalt modellieren, da die Anzahl der Schleifendurchläufe stets endlich ist und Ergebnisse eines Durchlaufs nicht in den nächsten einfließen. Somit ist es legititm, Schleifen implementierende Verbundmodule als gekapselte Einheiten anzusehen und die durch sie definierten Teilgraphen wiederum separat hinsichtlich Zyklenfreiheit zu betrachten.

Lediglich in einem Fall sollen Zyklen zwischen Modulen eines *VIOLA*–Programms unterstützt werden, nämlich zur Modellierung des Linking von Graphiken. Beim Linking werden oft mehrere Graphiken wechselseitig durch entsprechende Parameterdefinitionen modifiziert, was sich nur durch zyklische Abhängigkeiten im Analysegraphen modellieren läßt. Allerdings erfolgen alle Parameteränderungen stets interaktiv und nicht automatisch, so daß die interne Programmausführung weiterhin keine Zyklen zu betrachten hat. In Verallgemeinerung des Linking sind in *VIOLA* Zyklen immer dann erlaubt, wenn sie über mindestens zwei[40] *rein interaktive* Parameterdefinitionen bzw. -ports führen. Hierunter fallen vor allem Visualisierungsmodule ($\mathcal{MOD}^{\text{vis}}$), nicht aber etwa Module aus $\mathcal{MOD}^{\text{defpar}}$. Konkret bedeutet dies, daß jeder durch REL^{dat} und REL^{par} in einem Analysegraphen gegebene Zyklus mindestens zwei Kanäle, die von einem Parameterport *po* mit $in? = false$ in einem Modul mit $f^{\text{par}}_{po} = f^{\text{id}}$ ausgehen. Die folgende Formalisierung dieses Sachverhalts ist insofern etwas restriktiver als nötig, als nicht berücksichtigt wird, daß in den meisten Modulen ein Dateneingang jeweils nur zur Berechnung *eines* einzelnen Ausgangsports herangezogen wird: Zyklen werden im wesentlichen über Module und nicht über Ports definiert. Dies vereinfacht zum einen die Definition und spätere Prüfung und zielt zum anderen auch auf eine übersichtlichere Programmgestaltung.

[39] Auch der vorzeitige Abbruch wäre über eine entsprechende Auswahloperation zu realisieren, die aus den Ergebnissen all dieser Zweige selektiert.

[40] Somit können die Daten, die einer interaktiven Parametereingabe zugrunde liegen (wie etwa eine graphische Darstellung beim Linking) nicht direkt selbst durch die Interaktion modifiziert werden.

Definition 5.25 (Zyklenfreiheit von Analysegraphen) *Sei $ag \in \mathcal{AG}$ ein Analysegraph und*

$$(po_1, mod_1, po_1'), \ldots, (po_n, mod_n, po_n') = (po_1, mod_1, po_1') \in ag.PO \times ag.MOD \times ag.PO,\ n \geq 2 \ ,$$

eine Folge von Ports und Modulen, die einen Zyklus von ag spezifiziert, also

$$\forall i = 1, \ldots, n-1 : po_i.mod = po_i'.mod = mod_i \wedge (po_i', po_{i+1}) \in REL^{\text{dat}} \cup REL^{\text{par}} \ .$$

Dann gelte stets $\exists i \neq j \in \{1, \ldots, n-1\}$:

$$mod_i.f^{\text{par}}_{po_i'} = mod_j.f^{\text{par}}_{po_j'} = f^{\text{id}} \wedge po_i', po_j' \in \mathcal{PO}^{\text{par}} \wedge po_i'.in? = po_j'.in? = false \ .$$

Da typische Analysegraphen — wie wir etwa in Abschnitt 6.3 anhand einiger Beispiele zur Modellierung umfangreicher Analyseanwendungen sehen werden — kaum mehr als hundert Module enthalten[41], kann die entsprechende Prüfung auf Zyklenfreiheit einfach implementiert und jeweils direkt zur Zeit der Programm–Manipulation durchgeführt werden (mit einem Aufwand, der maximal linear zur Modulanzahl wächst).

Ein *VIOLA*–Programm ist schließlich ein Analysegraph, der keine nur innerhalb von Verbundmodulen auftretenden Module enthält.

Definition 5.26 (*VIOLA*–Programme) *Ein VIOLA–*PROGRAMM *sei definiert als ein Analysegraph $ag \in \mathcal{AG}$, für den gilt $ag.MOD \cap (\mathcal{MOD}^{\text{loop}} \cup \mathcal{MOD}^{\text{stop}} \cup \mathcal{MOD}^{\text{in}} \cup \mathcal{MOD}^{\text{out}} \cup \mathcal{MOD}^{\text{iopar}}) = \emptyset$.*

Natürlich ist ein *VIOLA*–Programm nur dann auch tatsächlich ausführbar, wenn es Datenquellmodule aus $\mathcal{MOD}^{\text{src}}$ enthält und alle Berechnungsfunktionen auf die am jeweiligen Modul anliegenden Daten anwendbar sind. Hierauf sowie auf ähnliche Zusammenhänge wird in Abschnitt 5.4 näher eingegangen.

5.3.2 Prozedurale Abstraktion in Gestalt von Verbundmodulen

Bereits mehrfach in diesem Abschnitt wurde im Kontext der Definition anderer Modultypen der Bezug zu Verbundmodulen angedeutet. Mit obiger Definition von Analysegraphen kann nun auch deren Gestalt genau formalisiert werden: Verbundmodule kapseln Analysegraphen zu „Unterprogrammen“, um größere *VIOLA*–Programme besser zu strukturieren und übersichtlicher zu machen. Eine derartige Möglichkeit der Abstraktion und Gliederung wird von vielen Autoren — etwa im Hinblick auf die Lösung des Screen–Space–Problems — als ein wesentliches Qualitätsmerkmal visueller Programmiersprachen angesehen [Mye90, Hil93, GP96].

Eingangs-, Ausgangs- und Parameterschnittstellenmodule stellen die Verbindung des Analysegraphen innerhalb eines Verbundmoduls zu seiner Außenwelt her. Ähnliches gilt auch für Schleifenmodule, die — stets durch einen Verbund gekapselt — über die Komponenten eines externen Verbundparameters iterieren, wobei u. U. ein Abbruchmodul die Verarbeitung des Verbundes vorzeitig beenden kann. Im Sinne einheitlicher Modellierung und Implementierung erhält *jedes* Verbundmodul ein Abbruchmodul, das auch die Abarbeitung von Verbunden, die keine Schleifen implementieren, steuert. Beim Fehlen von Schleifenmodulen kann ein Abbruchmodul auf der Benutzungsoberfläche auch vor dem Anwender verborgen werden.

Besitzt ein Verbundmodul mindestens ein Eingangsmodul, so kann es beliebig viele Eingangsports unterstützen — die Datenverarbeitung erfolgt, wie allgemein in *VIOLA* üblich, für alle Eingänge unabhängig; Berechnungsergebnisse werden über die den jeweiligen Eingängen zugeordneten Ausgangsports aus dem Verbundmodul exportiert. Somit wird klar, daß ein Verbundmodul maximal ein Ausgangsmodul enthalten kann. Auch Verbundmodule ohne Ausgangsports (und ohne Ausgangsmodul) sind möglich, sogar solche gänzlich ohne Ein- und Ausgänge (falls etwa das gesamte *VIOLA*–Programm eine Schleife bildet oder ein Teilprogramm allein über externe Parameter gesteuert wird).

Die Kardinalität von Eingangsports, also die Anzahl aufzunehmender Datenräume, richtet sich stets nach der Anzahl der Eingangsmodule eines Verbundmoduls. Durch die Parametrisierung eines Eingangsmoduls

[41] Mit steigender Modulanzahl kann eigentlich immer eine sinnvolle Kapselung von Teilgraphen in Verbundmodulen erfolgen.

wird festgelegt, welcher Datenraum eines Eingangsports durch dieses Modul repräsentiert und in den internen Analysegraphen propagiert wird. Details der Umsetzung, insbesondere die Gewährleistung einer *bijektiven* Abbildung zwischen Datenräumen und Eingangsmodulen, werden in Abschnitt 5.4 diskutiert. Diese Realisierung der „Daten–Parametrisierung“ von Unterprogrammen hat den Vorteil, daß die unabhängige Verarbeitung von Dateneingängen auch in Verbundmodulen analog zu den anderen Modulen von *MADEIRA* realisiert werden kann. Weiterhin gestaltet sich die Benutzung von Verbundmodulen einfacher: Der Anwender muß nicht zwischen Eingängen mit verschiedener Semantik differenzieren. Schließlich kann und muß intern die Semantik der verschiedenen Daten–Argumente durch entsprechende Parametrisierungen von Eingangsmodulen klarer spezifiziert werden, als es etwa durch differenzierte Namensvergabe zu mehreren getrennten Moduleingängen möglich wäre. Natürlich erkauft man sich mit diesen Vorzügen den Nachteil, nicht beliebige Parametrisierungen von Verbundmodulen durch Eingangsdaten umsetzen zu können, denn nicht immer lassen sich mit den gegebenen Mitteln die jeweiligen Anforderungen an Eingangsdaten durch Eingangsmodule eindeutig und vollständig festlegen. Auch sind die Zusammenhänge zwischen eingehenden Kanälen am Verbundmodul und den entsprechenden Eingangsmodulen nur schwer zu visualisieren und nur implizit vom Benutzer zu spezifizieren. Insgesamt sollten jedoch für typische Anwendungen[42] die Vorteile die Nachteile überwiegen.

Jeder externe Parameter *par* eines Verbundmoduls kann über maximal ein Parameterschnittstellenmodul mod^{iopar} dem internen Analysegraphen zugänglich gemacht werden. Um nach außen hin ein mit anderen Modularten konsistentes Erscheinungsbild zu gewähren, darf das Schnittstellenmodul für Verbundparameter, die eine Wertebelegung „von außen“ (also $in? = true$) zulassen, keinen schreibenden Zugriff durch andere Module innerhalb des gekapselten Analysegraphen gewähren — es gilt also $mod^{\text{iopar}}.po^{\text{par}}.in? = false$. Somit bleibt f^{par} für diesen Verbundparameter die Identität. Der Umkehrschluß gilt entsprechend: Wenn innerhalb des Verbundes ein Parameter neu belegt werden kann ($mod^{\text{iopar}}.po^{\text{par}}.in? = true$) und somit eine nicht–triviale Berechnungsfunktion f^{par} definierbar ist, muß der entsprechende Verbundparameter $po^{\text{par}}.in? = false$ erfüllen, damit keine Konflikte durch interne Berechnung und externe Belegung auftreten können. Wiederum sind manuelle Eingaben in jedem Fall möglich. Sie werden — wie auch die berechneten oder importierten Parameterwerte — in Abhängigkeit des Flags $po^{\text{par}}.in?$ unter Vermeidung von Zyklen *entweder* vom „äußeren“ Verbundparameter zum „inneren“ Parameter des Schnittstellenmoduls ($in? = true$) *oder* von „innen“ nach „außen“ ($in? = false$) propagiert. Zusätzlich erfolgt natürlich auch noch die übliche Propagierung gemäß der Verbindungsrelation des jeweiligen Analysegraphen. Entsprechend dient mod^{iopar} im ersten Fall quasi als Parametereingangs- und im zweiten Fall als Parameterausgangsmodul.[43] Gemäß dieser Unterscheidung gilt auch für Verbundparameter, die als Grundlage für Schleifenmodule dienen (deren Ports immer $po^{\text{par}}.in? = false$ erfüllen), stets $po^{\text{par}}.in? = true$.

Verbundmodule bieten die Möglichkeit, die Schnittstellen zur Parametrisierung der in ihnen enthaltenen Teilmodule zusammenzufassen. Für die externen Parameter ist dies ohnehin bereits über Definitionsmodule möglich, die Parameterwerte an verschiedene andere Module exportieren. Interne Parameter eines Verbundmoduls sollen zusätzlich den Zugriff auf interne Parameter von einigen (nicht notwendigerweise allen) Teilmodulen an zentraler Stelle ermöglichen. Hierbei können ggf. auch mehrere Parameter mit gleichem Wertebereich zu einem zusammengefaßt und gemeinsam modifizierbar gemacht werden.

Definition 5.27 (Verbundmodule) *Ein* VERBUNDMODUL $mod^{\text{sub}} = (b, PO^{\text{dat}}, PO^{\text{par}}, PAR^{\text{int}}, ag, f^{\text{dat}}, f^{\text{par}}, f^{\text{pp}})$ *sei definiert durch*

- *einen Namen* $b \in \mathcal{B}$,
- *endliche Mengen* $PO^{\text{par}} \subseteq \mathcal{PO}^{\text{par}}$ *und* $PAR^{\text{int}} \subseteq \mathcal{PAR}^{\text{int}}$ *von Parameterports bzw. internen Parametern,*
- *einen Analysegraphen* $ag \in \mathcal{AG}$*, der das durch* mod^{sub} *repräsentierte Teilprogramm spezifiziert,*

[42] Dies sind etwa Schleifen, Definitionen zusammengesetzter Maßzahlen, Kapselungen von Datenbankzugriff und nachfolgenden Datenmanagementschritten oder Gliederungen von Analysegruppen mit jeweils kleinen „Schnittstellen“ — einige Beispiele hierzu werden wir in nachfolgenden Abschnitten dieses Kapitels sowie bei der Diskussion eines größeren Anwendungsszenarios in Abschnitt 6.3 noch vorstellen.

[43] Für die Parameterports des Verbundmoduls ist diese Unterscheidung nicht ganz so klar. So leitet etwa ein Parameterport, der in obigem Sinne als Eingang fungiert, Parameterwerte nicht nur in den Verbund hinein, sondern auch an andere mit dem Port über Parameterkanäle verbundene Ports weiter.

- *Berechnungsfunktionen* f^{dat} *und* f^{par}*, die durch die Komponenten von ag definiert sind,*
- *eine partielle, surjektive Abbildung* f^{pp}*, die internen Parametern innerhalb von ag, also* $PAR^{\text{int}\prime} \stackrel{\text{def}}{=} \bigcup\{mod.PAR^{\text{int}} \mid mod \in ag.MOD\}$ *interne Parameter von* mod^{sub} *über die Funktion* $f^{\text{pp}}: PAR^{\text{int}\prime} \to PAR^{\text{int}}$ *zuordnet, wobei* $\forall par \in PAR^{\text{int}\prime}: par.dom = f^{\text{pp}}(par).dom$*, und*
- *eine endliche Menge* $PO^{\text{dat}} \subseteq \mathcal{PO}^{\text{dat}}$ *von Datenports, die sich schreiben lassen als*
  - *(a)* $PPO \subseteq \mathcal{PO}^{\text{dat}} \times \mathcal{PO}^{\text{dat}}$ *(parallele Ein- und Ausgänge),*
  - *(b)* $PO^{\text{in}} \subseteq \mathcal{PO}^{\text{dat}}$ *(nur parallele Eingänge),*
  - *(c)* $\{po^{\text{out}}\}$ *mit* $po^{\text{out}} \in \mathcal{PO}^{\text{dat}}$ *(nur ein Ausgang) oder*
  - *(d) die leere Menge* $\emptyset$*.*

  In den Fällen (a) und (b) erfolge die Verarbeitung der Dateneingänge, wie bei den meisten Modulen üblich, parallel und unabhängig. Falls $|PPO|$ *bzw.* $|PO^{\text{in}}| > 1$*, muß somit die automatische Parameterberechnung durch* f^{par} *entfallen, und in Fall (a) ist* f^{dat} *für alle Ausgänge gleich definiert. Für alle* $po^{\text{in}} \in PO^{\text{in}}$ *gelte* $po^{\text{in}}.card = |\{mod^{\text{in}} \in ag.MOD \mid mod^{\text{in}} \in \mathcal{MOD}^{\text{in}}\}|$*.*

  In den Fällen (c) und (d) enthalte ag.MOD keine Eingangsmodule. Die Zahl der Ausgangsmodule in ag.MOD sei in den Fällen (a) und (c) auf maximal[44] *eines festgelegt; in den anderen Fällen gebe es keine Ausgangsmodule.*

ag.MOD enthalte genau ein Abbruchmodul $mod^{\text{sub}}.mod^{\text{stop}}$*. Für alle Eingangs-, Ausgangs- und Abbruchmodule in ag.MOD gelte* $mod.mod^{\text{sub}} = mod^{\text{sub}}$*, analog für alle Parameterschnittstellen- und Schleifenmodule* $mod.po^{\text{sub}}.mod = mod^{\text{sub}}$*. Ein Verbundmodul mit mindestens einem Schleifenmodul heiße auch* SCHLEIFENVERBUND*.*

Für Schleifen- und Parameterschnittstellenmodule $mod \in (\mathcal{MOD}^{\text{loop}} \cup \mathcal{MOD}^{\text{iopar}}) \cap ag.MOD$ *und für alle* $po^{\text{par}} \in mod.PO^{\text{par}}$ *gelte ferner* $mod.po^{\text{sub}}.in? = false \Leftrightarrow po^{\text{par}}.in? = true$*. Zudem sei* $mod.po^{\text{sub}}$ *unterschiedlich für alle Parameterschnittstellenmodule aus ag.MOD.*

Schließlich enthalte der Analysegraph eines Verbundmoduls keine Module aus einem direkt oder transitiv „übergeordneten" Graphen, also

$$\forall ag \in \mathcal{AG}\ \forall mod^{\text{sub}} \in (ag.MOD \cap \mathcal{MOD}^{\text{sub}}) \cup \{mod^{\text{sub}\prime}.ag.MOD \cap \mathcal{MOD}^{\text{sub}} \mid mod^{\text{sub}\prime} \in (ag.MOD \cap \mathcal{MOD}^{\text{sub}})\} \cup \ldots: mod^{\text{sub}}.ag.MOD \cap ag.MOD = \emptyset \ .$$

$\mathcal{MOD}^{\text{sub}}$ *sei die Menge aller Verbundmodule.*

Abbildung 5.8 zeigt ein Beispiel für den Aufbau eines Verbundmoduls und verdeutlicht insbesondere die implizite Werteübergabe zwischen Ports des Verbundmoduls einerseits und Eingangs-, Ausgangs- und Parameterschnittstellenmodulen andererseits. Zudem sind noch Schleifenmodule und ein Abbruchmodul in die Darstellung einbezogen.

Die Darstellung der Modulports erfolgt in Anlehnung an Abb. 5.2: Datenports sind trapezförmig an den Seiten des Moduls angeordnet, während Parameterports als teilweise vom Modul verdeckte senkrechte Balken erscheinen. Die verwendeten Icons symbolisieren die unterschiedlichen Module, die gewissermaßen den Rahmen für den (hier nicht visualisierten) Rest des gekapselten Analysegraphen (aus beliebigen anderen Modulen, insbesondere Speichermodulen zur Aufnahme von Ergebnissen einzelner Iterationen) bilden. Im Uhrzeigersinn von oben links beginnend sieht man zwei Schleifen-, ein als Eingang genutztes Parameterschnittstellen-, ein Ausgangs-, ein Abbruch-, ein als Ausgang genutztes Parameterschnittstellen- und zwei Eingangsmodule (eine Gesamtübersicht über alle in *VIOLA* verwendeten Icons findet sich in Tab. 5.5 in Abschnitt 5.4.6). An

[44] Eigentlich *genau eines* Ausgangsmoduls, lediglich während der Konstruktion des Analysegraphen eines Verbundmoduls kann u. U. ein Ausgangsmodul noch fehlen. Für die korrekte Verarbeitung eines Verbundmoduls muß außerdem kein Ausgangsmodul vorhanden sein — ggf. bleiben evtl. vorhandene Ausgangsports einfach immer unbelegt.

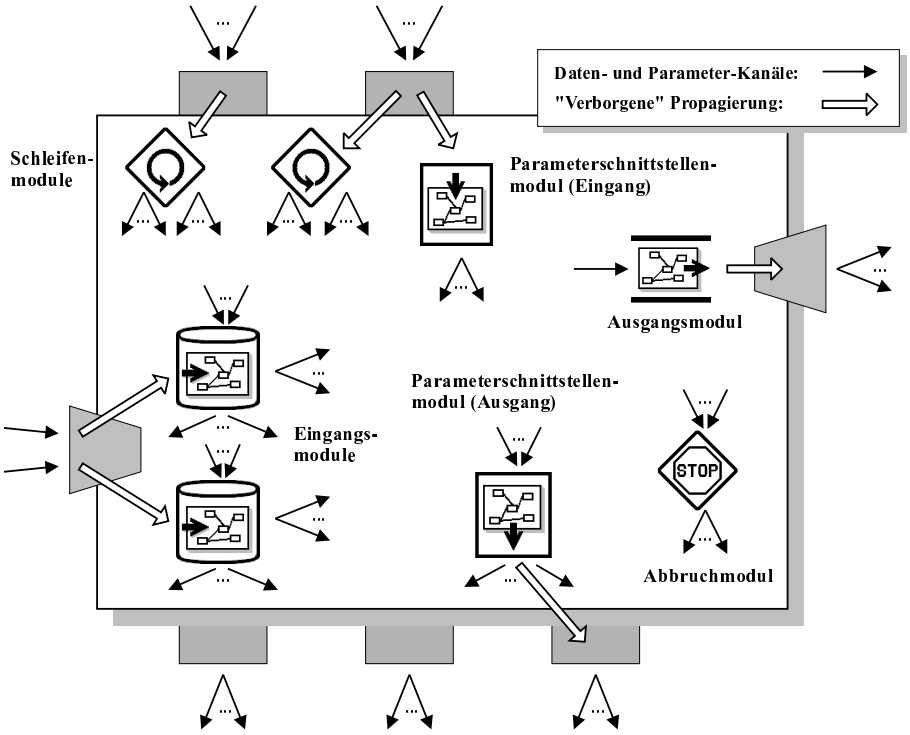

Abbildung 5.8: Ein Beispiel für die grobe Struktur eines Verbundmoduls

den Icons wurden jeweils keine Ports eingezeichnet, jedoch sollen auch hier Daten von links nach rechts und Parameter von oben nach unten fließen.

Das Verbundmodul hat in diesem Beispiel drei externe Parameter, von denen zwei Werte von anderen Modulen übernehmen können (*in*? = *true*). Beide sind mengenwertig und dienen zugleich als Grundlage für jeweils ein Schleifenmodul, das seinerseits innerhalb des Verbundes zwei Parameter *current* und *next* als Iteratoren über diese Wertelisten anbietet. Einer der beiden Parameterports liefert zusätzlich den Verbundparameter über ein Parameterschnittstellenmodul an den internen Analysegraphen. Der dritte externe Verbundparameter wird innerhalb des Verbundmoduls berechnet und ebenfalls aus dem Verbund exportiert, kann aber auch innerhalb desselben genutzt werden. Das dargestellte Verbundmodul besitzt sowohl Datenein- als auch (ebenso viele) Datenausgänge. Da jedes Portpaar gesondert gemäß des gekapselten Analysegraphen bearbeitet wird, ist hier nur ein derartiges Paar dargestellt. Der Eingangsport leitet jeden eingehenden Kanal an ein eigenes Eingangsmodul weiter; das Ausgangsmodul akzeptiert genau einen weiterzupropagierenden Datenraum.

Ohne explizite Verbindung zur „Außenwelt“ des Verbundes ist das Abbruchmodul. Dessen Parameter wird in der Regel lediglich verbundintern verwendet, kann prinzipiell natürlich auch über ein Parameterschnittstellenmodul exportiert werden — ein Import wäre demgegenüber nur selten sinnvoll.

Verbundmodule können sowohl dynamisch durch den Systemanwender zusammengestellt als auch über eine statische Bibliothek als Erweiterung von *VIOLA* um komplexere Analysefunktionen angeboten werden. Auch die Übernahme neu definierter Verbunde in eine persistente Bibliothek ist denkbar.

5.3.3 Übersicht über die Module von *VIOLA*

Abschließend zeigt Tab. 5.3 nochmals die zentralen Merkmale aller *VIOLA*–Module im Zusammenhang. Dargestellt sind jeweils

Modul	PO^{dat}	PAR^{ext}	PAR^{int}	**Funktion**	**det.**	**Anwendbarkeit**
$\mathcal{MOD}^{\text{mak}}$	0/1	$(\ldots)^3$	$\ldots$	$f^{\text{dat}} = \ldots$	$\surd$	–
$\mathcal{MOD}^{\text{mik}}$	0/1	$(\ldots)^3$	$\ldots$	$f^{\text{dat}} \approx f^{\text{kons}}$	$\surd$	–
$\mathcal{MOD}^{\text{in}}$	0/1	$2^{\mathcal{M}}, 2^{\mathcal{OR}}, 2^{\mathcal{K}}, 2^{\mathcal{E}\times\mathcal{OR}}$	$\mathbb{B}$	–	–	$MZ, K, EOR \neq \emptyset$
$\mathcal{MOD}^{\text{attr}}$	0/1	$\widetilde{2^{\mathcal{K}}}, \mathcal{R}, \mathcal{M}$	–	$f^{\text{dat}} = f^{\text{cube}}$	$\surd$	$D \supseteq K \neq \emptyset$ endlich; $mz \in f^{\text{mass}}(D)$
$\mathcal{MOD}^{\text{abl}}$	n/n	$\widetilde{2^{\mathcal{K}}}, \widetilde{2^{\mathcal{E}\times\mathcal{OR}}}$	{2,...,6} ($\mathbb{N}$)	$f^{\text{dat}} = f^{\text{abl}}$	$\surd$	$K, EOR \neq \emptyset$ über gleicher Dom., K endlich; $\exists$ betroffene Quellattribute; Zielkategorien (gemäß Variante) jeweils $\neq \emptyset$ und ableitbar aus Quellattributen; falls keine Restriktion: summarische Attribute aggregierbar
$\mathcal{MOD}^{\text{ablv}}$	n/n	$2^{\mathcal{E}\times\mathcal{OR}}$	–	$f^{\text{dat}} =^6 f^{\text{abl}}$	$\surd$	$EOR \neq \emptyset$; falls $\exists$ betroffene Quellattr.: summarische Attr. aggregierbar (für echt klassifizierende Attr.) und bei disjunkter Aggr. disjunkte Ableitung der Zielkategorien möglich
$\mathcal{MOD}^{\text{sel}}$	n/n	$2^{\mathcal{M}}, 2^{\mathcal{OR}}$	–	$f^{\text{dat}} = f^{\text{sel}}$	$\surd$	$MZ \neq \emptyset$; $\exists$ selektierte Attribute in dr
$\mathcal{MOD}^{\cup}$	n[1]/n	$2^{\mathcal{DO}}, 2^{\mathcal{E}\times\mathcal{OR}}, \mathcal{DR}$	–	$f^{\text{dat}} = f^{\cup}$	–	Klassifikation der kateg. Attr. möglich; jeweils gleiche Eigenschaften und Extensionen $\text{ext}(O,r)$ über gemeinsamen rollenbeh. Objektmengen
$\mathcal{MOD}^{\text{split}}$	n/n	$\mathcal{E} \times \mathcal{OR}, \mathcal{E}$	–	$f^{\text{dat}} =^6 f^{\text{split}}$	$\surd$	falls $\exists$ betroffenes kat. Quellattr.: $\nexists$ Attr. in dr über gleichen Eigenschaften wie Zielattr.
$\mathcal{MOD}^{\text{merge}}$	n/n	$\mathcal{E} \times \mathcal{OR}, \mathcal{E}$	–	$f^{\text{dat}} =^6 f^{\text{merge}}$	$\surd$	falls $\exists$ betroffenes kategorielles Quellattr.: Eigenschaften der Quellattr. sind bijektiv zuzuordnen, $\nexists$ Attr. in dr über gleiche Eigenschaften wie Zielattr.
$\mathcal{MOD}^{\text{role}}$	n/n	$\mathcal{R}, \mathcal{R}$	–	$f^{\text{dat}} = f^{\text{role}}$	$\surd$	Zielrolle in dr nicht vorhanden
$\mathcal{MOD}^{\text{cell}}$	n/n	$\mathcal{M}\ (+\ldots)^3$	–	$f^{\text{dat}} = f^{\text{cell}}$	–	mz aus Quelldaten berechenbar
$\mathcal{MOD}^{\text{aggr}}$	n/n	$\widetilde{2^{\mathcal{K}}}, \widetilde{2^{\mathcal{E}\times\mathcal{OR}}}, \mathcal{M}$ $(+\ldots)^3$	{2,3,6} ($\mathbb{N}$)	$f^{\text{dat}} = f^{\text{eaggr}}$	–	wie bei $\mathcal{MOD}^{\text{abl}}$ — statt Aggregierbarkeit: mz über alle summarischen Attr. berechenbar (wobei kat. Attr. geeignet zur Aggregierung)
$\mathcal{MOD}^{\text{join}}$	n/n	$\mathcal{M}\ (+\ldots)^3$	–	$f^{\text{dat}} = f^{\text{join}}$	–	mz aus Quelldaten berechenbar (unter Aggregierung über geeignete kategorielle Attr.)
$\mathcal{MOD}^{\text{defpar}}$	1/1	$\ldots + 1^4$	$\ldots$	$f^{\text{par}} = \ldots$	$\ldots$	$\ldots$
$\mathcal{MOD}^{\text{iopar}}$	0/0	$1^{(4)}$	–	–	$\surd$	–
$\mathcal{MOD}^{\text{out}}$	1/0	–	–	–	$\surd$	–
$\mathcal{MOD}^{\text{save}}$	n/n	–	$\ldots$	–	$\surd$	–
$\mathcal{MOD}^{\text{vis}}$	1/1	$\ldots^4$	$\ldots$	–	$\surd$	gemäß graphischer Attribute und Symboldefinitionen des Visualisierungsverfahrens
$\mathcal{MOD}^{\text{route}}$	n[1]/n	$\mathcal{M}, 2^{\mathcal{OR}}$	–	$f^{\text{dat}} \approx f^{\text{sel}}$	–	$\exists$ Datenraum mit gesuchten Merkmalen
$\mathcal{MOD}^{\text{if}}$	1/2	$\mathbb{B}$	–	$f^{\text{dat}} =$„if“	$\surd$	–
$\mathcal{MOD}^{\text{loop}}$	0/0	$2^{4,5}$	–	–	$\surd$	Werteliste nicht leer
$\mathcal{MOD}^{\text{stop}}$	0/0	$\mathbb{B}$	–	–	$\surd$	–
$\mathcal{MOD}^{\text{sub}}$	$\ldots$[2]	$\ldots$	$\ldots$	$\ldots$	$\ldots$	je nach externer Param. der Eingangsmodule; Parameter als Grundlage für Schleifenmodule dürfen keine leeren Listen enthalten

[1] mengenwertiger Datenport
[2] nahezu beliebige Kombinationen: 0/0, 0/1, n/0, n/0, n/n, n/n, wobei Eingangsports evtl. auch mengenwertig
[3] fast beliebige Parameter möglich, aber nur ergänzend und in der Regel ungenutzt
[4] Parameterport ohne Import
[5] beliebiger einwertiger Parameterwertebereich (also keine Wertemengen)
[6] Eingangsdaten bleiben unverändert, wenn „passende“ Quellattribute nicht existieren

Tabelle 5.3: *VIOLA*–Module im Überblick

- Anzahl von Datenein- und ausgangsports (*n* steht hierbei für beliebig viele) sowie (in der Fußnote) ob ein Eingangsport Mengen von Datenräumen aufnehmen kann,
- Typen (Wertebereiche) von externen und internen Parametern sowie (in der Fußnote) ob ein Parameterport gemäß des Flags *in*? den Werteimport von anderen Modulen ausschließt,
- die realisierten Funktionen auf Daten (f^{dat}) oder Parametern (f^{par}), wobei auf Nebeneffekte nicht eingegangen wird,
- ob die Anwendung der Funktionen sowie die Generierung von eventuellen Nebeneffekten (Visualisierungen, Kontrollfunktionen etc.) in jedem Fall deterministisch definiert ist oder evtl. zu alternativen Ergebnissen führen kann und schließlich
- welche Kriterien die Anwendbarkeit eines Moduls auf anliegende Daten beschreiben.

Ist eine Angabe je nach Instanz einer Modulklasse unterschiedlich definiert, ist dies durch „. . .“ gekennzeichnet. In der Spalte der externen Parametern steht „1“ oder „2“ dafür, daß es genau einen bzw. zwei Parameter gibt, deren genauer Typ aber nicht festgelegt ist.

Unter der Anwendbarkeit eines Moduls auf gegebene Eingangsdaten bzw. Parameterwerte soll (bis auf weiteres) allgemein verstanden werden, daß die Berechnungsfunktionen f^{dat} bzw. f^{par} auf diesen definiert sind, also stets ein gültiges Ergebnis liefern. Sind in Tab. 5.3 keine Anwendbarkeitsbedingungen angegeben, so ist das jeweilige Modul in jedem Fall anwendbar, sofern eingehende Datenkanäle nicht leer sind. Für jedes Modul sei ferner eine Hierarchisierung seiner externen Parameter gegeben, die spezifiziert, welcher als Grundlage der Anwendbarkeitsüberprüfung eines anderen dienen soll. Wird z. B. für Ableitungsmodule gefordert, daß Kategorien und Eigenschaften jeweils auf die gleiche Domäne bezogen sein müssen, so ist bei einem Verstoß gegen diese Regel zu entscheiden, welche der beiden unterschiedlichen Domänen als die „falsche“ anzusehen und somit welcher der Parameter vom Prüfverfahren als ungültig abzulehnen ist. Dateneingänge hängen in diesem Sinne stets von externen Parametern (und niemals von anderen Dateneingängen) ab; interne Parameter können als Grundlage aller Prüfungen dienen.

Nicht explizit aufgeführt ist als separate Anwendbarkeitsprüfung für alle Parameterports die Betrachtung der Werteübergabe zwischen nur *einwertig* kompatiblen Parametern. Die Einstufung der Anwendbarkeit einer Berechnungsfunktion bleibt hiervon jedoch unberührt, da — falls eine Konvertierung zwischen den Parametern nicht möglich ist — jeweils die letzte (gültige) Belegung des Parameterports vor Auftreten der Inkompatibilität bestehen bleibt (diesbezügliche Details werden in Abschnitt 5.4.2 diskutiert).

Die Anwendbarkeit eines Verbundmoduls auf multidimensionale Datenräume soll im obigen Sinne jeweils durch die Parameter seiner Eingangsmodule definiert sein, wobei ggf. optionale Eingänge zu berücksichtigen sind. Akzeptieren die Eingangsmodule also einen anliegenden Datenraum, so sollte dieser auch durch den gesamten Analysegraphen des Verbundes verarbeitet werden können.[45] Dagegen gestaltet sich die Betrachtung der Anwendbarkeit eines Verbundmoduls bzgl. seiner externen Parameter problematisch: Hier existiert kein einem Eingangsmodul entsprechender „Wächter“, der eine klare Anwendbarkeitsbedingung vorgibt. Somit wäre der interne Analysegraph näher zu betrachten, wobei jedoch häufig Anwendbarkeitsprobleme im Ablauf komplexerer Subnetze nicht klar bestimmten Parametern zuzuordnen sind. Dies entspricht auch dem Gesamtbild, das die Module von *VIOLA* bieten: Rein den Parametern zuzuordnende Anwendbarkeitsbedingungen sind selten. Dementsprechend wird für Verbundmodule selbst — abgesehen von der Forderung nach nicht–leeren Listen für Schleifen — auf die Berücksichtigung von Anwendbarkeitsprüfungen für externe Parameter verzichtet.

Die Information, ob die Berechnungen eines Verbundmoduls eindeutig sind, läßt sich dagegen wiederum recht einfach aus dem durch das Modul gekapselten Analysegraphen herleiten, d. h. mittels Betrachtung der über den jeweiligen Fan–in transitiv definierten und innerhalb des Verbundes liegenden Vorgänger–Module von Ausgangsmodulen, Datensenken sowie ggf. des Abbruchmoduls (falls dieses von den Schleifenmodulen abhängig ist).

[45] Eine solche Bedingung mag als Faustregel dienen — sie kann in der Regel weder exakt geprüft werden noch ist sie zwingend für die korrekte Verarbeitung eines *VIOLA*–Programms notwendig, wie wir noch in Abschnitt 5.4 sehen werden.

5.4 Datenflußbasierte Verarbeitung von Analyseprogrammen

VIOLA–Programme wurden als Netzwerke von Modulen, die durch Kanäle verbunden sind, definiert. Jedes Modul beschreibt ein Verfahren zur Verarbeitung seiner Eingaben bzw. zur Erzeugung neuer Ausgaben — andere als durch Module und deren Ports definierte Berechnungsschritte erfolgen nicht. Multidimensionale Datenräume und Parameterwerte haben keine direkte Repräsentation in dieser Darstellung, sondern werden als Zustände von Kanälen und Modulports modelliert.[46] Die Semantik eines *VIOLA*–Programms ergibt sich nun als Kombination der modulbasierten Verarbeitungsschritte in einer datenflußbasierten Abarbeitung. Aufgrund einer relativ engen Anlehnung an das grundlegende Datenflußparadigma, in das lediglich einige einfache Kontrollstrukturen eingebettet wurden, kann dieser Verarbeitungsprozeß mit recht einfachen Mitteln beschrieben werden. Nicht zuletzt trägt hierzu die Zyklenfreiheit der betrachteten Programme bei (vgl. Def. 5.25). Einige spezielle Überlegungen erfordern hingegen die unterschiedliche Behandlung von Datenräumen und Parameterwerten sowie die Unterstützung unterschiedlicher Verarbeitungsmodi, die eine flexible interaktive Programmerstellung und -bedienung gestatten.

Prinzipiell soll sich ein *VIOLA*–Programm zu jedem Zeitpunkt in einem gültigen (konsistenten) Zustand hinsichtlich der Belegung von Zwischen- und Endergebnissen der Analyse befinden. Somit wird dem eingangs dieser Arbeit motivierten Gedanken intelligenter Datenanalysen insofern Rechnung getragen, als jederzeit eine Überprüfung der Durchführbarkeit von Auswertungsfolgen sowie ein freier Zugriff auf die aktuelle Analysesituation als Grundlage der Entscheidung über nachfolgende Analyseschritte möglich sind.

Aus Gründen der Effizienz ist die Forderung nach vollständiger Berechnung jedoch in vielen Fällen dahingehend abzuschwächen, daß nicht alle Datenraum–Extensionen, sondern nur die jeweiligen Schemata ermittelt werden. Infolgedessen können auf Extensionen basierende Parameterdefinitionen in entsprechenden Situationen keine aktuellen Werte liefern, sondern müssen ihre jeweiligen Zielparameter (mit „veralteten“ Belegungen) als „vorläufig“ kennzeichnen. Als Konsequenz wird in Kauf genommen, daß nach Aktualisierung der Berechnungen hiervon abhängige Datenraumschemata zu ändern und evtl. sogar nachfolgende Berechnungsschritte nicht mehr anwendbar sind. Im Normalfall dürften sich jedoch grundlegende Veränderungen am gesamten Analyseablauf bei einem Wechsel von der schema- zur extensionsbasierten Verarbeitung im Rahmen halten, zumal extensionsbasierte Parameterdefinitionen auch nicht allzu häufig verwendet werden.[47]

Die Programmausführung erfolgt in einer Kombination anforderungs- und datengetriebener Verarbeitung: Datensenken bestimmen erforderliche Berechnungen, und Benutzerinteraktionen stoßen die Propagierung neuer Zwischenergebnisse im Zuge des Datenflusses an. Zur Beschreibung der Programmsemantik käme zunächst einmal ein funktionaler (denotationaler) Ansatz in Frage, der die funktionsbasierten Operator- und Moduldefinitionen von *MADEIRA* bzw. *VIOLA* fortführt. Hierbei würden sich jedoch schnell Probleme bei der Darstellung von Seiteneffekten (in Visualisierungsverfahren) und der Schleifenverarbeitung ergeben. Die Definitionen wären — bei Einbeziehung der Details von Summierungsfunktionen summarischer Attribute und von datenbasierten Parameterdefinitionen — unnötig komplex, und auch der dynamische Ablauf einer Analyse in all ihren Einzelschritten und die direkte Interaktion des Analysten mit diesen würde unangemessen stark in den Hintergrund treten. Statt dessen wollen wir aus diesen Gründen die Sprachsemantik auf operationaler Basis definieren, indem die Programmausführung durch modulare Verarbeitungsschritte auf einem Zustandsraum spezifiziert wird. Somit wird auch eine direkte Grundlage für eine effiziente objektorientierte Sprachimplementierung geliefert.

Einige Grundlagen der nachfolgenden Ausführungen entstammen dem Forschungsgebiet der Datenflußarchitekturen, genauer der Diskussion des Scheduling von Datenflußprogrammen [Sha92b, HYT+92, HYM+92, Sri92]. Der Abstraktionsgrad der Betrachtung ist in diesen Arbeiten jedoch in der Regel viel geringer. Weiterhin werden dort Datenflüsse über Token definiert, die auf Kanälen (auch in direkter Folge) dynamisch propa-

[46] Dies ist durchaus nicht selbstverständlich, vgl. etwa das System *ViSta*, das die Daten selbst in den Vordergrund stellt und sogar für einfache Berechnungen gar keine Repräsentation anbietet [You96].

[47] Meistens werden durch Parametermodule lediglich Komponenten des Schemas eines Datenraums zur „Synchronisation“ anderer Datenräume extrahiert. Ausnahmen bilden vor allem die bereits diskutierten Anwendungsfelder der datenbasierten Restriktion oder Klassifikation (siehe Abschnitt 5.2.4).

giert werden. In *VIOLA* dagegen wird einem Kanal ein Zustand zur Repräsentation von Daten zugeordnet, der im Kontext einer vorgenommenen Parametrisierung eines Modulgraphen eher als statisch anzusehen ist.[48]

Zur Definition der Programmausführung werden im folgenden jeweils Module innerhalb eines Programms bzw. Analysegraphen $ag = (MOD, REL^{\text{dat}}, REL^{\text{par}}) \in \mathcal{AG}$ mit einer wie in Def. 5.24 über ganz *MOD* definierten Gesamtmenge von Eingangsports PO^{in}, Ausgangsports PO^{out} und Parameterports PO^{par} betrachtet. Alle Module sind jeweils exklusiv nur diesem Analysegraphen zugeordnet und nicht gleichzeitig Teil mehrerer, evtl. sogar parallel auszuführender Programme. *PO* sei als Vereinigung aller drei Portmengen und *REL* als Vereinigung der Daten- und Parameterkanäle definiert.

Zunächst werden in Abschnitt 5.4.1 einige grundlegende Besonderheiten bei der Betrachtung von Verbundmodulen als eine Art von Unterprogrammen diskutiert, auf die in den weiteren Abschnitten noch mehrfach zurückgegriffen wird. Abschnitt 5.4.2 stellt den durch Ports und Kanäle gegebenen Zustandsraum eines *VIOLA*–Programms vor, auf dem dann in Abschnitt 5.4.3 der prinzipielle Ablauf der Programmausführung spezifiziert werden kann. Die hierzu nötigen Verarbeitungsprimitive werden in Abschnitt 5.4.4 in Form von Algorithmen in Pseudo–Code konkretisiert. Abschließend präsentiert Abschnitt 5.4.5 die wesentlichen Benutzerinteraktionen bei der Bedienung der *VIOLA*–Programmier- und –Analyseumgebung, und Abschnitt 5.4.6 geht auf einige spezielle Aspekte der Programmvisualisierung ein.

5.4.1 Behandlung von Verbundmodulen

Bevor der allgemeine Prozeß der Verarbeitung von *VIOLA*–Modulen diskutiert werden kann, sind zunächst einige Überlegungen dazu anzustellen, wie Verbundmodule in zu den übrigen Modulen kompatibler Weise berücksichtigt werden können. Wir gehen hierzu von dem allgemeinen Fall aus, daß ein Verbundmodul $mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}}$ n Dateneingänge und/oder n Datenausgänge besitzt ($n \in \mathbb{N}_0$). Falls sowohl Ein- als auch Ausgangsports vorliegen, sind es also jeweils gleich viele. Ein- und Ausgänge seien jeweils in der Form $po_1, \ldots, po_n$ durchnumeriert.

Die Verarbeitung von mod^{sub} soll in n identischen „Instanzen“ $ag_i, i = 1, \ldots, n$, (quasi Kopien) des Analysegraphen $ag = mod^{\text{sub}}.ag$ erfolgen, die jeweils, wie bereits in Abschnitt 5.3.2 dargestellt, Daten des i–ten Eingangs verarbeiten und die Ergebnisse zum i–ten Ausgang weiterleiten. Module $mod \in ag.MOD$, die explizit einer spezifischen Instanz ag_i des Analysegraphen zugeordnet werden sollen, werden als mod_i geschrieben.

Während Datenein- und -ausgangsports eines Verbundes sich somit jeweils stets auf nur eine einzige Instanz von *ag* beziehen, stehen alle internen und externen Parameter bzw. die Parameterports des Verbundes gleichzeitig allen Instanzen zur Verfügung. Dies ist völlig konform mit der Parameternutzung in anderen Modulen und hat auch hier (analog zu den in Abschnitt 5.2.1 angestellten Überlegungen) als logische Konsequenz, daß Verbundmodule, die intern Werte ihrer externen Parameter neu berechnen[49], nicht mehr als einen Dateneingang bedienen können. Vor dem Hintergrund, ein Verbundmodul als eine für alle seine Dateneingänge einheitliche Verarbeitungseinheit anzusehen, werden manuelle Änderungen interner oder externer Parameter innerhalb des Analysegraphen *ag* stets in allen seinen Instanzen synchron durchgeführt. Automatisch in Modulen von *ag* berechnete Parameter können natürlich in verschiedenen Graphinstanzen auch unterschiedliche Werte annehmen.

Weiterhin gebe eine Funktion $is\_loop : \mathcal{MOD}^{\text{sub}} \to \mathbb{B}$ an, ob ein Verbundmodul eine Schleife realisiert, also sein Analysegraph mindestens ein Schleifenmodul enthält; und eine dynamisch belegte Funktion $finished : \mathcal{MOD}^{\text{loop}} \to \mathbb{B}$ signalisiere, ob die aktuelle Belegung der Parameterwerte eines Schleifenmoduls bereits am Ende der jeweiligen Werteliste angelangt (genauer: ob für den *current*–Parameter noch ein weiterer Wert verfügbar) ist.

Schließlich ist noch ein Aspekt zu konkretisieren, der in Abschnitt 5.3.2 lediglich kurz angerissen wurde, nämlich die Zuordnung der in einem Daten–Eingangsport eines Verbundmoduls anliegenden Datenkanäle zu

[48] Aufgrund interaktiver Manipulationen wird natürlich auch dieser Zustand dynamisch. Weiterhin definieren Schleifen Folgen von Kanalzuständen.

[49] Gemäß Def. 5.27 sind dies Verbundmodule mit einem Parameterport po^{par}, für den $po^{\text{par}}.in? = false$ gilt, der also keine Werte von anderen Modulen importieren kann.

den Eingangsmodulen der entsprechenden Instanz des Verbund–Analysegraphen. Generell gibt ein Eingangsmodul über seine Parameter EOR, K, MZ und OR an, daß es (gemäß Def. 5.6) gerade Datenräume dr mit $(dr.KA \cap \mathcal{KA}_{EOR,K} \neq \emptyset) \wedge (dr.SA \cap \mathcal{SA}_{MZ,OR} \neq \emptyset)$ zur Weiterleitung in den Verbund akzeptiert. Hierbei sind Mehrdeutigkeiten, die durch „überlappende" Parametersätze von Eingangsmodulen auftreten, aufzulösen. Es wird folgender einfache Algorithmus verfolgt, dessen Ergebnis über eine Funktion $quellkanal : \mathcal{MOD}^{\text{in}} \rightarrow \mathcal{PO}^{\text{dat}}$ festgehalten wird[50]:

1. Weise alle eindeutig einem nicht–optionalen Eingangsmodul zuzuordnenden Datenkanäle jeweils diesem zu.
2. Weise umgekehrt alle eindeutig einem Datenkanal zuzuordnenden nicht–optionalen Eingangsmodule jeweils diesem zu.
3. Verteile die übrigen Datenkanäle der Reihe nach auf das jeweils erste mögliche nicht–optionale Eingangsmodul.
4. Verfahre analog mit den verbleibenden Datenkanälen und den optionalen Eingangsmodulen.

Die Anwendung der einzelnen Verfahrensschritte auf die als Kandidaten zur Verfügung stehenden Datenkanäle bzw. Datenräume erfolgt jeweils einfach in der chronologischen Reihenfolge ihrer Verbindung mit dem jeweiligen Eingangsport. Unter Umständen können bei diesem Vorgehen Fälle eintreten, in denen anliegende Datenraummengen nicht vollständig aufgeteilt werden können, obwohl dies prinzipiell (also bei einer vollständigen Überprüfung aller möglichen Permutationen von Zuordnungen zwischen Datenraum- und Eingangsmodulmenge) möglich wäre. Da eine derartige Situation mit mehrfachen mehrdeutigen Eingangsmodulen aber in der Regel auf eine unvollständige bzw. ungenaue Parametrisierung hinweist, soll hier kein komplexerer Zuordnungsalgorithmus zum Einsatz kommen. Es bleibt dem Systemanwender überlassen, die Verbundeingänge genauer zu definieren und somit Mehrdeutigkeiten zu vermeiden.

Auf der Basis dieser Zuweisungen kann für jeden Datenkanal und entsprechend für jedes Eingangsmodul eine spezifische Anwendbarkeit definiert werden. Auch wenn ein Verbund auf einzelne Eingänge nicht anwendbar sein sollte bzw. Eingangsmodule nicht belegt werden konnten, wird im Rahmen der Programmausführung der Analysegraph des Verbundes zumindest soweit verarbeitet, wie für die Anwendbarkeit von Modulen und Teilgraphen erforderliche Daten vorliegen. Sind nur optionale Eingänge betroffen, ist dies natürlich so gewünscht. Aber auch in anderen Fällen kann die Berechnung von Teilergebnissen eines Verbundmoduls für den Anwender hilfreich sein.

5.4.2 Zustände von Ports und Kanälen

Während der Verarbeitung eines *VIOLA*–Programms nehmen Ports und Kanäle dynamisch unterschiedliche Zustände an, die primär durch Datenraum- bzw. Wertebelegungen, aber auch in Form interner, die Verarbeitung steuernder Variablen gegeben sind. Um die Herleitung dieser Zustände modellieren zu können, sind zunächst die Beziehungen zwischen den Ports der Module eines *VIOLA*–Programms genauer zu beschreiben. Definition 5.24 beschränkte sich in diesem Kontext auf die Betrachtung der Verbindungsrelationen zwischen Ports. Der so definierte *externe* Fan–in bzw. Fan–out wird nun durch einen *internen* Fan–in bzw. Fan–out ergänzt, der die Abhängigkeiten zwischen Ports widerspiegelt, die über die jeweiligen Berechnungsfunktionen gegebenen sind, und zusätzlich die Parameter- und Werteübergabe an den Schnittstellen eines Verbundmoduls mit einbezieht.

Definition 5.28 ((Interner) Fan–in und Fan–out) *Der* INTERNE FAN–IN $f^{\text{in(i)}} : \mathcal{PO} \rightarrow 2^{\mathcal{PO}}$ *sei definiert, indem einem Port $po \in \mathcal{PO}$ eines Moduls $mod = po.mod$ die seiner Berechnung zugrundeliegenden Ports innerhalb desselben Moduls zugeordnet werden, also*

$$f^{\text{in(i)}}(po) \stackrel{\text{def}}{=} \{po' \in mod.PO^{\text{in}} \cup mod.PO^{\text{out}} \cup mod.PO^{\text{par}} \mid po' \text{ geht in } mod.f^{\text{dat}}_{po} \text{ bzw. } mod.f^{\text{par}}_{po} \text{ ein}\}.$$

[50] Zu beachten ist, daß diese Funktion natürlich auf den Modulen verschiedener Instanzen des Verbund–Analysegraphen unterschiedlich definiert ist. Der Funktionswert gibt jeweils den Quellport des eingehenden Kanals an.

Zusätzlich seien für Ports eines Verbundmoduls $mod = mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}}$ *mit n Instanzen seines Analysegraphen,*

- *falls* $po = po_i^{\text{in}} \in mod.PO^{\text{in}}$*:*
  $\forall mod_i^{\text{in}} \in mod^{\text{sub}}.ag_i.MOD \cap \mathcal{MOD}^{\text{in}}: f^{\text{in(i)}}(mod_i^{\text{in}}.po^{\text{out}}) \stackrel{\text{def}}{=} po_i^{\text{in}}$ *(Verknüpfung des i–ten Eingangsports* po_i^{in} *eines Verbundmoduls mit den Ausgangsports der Eingangsmodule der i–ten Instanz seines Analysegraphen),*
- *falls* $po = po_i^{\text{out}} \in mod.PO^{\text{out}}$*:*
  $\forall mod_i^{\text{out}} \in mod^{\text{sub}}.ag_i.MOD \cap \mathcal{MOD}^{\text{out}}: f^{\text{in(i)}}(po_i^{\text{out}}) \stackrel{\text{def}}{=} mod_i^{\text{out}}.po^{\text{in}}$ *(Verknüpfung des i–ten Ausgangsports* po_i^{out} *eines Verbundmoduls mit dem Eingangsport des (eindeutigen) Ausgangsmoduls der i–ten Instanz seines Analysegraphen),*
- *falls* $po = po^{\text{par}} \in mod.PO^{\text{par}}$*:*
  $\forall mod^{\text{iopar}} \in mod^{\text{sub}}.ag.MOD \cap \mathcal{MOD}^{\text{iopar}}$ *mit* $mod^{\text{iopar}}.po^{\text{sub}} = po^{\text{par}}$ *und* $\forall i = 1, \ldots, n$*:*
  - *falls* $po^{\text{par}}.in? = true$*:* $f^{\text{in(i)}}(mod^{\text{iopar}}.po^{\text{par}}) \stackrel{\text{def}}{=} po^{\text{par}}$*,*
  - *falls* $po^{\text{par}}.in? = false$*:* $f^{\text{in(i)}}(po^{\text{par}}) \stackrel{\text{def}}{=} mod^{\text{iopar}}.po^{\text{par}}$

  (Verknüpfung eines Parameterports po^{par} *eines Verbundmoduls mit dem Parameterport des entsprechenden (falls existent, eindeutigen) Parameterschnittstellenmoduls aller Instanzen des Verbundmodul–Analysegraphen) sowie*
- *wiederum falls* $po = po^{\text{par}} \in mod.PO^{\text{par}}$*:*
  $\forall mod^{\text{loop}} \in mod^{\text{sub}}.ag.MOD \cap \mathcal{MOD}^{\text{loop}}$ *mit* $mod^{\text{loop}}.po^{\text{sub}} = po^{\text{par}}$ *und* $\forall i = 1, \ldots, n$*:*
  $\forall po^{\text{loop}} \in mod^{\text{loop}}.PO^{\text{par}}: f^{\text{in(i)}}(po^{\text{loop}}) \stackrel{\text{def}}{=} po^{\text{par}}$ *(Verknüpfung eines Parameterports* po^{par} *eines Verbundmoduls mit Parameterports abhängiger Schleifenmodule in allen Instanzen des Verbundmodul–Analysegraphen).*

Der INTERNE FAN–OUT $f^{\text{out(i)}}: \mathcal{PO} \to 2^{\mathcal{PO}}$ *sei genau symmetrisch zum Fan–in definiert, also* $po \in f^{\text{in(i)}}(po') \Leftrightarrow po' \in f^{\text{out(i)}}(po)$*.*

FAN–IN f^{in} *und* FAN–OUT f^{out} *seien definiert als die Vereinigung von internem und externem Fan–in bzw. Fan–out innerhalb des Analysegraphen ag.*

Als grundlegende Zustandsdefinitionen wurden in Abschnitt 5.2 (Def. 5.1 und 5.4) bereits Abbildungen $val()$ und $data()$ eingeführt, die einem Parameter seinen aktuellen Wert bzw. einem Datenport die Menge aller enthaltenen Datenräume zuordnen. Dieser Ansatz wird im folgenden nun noch verfeinert.

Zunächst einmal ist auch den Daten- und Parameterkanälen ihre aktuelle Belegung zuzuordnen. Für Datenkanäle ergibt sich diese einfach als Belegung des jeweiligen Quellports. Dateneingangsports vereinigen wiederum die Datenräume aller eingehenden Kanäle zu ihrem eigenen Zustand und definieren somit eine Abhängigkeit zwischen den Belegungen von Ein- und Ausgangsports.

Definition 5.29 (Belegung von Datenkanälen und -eingangsports) *Die partielle Abbildung* $data: PO^{\text{out}} \times PO^{\text{in}} \to \mathcal{DR}$ *liefere zu jedem Datenkanal den aktuell propagierten Datenraum: Für* $po^{\text{out}} \in PO^{\text{out}}$ *und* $po^{\text{in}} \in PO^{\text{in}}$ *sei* $data(po^{\text{out}}, po^{\text{in}}) \stackrel{\text{def}}{=} data(po^{\text{out}})$*, falls* $(po^{\text{out}}, po^{\text{in}}) \in REL^{\text{dat}}$ *und* $data(po^{\text{out}}) \neq \emptyset$*.*

In ähnlicher Weise gelte für $po^{\text{in}} \in PO^{\text{in}}$ *stets* $data(po^{\text{in}}) = \bigcup_{po^{\text{out}} \in f^{\text{in(e)}}(po^{\text{in}})} data(po^{\text{out}})$*.*

Über die Belegung von Datenausgängen kann in diesem Kontext nichts weiter ausgesagt werden: Sie erhalten im Verlauf der Ausführung eines *VIOLA*–Programms jeweils einen Datenraum als Berechnungsergebnis des entsprechenden Moduls zugewiesen.

Ähnlich wie mit Datenkanälen verhält es sich mit den Parameterkanälen. Hier können jedoch zum einen keine undefinierten Kanalbelegungen auftreten, und zum anderen sind multiple Eingänge von Parameterports geeignet zum Erhalt eines einzelnen Parameterwertes zu „vereinen“. Als Grundlage der folgenden Definition ist natürlich die Funktion $val()$ von Parametern auf Parameterports zu übertragen:

Definition 5.30 (Belegung von Parameterkanälen und Parameterports) *Die totale Abbildung* $val : PO^{\text{par}} \cup REL^{\text{par}} \rightarrow \Omega$ *ordne jedem Parameterport* $po^{\text{par}} \in PO^{\text{par}}$ *sowie auch jedem Parameterkanal* $(po^{\text{par}}, po^{\text{par}\prime}) \in REL^{\text{par}}$ *die aktuelle Belegung* $val(po^{\text{par}}.par) \in po^{\text{par}}.par.dom$ *des jeweiligen (Quellport–)Parameters* $po^{\text{par}}.par$ *zu.*

Für jeden Parameterport $po^{\text{par}} \in PO^{\text{par}}$ *mit* $po^{\text{par}}.in? = true$ *definiere eine (partielle) Abbildung* $f^{\text{parjoin}}_{po^{\text{par}}} : 2^{PO^{\text{par}}} \rightarrow po^{\text{par}}.par.dom$ *eine Vorschrift, nach der aus dem jeweiligen externen Fan–in* $f^{\text{in}(e)}(po^{\text{par}})$ *und seiner Konstruktions- und Verarbeitungshistorie eine aktuelle Parameterausprägung* $val(po^{\text{par}}.par)$ *zu definieren ist. Hierbei sind folgende Varianten vorgesehen:*

1. *Der Wert des Kanals, der bei der Programmerstellung zuerst erzeugt wurde, wird übernommen; spätere Verbindungen werden nicht berücksichtigt.*[51]
2. *Der zuletzt erstellte bzw. zuletzt aktualisierte Kanal liefert den aktuellen Parameterwert.*
3. *Für boolesche Parameter werden einfache boolesche Operationen angeboten* ($\wedge, \vee$)*.*
4. *Parameterports auf mengenwertigen Parametertypen können auch einfache Mengenoperationen* ($\cap, \cup$) *unterstützen.*

Falls po^{par} *nur einwertig kompatibel zu einem oder mehreren Quellports aus dem Fan–in ist und die zur Konvertierung geforderten Eindeutigkeiten (durch einen oder die Gesamtheit mehrerer Quellparameter) verletzt sind, werden die betreffenden Parameterkanäle jeweils als nicht existent erachtet.*

Bewußt werden in den Parameterports nur einfache Verknüpfungsfunktionen angeboten, um die intuitive Verständlichkeit von Datenflußprogrammen für den Betrachter und Entwickler nicht durch eine in den Ports verborgene Parameterverarbeitung zu verringern.

Einige spezielle Bezeichnungen für Parameterports werden im folgenden noch hilfreich sein: Ein Parameterport soll *automatisch* heißen, wenn er das Ergebnis einer Berechnung im selben Modul aufnimmt (also f^{par} für ihn definiert ist), wenn er Port eines Schleifenmoduls ist oder wenn er über eingehende Parameterkanäle ($in? = true$) Parameterwerte aus Ports erhält, die ihrerseits automatisch sind. Andernfalls heiße er *manuell*. Für die Klasse automatischer Ports wird unterschieden, ob die Berechnung rein *schemabasiert* erfolgt, d. h. es werden keine Extensionen von Datenräumen benötigt, oder ob zumindest für einen eingehenden Datenraum auch auf die Zellwerte zugegriffen wird. In letzterem Fall heiße der Port *datenbasiert*, genauer: bezüglich eines Datenports po^{dat}, aus dem die Datenrauminstanz benötigt wird, po^{dat}*–datenbasiert*.

Mit diesen Begriffsfestlegungen wäre zu obigen Bearbeitungsvarianten von Parameterports noch zu bemerken, daß, falls gleichzeitig Kanten von mehreren automatischen Ports zu einem gemeinsamen Zielport führen, der im Fall (2) erhaltene Parameterwert nicht mehr deterministisch festgelegt ist, da die Reihenfolge der Bearbeitung der Quellports durch *VIOLA* nicht explizit definiert wird. Hier bleibt es dem Anwender überlassen, eine geeignetere Verarbeitungsalternative zu wählen, falls ihn der Nichtdeterminismus stören sollte.

Wie bereits angedeutet, werden neben den Daten- und Wertebelegungen noch weitere Statusinformationen für Ports und Kanäle dynamisch definiert. Eine wichtige Rolle spielt hierbei die grundlegende Entscheidung, auf Wunsch des Benutzers Berechnungen auch auf die Schemata (die Attribute) multidimensionaler Datenräume beschränken zu können (also eine rein *schema-* oder *metadatenbasierte* Verarbeitung durchzuführen). Vor allem sollen so rein schemabasierte Betrachtungen der Anwendbarkeit von Verfahren auf gegebene Daten ermöglicht werden. Der Anwender erhält eine wertvolle Hilfestellung bei der Programmierung, ohne daß Extensionen von Datenräumen (in u. U. aufwendigen Verfahren) berechnet werden müssen. Insgesamt werden folgende Aspekte der Statusdefinition berücksichtigt:

- Daten- und Parameterwerte können temporär ungültig sein, wenn etwa bestimmte Modulparameter verändert, aber die entsprechenden Neuberechnungen noch nicht durchgeführt wurden. Weiterhin kann die Belegung eines Datenports „undefiniert“ sein: Wenn ein Modul nicht auf seine Eingänge anwendbar

[51] Dies kann natürlich im wesentlichen auch realisiert werden, indem ein mit einem eingehenden Kanal belegter Port alle weiteren Verbindungen ablehnt.

ist, bleiben die Ausgänge leer. Auch Parameterkanäle werden als „leer“ angesehen, wenn sie von ihrem Zielport nicht berücksichtigt werden, weil dessen Parameterwert manuell definiert oder nur aus einzelnen eingehenden Kanälen ermittelt wurde. Schließlich wollen wir auch unterscheiden, ob Datenräume nur als Metadaten, also mit ihrem Schema, oder einschließlich ihrer Extension (ihrer Zellinhalte) vorliegen. Insgesamt ergibt sich ein *Datenstatus* mit den Werten $\{$ *invalide*, *leer*, *schema*, *ok* $\}$.

- Infolge einer Benutzeranforderung von Analyseergebnissen (etwa zur Erzeugung einer bestimmten Graphik oder Tabelle) müssen Ports und Kanäle markiert werden können, um zu kennzeichnen, ob jeweils „aktiv“ Datenberechnungen zur Bestimmung von Parameterwerten oder Datenraumextensionen durchzuführen sind, ob eine „passive“ Propagierung von Metadaten (Datenraum–Schemata) ausreichend oder ob zur Erfüllung der gegebenen Anforderung keine Aktion nötig ist. Ein entsprechender *Propagierungsstatus* enthält die Werte $\{$ *offen*, *passiv*, *aktiv* $\}$. Dies entspricht etwa der Idee des „Flagging“ relevanter Graphkomponenten, wie sie in [DRR+96] vorgestellt wird.

- Die Berechnung von Datenräumen kann gemäß der Definition der *MADEIRA*–Operatoren stets sowohl daten- als auch metadatenbasiert erfolgen, d. h. liegen lediglich Datenraum–Schemata als Eingangsdaten vor, kann der Zieldatenraum in jedem Fall bzgl. seines Schemas vollständig definiert werden. Demgegenüber sind für bestimmte (datenbasierte) Parameterberechnungen Datenraum–Extensionen unabdingbar. Somit können Fälle auftreten, in denen eine Parameterdefinition nicht möglich ist und der alte Parameterwert „vorläufig“ beibehalten wird. Nachfolgende Berechnungen in anderen Modulen können natürlich somit in diesem Sinne ebenfalls nur vorläufige Ergebnisse liefern. Weiterhin werden datenbasierte (oftmals aufgrund des zellenweisen Zugriffs aufwendige) Parameterberechnungen auch nur dann durchgeführt, wenn der entsprechende Parameterport aktiv, also zur Erfüllung der aktuellen Benutzeranforderung relevant ist. Schließlich werden auch manuelle Änderungen automatischer Parameter sowie Ergebnisse von Verbundmodulen, die aufgrund lokaler nicht „nach außen“ propagierter Änderungen innerhalb des gekapselten Analysegraphen nicht mehr aktuell sind, und hiervon abhängige Berechnungen nur als vorläufig angesehen. Insgesamt werden als Stati unter dem Aspekt der *Aktualität* nur *vorläufig* und *aktuell* unterschieden.

- Als letzte Status–Komponente wird auch die Anwendbarkeit von Verfahren auf anliegende Daten und Parameterwerte sowie die Eindeutigkeit der Berechnung vermerkt. Anwendbarkeitsprüfungen auf multidimensionalen Datenräumen beziehen sich gemäß Tab. 5.3 nur auf deren Schemata. Prinzipiell wären jedoch auch datenbasierte Tests denkbar, die jeweils unberücksichtigt bleiben müßten, wenn nur Metadaten als Dateneingänge zur Verfügung stünden, so daß — analog zur Behandlung extensionsbasierter Parameterdefinitionen — bei nachträglicher Bestimmung der Datenraumextensionen ein Anwendbarkeitsurteil evtl. zu revidieren wäre.[52] Es ergibt sich ein *Anwendbarkeitsstatus* mit den Werten $\{$ *abgelehnt*, *mehrdeutig*, *eindeutig* $\}$.

Statusinformationen werden im folgenden durch die Funktion

$$status : PO \cup REL \rightarrow Datenstatus \times Propagierungsstatus \times Aktualität \times Anwendbarkeitsstatus$$

auf Ports und Kanälen repräsentiert. Wie schon bei der Definition der Daten- bzw. Wertebelegungen in Def. 5.29 und 5.30, so ergeben sich auch für die $status()$–Funktion Abhängigkeiten zwischen den Stati von gemäß Fan–in bzw. Fan–out verbundenen Ports einerseits und den entsprechenden Kanälen andererseits. Tabelle 5.4 gibt einen Überblick über allgemeine Regelungen — ein „$\surd$“ signalisiert jeweils, daß eine Ausprägung für eine Entität im Rahmen des Verarbeitungsprozesses explizit definiert wird. Andernfalls ist angegeben, aus den Stati welcher anderen Entitäten die Bestimmung des Status vorgenommen wird. Unter den Parameterports wird unterschieden, ob ein Port über Kanäle Werte von anderen Ports importieren kann ($in? = true$) oder ob seine Belegung nur durch lokale Berechnungen (oder natürlich manuell) erfolgt ($in? = false$). Die Spalte „Inhalt“ steht für den Daten- bzw. Parameterwert eines Ports oder Kanals.

[52] Derartige datenbasierte Prüfungen (etwa auf statistisch relevante Datencharakteristika) würden erst bei einer Erweiterung von *VIOLA* zu einem statistischen Expertensystem (vgl. Abschnitt 7.6) nötig werden, so daß wir auf diese hier nicht näher eingehen wollen.

Element	Inhalt	Datenstatus	Propagierung	Aktualität	Anwendbarkeit
$po \in PO^{\text{in}}$	$f^{\text{in(e)}}$	$f^{\text{in(e)}}$	$\surd$	$f^{\text{in(e)}}$	eingehende Kanäle
$po \in PO^{\text{out}}$	$\surd$	$\surd$	$\surd$	$\surd$	$f^{\text{in(i)}}$
$po \in PO^{\text{par}}$ (in?)	$\surd$	$\surd$	$\surd$	$\surd$	$\surd^2$ (nicht–leere Eingangskanäle)
$po \in PO^{\text{par}}$ ($\neg in$?)	$\surd$	$\surd$	$\surd$	$\surd$	$\surd^2$ ($f^{\text{in(i)}}$)
$(po, po') \in REL^{\text{dat}}$	po	po	po	po	$\surd$
$(po, po') \in REL^{\text{par}}$	po	$\surd^1$ (po)	po	po	$\surd$

[1] lediglich im Hinblick auf die Kennzeichnung „leerer“ Kanäle
[2] nur, falls der Parameterwert manuell definiert wurde

Tabelle 5.4: Lokale Definition bzw. Herleitung der Stati von Programmelementen

Die Übernahme eines Statuswertes erfolgt im Falle einer einzelnen Quellentität als Wertekopie, im Falle einer Menge von Quellentitäten für den Inhalt gemäß Def. 5.29 und 5.30 sowie für die vier Stati auf der Basis einer Wertepriorisierung entsprechend obiger Statusdefinitionen (erstgenannt ist jeweils die Ausprägung mit der höchsten Priorität). Dateneingangsports mit leerem Fan–in erhalten als Datenstatus stets *leer* und sind *aktuell*; die Anwendbarkeit von Datenports ist bei leerer Menge von Quellentitäten stets *eindeutig*; Parameterports ohne externen Fan–in müssen manuell definiert sein und haben somit einen eigenständigen Anwendbarkeitsstatus. Einige Ausnahmen von obiger allgemeiner Darstellung sind im im Rahmen der Daten- und Parameterübergabe im Verbundmodul erforderlich, um eine konsistente, intuitive Statusdefinition zu gewähren:

- Ausgangsports von Eingangsmodulen mod^{in} übernehmen die Anwendbarkeit des vom Eingang des Verbundmoduls zugeordneten Kanals $quellkanal(mod^{\text{in}})$. Falls kein Kanal zugeordnet werden konnte, ist der Anwendbarkeitsstatus *abgelehnt*.
- Ausgangsports von Verbundmodulen bestimmen (wie alle anderen Module auch) ihre Anwendbarkeit aus dem jeweils zugehörigen Eingangsport und den Parameterports des Verbundmoduls (und nicht aus ihrem Fan–in, also dem Ausgangsmodul des Verbundes).
- Eingangsports von Verbundmodulen berücksichtigen bei der Statusdefinition nicht–zuzuordnende optionale Eingänge (vgl. Abschnitt 5.4.1) nicht.

Keine spezielle Behandlung erfordern in diesem Kontext Eingänge von Ausgangsmodulen, Parameterschnittstellenmodule und Parameterports von Verbundmodulen; auch Schleifen- und Abbruchmodule fügen sich nahtlos in das allgemeine Schema ein.

5.4.3 Grundlegende Überlegungen zur Programmausführung

Bereits im vorigen Abschnitt wurde die Möglichkeit, multidimensionale Datenräume in bestimmten Phasen der Verarbeitung auf ihre Metadaten (Schemata) zu beschränken, vorgestellt. Bei der Konzeption der Verarbeitung von *VIOLA*–Programmen wollen wir diese Alternative insofern konsequent nutzen, als zu jeder Zeit[53] der Programm–Konstruktion und –Ausführung in allen Datenports jeweils zumindest das aufgrund der aktuell spezifizierten Quelldaten gültige (und relativ schnell zu bestimmende) Schema des Berechnungsergebnisses vorliegen soll. Leere Datenports können also nur auftreten, wenn Verfahren auf Eingangsdaten nicht anwendbar sind, nicht aber, weil eine Berechnung noch nicht durchgeführt wurde. Metadatenbasierte Parameterberechnungen werden stets ausgeführt. Da dies für datenbasierte Berechnungen nicht immer möglich ist, sind einzelne Ergebnisse (Datenräume oder Parameter) u. U. nur vorläufig zu bestimmen.

[53] Genauer gesagt soll dies für alle Zeitpunkte gelten, zu denen ein Anwender mit dem System interagieren kann — temporäre Stati mit invaliden Daten, die im Zuge der Verarbeitung einer Benutzeranforderung auftreten, sind natürlich möglich.

Durch explizite Vorgaben kann der Anwender steuern, ob für ihn die Metadatenpropagierung ausreichend oder eine (evtl. auf ausgewählte Module beschränkte) vollständige Berechnung gewünscht ist. Dieses Vorgehen bietet einen sinnvollen Kompromiß zwischen Bedienungskomfort durch das Angebot aktueller Metadaten und geringen Antwortzeiten bei der Programm–Manipulation.

Benutzerinteraktionen, die den Verarbeitungsprozeß von *VIOLA* anstoßen können, sind generell das Erzeugen, Löschen oder Verbinden von Modulen bzw. Ports, das Ändern von Parametern sowie das explizite Anfordern der etwa für Visualisierungen oder andere Datensenken nötigen Daten. In Abschnitt 5.4.5 wird noch genauer auf die Behandlung dieser Interaktionen eingegangen; hier soll zunächst nur das grobe Prinzip eines Verarbeitungsschrittes vorgestellt werden, der nach Abschluß wieder alle invaliden (Zwischen–)Ergebnisse durch „korrekte“ Belegungen ersetzt hat:

1. Es wird (unter transitiver Betrachtung über Kanäle verbundener Ports) festgelegt, welche Ports zur Erfüllung einer Anforderung Daten oder Metadaten erhalten sollen. Entsprechend wird der jeweilige Propagierungsstatus (von *offen*) auf *aktiv* oder *passiv* gesetzt.
2. Alle Ports werden invalidiert, deren (transitiv über den Fan–in definierte) Datengrundlage sich ändert, weil vorläufige Belegungen aktualisiert werden sollen oder Parameter geändert wurden. Wird nur ein Datenschema durch die zugehörige Extension ergänzt, ist keine Invalidierung nötig.
3. Ein oder mehrere Ausgangspunkte der Datenpropagierung werden gesucht, von denen aus alle nicht–offenen (also alle aktiven oder passiven) und alle invaliden Ports (transitiv) über den Fan–out erreichbar sind.
4. Von diesen aus werden die neuen (Zwischen–)Ergebnisse durch den Analysegraphen propagiert.

Hiermit werden Elemente anforderungsgetriebener (Teilschritt 1 und 3) und datengetriebener Verarbeitung (Teilschritt 2 und 4) in einer flexiblen Programmausführung kombiniert. Während des gesamten Ablaufs sind keine zwischenzeitlichen Benutzerinteraktionen möglich.

Alle Ports, die vor dem letzten Propagierungsschritt einen Datenstatus ungleich *invalide* aufweisen, behalten ihre Inhalte unverändert — allenfalls werden Schemata von Datenräumen um die zugehörige Extension ergänzt. Ist der Datenstatus eines Ports *invalide* oder der Propagierungsstatus *offen*, gilt dies per Definition jeweils auch (transitiv) für alle seine Nachfolger.

Während die ersten drei Schritte des obigen Ablaufs problemlos (für verschiedene Teilbereiche eines Analysegraphen) ineinander verzahnt ablaufen können, ist vor Beginn der letzten, vierten Phase stets der Abschluß der vorangehenden Phasen abzuwarten, um unnötige Mehrfachberechnungen eines Ports zu vermeiden. Darüber hinaus wird die Durchführung überflüssiger, nicht interessierender Berechnungen bereits durch den ersten (anforderungsgetriebenen) Schritt der Verarbeitung vermieden.

Zustandsübergänge in *VIOLA*–Komponenten

Auf Basis obiger Überlegungen lassen sich die Übergänge zwischen den Stati der Komponenten eines *VIOLA*–Programms anhand von Zustandsdiagrammen veranschaulichen. Wir verwenden die *UML*–Notation (vgl. [FS98]). Zustandsübergänge sind jeweils in der Form „Ereignis [Bedingung] / Aktion“ beschriftet, wobei nicht interessierende Komponenten auch entfallen können. Zur Vereinfachung der Darstellung werden nur zentrale Klassen von Zuständen durch ihre charakteristischen Komponenten aus dem Tripel (Datenstatus, Propagierungsstatus, Aktualität) angegeben. Für Zustandsübergänge nicht relevante Ausprägungen sind durch „. . .“ symbolisiert, beim Eintritt in einen Zustand bleiben sie jeweils unverändert. Angaben zur Anwendbarkeit von Modulen (bzw. zum entsprechenden Port– oder Kanal–Status) werden — sofern relevant — lediglich im Kontext von einzelnen Zustandsübergängen gemacht; die Aufnahme entsprechender zusätzlicher Stati bzw. Statuskomponenten hätte zu Unübersichtlichkeit ohne großen Erkenntnisgewinn geführt.

Zur Betrachtung der Zustände von Programmkomponenten zur Speicherung multidimensionaler Datenräume werden die Ausgangsports gewählt, da an ihnen unmittelbar der Zustandsübergang bei der Berechnung neuer Daten verdeutlicht werden kann und aus ihrem Zustand der aller anderen Komponenten (bis auf die

hier nicht betrachtete Anwendbarkeit) ableitbar ist. Die Transitionen eines Datenports gemäß Abb. 5.9 ergeben sich aus den Benutzerinteraktionen

- Kennzeichnung der *Anforderung* von Daten oder Schema durch das betrachtete oder ein nachfolgendes Modul,
- Propagierung der Meldung einer *Änderung* eines Parameter- oder Datenwertes aus dem Fan–in sowie
- Durchführung der *Berechnung* neuer Daten im betrachteten Modul (nur hiernach ändert sich der Dateninhalt des Ports)

und „durchlaufen“ die Module/Ports des Analysegraphen jeweils rückwärts (von den Senken zu den Quellen: Anforderung) bzw. vorwärts (von Quellen zu Senken: Änderung, Berechnung).[54]

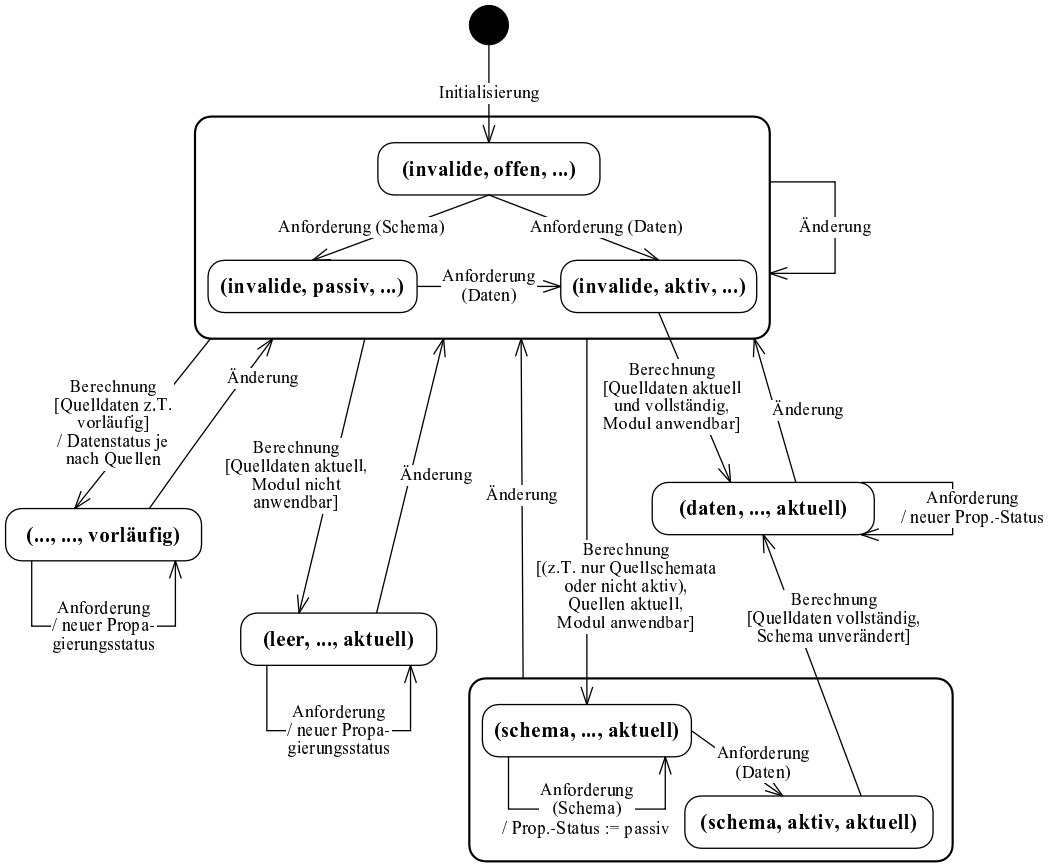

Abbildung 5.9: Zustandsübergänge in Datenausgangsports

Man sieht, daß jede für den Port relevante Änderung zunächst zu seiner Invalidierung führt; lediglich bei der Ergänzung eines Datenraum–Schemas um die jeweilige Extension erfolgt eine Berechnungstransaktion nicht ausgehend von einem invaliden Port. Nicht dargestellt ist das Zurücksetzen des Propagierungsstatus auf *offen* nach Abarbeitung einer Anforderung sowie das explizite Propagieren der Vorläufigkeit bei der Bearbeitung von Verbundmodulen, wie wir es weiter unten noch kennenlernen werden.

Abbildung 5.10 zeigt die Zustandsübergänge eines Parameterkanals. Gegenüber den Parameterports spielt der Datenstatus *leer* für Parameterkanäle eine besondere Rolle, weshalb ihnen hier der Vorzug gegeben wird. Anders als bei der Behandlung von Datenräumen ist in der Parameterverarbeitung die Betrachtung von Metadaten

[54] Im Rahmen der Verarbeitung von Verbundmodulen gibt es einzelne, geringfügige Ausnahmen von diesem Schema, die aber an dieser Stelle noch nicht von Bedeutung sind.

irrelevant; somit entfallen der Datenstatus *schema* und der Propagierungsstatus *passiv*.[55] Dementsprechend sind der Propagierungsstatus sowie die Anforderungs–Transition auch von keiner wesentlichen Bedeutung für die dargestellten Zustandsklassen. An interessierenden Übergängen ergeben sich vielmehr hier

- Propagierung der Meldung einer *Änderung* aus dem Fan–in,
- Werte–*Berechnung* im Quellmodul des Kanals, worunter auch eine Weiterleitung (ein Import) von Werten aus anderen Modulen an den Quellport (aufgrund von dortigen manuellen oder automatischen Definitionen) subsumiert werden soll — in letzterem Fall gibt es bzgl. der Anwendbarkeit des Quellmoduls natürlich keine Einschränkungen — sowie hier zusätzlich
- manuelle, *lokale Definition* des Parameterwertes im Quellport und
- *Trennung* vom Zielport po^{par} des Kanals, d. h. dieser berücksichtigt den betrachteten Kanal gemäß $f^{\text{parjoin}}_{po^{\text{par}}}$ nicht oder aber er wurde manuell definiert und ist nicht automatisch[56].

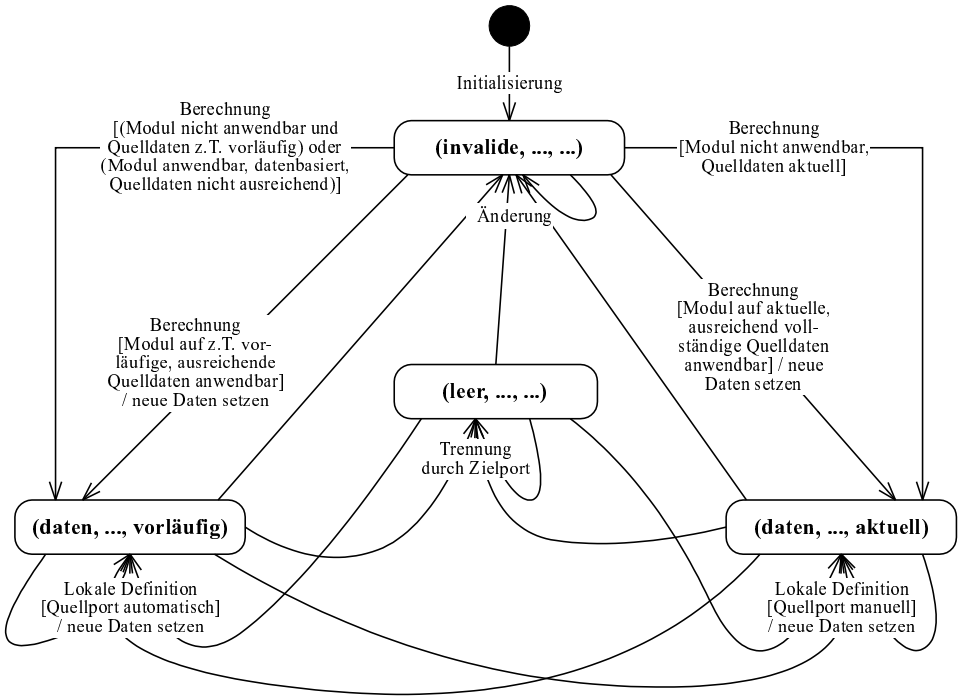

Abbildung 5.10: Zustandsübergänge in Parameterkanälen

Nur die hier nicht näher betrachtete Anforderungs–Transition breitet sich in der Parameterverarbeitung — wie auch schon über die Datenkanäle — rückwärts durch den Analysegraphen aus und setzt den Propagierungsstatus von Parameterkanälen stets auf *aktiv*. Änderungs- und Berechnungsmeldungen werden wieder vorwärts weitergeleitet, die Trennung erfolgt rein lokal, und die lokale Definition führt erst zur Meldung einer Änderung

[55] Auch Datenräume als Parameter sollen keine grundsätzliche Sonderbehandlung erfahren und werden entweder „vollständig“ oder gar nicht ermittelt, nicht jedoch — wie bei der Datenpropagierung — evtl. erst „nachträglich“ gefüllt. Allerdings kann diese „Vollständigkeit“ u. U. von unterschiedlichen Parameterdefinitionsmodulen entweder als Bereitstellung eines gesamten Datenraums mit Extension (in einem datenbasierten Parameter) oder lediglich als Extraktion des Datenraumschemas (in einem schemabasierten Parameter) aufgefaßt werden.

[56] Andernfalls — wenn der Zielport automatisch ist — soll aus folgendem Grund nicht von Trennung gesprochen werden: Mit dem Zielport muß per Definition auch der Quellport automatisch sein, denn aufgrund der Existenz eines verbindenden Kanals gilt $po^{\text{par}}.in? = true$, so daß im Zielmodul keine Parameterberechnung erfolgen kann. In diesem Fall bleibt der Kanal aber weiterhin relevant für den Zielport (ist nicht von ihm „getrennt“), da über ihn bei der nächsten automatischen Berechnung wieder ein neuer Parameterwert propagiert würde.

und anschließend zur Propagierung einer Berechnung. Anders als bei der Datenberechnung können in der Parameterverarbeitung durchaus auch einmal (vorläufig) alte Parameterwerte beibehalten werden. Aus diesem Grund ist in Abb. 5.10 explizit angegeben, ob der Inhalt eines Kanals neu gesetzt wird.

Es fällt auf, daß fast von jedem Zustand ein Übergang in jeden anderen Zustand möglich ist — lediglich eine Trennung kann nicht aus dem Zustand einer invaliden Belegung erfolgen. Klar erkennbar ist, daß der Zielzustand jeweils bereits durch die vorliegende Interaktion und nicht durch den Quellzustand bestimmt wird. Hierbei können jedoch nicht alle Ereignisse in jedem Zustand eintreten.

Auch für Parameterkanäle wurde (neben dem expliziten Setzen der Vorläufigkeit) ein möglicher Zustandsübergang nicht in die Darstellung aufgenommen: Wenn die Trennung vom Zielport wieder aufgehoben wird, übernimmt der jeweilige Kanal den Zustand seines Quellports.

Verarbeitungsprimitive

Es werden folgende grundlegende Algorithmen unterschieden, nach denen die Daten- und Parameterverarbeitung in *VIOLA* erfolgt und die oben vorgestellte Zustandsübergänge zur Folge haben:

Fordere(umfang) propagiert Anforderungen nach Daten, Parameterwerten oder Metadaten rückwärts (von Datensenken zu -quellen) durch den Analysegraphen. Im Rahmen von Verbundmodulen ist hierbei ein spezieller Auftrag an Abbruchmodule zur Abarbeitung des *gesamten* Teilgraphen eines Verbundes sowie eine ähnliche Forderung an Parameterschnittstellenmodule durch den zugeordneten Verbund–Parameterport von Anforderungen an diese beiden Modularten, die von Modulen *innerhalb* des Verbundes initiiert wurden, abzugrenzen. Entsprechend erhält die Prozedur ein Argument *umfang* mit den möglichen Ausprägungen $\{$ *extension*, *schema*, *verbund* $\}$. (Bei der Parameteranforderung bleibt die Unterscheidung zwischen Schema und Extension jeweils unberücksichtigt.)

Propagiere(modus) dient der Vorwärts–Propagierung von Änderungsmeldungen (also Invalidierungen), Daten/Parameterwerten sowie — lediglich innerhalb und ausgehend von Verbunden — des expliziten Setzens der Vorläufigkeit, sofern sie nicht (wie in den meisten Fällen) direkt bei der Wertepropagierung belegt wird. Ein Argument *modus* hat somit die möglichen Ausprägungen $\{$ *daten*, *invalide*, *vorläufig* $\}$.

Prüfe() führt jeweils in Daten- oder Parametereingängen vor der eigentlichen Berechnung und Propagierung nötige Anwendbarkeitsprüfungen und für Eingänge von Verbundmodulen sowie Parameterports mit mehreren Eingängen Zuordnungen von eingehenden Kanälen durch.

Definiere() behandelt die manuelle Parameterport–Belegung und stößt die Propagierung an.

Initialisiere(modus) spielt lediglich eine Rolle bei der Initialisierung von (Schleifen in) Verbundmodulen. Initialisiert werden müssen Abbruch- und Schleifenmodule. Über den *modus* ($\{$ *daten*, *invalide* $\}$) wird — analog zur Propagierung — angegeben, ob zunächst eine Invalidierung der Schleife oder der Beginn der Abarbeitung erfolgen soll.

Abbildung 5.11 gibt in einem *UML*–Sequenzdiagramm (vgl. [FS98]) anhand einiger exemplarischer Ports und unter Abstraktion von einzelnen Details[57] einen groben Überblick über das Zusammenwirken der verschiedenen Primitive bei der Bearbeitung einer Datenanforderung in einem beliebigen Port (rechts oben). Die Forderung wird transitiv gemäß des Fan–in der betrachteten Ports in Richtung der Datenquellen weitergeleitet, bis der Fan–in eines Ports leer ist oder bereits die gewünschten Daten bzw. Parameterwerte vorliegen. Während der Anforderungspropagierung erfolgt ggf. aus den dabei berührten Ausgängen eines Verbundes dessen Initialisierung über das Abbruchmodul. (Von hier aus werden auch die Schleifenmodule initialisiert — auf die Details wird in Abb. 5.11 nicht eingegangen.) Außerdem wird der Fan–out vorläufiger Parameterports als Vorbereitung auf eine vollständige Neuberechnung invalidiert — auch dies wird transitiv fortgesetzt.

[57] Dies betrifft vor allem Aspekte der Verbundbehandlung, die im nächsten Abschnitt noch genauer diskutiert wird.

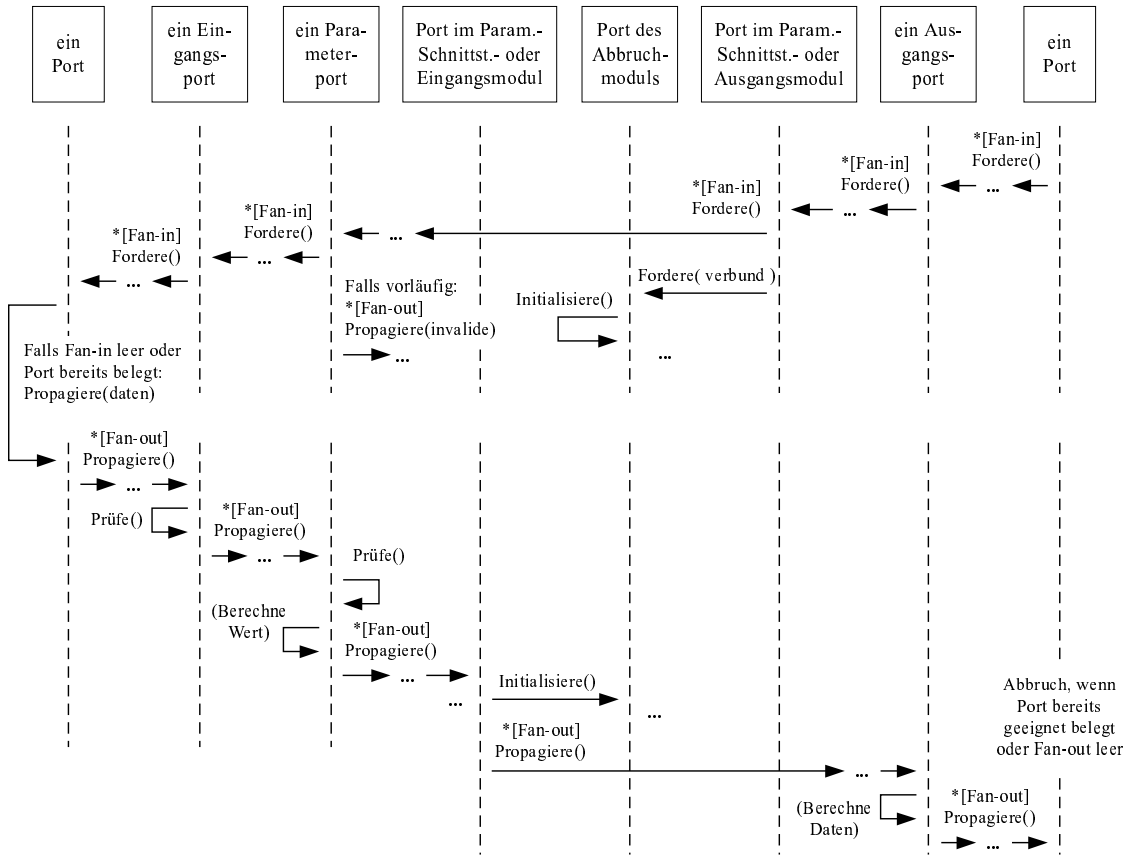

Abbildung 5.11: Bearbeitung einer Datenanforderung in *VIOLA* (*UML*–Sequenzdiagramm)

Wie bereits zu Beginn von Abschnitt 5.4.3 motiviert, ist die Datenpropagierungs- von der Anforderungsphase abgetrennt. In Eingangs- und Parameterschnittstellenports werden Anwendbarkeitsprüfungen durchgeführt, in Parameterschnittstellen- und Ausgangsports erfolgen Parameter- und Datenraumberechnungen. Ändern sich die Eingänge eines Verbundes, wird dieses wiederum initialisiert, bevor die Datenpropagierung das Verbundmodul durchläuft. Die Propagierung endet jeweils in den Ports, in denen bereits gültige Daten vorliegen bzw. deren Fan–out leer ist.

In Abb. 5.11 nicht aufgeführt ist die *Definiere*–Prozedur. Sie wird lediglich dann angestoßen, wenn eine *manuelle* Änderung in einem Parameterport erfolgt. Es schließen sich Invalidierung (*Propagiere*(*invalide*)) und Datenpropagierung entsprechend der unteren Hälfte der Abbildung an. Auf weitere Varianten der Verwendung der vorgestellten Verarbeitungsprimitive kommen wir im Überblick in Abschnitt 5.4.5 zu sprechen.

Verarbeitung von Schleifen bzw. Verbundmodulen

Da im Kontext der Verarbeitung von Verbundmodulen, insbesondere von Schleifen, einige Spezialfälle der Behandlung von Modulports zu berücksichtigen sind, wollen wir die Umsetzung der Verbundverarbeitung etwas genauer diskutieren. Wenn auch in *VIOLA* nur eine relativ einfache Form von Schleifen implementiert wird, so erfordern doch auch diese einen etwas komplexeren Verarbeitungsalgorithmus. Vor allem ist zu gewährleisten, daß die Schleife stets als Ganzes abgeschlossen wird, bevor ihre Ergebnisse an die „Außenwelt“ des jeweiligen Verbundmoduls propagiert werden. Fordert man diese stets vollständige Abarbeitung auch für Verbundmodule, die keine Schleifen implementieren (generell wohl auch ein durchaus einleuchtender Ansatz), so läßt sich die Verarbeitung von Verbunden stets in einheitlicher Weise realisieren. Dementsprechend werden Verbundmodule

im folgenden in der Regel nicht bzgl. des Vorliegens von Schleifenmodulen differenziert.

Immer wenn am Verbund neue Daten oder Parameterwerten an entsprechenden Eingängen anliegen, wird also der vollständige Analysegraph neu abgearbeitet. Es wird der Einfachheit halber nicht unterschieden, welche internen Module von welchen externen Parametern, Eingangsdaten oder Schleifenzählern abhängen; stets werden auch alle Schleifenzähler neu initialisiert. Im Prinzip ist dieses Vorgehen durchaus konform mit der Behandlung anderer Module, da auch dort nicht geprüft wird, inwiefern bestimmte Parameter einen Einfluß auf das Gesamtergebnis haben.

Umgekehrt richten sich Anforderungen zur Datenberechnung „von außerhalb“ an die Verbundausgänge (Daten oder Parameter) prinzipiell auch immer an das ganze Verbundmodul. Dies bedeutet, daß dieses nicht nur dann vollständig neu auszuführen ist, wenn das jeweilige Ausgangsmodul nur vorläufige oder unvollständige Daten enthält, sondern auch, wenn der Abbruchparameter (des Abbruchmoduls) nur vorläufig bestimmt werden konnte. Wenn auch das Ergebnis einer Schleife (abgesehen von intermediären Seiteneffekten wie Speicherungen von Zwischenergebnissen etc.) gerade das Ergebnis des letzten Schleifendurchlaufs ist, so ist doch zur Gewährleistung von dessen Aktualität immer auch zu prüfen, ob der Abbruchparameter *irgendwann* während der Iteration einmal nur vorläufig definiert war. In diesem Fall wäre mit aktuellen und vollständigen Angaben womöglich ein vorzeitiger Schleifenabbruch erfolgt, der somit auch zu anderen Schleifenergebnissen geführt haben könnte. Nicht erkannt werden kann durch die Betrachtung der Ergebnisse des *letzten* Durchlaufs natürlich, ob bestimmte Seiteneffekte im Laufe der Iteration nicht oder nur unvollständig erfolgt sind. Diese Einschränkung soll im Sinne geringerer Verarbeitungskomplexität akzeptiert werden.

Schließlich ist dem Anwender auch die Möglichkeit zu geben, nur „lokale“ Änderungen innerhalb eines Verbundes vornehmen zu können, ohne daß eine evtl. vorgesehene Iteration immer wieder von vorne begonnen wird und ohne daß Berechnungsergebnisse die Schleife „verlassen“. Insgesamt ergibt sich folgendes Bild:

- Anforderungen, die ihren Ursprung „hinter“ dem Verbundmodul haben, führen — sofern dessen angefordertes Ergebnis nicht bereits aktuell und vollständig ist — stets zu einer vollständigen Verbundausführung (also ggf. einer Schleifeninitialisierung und einer Bestimmung des Abbruchparameters).
- Anforderungen, die ihren Ursprung innerhalb des Verbundmoduls haben, haben eine vollständige Verbundausführung nur dann zur Folge, wenn sie bis zu Verbundeingängen zurückgeleitet werden. (Schleifenmodule werden hierbei nicht als Eingänge angesehen, da bei lokalen Anforderungen nur der aktuelle Wert des Iterationsparameters interessiert.) Andernfalls ist, wenn sie zu einer Neubestimmung des Abbruchparameters führen, der Fan–out der Schleifenmodule (transitiv) als vorläufig zu kennzeichnen. Bleibt die Bearbeitung lokal, sind Nachfolger betroffener Verbundausgänge auch nur vorläufig (aber nicht notwendigerweise die belegten Ausgangsmodule selbst).
- Nach Änderungen „vor“ dem Verbundmodul wird das gesamte Verbundmodul neu verarbeitet.
- Änderungen innerhalb des Verbundmoduls werden wiederum wie eventuelle Auswirkungen lokaler Anforderungen behandelt. Alle Neuberechnungen bleiben verbundintern; „nach außen“ wird nur ggf. Vorläufigkeit propagiert.

Zur Realisierung dieses Konzepts wird dem Abbruchmodul jeweils die Steuerung eines Verbundmoduls übertragen. Das Abbruchmodul wird von Daten- und Parametereingängen initialisiert, sorgt für die Ermittlung des Abbruchparameters und initialisiert seinerseits alle vorhandenen Schleifenmodule. Diese übernehmen ihre Werte somit erst auf Befehl des Abbruchmoduls, nicht bereits aufgrund neuer Daten im jeweiligen mengenwertigen Verbundparameter. Auch bei einer nicht bereits durch die vorliegenden Portinhalte zu erfüllenden Anforderung an die Verbundausgänge tritt das Abbruchmodul in Aktion und startet die Verbundverarbeitung neu.

Entsprechend leiten Daten- und Parameterausgangsmodule ihre Inhalte nach Ermittlung auch nicht selbständig über die Verbundgrenzen hinaus, sondern erst nach vollständiger Abarbeitung des gesamten Analysegraphen prüft das Abbruchmodul, ob eine weitere Iteration einer Schleife anzustoßen ist oder aber die Verbundausgänge neu belegt und deren Inhalte weiterpropagiert werden können. Zum Schleifenende werden außerdem alle Nachfolger der Schleifenmodule als vorläufig gekennzeichnet, wenn das Abbruchmodul selbst

nur vorläufig definiert wurde oder dies im Laufe der Iteration einmal war (vgl. Abb. 5.12). Schließlich kennzeichnen Ausgangsmodule mit jeder internen Belegung die entsprechenden Verbundmodulports (und deren Fan–out) als vorläufig, um auch die korrekte Behandlung verbundinterner Interaktionen zu gewährleisten. Diese führen nämlich nicht zu einer Initialisierung des Abbruchmoduls, so daß auch später keine Propagierung von Portinhalten aus dem Verbundmodul heraus erfolgt.

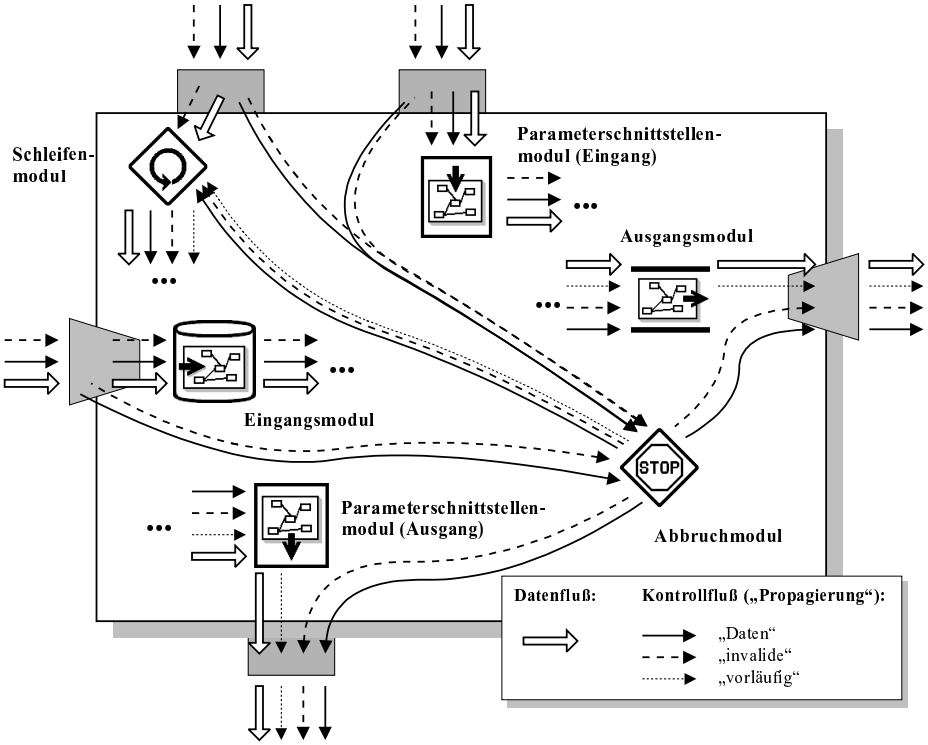

Abbildung 5.12: Verarbeitung von Verbundmodulen bzw. Schleifen

Abbildung 5.12 gibt einen Überblick über den Daten- und Kontrollfluß im Rahmen der (vollständigen) Abarbeitung eines Verbundmoduls. Der Kontrollfluß ist nach den verschiedenen Argument–Ausprägungen der *Propagiere*–Prozedur differenziert. Die verwendeten Icons für Untermodule sind die gleichen wie schon in Abb. 5.8. Wenn auch die beiden Parameterports am oberen Rand des Verbundmoduls hier in erster Linie zur Aufnahme von außen eingehender Werte fungieren, so können sie gemäß Definition natürlich stets auch über einen externen Fan–out ihre Inhalte an andere Ports propagieren. Auf entsprechende graue Kästen am unteren Bildrand wurde in Abb. 5.12 der Übersichtlichkeit halber verzichtet. Weiterhin besteht i. a. auch ein verbundinterner Datenfluß von und zu Abbruchmodulen — auch dieser fehlt im Bild. Man sieht insbesondere

- die weitgehend gleichartige Behandlung von Invalidierung und Datenpropagierung — nur die Änderung der Listenparameter, auf die Schleifenmodule zugreifen, wird nicht direkt in die Schleife geleitet, um eine Umgehung der Schleifeninitialisierung zu vermeiden,
- die zentrale Rolle des Abbruchmoduls, das jeweils bei Änderung beliebiger Verbundeingänge initialisiert wird, seinerseits die Schleifenmodule kontrolliert und für die Propagierung von Invalidierungen und Berechnungsergebnissen aus dem Verbund heraus verantwortlich ist, und
- die durch das Abbruchmodul angestoßene Kennzeichnung von Vorläufigkeit der Nachfolger der Schleifenmodule im Rahmen verbundlokaler Berechnungen — nicht dargestellt ist, daß auch andere Module innerhalb des Verbundes (insbesondere auch die Ausgangsmodule) ihren Fan–out als vorläufig markieren

können und daß auch über die Verbundeingänge Vorläufigkeit von auf anderen Verbunden basierenden Berechnungen in das Verbundmodul hineinpropagiert werden kann.

Eine weitere Entwurfsentscheidung soll nicht unerwähnt bleiben: Auch wenn nur vorläufige Daten oder nur Datenschemata zum Beginn einer Schleifenausführung als Eingangsdaten zur Verfügung stehen, werden stets alle Iterationen vollständig durchlaufen. Denkbar wäre es, in derartigen Fällen den Schleifenrumpf nur exemplarisch auf den jeweils ersten oder letzten Listenelementen des Schleifenparameters auszuführen und hierdurch oftmals unnötig für lediglich vorläufige Berechnungen eingesetzte Rechenzeit einzusparen. Um die Schleifenbearbeitung aber nicht zu kompliziert werden zu lassen und da zudem ein derartiger Ansatz auch nicht unbedingt für den Anwender intuitiv wäre, soll es ihm vorbehalten sein, zur Vermeidung unnötiger Schleifendurchläufe entweder für einelementige Schleifenparameter zu sorgen oder aber die Schleife durch vorübergehendes Löschen von Kanälen ganz von der Datenpropagierung abzutrennen.[58]

5.4.4 Portbezogene Algorithmen für Verarbeitungsprimitive

Wie man bereits in Abb. 5.11 sehen konnte, resultiert der *VIOLA*–Verarbeitungsprozeß vollständig aus dem Zusammenwirken von Eingangs-, Ausgangs- und Parameterports. Stati und Inhalte von Ports und Kanälen ergeben sich im Rahmen der Portverarbeitung gemäß Tab. 5.4. Die dort implizit definierten, aus den Stati anderer Programmkomponenten ableitbaren Belegungen werden in den folgenden Darstellungen nicht mehr explizit angeführt.

Die in diesem Abschnitt vorgestellten Algorithmen zur Realisierung der *Fordere–*, *Propagiere–*, *Prüfe–*, *Definiere–* und *Initialisiere–*Prozeduren sind in bestimmten Fällen eher etwas konservativer als nötig, es werden teilweise häufiger und mehr Bedingungen abgeprüft, als unbedingt erforderlich wäre. Dies dient allein zur Verdeutlichung jeweils bestehender Nebenbedingungen und hat keinen Einfluß auf Korrektheit oder Größenordnung des Bearbeitungsaufwandes. Auch erfolgt die Abfolge und Gruppierung von Bedingungsprüfungen und Zustandsveränderungen nicht unter Effizienzaspekten, sondern im Sinne bestmöglicher Übersichtlichkeit. Um einen möglichst kompakten Überblick — auch über Unterschiede in der Verarbeitung von Ports unterschiedlicher Module — zu ermöglichen, wird auf Unterprozeduren verzichtet. Aus dem gleichen Grund verwenden wir auch eine prozedurale Darstellung, obwohl sich — aufgrund der stark modul- bzw. portzentrierten und nach Modulklassen differenzierten Verarbeitung — für eine Implementierung sicher ein objektorientierter Ansatz anbieten würde.

Eine Verifizierung der vorgestellten Algorithmen müssen wir leider schuldig bleiben, allerdings lassen sich viele Aspekte direkt anhand der vorangestellten grundlegenden Konzepte (insbesondere auch anhand obiger Zustands- und Sequenzdiagramme) motivieren.

Wenden wir uns nun der Behandlung der verschiedenen Ports zu. Das dem jeweils betrachteten Port *po* zugehörige Modul $po.mod$ wird im folgenden kurz mit *mod* bezeichnet. Sofern die Instanz eines Analysegraphen eines Verbundmoduls relevant ist, wird dies durch einen Index *i* gekennzeichnet, ansonsten beziehen sich — sofern ein Modul innerhalb eines Verbundmoduls betrachtet wird — alle Angaben immer auf die gleiche Instanz, zu dem auch behandelter Port bzw. dessen Modul gehören.

Dateneingänge

Die Verarbeitung von Dateneingangsports $po = po^{\text{in}} \in mod.PO^{\text{in}}$ gestaltet sich recht einfach: Forderung und Propagierung werden im wesentlichen lediglich an die Ports im Fan–in bzw. Fan–out weitergeleitet (Alg. 5.2 und 5.3), und die Anwendbarkeitsprüfung besteht eigentlich nur in der Anwendung der durch Tab. 5.3 gegebenen Regeln (Alg. 5.4).

Denkbar wäre es, aus der Behandlung der Ausgangsports in Alg. 5.6 eine Abbruchbedingung für die Weiterleitung der Forderung zu übernehmen. In der derzeitigen Darstellung verschiebt sich ein eventueller Abbruch

[58] Eine Option, die wir in Abschnitt 5.4.5 noch einmal aufgreifen werden, wäre es zudem, auch die Metadatenpropagierung im gesamten Programm vollständig abzuschalten.

procedure Fordere(po^{in}: $\mathcal{PO}^{\text{dat}}$, *umfang*: *Umfang*) :
  {Umfang der Anforderung vermerken:}
  $status(po^{\text{in}}) \stackrel{\text{def}}{=}$ **if** $umfang = extension$ **then** $(\ldots, aktiv, \ldots)$ **else** $(\ldots, passiv, \ldots)$;
  {Vorläufige/unvollständige Ergebnisse in Ausgangsmodulen führen zur Verbund–Verarbeitung:}
  if $mod \in \mathcal{MOD}^{\text{out}} \wedge (status(po^{\text{in}}) = (\ldots, vorläufig, \ldots) \vee$
    $(umfang = extension \wedge status(po^{\text{out}}) = (schema, \ldots)))$ **then**
    Fordere($mod.mod^{\text{sub}}.mod^{\text{stop}}.po^{\text{par}}$, *verbund*);
  end if
  if $f^{\text{in}}(po^{\text{in}}) = \emptyset$ **then** {Beginn der Datenpropagierung ...}
    Propagiere(po^{in}, *daten*);
  else {... oder Weiterleitung der Anforderung an den gesamten Fan–in.}
    for all $po \in f^{\text{in}}(po^{\text{in}})$ **do** Fordere(po, *umfang*) **end for**;
  end if
end procedure

Algorithmus 5.2: (Ergebnis–)Anforderung an Dateneingangsports

procedure Propagiere(po^{in}: $\mathcal{PO}^{\text{dat}}$, *modus*: *Modus*) :
  {Anwendbarkeitsprüfung vor Datenpropagierung:}
  if $modus = daten$ **then** Prüfe(po^{in}) **end if**;
  {Der i–te Eingangsport eines Verbundmoduls initialisiert ggf. das i–te Abbruchmodul:}
  if $modus \neq vorläufig \wedge mod = mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}} \wedge po^{\text{in}} = po_i^{\text{in}}$ **then**
    Initialisiere($mod^{\text{sub}}.mod_i^{\text{stop}}.po^{\text{par}}$, *modus*);
  end if
  if $mod \in \mathcal{MOD}^{\text{out}}$ **then** {Ausgangsmodule propagieren stets Vorläufigkeit.}
    for all $po \in f^{\text{out}}(po^{\text{in}})$ **do** Propagiere(po, *vorläufig*) **end for**;
  else {Normalfall: Weiterpropagierung des *modus* an den Fan–out.}
    for all $po \in f^{\text{out}}(po^{\text{in}})$ **do** Propagiere(po, *modus*) **end for**;
  end if
end procedure

Algorithmus 5.3: Propagierung über Dateneingangsports

jedoch lediglich auf den jeweils nächsten (Ausgangs–)Port im Fan–in, so daß hier kein zwingender Bedarf besteht. Aus dem gleichen Grund (zugunsten einer kompakteren Präsentation) wird auch die Propagierung nicht abgebrochen, wenn ein propagierter Status (*invalide* oder *vorläufig*) bereits vorliegt.

Bei Beendigung der rückwärtigen Anforderung schließt sich in der Darstellung in Alg. 5.2 unmittelbar die Datenpropagierung an. Somit wird garantiert, daß alle relevanten, evtl. während der Bearbeitung der Anforderung als *invalide* gekennzeichneten Daten neu bestimmt werden. Das Abwarten der Propagierungsphase (wie zu Beginn von Abschnitt 5.4.3 postuliert) wird in Alg. 5.3 nicht direkt modelliert — dessen Umsetzung wird weiter unten im Kontext der Ausgangsports (in der Erläuterung zu Alg. 5.5) beschrieben. In allen angeführten Algorithmen sieht man einige Sonderfälle, die Verbundmodule oder deren Teilmodule betreffen und gemäß der Überlegungen aus dem vorigen Abschnitt umgesetzt sind.

Zur Prüfprozedur sei noch erwähnt, daß im Rahmen der Abtrennung der Daten–Propagierungsphase von den übrigen Phasen der Programmausführung gewährleistet werden wird, daß früher oder später jeder einmal invalidierte Port neu berechnet wird. Aufgrund der Überlegungen am Ende von Abschnitt 5.3.3 zur hierarchischen (also insbesondere azyklischen) Strukturierung der Abhängigkeiten zwischen Ports im Rahmen der Anwendbarkeitsprüfung, ergibt sich in Alg. 5.4 nicht die Gefahr eines Deadlocks.

procedure Prüfe(po^{in}: $\mathcal{PO}^{\text{dat}}$) :
  {Bestimmung für die Prüfung von po^{in} nötiger Parameter abwarten:}
  wait until $\forall po^{\text{par}} \in mod.PO^{\text{par}}$, die zur Prüfung relevant: $status(po^{\text{par}}) \neq (invalide, \ldots)$;
  if $mod = mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}}$ **then** {Spezielle Behandlung der Eingänge von Verbundmodulen.}
    Ordne eingehende Kanäle den Eingangsmodulen von $mod^{\text{sub}}.ag$ über die Funktion
    quellkanal() zu (vgl. Abschnitt 5.4.1);
  end if
  Weise eingehenden Kanälen Anwendbarkeiten gemäß Tab. 5.3 zu, wobei nur Bedingungen,
  die sich *nicht* ausschließlich auf Parameterwerte beziehen, relevant sind;
end procedure

Algorithmus 5.4: (Anwendbarkeits–)Prüfung in Dateneingangsports

Datenausgänge

Die Formalisierung der Datenpropagierung in Ausgangsports $po^{\text{out}} \in mod.PO^{\text{out}}$ (siehe Alg. 5.5) deutet durch eine Wait–Anweisung den Aufschub der Berechnung in die Propagierungsphase an.

procedure Propagiere(po^{out}: $\mathcal{PO}^{\text{dat}}$, *modus*: *Modus*) :
  if $modus = invalide$ **then** {Falls nötig, Stati setzen und propagieren.}
    if $status(po^{\text{out}}) \neq (invalide, \ldots)$ **then**
      $status(po^{\text{out}}) \stackrel{\text{def}}{=} (invalide, \ldots)$;
      for all $po \in f^{\text{out}}(po^{\text{out}})$ **do** Propagiere($po, invalide$) **end for**;
    end if
  else if $modus = vorläufig$ **then**
    ... {Zur Invalidierung analoge Behandlung.}
    {Nun Datenbestimmung und -propagierung — Abbruch, wenn Port schon geeignet belegt:}
  else if $status(po^{\text{out}}) \notin \{(ok, \ldots), (leer, \ldots), (schema, passiv, \ldots), (schema, offen, \ldots)\}$ **then**
    {Auf geforderte Daten warten und anschließend verarbeiten:}
    wait until $\forall po \in f^{\text{in}}(po^{\text{out}}) : status(po) \notin \{(invalide, \ldots), (schema, aktiv, \ldots)\}$;
    if $status(po^{\text{out}}) = (\ldots, abgelehnt)$ **then** {Verfahren nicht anwendbar.}
      $data(po^{\text{out}}) \stackrel{\text{def}}{=} \emptyset$; $status(po^{\text{out}}) \stackrel{\text{def}}{=} (leer, \ldots)$;
      {Ansonsten Extension berechnen, falls gewünscht und möglich:}
    else if $status(po^{\text{out}}) = (\ldots, aktiv, \ldots) \wedge \forall po \in f^{\text{in}}(po^{\text{out}}) : status(po) \neq (schema, \ldots)$ **then**
      Belege $data(po^{\text{out}})$ mit Daten; $status(po^{\text{out}}) \stackrel{\text{def}}{=} (ok, \ldots)$;
      Generiere ggf. Nebeneffekte (bei $\mathcal{MOD}^{\text{vis}}$, $\mathcal{MOD}^{\text{save}}$);
    else {Extension nicht gefordert oder nicht berechenbar.}
      Belege $data(po^{\text{out}})$ mit Metadaten; $status(po^{\text{out}}) \stackrel{\text{def}}{=} (schema, \ldots)$;
    end if
    {Noch Vorläufigkeit übernehmen und weiter propagieren:}
    $status(po^{\text{out}}) \stackrel{\text{def}}{=}$ **if** $\exists po \in f^{\text{in}}(po^{\text{out}}) : status(po) = (\ldots, vorläufig, \ldots)$ **then**
      $(\ldots, vorläufig, \ldots)$ **else** $(\ldots, aktuell, \ldots)$;
    for all $po \in f^{\text{out}}(po^{\text{out}})$ **do** Propagiere($po, daten$) **end for**;
  end if
end procedure

Algorithmus 5.5: Propagierung über Datenausgangsports

Allgemein werden alle *Propagiere(daten)*–Aufrufe in einer Warteschlange gesammelt, deren Abarbeitung erst begonnen wird, wenn alle übrigen Prozeduraufrufe vollständig behandelt sind. Beim Eintragen in diese

Queue können zur Effizienzsteigerung eventuelle Duplikate eliminiert werden. Aus der Warteschlange werden dann sukzessive nur solche Ports betreffende Aufrufe entnommen, deren Module keine invaliden Eingänge mehr aufweisen.[59] Außerdem sollte natürlich prinzipiell die jeweilige Wait–Anweisung erfüllt sein. Unter Umständen können jedoch Fälle auftreten, in denen deren Bedingung niemals gültig werden kann. Dies tritt genau dann ein, wenn auf das Vorliegen von Daten gewartet wird, aber nur Metadaten angeboten werden. Somit wird stets dann, wenn die Warteschlange nur noch Kandidaten mit unerfüllten Wait–Bedingungen enthält, ein beliebiger Port zur Bearbeitung entnommen und dessen Wait–Anweisung übergangen, sofern kein Eingang des zugehörigen Moduls invalide ist. Anders ausgedrückt: Die Wait–Anweisung wird jeweils auf die Forderung nicht–invalider Portbelegungen eingeschränkt. Auf diese Weise kann die Queue stets vollständig abgearbeitet werden.

Die Datenbelegung im Laufe der Propagierung erfolgt jeweils gemäß der Berechnungsfunktion f^{dat} oder — falls diese nicht definiert sein sollte — durch Übernahme des Datenraums aus dem Fan–in. Typische Nebeneffekte bilden die Abspeicherung von Ergebnissen sowie die Erstellung von Graphiken inklusive der automatischen Belegung der dafür nötigen internen Parameter.

Bisher gehen wir davon aus, daß immer dann, wenn die Anwendbarkeitsprüfung erfolgreich ist, auch ein Ergebnis–Datenraum ermittelt werden kann. In der Praxis sind sicherlich Ausnahmen denkbar. So lassen sich z. B. durch Verbindungsprobleme auftretende Fehler beim Zugriff auf physische Datenquellen (von zugrundeliegenden DBMS verwaltete Datenbanken) oder andere in Analyse- oder Visualisierungsmodulen genutzte externe Systeme nicht über Anwendbarkeitstests in *VIOLA* modellieren. Auch in Verbundmodulen können u. U. Anforderungen an eingehende Datenräume und Parameter nicht immer vollständig über die Parameter der Eingangsmodule spezifiziert werden. Vor diesem Hintergrund ließe sich Alg. 5.5 leicht anpassen, so daß auch nach einem Fehlschlagen der Berechnung der Datenstatus *leer* vorliegt.

Die Anforderungsprozedur (Alg. 5.6) für Datenausgänge bietet gegenüber der Behandlung der Dateneingänge eine differenzierte Abbruchbedingung.

procedure Fordere(po^{out}: $\mathcal{PO}^{\mathrm{dat}}$, *umfang*: *Umfang*) :
  {Abbruch, wenn Port bereits behandelt:}
  if $status(po^{\mathrm{out}}) = (\ldots, offen, \ldots)$ **then**
    {Drei Fälle, die eine Aktion/Statusänderung erforderlich machen:}
    if $status(po^{\mathrm{out}}) = (\ldots, vorläufig, \ldots) \vee status(po^{\mathrm{out}}) = (invalide, \ldots) \vee$
        $(umfang = extension \wedge status(po^{\mathrm{out}}) = (schema, \ldots))$ **then**
      {Neuen Propagierungsstatus setzen:}
      $status(po^{\mathrm{out}}) \stackrel{\mathrm{def}}{=}$ **if** $umfang = extension$ **then** $(\ldots, aktiv, \ldots)$ **else** $(\ldots, passiv, \ldots)$;
      if $f^{\mathrm{in}}(po^{\mathrm{out}}) = \emptyset$ **then** {Propagierung in Datenquellen ($\mathcal{MOD}^{\mathrm{mik}}$, $\mathcal{MOD}^{\mathrm{mak}}$) beginnen.}
        Propagiere(po^{out}, *daten*);
      else {Weiterleitung der Anforderung an den gesamten Fan–in.}
        for all $po \in f^{\mathrm{in}}(po^{\mathrm{out}})$ **do** Fordere(po, *umfang*) **end for**;
      end if
    else {Anforderung erfüllt, da Port bereits geeignet belegt — Propagierung beginnen.}
      Propagiere(po^{out}, *daten*);
    end if
  end if
end procedure

Algorithmus 5.6: (Ergebnis–)Anforderung an Datenausgangsports

Abschließend soll noch auf drei Möglichkeiten zur „Verbesserung“ der vorgestellten Algorithmen hingewiesen werden, die sich recht nahtlos einpassen ließen:

[59] Hieraus können definitiv keine Deadlocks entstehen, da *VIOLA*–Programme zyklenfrei sind (vgl. Def. 5.25). Auf einen exakten Beweis verzichten wir, die Definition und Anwendung der Verarbeitungsprozeduren macht die Gültigkeit dieser Aussage jedoch plausibel.

- Zum einen könnte die Berechnung von Verzweigungsmodulen, falls der jeweilige boolesche Parameter nur vorläufig ermittelt werden konnte, die Eingangsdaten (wiederum „vorläufig“) gleichzeitig an *beide* Ausgänge propagieren und somit dem Anwender erweiterte Informationen zur Durchführbarkeit der weiteren Analyse liefern. Diese Option könnte auch vom Benutzer einstellbar sein.
- Zum anderen ließe sich die Daten–Anforderung in Routing–Modulen bei Vorliegen aktueller Metadaten und aktueller Modulparameter auf diejenigen Eingänge beschränken, die gemäß der Parameter weiterzuleiten sind. Somit könnten unnötige Berechnungen eingespart werden. In der Literatur [AB92, AT95] werden vor diesem Hintergrund Ansätze zur Implementierung des Routing–Operators diskutiert, generell erst den Routingparameter und dann die erforderlichen Eingänge zu ermitteln. Dort ergibt sich jedoch stets aus dem Parameterwert direkt die Menge der relevanten Eingänge; in *VIOLA* ist die Kenntnis der von den Eingängen gelieferten Metadaten zusätzlich erforderlich. Prinzipiell wäre es auch hier denkbar, stets erst Parameter und alle Metadaten zu bestimmen und dann erst nötige Daten anzufordern. Dieser Spezialfall würde den gesamten *VIOLA*–Verarbeitungsalgorithmus jedoch deutlich komplizieren, so daß wir uns auf oben genannte einfache Option beschränken.
- Gibt es im Fan–in eines Ausgangsports Datenports mit aktuellen Metadaten und hat der Ausgangsport den Anwendbarkeitsstatus *abgelehnt*, so könnte schließlich theoretisch darauf verzichtet werden, einen *Fordere(extension)*–Aufruf an diese weiterzuleiten, falls sich hierdurch an der Anwendbarkeit nichts ändern würde. Gilt dies für den gesamten Fan–in, wäre im betrachteten Port dann die Propagierung zu starten. Da sich in der Tat alle bisher vorgestellten Anwendbarkeitstests lediglich auf das Schema von Datenräumen beziehen, wäre eine entsprechende Modifikation des Verarbeitungsalgorithmus direkt umzusetzen.

Parameterports

Parameterports $po^{\text{par}} \in mod.PO^{\text{par}}$ kombinieren in vielerlei Hinsicht die Prüf–, Anforderungs– und Propagierungs–Algorithmen von Datenein- und -ausgängen (Alg. 5.7, 5.8 und 5.9). Außerdem ist hier teilweise Steuerungsfunktionalität von Kontrollstrukturen umzusetzen, so daß die einzelnen Prozeduren etwas umfangreicher ausfallen.

Die Prüfung von Parameterports mit nicht–leerem externen Fan–in verläuft im wesentlichen analog zu den Dateneingangsports, nur daß jetzt noch multiple eingehende Kanäle besonders zu behandeln sind (siehe Alg. 5.7). Außerdem wird hierbei der Wert des Parameterports bereits gesetzt (Zeile 10). Die Anwendbarkeit von Parameterports mit gänzlich leerem Fan–in ist dagegen explizit anhand des jeweiligen Parameterwertes zu definieren (Zeile 19). Für Ports mit $in? = false$ und nicht–leerem internen Fan–in schließlich ist keine Aktion erforderlich, da sich die Anwendbarkeit gemäß Tab. 5.4 aus den jeweiligen Quellports ergibt.

In Anforderungen an Parameterports spielt eine Unterscheidung von *schema* und *extension* im übergebenen Parameter *umfang* keine Rolle. So ergibt sich bei der Weiterleitung des Aufrufs an andere (Daten–)Ports der verwendete Parameterwert allein daraus, ob der aktuell betrachtete Port po^{par} bezüglich des anderen daten- oder schemabasiert ist (Zeile 29 in Alg. 5.8).

Zwei Varianten von *Fordere(verbund)* werden in Alg. 5.8 behandelt: zum einen ein Abbruchmodul und zum anderen ein Parameterschnittstellenmodul betreffend. Im ersten Fall (Zeile 3–4) wird hierdurch explizit das Verbundmodul, also vor allem etwaige Schleifenmodule, initialisiert. Eine direkte Anforderung an eine Neu–Berechnung des booleschen Parameters des Abbruchmoduls ist hieran nicht geknüpft, so daß auch der Propagierungsstatus nicht zu setzen ist. Dem Parameterschnittstellenmodul dagegen (Zeile 8–9) wird durch diesen Aufruf angezeigt, daß eine Anforderung zur Berechnung seines Wertes von außerhalb des Verbundes kommt (Zeile 27). Hier wird *zusätzlich* zu dessen Bestimmung unter gewissen Umständen (wenn der Parameterwert bisher vorläufig ist) die Neuberechnung des gesamten Verbundes initiiert.

Zur Behandlung der Schleifenmodule in Zeile 16–17 läßt sich noch folgendes sagen: Ist ihr Parameterwert selbst vorläufig, nicht aber der zugrundeliegende Verbundparameter, so kann dies nur daran liegen, daß entweder die Schleifenvariable manuell geändert oder vom Abbruchmodul explizit auf vorläufig gesetzt wurde.

procedure Prüfe(po^{par}: $\mathcal{PO}^{\text{par}}$) :
  {Für die Prüfung sind evtl. andere Parameter-, nicht aber Datenports wichtig:}
  wait until $\forall po^{\text{par}} \neq po \in mod.PO^{\text{par}}$, die zur po^{par}–Prüfung relevant: $status(po) \neq (invalide, \ldots)$;
  if $f^{\text{in(e)}}(po^{\text{par}}) \neq \emptyset$ **then** {Bei $po^{\text{par}}.in? = true$ eingehende Kanäle betrachten.}
5: {Keine Anwendbarkeit auf invalide Parametereingänge:}
  for all $po \in f^{\text{in}}(po^{\text{par}})$ mit $status(po) = (invalide, \ldots)$ **do**
    $status(po, po^{\text{par}}) \stackrel{\text{def}}{=} (\ldots, abgelehnt)$;
  end for
  {Auswahl der für den Parameterwert relevanten Eingänge gemäß Def. 5.30:}
10: Bestimme $f^{\text{val}}(po^{\text{par}})$ gemäß $f^{\text{parjoin}}_{po^{\text{par}}}$;
  for all $po \in f^{\text{in}}(po^{\text{par}})$, die entsprechend nicht in $f^{\text{val}}(po^{\text{par}})$ eingehen **do**
    $status(po, po^{\text{par}}) \stackrel{\text{def}}{=} (leer, \ldots)$;
  end for
  {Bestimme Anwendbarkeit für „verbleibende“ Kanäle:}
15: **for all** $po \in f^{\text{in}}(po^{\text{par}})$ mit $status(po) \notin \{(leer, \ldots), (invalide, \ldots)\}$ **do**
    Weise Kanälen Anwendbarkeiten gemäß der Bedingungen aus Tab. 5.3 zu;
  end for
  else if $f^{\text{in}}(po^{\text{par}}) = \emptyset$ **then** {Port „unabhängig“, nur manuell definiert.}
    Weise po^{par} selbst seinem Wert entsprechende Anwendbarkeit zu;
20: **end if**
  {Für automatisch berechnete (Ausgangs–)Parameterports ist nichts zu tun.}
end procedure

Algorithmus 5.7: (Anwendbarkeits–)Prüfung in Parameterports

In diesem Fall ist hier die Propagierung auf der Basis des vorläufigen Wertes zu starten, um bei Benutzerinteraktionen innerhalb des Verbundes nicht die gesamte Schleife neu zu initialisieren (was beim Rückgriff auf den Verbundparameter geschehen würde). Bewußt wird die Propagierung erst beim Fan–out und nicht im Schleifenmodul selbst gestartet, um die Vorläufigkeit beizubehalten.

Algorithmus 5.9 zeigt — wiederum analog zu den Datenports — die Propagierung in Parameterports, deren Modul kein Abbruchmodul ist.[60] Man beachte, daß der Port bei der Wertepropagierung auf jeden Fall den Datenstatus *ok* erhält (Zeile 15, 27, 30, 32).

Im Gegensatz zur Behandlung von Dateneingängen darf sich bei der Propagierung der Werte eines Parameterports die zugehörige Anwendbarkeitsprüfung nicht nur auf den jeweiligen Parameterport po^{par} selbst beschränken. Vielmehr ist die Prüfprozedur zusätzlich auch für alle Dateneingänge und andere Parameterports des gleichen Moduls *mod* aufzurufen, in deren Anwendbarkeitsbetrachtung der Wert von po^{par} eingeht. Mit der Neubestimmung von po^{par} kann sich das Anwendbarkeitsurteil für diese Ports nämlich ändern, *ohne* daß ihre Belegung selbst im Rahmen der aktuellen Verarbeitungstransaktion neu bestimmt wird. Liegen sie nicht im Fan–out von po^{par}, erfolgt über sie u. U. keine Propagierung und somit auch keine damit zusammenhängende Anwendbarkeitsprüfung. Um nun aber Berechnungen im internen Fan–out von po^{par}, die der Propagierung von po^{par} nachfolgen, eine korrekte Einstufung ihrer Anwendbarkeit auf all ihre Quelldaten (auch aus nicht neu berechneten Ports mit aber neuer Anwendbarkeitsbeurteilung) zu erlauben, ist die Schleife zur Prüfung von po^{par} abhängiger Ports in Zeile 3–4 nötig (vgl. auch die Diskussion zu Tab. 5.3 hinsichtlich azyklischer Abhängigkeiten von Anwendbarkeitsprüfungen). Durch die Überlappungen von portbezogenen Anwendbarkeitsprüfungen können evtl. redundante Mehrfachaufrufe von Prüfungen für ein und denselben Port im Rahmen der Bearbeitung einer Benutzerinteraktion auftreten. Dies ließe sich leicht durch Einführung eines Flags, das kontrolliert, ob eine Prüfung auf bestimmte Eingangsdaten schon durchgeführt wurde, bzw. durch eine Erwei-

[60] Aufgrund der komplexen Aufgabe des Abbruchmoduls wird dessen Verarbeitung gesondert weiter unten (in Alg. 5.13) vorgestellt.

```
    procedure Fordere( po^par: PO^par, umfang: Umfang ) :
      {Zunächst Verbund–Anforderungen an Abbruchmodule — Verbund–Initialisierung:}
      if mod ∈ MOD^stop ∧ umfang = verbund then
        Initialisiere( po^par, invalide ); Initialisiere( po^par, daten );
 5:     {Ansonsten Abbruch, wenn Port bereits bearbeitet:}
      else if status(po^par) = (..., offen, ...) then
        {Verbund–Anforderungen an vorläufige Parameterports ergeben Verbund–Bearbeitung:}
        if umfang = verbund ∧ mod ∈ MOD^iopar ∧ status(po^par) = (..., vorläufig, ...) then
          Fordere( mod.mod^sub.mod^stop.po^par, verbund );
10:     end if
        {Zwei Fälle, die weiterhin eine Aktion/Statusänderung erforderlich machen:}
        if status(po^par) = (..., vorläufig, ...) ∨ status(po^par) = (invalide, ...) then
          {Bei Parameterports werden aktiv und passiv nicht unterschieden:}
          status(po^par) ≝ (..., aktiv, ...);
15:       {Propagierung in Schleifenmodulen, die nur aufgrund lokaler Interaktionen vorläufig:}
          if mod ∈ MOD^loop ∧ status(po^par) = (..., vorläufig, ...) ≠ status(mod.po^sub) then
            for all po ∈ f^out(po^par) do Propagiere( po, daten ) end for;
          else {Übrige vorläufige oder invalide Ports.}
            {Vorläufige Daten werden vor Aktualisierung invalidiert:}
20:         if status(po^par) = (..., vorläufig, ...) then Propagiere( po^par, invalide ) end if;
            if f^in(po^par) = ∅ then {Propagierung bei nur manuell belegten Ports beginnen.}
              Propagiere( po^par, daten );
            else {Weiterleitung der Anforderung an den gesamten Fan–in.}
              for all po ∈ f^in(po^par) do
25:             {Spezielle Anforderung von Verbundports an Parameterschnittstellenmodule:}
                if mod ∈ mod^sub ∧ po.mod ∈ MOD^iopar then
                  Fordere( po, verbund );
                else {Normalfall: Umfang nur von po^par abhängig.}
                  Fordere( po, if po^par automatisch ∧ po–datenbas. then extension else schema );
30:             end if
              end for
            end if
          end if
        else {Anforderung erfüllt, da Port bereits geeignet belegt — Propagierung beginnen.}
35:       Propagiere( po^par, daten );
        end if
      end if
    end procedure
```

Algorithmus 5.8: (Ergebnis–)Anforderung an Parameterports

terung des Anwendbarkeitsstatus um eine Ausprägung *ungeprüft* verhindern.

Wenn im Rahmen der Propagierung von Berechnung oder Belegung der Parameterwerte gesprochen wird (Zeile 29), soll hierunter ggf. auch die unveränderte Übernahme eines Wertes aus dem (dann einelementigen) Fan–in verstanden werden, z. B. in Ausgangsports von Verbundmodulen. In Schleifenmodulen bedeutet Berechnung gerade die Auswahl des nächsten Elements aus der Liste der zu durchlaufenden Parameterwerte, wobei eine hierauf definierte Ordnung ausgenutzt wird. Gibt es kein nächstes Element, bleibt der Wert unverändert. Ein derartiger Fall tritt gemäß der Konzeption von Schleifen in *VIOLA* lediglich für den *next*–Parameter eines Schleifenmoduls auf.

procedure Propagiere(po^{par}: $\mathcal{PO}^{\text{par}}$, *modus*: *Modus*) :
  if $modus = daten$ **then** {Anwendbarkeit in po^{par} und anderen betroffenen Ports prüfen.}
    for all $po \in mod.PO^{\text{in}} \cup mod.PO^{\text{par}}$, deren Anwendbarkeit von po^{par} abhängt **do**
      Prüfe(*po*);
5: **end for**
  end if
  if $modus \neq vorläufig \wedge mod = mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}} \wedge po^{\text{par}}.in?$ **then**
    {Ein Parameter(eingangs)port eines Verbundmoduls initialisiert ggf. das Abbruchmodul.}
    for all $i = 1, \ldots, n$ **do** Initialisiere($mod^{\text{sub}}.mod_i^{\text{stop}}.po^{\text{par}}$, *modus*) **end for**;
10: **end if**
  if $modus \in \{invalide, vorläufig\}$ **then**
    ... {wie bei Datenausgängen (Beginn von Alg. 5.5); jedoch eine Ausnahme: Ports po^{par} von Parameterschnittstellenmodulen $mod \in \mathcal{MOD}^{\text{iopar}}$ mit $po^{\text{par}}.in? = true$ leiten (wie Ausgangsmodule) statt Propagiere(*invalide*) stets Propagiere(*vorläufig*) an ihren Fan–out.}
  else {Falls nun ein neuer Wert nicht berechnet werden kann, wird der alte beibehalten.}
    if $po^{\text{par}}.in?$ **then** {Importierte Werte (in Zeile 4 bei Prüfung übernommen) weiterreichen.}
15:   $status(po^{\text{par}}) \stackrel{\text{def}}{=}$ **if** $\exists po \in f^{\text{in}}(po^{\text{par}}) : status(po) = (\ldots, vorläufig, \ldots)$ **then**
                $(ok, \ldots, vorläufig, \ldots)$ **else** $(ok, \ldots, aktuell, \ldots)$;
      if $mod \in \mathcal{MOD}^{\text{iopar}}$ **then** {Parameter(ausgangs)module propagieren nur Vorläufigkeit.}
        for all $po \in f^{\text{out}}(po^{\text{par}})$ **do** Propagiere($po, vorläufig$) **end for**;
      else {Normale Weiterpropagierung, jedoch nicht an Schleifenmodule.}
        for all $po \in f^{\text{out}}(po^{\text{par}})$ mit $po.mod \notin \mathcal{MOD}^{\text{loop}}$ **do** Propagiere($po, daten$) **end for**;
20:   **end if**
      {Eine Berechnung erfolgt nur, wenn nötig:}
    else if $status(po^{\text{par}}) \notin \{(ok, \ldots, aktuell, \ldots), (ok, offen, vorläufig, \ldots)\}$ **then**
      {Zunächst auf geforderte Daten warten:}
      wait until $\forall po \in f^{\text{in}}(po^{\text{par}}) : status(po) \notin \{(invalide, \ldots), (schema, aktiv, \ldots)\}$;
25:   {Berechnung bei Anwendbarkeit und Vorliegen nötiger Angaben:}
      if $status(po^{\text{par}}) = (\ldots, abgelehnt)$ **then** {Wert bleibt unverändert.}
        Übernimm (wie in Zeile 15) Aktualität vom Fan–in und setze Datenstatus auf *ok*;
      else if ($mod.f^{\text{par}}$ metadatenbasiert) $\vee$
            ($status(po^{\text{par}}) = (\ldots, aktiv, \ldots) \wedge$ nötige Daten in $f^{\text{in}}(po^{\text{par}})$ sind *ok*) **then**
        Belege $val(po^{\text{par}})$ neu;
30:     Übernimm (wie in Zeile 15) Aktualität vom Fan–in und setze Datenstatus auf *ok*;
      else {Alten Wert vorläufig beibehalten, da keine neue Berechnung möglich.}
        $status(po^{\text{par}}) \stackrel{\text{def}}{=} (ok, \ldots, vorläufig, \ldots)$;
      end if
      {Propagierung berechneter Werte:}
35:   **for all** $po \in f^{\text{out}}(po^{\text{par}})$ **do** Propagiere($po, daten$) **end for**;
    end if
  end if
end procedure

Algorithmus 5.9: Propagierung über Parameterports

Optimieren ließe sich die Parameterverarbeitung noch geringfügig, indem, falls neu bestimmte Parameterwerte gegenüber ihrer letzten Belegung unverändert bleiben, auf diesen aufbauende Berechnungen in nachfolgenden Modulen nicht neu durchgeführt, sondern nur die jeweiligen Statusinformationen angepaßt würden. Dieser Fall tritt insbesondere dann auf, wenn automatische Berechnungen „fehlschlagen“ und deshalb vorläufig

der bestehende Wert weiterverwendet wird. Natürlich können auch „zufällig“ neu durchgeführte Berechnungen zum gleichen Wert wie bisher führen, aber diese Fälle abzufangen, würde — ebenso wie bei der Verarbeitung von Datenräumen — den erhöhten Verwaltungsaufwand kaum lohnen.

Die manuelle Änderung von Parameterwerten erfordert ggf. eine Trennung des betrachteten Ports von eingehenden Kanälen (siehe Alg. 5.10, Zeile 16). Außerdem ist sie gegenüber automatischen Berechnungen nur vorläufig (Zeile 14). Aufgrund dieser speziellen Behandlung beginnt die Propagierung der lokalen Wertedefinition erst mit dem Fan–in und nicht mit dem Port selbst (Zeile 24). Weiterhin werden auch hier wiederum abhängige Anwendbarkeitsprüfungen in anderen Ports des gleichen Moduls berücksichtigt (Zeile 9–10).

procedure Definiere(po^{par}: $\mathcal{PO}^{\text{par}}$) :
  {Der neue Parameterwert wurde bereits gesetzt.}
  {Zunächst Nachfolger invalidieren:}
  Propagiere(po^{par}, *invalide*);
5: {Dateneingabe ist stets ok und wird übernommen:}
  $status(po^{\text{par}}) \stackrel{\text{def}}{=} (ok, \ldots)$;
  {Anwendbarkeit lokal und für abhängige Ports definieren:}
  Weise po^{par} Anwendbarkeit gemäß Tab. 5.3 zu;
  for all $po^{\text{par}} \neq po \in mod.PO^{\text{in}} \cup mod.PO^{\text{par}}$, deren Anwendbarkeit von po^{par} abhängt **do**
10: Prüfe(po);
  end for
  {Manuelle Änderung ist gegenüber automatischer Berechnung jedoch nur vorläufig:}
  if po^{par} automatisch **then**
    $status(po^{\text{par}}) \stackrel{\text{def}}{=} (\ldots, vorläufig, \ldots)$;
15: **else** {Eingehende Parameterkanäle werden jetzt als leer angesehen.}
    for all $po \in f^{\text{in}}(po^{\text{par}})$ **do** $status(po, po^{\text{par}}) \stackrel{\text{def}}{=} (leer, \ldots)$ **end for**;
  end if
  {Innerhalb eines Verbundes werden Eingaben zwischen Analysegraph–Instanzen ausgetauscht:}
  if $\exists mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}} : mod \in mod^{\text{sub}}.ag.MOD \wedge po^{\text{par}} = po_i^{\text{par}}$ **then**
20: **for all** $j = 1, \ldots, n$ mit $j \neq i$ **do** Definiere(po_j^{par}) **end for**;
  end if
  {Jetzt Daten propagieren:}
  for all $po \in f^{\text{out}}(po^{\text{par}})$ **do** Propagiere($po, daten$) **end for**;
end procedure

Algorithmus 5.10: (Manuelle) Änderung in Parameterports

Bereits mehrfach wurde in den obenstehenden Algorithmen die Initialisierung von Abbruchmodulen $mod^{\text{stop}} \in \mathcal{MOD}^{\text{stop}}$ aufgerufen, die in Alg. 5.11 nun konkretisiert wird. Hierbei wird über eine Status–Funktion $iteriere : \mathcal{MOD}^{\text{stop}} \rightarrow \mathbb{B}$ vermerkt, daß der Verbund (mit allen Iterationen) vollständig auszuführen ist. Dieses Flag wird erst beim Abschluß des Verbundes bzw. der Schleife wieder zurückgesetzt. Da nach lokalen Benutzerinteraktionen innerhalb des Verbundmodul–Analysegraphen das Abbruchmodul nicht initialisiert wird, ist in diesen Fällen das Flag nicht gesetzt, so daß nicht iteriert wird und keine Ergebnisse aus dem Verbund herausgeleitet werden.

Während einer Iteration (also mit $iteriere(mod^{\text{stop}}) = true$) soll sich die Aktualität des Abbruchparameters stets daraus ergeben, ob dieser in *irgendeinem* Durchlauf seit der Initialisierung der Schleife nur vorläufig bestimmt werden konnte. Die Verwaltung der entsprechenden historischen Information leistet die Status–Funktion $vorlaeufig : \mathcal{MOD}^{\text{stop}} \rightarrow \mathbb{B}$. Da dieser Status im Verlauf der vollständigen Verbundmodul–Ausführung nicht wieder von *vorläufig* auf *aktuell* zurückgesetzt wird, ist er bei der Initialisierung des Abbruchmoduls jeweils neu gemäß der Aktualität des Abbruchparameters zu belegen.

Im folgenden sei $mod^{\text{sub}} \in \mathcal{MOD}^{\text{sub}}$ jeweils das Verbundmodul, in dessen Analysegraphen die betrachteten Schleifen- und Abbruchmodule liegen. Die Menge aller Ports von Schleifenmodulen im Analysegraphen von mod^{sub} sei $PO^{\text{loop}} \stackrel{\text{def}}{=} \{po^{\text{par}} \in mod^{\text{loop}}.PO^{\text{par}} \mid mod^{\text{loop}} \in mod^{\text{sub}}.ag \cap \mathcal{MOD}^{\text{loop}}\}$. Sofern relevant, sei i der Index der jeweiligen Instanz des Analysegraphen.

```
    procedure Initialisiere( po^par: PO^par, modus: Modus ) :
      {Mit der Invalidierung erfolgt die eigentliche Initialisierung:}
      if modus = invalide then
        if iteriere(mod) = false then {Schleife bzw. Verbund ist noch nicht initialisiert.}
 5:       {Aktualität bestimmen, Iterierungsflag setzen und Schleifenmodule initialisieren:}
          vorlaeufig(mod) ≝ (status(po^par) = (..., vorläufig, ...));
          iteriere(mod) ≝ true;
          for all po ∈ PO^loop do Initialisiere( po, invalide ) end for;
          {Bestimmung des Abbruchparameters sichern:}
10:       Fordere ( po^par, extension );
          {Invalidierung an Ausgänge des Verbundmoduls leiten:}
          if mod^sub.ag.MOD ∩ MOD^out ≠ ∅ then Propagiere( mod^sub.po_i^out, invalide ) end if;
          for all mod^iopar ∈ mod^sub.ag.MOD ∩ MOD^iopar mit mod^iopar.po^par.in? = true do
            Propagiere( mod^iopar.po^sub, invalide );
15:       end for
        end if
      else {modus = daten}
        {Abbruchmodul selbst propagieren, um das Schleifenende zu sichern:}
        Propagiere( po^par, daten );
20:     {Und die Schleifenmodule anstoßen:}
        for all po ∈ PO^loop do Propagiere( po, daten ) end for;
      end if
    end procedure
```

Algorithmus 5.11: Initialisierung von Abbruchmodulen

Der Aufruf der Initialisierung eines Abbruchmoduls gemäß Alg. 5.11 wird jeweils aus dem Umfeld der Schnittstellen eines Verbundmoduls vorgenommen — entweder im Rahmen einer Anforderung an den Verbund als Ganzes (siehe Alg. 5.8, Zeile 4) oder bei der Propagierung in einen Verbund hinein (Alg. 5.3, Zeile 6, und Alg. 5.9, Zeile 9). In jedem Fall geschieht dies erst als Invalidierung, anschließend als Daten–Initialisierung. Der Wert *vorläufig* als übergebener *modus* ist nicht vorgesehen.

Neben Abbruchmodulen sind auch Schleifenmodule zu initialisieren (siehe Alg. 5.12). Dies erfolgt stets aus der Initialisierung des Abbruchmoduls heraus und besteht im wesentlichen im Setzen der Schleifenparameters auf die jeweiligen Anfangswerte. Andere Ports haben keine Initialisierungsroutine.

Es wird bei der Initialisierung von Abbruchmodulen nicht unterschieden, ob es sich jeweils tatsächlich um eine Schleife (ein Verbundmodul mit Schleifenmodulen) oder einen „normalen“ Verbund handelt. Dies spielt lediglich bei der Prüfung der Abbruchbedingung im Rahmen der Propagierung von Abbruchmodulen (s. u.) eine Rolle. Damit diese bei der vollständigen Verbundausführung stets zum Zuge kommt, fordert der Port des Abbruchmoduls in Alg. 5.11 explizit die Bestimmung des Abbruchflags (Zeile 10) und setzt sich nach der Durchführung der Initialisierung selbst auf die Propagierungsliste (Zeile 19).

Die Initialisierung eines Abbruchmoduls erfolgt immer zusätzlich zu einer Propagierung von Daten und Parameterwerten über die Verbundeingänge oder deren Anforderung an den Verbundausgängen. Von diesen ausgehend wird jeweils spezifiziert, welche Teilmodule konkret zu verarbeiten sind, während sich das Abbruchmodul lediglich um Start und Ende der Bearbeitung des gesamten Verbundes kümmert.

Die Umsetzung der Wait–Anweisung bei der Schleifenmodul–Initialisierung (Zeile 6), also deren Aufschub

```
    procedure Initialisiere( po^par: PO^par, modus: Modus ) :
      if modus = invalide then {Nur Nachfolger invalidieren.}
        Propagiere( po^par, invalide );
      else {modus = daten}
 5:     {Nach Vorliegen valider Eingangsdaten Werte initialisieren:}
        wait until status(mod.po^sub) ≠ (invalide, ...);
        {Anfangswerte belegen — falls nicht möglich, alte Werte beibehalten:}
        if status(po^par) ≠ (..., abgelehnt) then
          Setze val(po^par) auf den ersten Wert von val(mod.po^sub);
10:       if po^par.par = next then falls möglich, setze val(po^par) auf den nächsten Wert end if;
        end if
        {Status wie gehabt setzen und weiterpropagieren:}
        status(po^par) ≝ if status(mod.po^sub) = (..., vorläufig, ...) then
                           (ok, ..., vorläufig, ...) else (ok, ..., aktuell, ...);
        for all po ∈ f^out(po^par) do Propagiere( po, daten ) end for;
15:   end if
    end procedure
```

Algorithmus 5.12: Initialisierung von Schleifenmodulen

in eine abgetrennte letzte Verarbeitungsphase, erfolgt analog zum Propagierungsalgorithmus. Sind Schleifenmodule einmal nicht anwendbar, ist also die zugeordnete Werteliste leer, wird dies wie auch in anderen nicht anwendbaren Parameterports gehandhabt: Der bestehende Parameterwert wird einfach beibehalten; entsprechend endet die Schleife nach einem Durchlauf.

Wie bereits erwähnt, unterscheidet sich die Propagierung des Parameterwertes eines Abbruchmoduls $mod = mod^{\text{stop}} \in \mathcal{MOD}^{\text{stop}}$ grundsätzlich von der Implementierung anderer Module. Vor allem ist nach einem Durchlauf durch den Verbundmodul–Analysegraphen der Schleifenabbruch und die Werte–Propagierung der Schleifenergebnisse bzw. die Durchführung des nächsten Schleifendurchlaufs zu organisieren. Algorithmus 5.13 gibt hierzu die Details.

Das Abbruchmodul führt zwei Aktualitätsstati: zum einen die modulbezogene Aktualität, wie sie auch in allen anderen Modulen genutzt wird, zum anderen aber auch mit der Funktion *vorlaeufig*() die verbundbezogene Aktualität, die während der Iteration nicht wieder zurückgesetzt wird. Falls Berechnungen nur verbundlokal, ohne Iteration erfolgen, entspricht der Wert von *vorlaeufig*() dem üblichen Aktualitätsstatus (Zeile 9–14).

Die Behandlung der Wait–Anweisung in Zeile 16 ist für Abbruchmodule insofern getrennt von der Warteschlange für andere Propagierungsanweisungen zu realisieren, als sie erst bearbeitet werden darf, wenn sich kein Port aus dem jeweiligen Analysegraphen mehr in der Queue befindet. Auch muß natürlich der vor der Wait–Anweisung liegende Teil des Propagiere–Algorithmus ausgeführt werden, *bevor* ein Eintrag des übrigen Ablaufs in die Warteschlange erfolgt. Andernfalls wäre die dort gegebene Bedingung unter Umständen nicht erfüllbar: Erst wenn auch (transitiv) Nachfolger des Abbruchmodul–Parameterports innerhalb des Verbundmoduls bearbeitet sind (Zeile 6), kann die Verbundverarbeitung abgeschlossen werden.

Bei der Weiterleitung berechneter Parameterwerte aus den Parameterschnittstellenmodulen an die Verbundmodulports fällt auf, daß keine spezielle Instanz des Analysegraphen spezifiziert ist (Zeile 27). Wie bereits in Abschnitt 5.4.1 dargestellt, dürfen derartige Modulparameter nur dann auftreten, wenn das Verbundmodul lediglich einen Dateneingang und somit auch nur eine Instanz des Analysegraphen besitzt. Somit ist der Parameterwert in jedem Fall eindeutig.

Schließlich sei noch die Korrektheit der letzten Zeilen des Algorithmus 5.13 (Zeile 36–38) motiviert: Wenn die Propagierung von Werten innerhalb eines Verbundes bis zum Abbruchmodul führt, ergibt sich hier eine neue Belegung des Abbruchparameters. Da das *iteriere*()–Flag jedoch (im Else–Zweig der If–Anweisung in Zeile 36) nicht gesetzt ist (d. h. die Berechnung soll lokal im Verbundmodul, ohne Iteration und ohne Propagierung

```
    procedure Propagiere( po^par: PO^par, modus: Modus ) :
      if modus ∈ {invalide, vorläufig} then
        ... {Propagierung der Stati wie bei Datenausgängen.}
      else {Wertepropagierung; entspricht Fall po^par.in? = true bei anderen Parameterports.}
 5:     Prüfe( po^par );
        for all po ∈ f^out(po^par) do Propagiere( po, daten ) end for;
        status(po^par) ≝ (ok, ...);
        {Vorläufigkeit für Parameter und Historie setzen:}
        if ∃po ∈ f^in(po^par) : status(po) = (..., vorläufig, ...) then
10:       status(po^par) ≝ (..., vorläufig, ...); vorlaeufig(mod^stop) ≝ true;
        else
          status(po^par) ≝ (..., aktuell, ...);
          if ¬iteriere(mod^stop) then vorlaeufig(mod^stop) ≝ false end if;
        end if
15:     {Vollständige Abarbeitung des Verbundmoduls (des Schleifenrumpfes) abwarten:}
        wait until ∀mod ∈ mod^sub.ag.MOD ∀po ∈ mod.PO^in ∪ mod.PO^out ∪ mod.PO^par :
                   status(po) ∉ {(invalide, ...), (schema, aktiv, ...)};
        if iteriere(mod^stop) then {Schleife bzw. Verbund vollständig zu durchlaufen.}
          {Test auf Schleifenende (Abbruch auch, wenn Verbund keine Schleife ist):}
          if mod^stop.stop? ∨ ¬is_loop(mod^sub) ∨
            (∃mod^loop ∈ mod^sub.ag.MOD ∩ MOD^loop : finished(mod^loop)) then
20:         iteriere(mod^stop) ≝ false;
            {Falls Schleifenabbruchkriterium vorläufig, Schleifeninhalte als vorläufig markieren:}
            if is_loop(mod^sub) ∧ vorlaeufig(mod^stop) then
              for all po ∈ PO^loop do Propagiere( po, vorläufig ) end for;
            end if
25:         {Propagierung an Daten- und Parameterausgänge:}
            if mod^sub.ag.MOD ∩ MOD^out ≠ ∅ then Propagiere( mod^sub.po_i^out, daten ) end if;
            for all mod^iopar ∈ mod^sub.ag.MOD ∩ MOD^iopar mit mod^iopar.po^par.in? = true do
              Propagiere( mod^iopar.po^sub, daten );
            end for
30:       else {Nächste Schleifeniteration.}
            for all po ∈ PO^loop do Propagiere( po, invalide ) end for;
            {Zur Behandlung des Schleifenendes:}
            Propagiere( mod^stop, daten );
            for all po ∈ PO^loop do Propagiere( po, daten ) end for;
35:       end if
        else {Iteration vorläufig unterbrechen, Schleifeninhalte sind vorläufig.}
          for all po ∈ PO^loop do Propagiere( po, vorläufig ) end for;
        end if
      end if
40: end procedure
```

Algorithmus 5.13: Propagierung über Parameterports von Abbruchmodulen

von Ergebnissen aus dem Verbund heraus erfolgen), kann die Auswirkung dieser neuen Belegung auf die Schleifenverarbeitung nicht evaluiert werden — also sind alle von den Schleifenmodulen abhängigen Verbund–Inhalte vorläufig.

5.4.5 Konstruktion und Verarbeitung von Analysegraphen

Mit der Definition grundlegender Verarbeitungsprimitive im vorigen Abschnitt kann nun leicht die Nutzung von *VIOLA*, d. h. die Erstellung visueller Programme sowie deren Abarbeitung über eine Reihe möglicher Benutzerinteraktionen skizziert werden. Anschließend gehen wir noch auf einige Aspekte der Benutzungsoberfläche von *VIOLA* ein, die dem Anwender einen umfassenden Überblick über Eigenschaften des von ihm erstellten Programms sowie der damit verarbeiteten Daten gewährt.

Um dem Datenanalysten eine Trennung bzw. genauer gesagt eine Verlagerung des Schwerpunkts zwischen Entwurfs-, Implementierungs- und Anwendungsphase eines *VIOLA*–Programms zu erlauben, wird dem Programm ein Verarbeitungsstatus *prg_status*() zugeordnet. Dieser definiert, welchen Propagierungsstatus Ports direkt nach Erzeugung ihres Moduls sowie nach vollständiger Abarbeitung einer Benutzerinteraktion erhalten sollen. Es werden folgende Fälle unterschieden:

1. Der Status aller Ports ist offen; somit werden über die direkte Abarbeitung einer Benutzeranforderung hinaus nur Metadaten propagiert.
2. Der Status aller Ports richtet sich danach, ob sie als Grundlage einer beliebigen Datensenke dienen. Dies entspricht der gleichzeitigen und ständigen Erfüllung von Anforderungen („*Fordere(extension)*“) aus allen Ports von Datensenken (auch solchen innerhalb von Verbundmodulen). Nach jeder Benutzerinteraktion werden also die Inhalte aller betroffenen Datensenken vollständig aktualisiert. Eine entsprechende Festlegung der Relevanz von Ports kann entweder in laufender Aktualisierung während der Programmierung oder lediglich bei Bedarf erfolgen.
3. Alle Ports sind aktiv; somit werden auch jeweils nicht „benötigte“ Zwischenergebnisse vollständig definiert.

Schließlich soll über *prg_status*() auch die Propagierung von Metadaten zeitweise völlig abzuschalten sein, d. h. die Abarbeitung der Warteschlange aller *Propagiere(daten)*–Aufrufe aus der Abarbeitung mehrerer Benutzerinteraktionen wird bis zur expliziten Auslösung durch den Anwender aufgeschoben. Eine Invalidierung sowie eine Kennzeichnung vorläufiger Resultate muß auch in diesem Zustand bei Bedarf in jedem Fall sofort vorgenommen werden.

Visuelle Programmierung — die Erstellung von Analysegraphen

Der während einer Programmkonstruktion vorliegende Analysegraph soll zu jedem Zeitpunkt gemäß Def. 5.24 und 5.25 gültig und zyklenfrei sei, um eine sofortige Interpretation des Programms zu ermöglichen. Auch Verbundmodule sollen stets Def. 5.27 entsprechen. Folgende Schritte der Programm–Erzeugung lassen sich von der interaktiven Programm–Bedienung, die im nächsten Abschnitt näher betrachtet wird, abgrenzen:

Erzeugen von Modulen Die Statusinformationen der Ports werden jeweils gemäß *prg_status*() und Abb. 5.9 bzw. 5.10 initialisiert. Der Datenstatus ist *invalide*, der Port *aktuell* und die Anwendbarkeit *abgelehnt*. Parameter werden mit vordefinierten Default–Werten belegt, Dateninhalte sind undefiniert. In Visualisierungsmodulen kann evtl. die Menge anzubietender, durch Interaktion mit der Graphik belegter Parameter ausgewählt werden. Anschließend wird für alle Parameterports und Datenausgänge gemäß ihrem Propagierungsstatus *Fordere(extension)* oder *Fordere(schema)* aufgerufen.

Verbinden von Ports über Kanäle Hierbei ist zu beachten, daß ein gültiger Analysegraph resultiert, also Datenein- und -ausgänge oder Parameterports untereinander (wobei der Zielport $in? = true$ erfüllt) verbunden werden, keine Zyklen entstehen (also auch keine Ports desselben Moduls einander zugeordnet werden) und ggf. die Kompatibilität von Parametern berücksichtigt wird. Andernfalls ist die Verbindung abzulehnen. Für den jeweiligen Zielport wird nach erfolgreicher Zuordnung zunächst *invalide* propagiert und anschließend wiederum der *Fordere*–Algorithmus mit einem dem Datenstatus des Ports entsprechenden Argument angestoßen.

Entfernen von Kanälen Nach Löschen eines Datenkanals werden analog zur Kanalerzeugung zunächst die Nachfolger des (ehemaligen) Zielports invalidiert, und anschließend wird für diesen *Propagiere(daten)* initiiert. Da durch das Entfernen eines Parameterkanals der Wert im Zielport normalerweise unverändert bleibt, braucht hier gar keine Aktion zu erfolgen. Lediglich, wenn der gelöschte Kanal einer von mehreren Eingängen des Zielports war und einen Beitrag zur Parameterdefinition geleistet hat, sind für diesen Port erst eine Invalidierung und dann ein *Fordere(extension)*–Aufruf erforderlich.

Entfernen von Modulen Vor dem Löschen eines Moduls werden alle anliegenden Kanäle entfernt, so daß anschließend keine besondere Aktion mehr auszuführen ist.

Kapseln eines Teilgraphen in einem Verbundmodul Die Statusinformationen von Verbundmodulen werden analog zur Erzeugung anderer Module initialisiert, eine Wertepropagierung bzw. Anforderung erfolgt jedoch zunächst nicht. Aus den bisherigen Verbindungen des ausgewählten Teilgraphen zu nicht einbezogenen Modulen können (halb-)automatisch Ein- und Ausgangsmodule, Parameterschnittstellenmodule und entsprechende externe sowie auch interne Modulparameter hinzugefügt werden — auf Details wollen wir an dieser Stelle nicht eingehen. An vielen Stellen, etwa bei der Zusammenfassung interner Parameter mittels der Funktion f^{pp} oder der Parametrisierung von Eingangsmodulen, dürfte eine explizite Benutzerinteraktion unabdingbar sein. Automatisch wird in jedem Fall ein Abbruchmodul hinzugefügt, das zusätzlich zur üblichen Statusinitialisierung im *iteriere*()–Flag den Wert *false* erhält.

Hinzufügen und Löschen von Modulen in Verbunden Egal, ob automatisch bei der initialen Erzeugung eines Verbundmoduls oder ob nachträglich, das Erzeugen (und Löschen) eines Moduls innerhalb eines Verbundes erfolgt prinzipiell genau wie in ungekapselten Analysegraphen. Zusätzlich wird jedoch stets auf die Einhaltung aller Konsistenzbedingungen aus Def. 5.27 geachtet, wodurch auch Aktionen abgelehnt werden können. Gegebenenfalls werden Ein- oder Ausgangsports oder Verbundparameter hinzugefügt oder gelöscht bzw. die Kardinalität von Dateneingängen geändert. Hierbei werden, falls nötig, ein- oder ausgehende Kanäle gleichermaßen entfernt; bei der „Verkleinerung“ von Eingangsports sollte der Anwender ggf. einen zu löschenden Kanal auswählen können. Nach der Definition eines Eingangsmoduls ist auf dem Verbundeingang jeweils einmal die *Prüfe*–Routine auszuführen, um dem Modul den zugehörigen Datenkanal zuzuordnen.

Abschließen eines Verbundmoduls Erzeugung und lokale Modifikation eines Verbundmoduls erfolgen jeweils (mit *iteriere*() = *false*) ohne Weiterleitung von Daten und Parameterwerten aus dem Verbundmodul heraus.[61] Mit dem „Schließen“ eines Verbundmoduls wird dies über eine Anforderung aus den Daten- und Parameterausgängen analog zur allgemeinen Modulerzeugung nachgeholt.

Hinzufügen und Löschen von Datenport–Paaren Zu allen Modulen, in denen Datenein- und ausgänge einander paarweise zugeordnet sind, können in beliebiger Anzahl Portpaare hinzugefügt und — nach Entfernung aller Kanäle — auch wieder gelöscht werden. Vorzunehmende Aktionen ergeben sich wiederum wie bei der Modulerzeugung (bzw. -löschung).

Interaktive Nutzung von Analyseprogrammen

Nachdem ein *VIOLA*–Programm mit oben beschriebenen Mitteln erstellt wurde, kann der Anwender — in Kombination mit den vorgestellten Verarbeitungsmodi — in sehr flexibler Weise mit diesem interagieren, Auswertungen parametrisieren und ausschnittsweise oder vollständig durchführen. Natürlich können sich diese Interaktionen wiederum mit weiteren Modifikationen an der Programmstruktur abwechseln. Im einzelnen werden folgende Möglichkeiten geboten, wobei jeweils die zugehörige Realisierung kurz umrissen wird:

Manipulation externer Parameter Hierbei erfolgt lediglich ein *Definiere*–Aufruf für den jeweiligen Parameterport.

[61] Man beachte, daß demgegenüber jedoch bereits durch Eingangs- und Parameterschnittstellenmodule auf die jeweiligen Quelldaten zugegriffen wird.

Manipulation interner Parameter Die Änderung interner Parameter entspricht der gleichzeitigen Änderung aller abhängigen Ports des gleichen Moduls, d. h. all diese werden zunächst mittels *Propagiere(invalide)* invalidiert und anschließend über einen *Propagiere(daten)*–Aufruf zur Datenweitergabe aufgefordert. Unter Gleichzeitigkeit soll hier verstanden werden, daß mit der Abarbeitung der Warteschlange von Propagierungsaufrufen bis zum Eintrag des letzten betroffenen Ports gewartet wird.[62] *Vor* der Propagierung muß jedoch (ähnlich wie in Alg. 5.9) eine Prüfung für alle Ports erfolgen, deren Anwendbarkeit vom manipulierten internen Parameter abhängt. Innerhalb von Verbundmodulen erfolgt, wie auch schon im Fall der externen Parameter, eine synchrone Parameterdefinition in allen Instanzen des Analysegraphen. Wird ein interner Parameter eines Verbundmoduls selbst manuell geändert, ist diese Manipulation zudem an alle gemäß der Funktion f^{pp} abhängigen Parameter von Modulen innerhalb des gekapselten Analysegraphen weiterzuleiten und wiederum gleichzeitig zu verarbeiten.

Anfordern der Belegung eines Ports Mittels *Fordere(extension)* kann explizit die Bestimmung des Inhalts eines Ports initiiert werden. Falls dieser schon aktuell ist und den Datenstatus *ok* besitzt, ist keine Aktion nötig. Nach Abarbeitung der Anforderung werden die Propagierungsstati aller betroffenen Ports unter Berücksichtigung des Verarbeitungsstatus des Programms wieder zurückgesetzt.

Explizite Propagierung eines Portinhalts Neben der im letzten Punkt gegebenen anforderungsgetriebenen wird auch die datengetriebene Verarbeitung unterstützt. Unter Angabe eines Dateneingangs- oder Parameterports können alle Ports im gleichen Modul, die von diesem abhängen, explizit berechnet werden. Hierzu wird vom ausgewählten Port zunächst *invalide* und dann *daten* propagiert, nachdem der Datenstatus seines internen Fan–out auf *aktiv* gesetzt wurde. Die Invalidierung kann entfallen, falls ein betrachteter Zielport bereits aktuelle Werte enthält. „Hinter“ dem betrachteten Modul setzt sich die Propagierung gemäß der bereits zuvor bestehenden Propagierungsstati fort. Eine Alternative wäre, stets alle weiteren, abhängigen Ports zu aktivieren, so daß sich die Datenberechnung durch den gesamten Analysegraphen fortpflanzt. Der Schwerpunkt soll jedoch in *VIOLA* auf der anforderungsgetriebenen, auf Datensenken bezogenen Verarbeitung liegen, die explizite Propagierung dient lediglich als zusätzliche Möglichkeit, einzelne Module in einer schrittweisen Verarbeitung in den Mittelpunkt der Betrachtung zu stellen.

Gerade für Verbundmodule kann diese Aktion sehr nützlich sein, wenn nach lokalen Interaktionen im gekapselten Analysegraphen der Verbund vollständig abgearbeitet werden soll. Anschließend erfolgt wiederum ein Rücksetzen der Propagierungsstati der Zielports.

Ändern des Verarbeitungsstatus Wird der programmbezogene Verarbeitungsstatus so verändert, daß *mehr* Ports „aktiviert“ werden sollen, ist von allen entsprechenden Datensenken bzw. allen Ports mit leerem Fan–out ein *Fordere(extension)*–Aufruf zu starten.

Aufheben einer Propagierungspause Wurden im Verarbeitungsstatus, der auch sämtliche Metadaten–Propagierungen unterbindet, ein oder mehrere Benutzerinteraktionen vorgenommen, ist anschließend die Warteschlange an Propagierungsaufrufen explizit zur Abarbeitung freizugeben.

Neuberechnung des gesamten Programms Um alle lokal veränderten Verbundmodule bei der Interpretation des Gesamtprogramms zu berücksichtigen und alle manuellen Änderungen automatischer Parameterports wieder zu überschreiben, kann auch der gesamte Analysegraph neu berechnet werden. Dies läßt sich am einfachsten realisieren, indem alle Ports mit leerem Fan–in invalidiert werden und von dort aus die Datenpropagierung gestartet wird. Natürlich könnte man auch lediglich von den entsprechenden Ursprüngen vorläufiger Ergebnisse ausgehen — hier wird jedoch die „radikale“ Methode gewählt, um etwa auch veränderte Datenbasen oder andere unvorhergesehene, externe „Störungen“ der vollständigen und bzgl. aller Portzustände korrekten Datenverarbeitung behandeln zu können.

Zusätzlich ist auch der Propagierungsstatus einzelner Ports zur Berücksichtigung bei nachfolgenden Aktionen manuell modifizierbar. Die Nutzung dieser Möglichkeit dürfte jedoch eher auf Einzelfälle beschränkt sein

[62] In diesem Sinne werden auch alle in den nachfolgenden Aufzählungspunkten beschriebenen Interaktionen verarbeitet.

(etwa die Aktivierung zentraler datenbasierter Parameterberechnungen), da die meisten typischen Zielsetzungen bereits durch oben aufgeführte Interaktionen abgedeckt werden.

5.4.6 Ausgestaltung der Benutzungsoberfläche

Neben der flexiblen Unterstützung der Programmerstellung und -ausführung durch eine Reihe unterschiedlicher Interaktionen ergeben sich noch eine Vielzahl weiterer Anforderungen an einen graphischen Editor als Benutzungsoberfläche von *VIOLA*. Im Vordergrund steht hierbei — ganz im Sinne *intelligenter* Datenanalyse — die umfassende Information des Anwenders über Analyseablauf und betrachtete Daten.

Modulicons

Zunächst einmal sind die verschiedenen Modulklassen von *VIOLA* durch unterschiedliche Icons zu symbolisieren, die dem Benutzer einen schnelleren Überblick über die Struktur der durchgeführten Analyse gewähren. Tabelle 5.5 stellt die gewählten Darstellungen vor. Die Icons repräsentieren jeweils durch ihre äußere Form eine Obergruppe von Modulen gemäß Def. 5.3 (Datenquellen, Datensenken, Datenmanagement, Analyse, Parameterbearbeitung, Kontrollstrukturen, Verbunde) und durch weitere Details die spezifische Modulklasse.

Datenquellen und -senken		**Daten-management**		**Berechnungen und Parameter**		**Kontrollstrukturen und Verbunde**	
	$\mathcal{MOD}^{\text{mak}}$		$\mathcal{MOD}^{\text{abl}}$		$\mathcal{MOD}^{\text{cell}}$		$\mathcal{MOD}^{\text{route}}$
	$\mathcal{MOD}^{\text{mik}}$		$\mathcal{MOD}^{\text{ablv}}$		$\mathcal{MOD}^{\text{aggr}}$		$\mathcal{MOD}^{\text{if}}$
	$\mathcal{MOD}^{\text{attr}}$		$\mathcal{MOD}^{\text{sel}}$		$\mathcal{MOD}^{\text{join}}$		$\mathcal{MOD}^{\text{loop}}$
	$\mathcal{MOD}^{\text{in}}$		$\mathcal{MOD}^{\cup}$				$\mathcal{MOD}^{\text{stop}}$
	$\mathcal{MOD}^{\text{save}}$		$\mathcal{MOD}^{\text{merge}}$		$\mathcal{MOD}^{\text{defpar}}$		
	$\mathcal{MOD}^{\text{vis}}$		$\mathcal{MOD}^{\text{split}}$		$\mathcal{MOD}^{\text{iopar}}$ $(in? = true)$		$\mathcal{MOD}^{\text{sub}}$
	$\mathcal{MOD}^{\text{out}}$		$\mathcal{MOD}^{\text{role}}$		$\mathcal{MOD}^{\text{iopar}}$ $(in? = false)$		

Tabelle 5.5: Icons zur Repräsentation der *VIOLA*–Module

Die Wahl bzw. der Entwurf „sprechender“, intuitiv verständlicher Icons ist ein zentrales Thema nicht nur in der iconbasierten visuellen Programmierung [Cha87, TG86], sondern allgemein im Rahmen der Programmvisualisierung [GP96, Sch98] (vgl. auch Abschnitt 2.2.2 und 2.2.3) sowie der Gestaltung von interaktiven Benutzungsoberflächen [P$^+$94, Lod83, Shn87]. Icons sollten gut zu identifizieren, voneinander zu unterscheiden, konsistent, leicht erlernbar und nicht zuletzt ansprechend sein sowie gut im Gedächtnis haften bleiben. Ihre

Gestaltung sollte sich am jeweiligen Anwendungskontext bzw. den entsprechenden Benutzergruppen orientieren

Die hier vorgestellten Icons lehnen sich zum einen — wo immer möglich — an in der multidimensionalen Modellierung und der datenflußbasierten Programmierung gängige Notationen (für Datenquellen und -senken, Datenwürfel, Kategorienhierarchien und Kontrollstrukturen) an und versuchen außerdem, durch ihre zweistufige, strukturierte Gestaltung mittels Rahmen und Inhalt ein leichteres Erlernen und Erinnern zu ermöglichen.

Eigenschaften von Ports und Kanälen

Weiterhin werden port- und kanalbezogene statische Charakteristika und dynamische Programmzustände in geeigneter Form, also möglichst intuitiv verständlich, visualisiert, ohne jedoch die Programmdarstellung zu überfrachten. Hierdurch erhält der Benutzer zu jeder Zeit Antworten auf wichtige Fragen wie „Was wurde bereits berechnet, was noch nicht?“, „Wo gibt es Anwendbarkeitsprobleme?“, „Welche Parameterangaben sind manuell definiert und überschreiben vorläufig automatische Berechnungsergebnisse?“. Tabelle 5.6 faßt die entsprechenden Vorschläge zusammen.

Generell erscheinen Datenports in Trapezform auf der linken und rechten Seite eines Moduls (unterschieden nach Ein- und Ausgängen) sowie Parameterports am unteren Modulrand als kleine Rechtecke („Parameterausgänge“), die um eine jeweils gegenüberliegende (in der Grundform ebenfalls rechteckige) Form am oberen Modulrand (einen „Parametereingang“) ergänzt werden, sofern der Port *in*? = *true* erfüllt.

Information	**Umsetzung (Entitätsmenge)**	**Ausprägungen**
Datenstatus	Liniendicke (Ports und Kanäle)	ok, schema, leer
Propagierungsstatus	Helligkeit (Ports)	aktiv, passiv, offen
Aktualität	Linienstil (Ports und Kanäle)	aktuell, vorläufig
Anwendbarkeit	Färbung (Ports und Kanäle)	(grün) eindeutig — (gelb) mehrdeutig — (rot) abgelehnt
Kardinalitäten	Form und Text (Dateneingangsports)	0, 1, 2, n
Typen (Wertebereiche)	Text und Form (Parameterporteingänge)	*K*, ***D***, *mz*, •••
Zusammenführungsfunktion	Form (Parameterporteingänge)	erster, letzter, Union / oder, Schnitt / und
Unveränderte Eingangsdaten	=–Symbol (Datenausgangsports)	=
Automatische Ports	Form (Parameterports)	

Tabelle 5.6: Visualisierung der Stati und Charakteristika von Ports und Kanälen

Die vier in Abschnitt 5.4.2 eingeführten Zustandskomponenten Anwendbarkeit, Aktualität, Daten- und

Propagierungsstatus werden im wesentlichen durch vier unterschiedliche graphische Attribute von Ports und Kanälen visualisiert: Farbe, Linienstil und -dicke sowie Flächenhelligkeit. Invalide Daten werden stets durch schwarze, durchgezogene, dünne Linien dargestellt, da für diese Anwendbarkeit und Aktualität keine Rolle spielen. Je nach Bedarf kann die für eine eingeschränkte Anwendbarkeit stehende gelbe Einfärbung auch dazu genutzt werden, weitere (potentielle) „Probleme“ bei der Verfahrensanwendung zu signalisieren, etwa den Zugriff auf Nullwerte, Divisionen durch Null oder die aggregierende Nutzung rein restriktiver Attribute in allgemeinen Maßzahlberechnungen (Modulen aus $\mathcal{MOD}^{\text{join}}$). Hier sollen jedoch keine allgemeinen Regelungen festgelegt werden; die konkrete Definition der Nutzung des Anwendbarkeitskriteriums bleibt einzelnen Modulen bzw. Maßzahlklassen vorbehalten. Auch erscheint es nicht sinnvoll, weitere unterschiedliche Einfärbungen (orange, hellgrün etc.) einzuführen, da eine klare Abgrenzung schwierig wäre oder zu einer zu großen Menge verschiedener Farben führen würde. Statt dessen können nähere Angaben und Erläuterungen jeweils textuell abgefragt werden (s. u.).

Unter Umständen könnten die Darstellungsarten von Datenstatus und Aktualität, Liniendicke und -stil, auch vertauscht werden. Zwar spiegelt die Liniendicke eher den „Umfang“ propagierter Inhalte wieder, jedoch ließen sich drei verschiedene Ausprägungen besser durch eine unterschiedlich feine Strichelung als durch die visuell schwerer zu differenzierende Breite von Linien visualisieren. Letztendlich sind hier konkrete Erfahrungen im routinemäßigen Einsatz von *VIOLA* abzuwarten. Im Gegensatz zu Datenstatus, Aktualität und Anwendbarkeit ist es im Falle des Propagierungsstatus ausreichend, diesen nur für Ports, nicht jedoch für Kanäle zu visualisieren, zumal er sich nicht direkt auf die (über die Kanäle fließenden) Daten, sondern vielmehr auf den Ort ihrer Ermittlung bezieht.

Neben den allgemeinen Statusinformationen werden auch Kardinalitäten von Eingangsports sowie die Wertebereiche von Parameterports (im wesentlichen durch knappe textuelle Angaben) dargestellt. Mengenwertige Parameter werden speziell durch einen Balken auf dem Parameterport gekennzeichnet. Es wäre zusätzlich denkbar, Parameterwertebereiche auch durch spezielle Einfärbungen der Portflächen identifizierbar zu machen. In Anbetracht der vielfältigen Kombinationsmöglichkeiten zueinander kompatibler Parameter könnte man hierbei jeweils den Ausgangsteil eines Parameterports einheitlich färben und im Eingangsteil die Farben akzeptierter Wertebereiche nebeneinander kombinieren. Insgesamt bestünde mit derartigen Einfärbungen jedoch die Gefahr der Überladung der Programmvisualisierung, so daß wir zunächst einmal darauf verzichten wollen. Die über die Art der Zusammenfassung mehrerer eingehender Parameterkanäle gemäß der Funktion f^{parjoin} definierten Typen von Parameterports (vgl. Abschnitt 5.4.2) lassen sich wiederum einfach durch leicht zu memorierende Form–Varianten der Parametereingänge repräsentieren.

Durch eine Kennzeichnung eines Datenausgangs wird dem Anwender jeweils signalisiert, ob die Daten aus einem entsprechenden Dateneingang unverändert weitergeleitet wurden (generell für Datensenken oder Parameterdefinitionsmodule, in bestimmten Fällen aber auch bei vollständiger Ableitung, Split oder Merge), so daß umgekehrt eine klare Identifikation von Datenmanipulationen möglich ist. Außerdem werden automatische Parameterports durch eine doppelte Umrahmung hervorgehoben. Aufgrund seiner herausragenden Bedeutung wird schließlich auch das Icon von Abbruchmodulen besonders (etwa rot) eingefärbt, wenn der Abbruchparameter den Wert *true* annimmt.

Interaktive Abfrage weiterer Informationen

Neben der beschriebenen visuellen Repräsentation, die auch eine geeignete Basis für eine Animation bzw. Visualisierung der Programmausführung bildet, bieten Pop–up Menüs die Möglichkeit der direkten Inspektion von Port- und Kanalzuständen. Auf die gleiche Weise ergibt sich ein lesender und teilweise auch schreibender Zugriff auf die Metadaten–Beschreibungen vorliegender Datenräume und ihrer Komponenten, also auf Schemata und weitere Metadaten gemäß Abschnitt 4.4. Auch Parametern, vor allem berechneten multidimensionalen Parametern, können so Metainformationen zugeordnet und wieder abgefragt werden. Gerade textuelle Metadaten sind oftmals manuell zu definieren; eine zukünftige Erweiterung von *VIOLA* könnte hier evtl. eine (halb-)automatische Unterstützung bei der Festlegung beschreibender Metadaten von aus anderen Datenräumen abgeleiteten Datenwürfeln leisten.

Weiterer Bedarf zur Benutzerinformation besteht noch bzgl. der in Modulen eingesetzten Verfahren und ihren jeweiligen Anwendbarkeitsbedingungen sowie im Hinblick auf die Erklärung der Ergebnisse fehlgeschlagener oder nur eingeschränkt erfolgreicher Anwendbarkeitsüberprüfungen. Hierfür werden über die jeweiligen Programmkomponenten Möglichkeiten zum Aufruf einer On–line–Hilfe geboten. Auch auf Mehrdeutigkeiten in Berechnungen kann so hingewiesen werden. Denkbar wäre zudem eine manuelle Auflösung von Mehrdeutigkeiten. Da diese oftmals jedoch nur schwer im Detail darzustellen sind (etwa die Ableitung von Kategorien für die explizite Aggregierung) und eine eindeutige Alternative jeweils entsprechend schwer zu spezifizieren wäre, beschränkt sich *VIOLA* derzeit auf die automatische Auswahl einer möglichen Variante. Eine Konfliktlösung besteht natürlich stets in einer Programm- oder Datenmanipulation, die die gewünschte Berechnung eindeutig macht und in der Regel als Korrektur eines Programmierfehlers anzusehen ist.

Sonstiges

Zum Abschluß dieses Abschnitts greifen wir noch einmal einige weitere wichtige Aspekte heraus, die als Anforderungen an einen komfortablen graphischen Editor zur Programmerstellung und -ausführung gemäß Abschnitt 5.4.5 zu sehen sind:

- Bei der Auswahl von Parameterwerten können Ausprägungen hervorgehoben werden, die keinen Anwendbarkeitsbedingungen innerhalb des jeweiligen Moduls widersprechen.

- Zur genauen Inspektion eines Verbundmoduls ist ein „Blättern“ zwischen den Instanzen des jeweiligen Analysegraphen erforderlich.

- Um ein dynamisches Linking von Graphiken bzw. Dynamic Queries im gängigen Sinne zu realisieren, müssen Oberflächenkomponenten (insbesondere Schieberegler) angeboten werden, die noch während der Interaktion und mit jeder Änderung der Belegung die Verarbeitungsalgorithmen von *VIOLA* initiieren.[63] Diese Echtzeit–Propagierung sollte vom Benutzer auch unterbunden werden können. Allgemeiner sollten auch mehrere Parameter eines Moduls direkt nacheinander geändert werden können, ohne daß sofort nach der ersten Manipulation eine Neuberechnung abhängiger Ergebnisse erfolgt. Dies entspricht in etwa einem Umschalten auf den programmweiten Verarbeitungsstatus ohne Propagierung, solange ein einzelnes Modul gerade in manueller „Bearbeitung“ ist.

- Bei der Plazierung und übersichtlichen Verbindung von Modulen (durch mehrteilige Streckenzüge) kann und soll der Anwender vom Editor durch entsprechende (halb-)automatische Layoutalgorithmen unterstützt werden (vgl. [GN95, San96, BJM97] sowie Arbeiten zum Graphen–Visualisierungssystem *daVinci* [Frö98]).

- Sowohl die parallele, unabhängige Verarbeitung verschiedener Eingänge von *VIOLA*–Modulen als auch die flexible Weiterleitung von Parametereinstellungen von einem zum anderen Modul dienen der übersichtlichen, strukturierten Gestaltung von *VIOLA*–Programmen. Zusätzlich kann es sinnvoll sein, speziell Datenquellen und Parameterschnittstellenmodule rein visuell zu duplizieren, d. h. anstatt mehrere, teilweise zu entfernten Zielmodulen führende Kanäle an die jeweiligen Datenausgänge anzulegen, werden Repräsentanten *derselben* Modulinstanz an unterschiedlichen Stellen bedarfsgemäß im Programm plaziert, so daß dort jeweils nur ein kurzer Kanal zum Zielmodul erforderlich ist. Intern wird weiterhin jedoch nur ein Modul mit mehreren ausgehenden Kanälen verwaltet. Ein beliebiger Einsatz dieses Konzepts für alle Modularten erscheint zum einen in Anbetracht oben geschilderter Alternativen nicht nötig und würde zum anderen aufgrund der versteckten Zusammenhänge zu Interpretationsproblemen sowie Programmierfehlern führen. Für die genannten Module ohne Dateneingänge sollte der Einsatz jedoch vertretbar sein, sofern die Kopplung duplizierter Module für den Benutzer zumindest auf Anfrage ersichtlich ist.

[63] Auf entsprechende Überlegungen zur effizienten Propagierung kommen wir in Abschnitt 6.2.2 noch zu sprechen.

- Frei plazierbare textuelle Kommentare, gerade auch als Beschriftung von Modulen, ermöglichen schließlich allgemein ein leichteres Verständnis erstellter Analysegraphen.

Das allgemeine Problem der Skalierbarkeit visueller Programme bzw. Programmierumgebungen (vgl. [GQ94, BBB+95, SKA94]) wird in *VIOLA* an mehreren Stellen angegangen. Zu nennen sind hier die parallele Verarbeitung mehrerer Eingänge von Modulen mit dem jeweils gleichen Verfahren oder alternativ die Zusammenfassung gleichartig zu behandelnder Daten durch Vereinigungsoperationen, der Einsatz von Verbundmodulen, die flexible Parametrisierung sowie die gerade angeführte Kommentierung. Zusätzlich könnten auch wahlweise Parameterkanäle aus einem Analysegraphen ausgeblendet werden, um eine Fokussierung auf die Datenverarbeitung zu ermöglichen. Auch ein explizites Hervorheben von Kanälen bestimmten Typs kann sinnvoll sein. Zooming–Verfahren schließlich kommen in *VIOLA* nicht zum Einsatz — hier wird der prozeduralen Abstraktion der Vorzug gegeben.

5.5 Zusammenfassung und Fazit

Mit dem Ziel einer besseren Unterstützung interaktiver Datenverarbeitung entstehen derzeit — im Multimediabereich, im Workflowmanagement oder auch in der Datenanalyse — immer neue Systeme zur visuellen Programmierung. Gerade das Gebiet des On–line Analytical Processing erfordert in hohem Maße eine flexible Interaktion mit dem zu betrachtenden Datenbestand [WKC96]. Zwar bieten OLAP–Systeme immer komfortablere menübasierte Oberflächen sowie direkt zu manipulierende und zu parametrisierende graphische Darstellungen; ein flexibler, auf einer visuellen Repräsentation basierender Umgang mit dem jeweils unterliegenden explorativen Analyseprozeß, also mit der Historie einer Datenanalyse, wird derzeit jedoch so gut wie gar nicht unterstützt. Diese Lücke schließt *VIOLA*, ein System zur visuellen Analyse multidimensionaler Daten.

Durch vielfältige Informations- und Interaktionsmöglichkeiten sowie die flexible Wahl verschiedener Ausführungsmodi im Spektrum daten- und anforderungsgetriebener Datenverarbeitung bietet *VIOLA* eine solide Basis für eine direkte und differenzierte Kommunikation zwischen menschlichem Datenanalysten und rechnerbasiertem Analysesystem und somit für die Durchführung *intelligenter* Datenanalysen. Die Visualisierung des jeweiligen Analyseprogramms gewährt jederzeit einen umfassenden Einblick in den aktuellen inhaltlichen und organisatorischen Zustand einer Analysesitzung. Weiterhin erlaubt sie die direkte Manipulation des Analyseprozesses, so wie dynamische Visualisierungen die direkte Interaktion mit den verarbeiteten Daten anbieten.

Während den meisten bestehenden visuellen Datenanalysesystemen eher eine funktionsorientierte Sicht auf den Analyseprozeß zugrunde liegt, baut *VIOLA* auf die direkte Umsetzung des multidimensionalen Datenmodells *MADEIRA*. In einheitlich aufgebauten Modulklassen werden die Operatoren der durch *MADEIRA* definierten Algebra umgesetzt. Somit ergibt sich ein überschaubarer, leicht verständlicher Satz an wenigen grundsätzlich unterschiedlichen Programmbausteinen, wobei die *MADEIRA*–Operatoren noch gemäß der Palette gängiger Analyseinteraktionen um Visualisierungen, Parametermanipulationen und Kontrollstrukturen ergänzt werden.

Mit der formalen, durch die *MADEIRA*–Algebra definierten funktionalen Grundlage, der objektorientierten Kapselung verschiedener Verfahrensklassen und der datenflußbasierten Verarbeitung von Analyseprogrammen kombiniert *VIOLA* Konzepte mehrerer visueller Programmierparadigmen, wobei das Datenflußparadigma sicherlich im Vordergrund steht. *VIOLA* unterscheidet klar zwischen Daten- und Parameterverarbeitung. Die Analyse multidimensionaler Daten bildet die zentrale Aufgabenstellung, Parametermanipulationen werden lediglich soweit unterstützt, wie es für die flexible Parametrisierung der *MADEIRA*–Operatoren notwendig ist. Dieser Grundsatz hält die Systemfunktionalität überschaubar.

Aufgrund der Granularität der angebotenen Funktionsmodule sowie des Aufbaus typischer explorativer Analysesitzungen kann davon ausgegangen werden, daß die Größe von interaktiv genutzten *VIOLA*–Programmen auf Graphen von höchstens wenigen hundert Modulen beschränkt ist.[64] Themen, die die Effizienz

[64] Diese Abschätzung erscheint vor dem Hintergrund plausibel, daß der Anwender in einer Datenanalysesitzung jeweils eine abgegrenzte, mehr oder weniger komplexe Fragestellung bearbeitet, deren Bearbeitung durch einen Analysegraphen gerade eine bestimmte, auch noch im Gesamtüberblick kognitiv erfaßbare Analysestrategie widerspiegelt.

der Programmausführung sowie der Aktualisierung von Programmstati betreffen, wurden aus diesem Grund hier bewußt vernachlässigt. Bei der Ausführung einer Benutzerinteraktion wird jeder (transitiv betroffene) Port des betrachteten Analysegraphen jeweils maximal einmal[65] im Rahmen der Anforderung, Invalidierung, Vorläufigkeits- und vor allem Datenpropagierung bearbeitet. Dieser lineare Aufwand läßt sich prinzipiell auch nicht reduzieren, wenn stets alle von einer Änderung oder Anfrage betroffenen Belegungen auf dem aktuellsten Stand gehalten werden sollen. Allein Detailfragen zur Realisierung der Kommunikation zwischen Ports können den Laufzeitaufwand um einen konstanten Faktor beeinflussen.

In Abschnitt 2.2.2 wurden einige von Schiffer in [Sch98] formulierte Kriterien zur Bewertung der Sinnhaftigkeit des Einsatzes visueller Programmierung untersucht. Ihre Anwendung auf *VIOLA* ergibt im wesentlichen ein positives Urteil:

- Mit der Darstellung grober Abläufe sowie der Unterstützung des Analyseentwurfs erhöht *VIOLA* die Verständlichkeit von Datenanalysen und erleichtert die Kommunikation zwischen Datenanalyst und Analysesystem einerseits sowie Nutzern/Auftraggebern der Analysen andererseits.
- In Anbetracht der eher geringen Größe typischer Programme und der angebotenen, breit gefächerten Funktionalität ist der „spielerische“, interaktive Umgang mit dem System als signifikanter Vorteil zu sehen, ohne daß die Gefahr einer schnellen „Benutzerfrustration“ besteht.
- Viele Konzepte von *VIOLA* zielen speziell darauf ab, dem Anwender einen umfassenden Überblick über und Zugriff auf die betrachtete Analysesitzung zu gewähren.

Weiterhin soll hier noch ein kurzer kritischer Abgleich mit den von Green und Petre vorgeschlagenen „kognitiven Dimensionen“ zur Beurteilung visueller Programmierumgebungen (siehe Abschnitt 2.2.3) erfolgen:

Die Granularität der Basismodule sowie das Konzept der Verbundmodule scheinen ein geeignetes Abstraktionsniveau zu bilden, um Datenanalysen strukturieren und verständlich darstellen zu können. Nicht zuletzt durch den direkten Aufbau auf dem multidimensionalen Datenmodell *MADEIRA* ergibt sich eine konsistente Sprachdefinition, die lediglich auf wenigen grundlegenden Konzepten basiert. Außerdem besteht so eine sehr enge und intuitive Abbildung zwischen visueller Programmiersprache und Problemdomäne (der explorativen Analyse von Daten in vielen miteinander verketteten einzelnen Analyseschritten). Die Programmierung kann im Hinblick auf Reihenfolge und Layout recht frei und flexibel erfolgen. Auch „unvollständige“ Programme (sofern es sie in *VIOLA* überhaupt gibt) sind jederzeit ausführbar bzw. werden sogar laufend ausgeführt.

VIOLA enthält allenfalls die Schleife als (aufgrund ihrer visuellen Umsetzung) „mental schwierige“ Operation. Die *MADEIRA*–Operatoren an sich wurden mit dem Ziel möglichst flexibler und eingängiger Nutzbarkeit entworfen. Ob dies gelungen ist oder ob sich bei ihrer Nutzung in *VIOLA* Unklarheiten ergeben können, muß die Erfahrung in der Anwendung von *VIOLA* zeigen. Abhängigkeiten zwischen Komponenten von *VIOLA*–Programmen sowie deren jeweilige Aufgabe im Gesamtkontext sind größtenteils klar und offen dargestellt. Einige verdeckte Zusammenhänge treten im Kontext von Verbundmodulen (an den Schnittstellen und mit dem Abbruchmodul) auf. Sekundäre Notationen werden in vielerlei Hinsicht, evtl. sogar *zu* exzessiv in *VIOLA* genutzt. Hier spielt sicherlich auch das subjektive Empfinden des einzelnen Anwenders eine zentrale Rolle. Gegebenenfalls können einzelne graphische Attribute von Programmkomponenten auf Wunsch auch deaktiviert und die hinterliegenden Informationen lediglich textuell auf Abruf angeboten werden.

An syntaktischen Feinheiten, deren Mißachtung zu Fehlern in der Programmierung führen könnte, sind evtl. die unterschiedlichen Typen von Parameterporteingängen sowie die Verwendung mehrerer Schleifenmodule in einem Verbund zu nennen. Auch sind vielleicht nicht alle Arten der externen Parametrisierung von Modulen für jeden Anwender gleichermaßen intuitiv (etwa im Fall der Vereinigungsmodule die Angabe von Eigenschaften, über die kategorielle Attribute gerade *nicht* zusammengeführt werden sollen). Jedoch sollten die Ursachen bzw. Quellen ungewollter Analyseaktionen aufgrund des umfassenden Informationsangebots in *VIOLA* schnell aufzuspüren, nachzuvollziehen und ggf. zu korrigieren sein. Auch das Auftreten *vorläufiger* Zwischenergebnisse könnte möglicherweise mißverständlich sein. Weiterhin führt die Vielzahl der mit einem

[65] Innerhalb von Schleifen natürlich maximal so oft, wie es der Anzahl der Iterationen entspricht.

Modul und seinen Komponenten dargestellten Details zu Abstrichen in der Kompaktheit der Notation, was jedoch nur bei sehr großen Programmen ins Gewicht fallen dürfte.

Wie schon Green und Petre feststellten, kann ein Sprachentwurf naturgemäß nicht in jeder Hinsicht alle Anforderungen erfüllen, stets sind Kompromisse zwischen verschiedenen Zielsetzungen einzugehen. Gerade mit der größtmöglichen Offenlegung aller für eine Analysesitzung relevanten Informationen sollte *VIOLA* eine gute Basis für eine hohe Benutzerakzeptanz legen.

Im nachfolgenden Kapitel 6 werden wir über die angeführten Aspekte hinaus sehen, wie *VIOLA* aufgrund der hier definierten Strukturierung von Analyseabläufen als Grundlage verschiedener alternativer Wege zur komfortablen Datenanalyse, die unterschiedliche Anwendergruppen und Informationsbedürfnisse ansprechen, dienen kann.

Kapitel 6

Implementierung und Anwendung von *VIOLA*

Die vollständige Implementierung eines flexibel einsetzbaren, komfortabel nutzbaren Datenanalysesystems geht über den Rahmen dieser Arbeit hinaus. Wenn auch die in den vorangegangenen Kapiteln vorgestellten Strukturen und Algorithmen bereits recht detailliert beschrieben und großteils direkt in eine Systemimplementierung umsetzbar bzw. bereits prototypisch realisiert sind, so erfordert die Erstellung eines real einsatzfähigen Systems doch noch umfangreiche Betrachtungen vor allem hinsichtlich der Konzeption und physischen Realisierung der Schnittstellen zu Datenbankmanagement-, externen Analyse- und Visualisierungssystemen sowie zum Anwender selbst. Nichtsdestoweniger soll in diesem Kapitel bereits diesbezüglich eine gewisse Grundlage als Vision für Anschlußarbeiten gelegt werden; einige Aspekte wurden auch bereits im Verlauf von Kapitel 4 und 5 kurz angerissen.

Abschnitt 6.1 skizziert eine mögliche Gesamtarchitektur eines auf Basis von *VIOLA* entworfenen Analysesystems und geht auf dessen externe Schnittstellen ein. Abschnitt 6.2 stellt — ohne jedoch alle Details auszuarbeiten und zu evaluieren — einige Ansätze zur Optimierung der Ausführung von *VIOLA*–Programmen vor, die insbesondere für einen Einsatz im Umfeld des Data Warehousing, also für die Auswertung *großer* Datenbestände, sinnvoll anwendbar sein dürften. In Abschnitt 6.3 wird schließlich als konkretes Anwendungsbeispiel gezeigt, wie *MADEIRA* und *VIOLA* in der praktischen Arbeit des Niedersächsischen Krebsregisters eingesetzt werden können. Hiermit wird auch nochmals ein spezifischer Rahmen mit speziellen Anforderungen an eine erste *VIOLA*–Implementierung definiert und zugleich eine weitreichende Übertragbarkeit auf andere Anwendungsbereiche motiviert.

6.1 Eine offene Client–Server–Architektur

Die Implementierung von *VIOLA* soll mehr darstellen als „nur“ eine visuelle Programmiersprache mit entsprechender Programmierumgebung. Vielmehr kann die datenflußbasierte Modellierung und Verarbeitung als solide und einheitliche Basis unterschiedlicher Benutzungsschnittstellen für unterschiedliche Nutzergruppen und Anwendungsziele dienen (vgl. entsprechende Gedanken in [UFK+89] und [YL95]). Vor diesem Hintergrund ergibt sich eine dreischichtige Architektur eines verteilten Systems, dessen Teilkomponenten und Daten- bzw. Informationsflüsse in Abb. 6.1 skizziert sind:

- Verschiedene, potentiell verteilte Datenbankserver stellen — koordiniert durch einen Datenbasen–Manager — Datenbasen als Eingangsdaten von Datenquellmodulen bereit. An zentraler Stelle finden sich ein für alle Datenserver einheitliches Dimensionsschema, ein Datenbank–Cache für effizienteren Datenzugriff sowie die Möglichkeit zur persistenten Ablage von Analysegraphen.

- Der *VIOLA*–Server selbst implementiert die Verarbeitung von *VIOLA*–Programmen, der — in Analogie zur Anfrageverarbeitung in DBMS — eine Optimierung der jeweiligen Analysegraphen vorgeschaltet werden kann. Neben einem hauptspeicherbasierten Caching ist hier eine Vielzahl von Schnittstellen zur Spezifikation und Manipulation von zu bearbeitenden Programmen und zu verwendenden Datenbasen sowie zur Erweiterung der Systemfunktionalität um neue (auch extern realisierte) Verarbeitungsverfahren und weitere Optimierungsregeln angesiedelt.
- Über verschiedene Arten von clientseitigen Benutzungsschnittstellen, unter anderem natürlich auch einen graphischen Editor zur visuellen Programmierung, kann schließlich mit dem „Datenanalyseserver“ kommuniziert werden, um konkrete Datenanalysen durchzuführen.

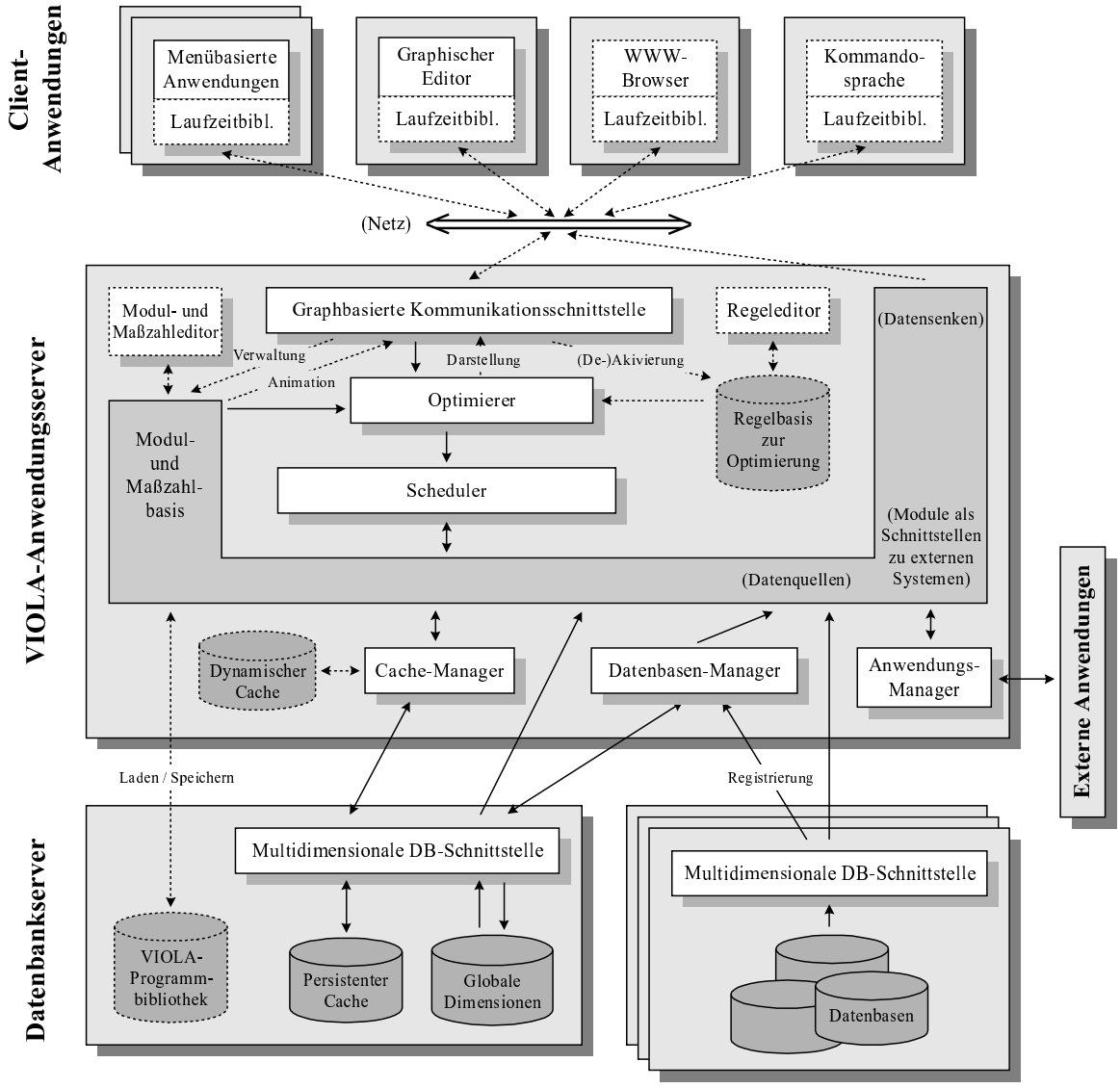

Abbildung 6.1: Architektur des *VIOLA*–Systems

Derzeit existiert noch keine vollständige Realisierung des in Abb. 6.1 dargestellten Systems. Mehrere Teilkomponenten sind jedoch bereits prototypisch unter Verwendung der Programmiersprache *C++* umgesetzt und

in ein Gesamtsystem unter *SUN Solaris 2.5* mit *ORACLE* 7 als verwendetem DBMS integriert.[1] Komponenten, die bisher noch nicht näher betrachtet wurden, sind in Abb. 6.1 gestrichelt dargestellt.

In [Mei99] wird die verteilte Lösung zur Implementierung von *VIOLA* im Detail vorgestellt. Als Middleware zur Kommunikation zwischen Datenbankservern und Anwendungsserver wird *CORBA* vorgeschlagen (vgl. [OHE98]), um möglichst flexibel unterschiedliche Plattformen unterstützen zu können. Die Kommunikation zwischen Client und *VIOLA*–Server erfolgt vor allem aus Effizienzgründen über eine Programmierschnittstelle im gleichen Adreßraum. Im Sinne eines flexibleren Einsatzes von *VIOLA* in einer mehrbenutzerfähigen Umgebung wäre jedoch noch eine weitere Öffnung des Systems durch dynamische Anbindung unterschiedlicher Clients nötig. Auch könnte hierdurch ein wechselseitiger Zugriff auf von mehreren Anwendern gemeinsam genutzte Zwischenergebnisse erfolgen und somit die Gesamteffizienz der Verarbeitung erhöht werden. Auch die Idee der *kooperativen* Datenanalyse, also der verteilten, gemeinsamen Erstellung und Manipulation von Analysegraphen, wäre untersuchenswert: Können sich mehrere auf der Basis eines Datenanalyseprogramms zusammenarbeitende Datenanalysten ergänzen oder behindern sie sich eher (vgl. Grundlagen der rechnergestützten Gruppenarbeit, etwa in [TSMB95])?

Im Kern realisiert der Anwendungsserver durch den *Scheduler* (im Zusammenwirken mit Modulen und Ports) gerade den in Abschnitt 5.4 vorgestellten Verarbeitungsalgorithmus für *VIOLA*–Programme (zur Implementierung siehe [Kel98, Hüs99]). Entsprechend steckt die eigentliche Datenanalyse–Funktionalität in einer umfassenden Basis an Modulklassen, die gemäß Abschnitt 5.2 definiert sind bzw. die dort eingeführten Oberklassen noch spezialisieren, sowie einer Menge von Maßzahlen mit ihren jeweiligen Berechnungs- und Aggregierungsfunktionen.[2] Über Modulinstanzen erfolgen auch der Zugriff auf Datenquellen, die Datenvisualisierung und — unter Nutzung spezieller Adapter, die ein Anwendungsmanager bereitstellt — die Kommunikation mit externen Systemen. Die graphbasierte Kommunikationsschnittstelle ermöglicht gemäß Abschnitt 5.4.5 die Konstruktion von *VIOLA*–Programmen sowie deren Manipulation, Parametrisierung und Ausführung. Teilkomponenten zur Optimierung von Analysegraphen (zu deren Transformation und zur Wahl der physischen Umsetzung von Moduloperationen) sowie zum Caching von Zwischenergebnissen runden die Funktionalität des Gesamtsystems ab. Bevor in Abschnitt 6.2 auf Optimierungsaspekte noch genauer eingegangen wird, werden im folgenden einige Gedanken zu den Systemschnittstellen zu Datenbank und Benutzungsoberfläche sowie zur Erweiterung der Systemfunktionalität ausgeführt. Hierbei kommen wir auch auf die jeweiligen Benutzergruppen von *VIOLA* zu sprechen: Administratoren, die Datenbasen zur Verfügung stellen und in Zusammenarbeit mit Analyseexperten Systemerweiterungen vornehmen, sowie die eigentlichen Systemanwender, die *VIOLA* zur Durchführung konkreter Analysen einsetzen.

6.1.1 Aspekte der Datenbankanbindung

Wie bereits erwähnt, wurde *ORACLE* 7 als DBMS zur Verwaltung der auszuwertenden Daten ausgewählt. Zur Kapselung der internen Repräsentation multidimensionaler Daten sowie zur Bereitstellung der durch *MADEIRA* definierten Schnittstelle zum Zugriff auf Datenräume und Dimensionen dient eine *multidimensionale Datenbankschnittstelle* als Aufsatz auf *ORACLE*.[3] Diese kommt für sich auch bereits im Auswertungssystem *CARESS*, das aktuell im Niedersächsischen Krebsregister genutzt wird, zum Einsatz. Sicherlich wäre es in Zukunft lohnenswert, neue Funktionalitäten zur Verwaltung multidimensionaler Daten in *ORACLE* 8 näher zu betrachten oder den Einsatz eines OLAP–Tools, etwa *ORACLE Express*, in Erwägung zu ziehen. Neben dem effizienten Zugriff auf Datenwürfel

- sind jedoch stets auch Möglichkeiten zur Ablage und Nutzung der mit *MADEIRA* eingeführten zusätzlichen semantischen Metadaten zu gewähren, wobei ebenfalls die Nutzung objektrelationaler Kon-

[1] Die Wahl dieser Umgebung ergab sich im wesentlichen aus dem Bezug zum Niedersächsischen Krebsregister und der dort vorliegenden Systemlandschaft.

[2] Um die Mächtigkeit der Verarbeitungsfunktionalität sowie die Möglichkeiten der Systemerweiterung nicht zu beschränken, ist keine datenbankseitige Ablage dieser Funktionen, etwa auf der Basis eines erweiterbaren DBMS, angedacht.

[3] Eine derartige Architektur entspricht der Realisierung eines durch ein Repository definierten Informationsmodells — hier zur Analyse multidimensionaler Daten — auf Basis eines universellen Datenbanksystems, wie sie in [Ber97] vorgeschlagen wird.

zepte aus *ORACLE* 8 hilfreich sein könnte,

- muß ein eng mit dem datenflußbasierten Verarbeitungsprozeß verknüpftes, teilweise benutzergesteuertes Caching zu realisieren sein, und
- müssen schließlich erstellte *VIOLA*–Programme auch im Datenbanksystem in einer Programmbibliothek abgelegt und wieder geladen werden können.

Vor diesem Hintergrund erscheint der Einsatz auf dem Markt existierender OLAP–Tools zumindest problematisch oder nur im Zusammenspiel mit einem „universellen“ DBMS sinnvoll.

Innerhalb eines *VIOLA*–Programms sind unterschiedliche, potentiell auch verteilte Datenbasen nutzbar. Datenbasen repräsentieren multidimensionale Datenräume, wobei u. U. bestimmte Datenbasen auch einschränkende oder aggregierende Sichten auf andere Datenbasen darstellen, die besonders häufig selektiert werden. Wir wollen davon ausgehen, daß alle diese Datenbasen über eine einheitliche Schnittstelle zur Bereitstellung multidimensionaler Daten im durch *MADEIRA* definierten Format angesprochen werden können. Somit lassen sich auch die Datenquellmodule auf einheitliche Weise realisieren — es brauchen in der Regel lediglich einige Parameter zur Steuerung von Netzkommunikation und Datenbankzugriff definiert zu werden.

Vor der Nutzung einer Datenbasis (einer physischen, durch ein DBMS verwalteten Datenquelle) ist jeweils eine einmalige Registrierung bei einem *Datenbasen–Manager* in *VIOLA* erforderlich. Dieser gleicht die von den verteilten Datenbasen genutzten Kategorienhierarchien mit einer Menge von Dimensionen in einer globalen Metadatenbasis ab, erweitert diese ggf. und speichert Tabellen zur Umcodierung von Identifikatoren. Bei der Definition der verteilten Datenbasen–Schemata ist jeweils bereits darauf zu achten, daß eine derartige Eingliederung in globale Dimensionen stets „problemlos“ möglich ist, etwa durch Verwendung der gleichen Mengen „feinster“ Kategorien bzw. Aggregierungsebenen. Analog wird global eine Menge weiterer Metadaten (Objektmengen, Eigenschaften, Rollen) vorgegeben, die in den verteilten Datenbeständen referenziert und ggf. um nur lokal genutzte Entitäten erweitert werden können. Auch hier ist ggf. ein globaler Abgleich von Identifikatoren erforderlich. Da die Verarbeitungsfunktionalität von Maßzahlen nicht in der Datenbank abgelegt wird, speichert jede Datenbasis als Komponenten summarischer Attribute eindeutige Kennungen global definierter Maßzahlen.

Nach erfolgter Registrierung stellt der Datenbasen–Manager den Datenquellmodulen in *VIOLA*–Programmen die Zugriffsparameter und Schemata der verfügbaren Datenbasen zur Verfügung, damit diese dem Anwender geeignete Auswahlmenüs zusammenstellen können. Auf die globalen Metadaten wird direkt aus all denjenigen Modulen zugegriffen, die entsprechende Angaben zu ihrer Parametrisierung vorsehen.

VIOLA soll nicht die Funktionalität eines verteilten DBMS realisieren. So wie alle Verarbeitungsschritte explizit spezifiziert werden, ist auch in einem Datenquellmodul jeweils direkt die interessierende Datenbasis auszuwählen, zu der dann die entsprechende Anfrage geleitet wird. Eine Unterstützung weitgehend „freier“ Anfragen und ein „intelligentes Zusammenstellen“ der zu ihrer Beantwortung nötigen Datenquellen sowie eine Aufteilung der Anfrage in mehrere Teilanfragen, wie z. B. im *CubeStar*– oder im METASTASYS–Projekt [AGL98, Fro97], ist nicht Ziel von *VIOLA*. Im Sinne von [Con97] bilden die von *VIOLA* angesprochenen (für sich autonomen) Datenserver ein lose (also spezifisch durch den Anwender, nämlich hier *VIOLA*) gekoppeltes föderiertes Datenbanksystem ohne Verteilungstransparenz; die Registrierung der Datenbasen mit Abgleich der Dimensionsinformationen entspricht der Integration verteilter Schemata zu einem föderierten Schema innerhalb einer Import–/Export–Schema–Architektur [Dad96, Con97].

6.1.2 Verschiedene Benutzungsschnittstellen

Die Datenanalyse stellt je nach Aufgabenstellung, Nutzergruppe und Analysesituation ganz unterschiedliche Anforderungen an die Benutzungsoberfläche eines Analysesystems. Young unterscheidet hierbei in [YS91] zwischen verschiedenen Kognitionsmodi (explorativ, konfirmatorisch, strukturierend), die jeweils andere Formen der Interaktion mit dem von ihm entwickelten *ViSta*–System erfordern. In ähnlicher Weise sollen über

verschiedene Clients mehrere Varianten der Steuerung des datenflußbasierten Analyseprozesses von *VIOLA* angeboten werden:

- Ein *graphbasierter Editor* (vgl. die Beschreibung eines Prototypen in [Kel98], der jedoch noch nicht alle in Abschnitt 5.4.5 formulierten Anforderungen erfüllt) dient Analyseexperten zur direkten Nutzung der visuellen Programmiersprache *VIOLA* — insbesondere zur strukturierten Exploration von Datenbeständen.

- Über eine Kommandosprache können vor allem Systemadministratoren einzelne vorgegebene, als konfirmatorisch einzustufende Anfragen durchführen sowie mittels einer Zusammenfassung von Einzelbefehlen in Skripten eine Deskription des Datenbestandes vornehmen.

- Indem Analyseabläufe hinter einer menübasierten Benutzungsoberfläche (wie sie etwa das Auswertungssystem *CARESS* des Niedersächsischen Krebsregisters [Wie99, RWW99] bietet[4]) gekapselt werden, kann dem Dokumentationspersonal die routinemäßige Durchführung von Standardauswertungen erleichtert werden.

- In ähnlicher Form kann eine etwa über WWW–Browser leicht zu bedienende Internet–Anwendung die Analysefunktionalität von *VIOLA* einem breiteren Publikum mit unterschiedlicher Expertise zur Verfügung stellen.

Für all diese Arten von Clients dient eine *graphbasierte Kommunikationsschnittstelle* der Spezifikation, Verwaltung und Ausführung von *VIOLA*–Programmen gemäß Abschnitt 5.4.5. Sie interagiert direkt mit den jeweils relevanten Modulen, leitet Analysegraphen zur Ausführung über einen Optimierer an einen Scheduler weiter und meldet in Form von Statusangaben den Stand der Verarbeitung an die Clients zurück. Weiterhin ermöglichen clientseitige *Laufzeitbibliotheken* den *VIOLA*–Datensenken den Zugriff auf Client–Ressourcen.

Auf diese Weise können parallel mehrere Analysegraphen von verschiedenen Nutzern verarbeitet werden. Indem die Kommunikationsschnittstelle auch den Zugriff auf bereits bestehende Programme gewährt, kann ein und dasselbe Programm (nacheinander, evtl. sogar gleichzeitig) von mehreren Anwendern bzw. auf unterschiedlichen Betrachtungsebenen angesprochen werden. So ist es möglich, über einen menübasierten Client vorgenommene Standardauswertungen durch einen Wechsel in die graphbasierte, visuelle Programmierumgebung zu verfeinern oder mittels einzelner Befehle einer Kommandosprache zu ergänzen. Natürlich sind hierbei — auch unter Berücksichtigung von Grundideen des kooperativen Arbeitens [TSMB95] — eine Abstimmung zwischen den gleichzeitigen Interaktionen verschiedener Nutzer zu gewährleisten bzw. Konflikte über Sperren zu verhindern. Zu beachten ist zudem, daß Benutzer die Metadaten generierter Analyseergebnisse individuell modifizieren können — evtl. sollten entsprechende Manipulationen jeweils benutzerspezifisch lokal pro Sitzung verwaltet werden. Häufig mag es natürlich auch sinnvoll sein, ein *VIOLA*–Programm generell jeweils nur einem Client exklusiv zugänglich zu machen.

Wenn auch die funktionale Programmierung i. a. eine direktere textuelle Umsetzung von datenflußbasierten Programmen ermöglicht (vgl. [HYM+92]), sollte die Kommando- bzw. Skriptsprache von *VIOLA* aufgrund der Funktionalität der Kommunikationsschnittstelle doch dem objektorientierten Paradigma folgen (wie etwa in *AVS* [AVS96, KAW+93] oder im *IBM Data Explorer* [LAC+92]): Es stehen eindeutig Module, deren Parametrisierung sowie gegenseitige Verbindungen im Vordergrund, nicht jedoch die Verarbeitungs- und Analysealgorithmen, die auf dieser Ebene nicht sichtbar sind. Ein entsprechendes Format kann auch für die Ablage von *VIOLA*–Programmen in der Datenbank verwendet werden.

6.1.3 Erweiterbarkeit des *VIOLA*–Systems

Bereits in Abschnitt 6.1.1 wurde darauf eingegangen, wie *VIOLA* um neue Datenbasen und auf deren Grundlage auch um neue Dimensionen, Kategorien, Eigenschaften, Objektmengen und Rollen erweitert werden kann.

[4] Dieses Beispiel werden wir in Abschnitt 6.3.2 noch vertiefen.

Zusätzlich sollte natürlich auch ein direkter Zugriff auf die globale Metadatenbasis mittels eines separaten, konsistenzsichernden Werkzeugs angeboten werden, um unabhängig von Datenbasen neue Metadaten einzufügen.

Auch externe Applikationen zur Datenanalyse und -visualisierung sollen an *VIOLA* anzubinden sein, wobei Module als Einheit der Kapselung dienen. Für den Anwender kann es somit im wesentlichen transparent bleiben, *wo* das durch ein Modul repräsentierte Verfahren ausgeführt wird. Die Umsetzung der datenflußbasierten Verarbeitung, also die Werteanforderung und -propagierung zwischen den Modulen, bleibt jedoch stets auf dem Anwendungsserver angesiedelt. In diesem Sinne kann *VIOLA* als Grundlage zur Koordination anderer Datenanalyseanwendungen genutzt werden.[5] Erste Ansätze zur Umsetzung dieser Idee in *VIOLA*, speziell zur externen Visualisierung von Analyseergebnissen unter Nutzung der Automatisierung auf Basis von *DCOM* (des *Distributed Component Object Model* von *Microsoft*) [EE98], finden sich in [Sch99].

Konkret wird die modulbasierte Verteilung von Analysefunktionalität in *VIOLA* bzw. *MADEIRA* integriert, indem den entsprechenden Maßzahlen (genauer: einzelnen Berechnungsvorschriften) und Visualisierungsverfahren intern anstelle einer konkreten Implementierung einer Funktion nur eine Identifikation des Ortes der Berechnung sowie zum Ansprechen des jeweiligen Programms nötige Aufrufparameter zugeordnet werden. Weiterhin ist ein von einem *Anwendungsmanager* bereitgestellter Adapter zu identifizieren, der jeweils ein Kommunikationsprotokoll mit dem Anwendungssystem realisiert. Somit muß auch keine Standardisierung auf ein bestimmtes Protokoll erfolgen. Der Propagierungsalgorithmus von *VIOLA* läßt sich schließlich einfach auch auf eine asynchrone Kommunikation mit externen Applikationen anpassen, indem eine weitere Warteschlange von Ports eingeführt wird. Dies impliziert gleichzeitig die Offenheit für eine parallele (zeitgleiche) Ausführung von Modulen auf verteilten Ressourcen (vgl. *AVS*, *IBM Data Explorer* oder *Weaves* [UFK+89, LAC+92, GR91]).

Auch ohne auf zusätzliche externe Applikationen zurückgreifen zu müssen, soll die Funktionalität von *VIOLA* durch die Definition neuer Modultypen zur Parameterdefinition, neuer Datensenken und neuer Maßzahlen bzw. neuer Metadaten für bestehende Maßzahlen erweitert werden können.[6] Die neuen Module spezifizieren lediglich neue Berechnungs- bzw. Visualisierungsverfahren sowie hierfür nötige Parameter und spezialisieren somit bereits bestehende Klassen. Gleichermaßen werden auch neue Maßzahlen in die gegebene Maßzahlhierarchie eingeordnet. Ein *Modul- und Maßzahleditor* bietet — ähnlich wie in *AVS* [UFK+89, LP98] oder *KHOROS* [RW91, YAK95] — dem Nutzer von *VIOLA* eine entsprechende Umgebung, in der die neuen Verfahren und Zuordnungen in Gestalt von *C++*–Unterklassen spezifiziert werden können. Diese werden automatisch in eine (durch ihre jeweiligen Oberklassen definierte) Hülle mit der internen Verwaltungsfunktionalität eingebettet und zum Gesamtsystem dynamisch hinzugebunden. Weiterhin sollen hier auch Bibliotheken von Verbundmodulen angelegt sowie Unterklassen von Analysemodulen, die zur klareren Strukturierung von Analysen nur eine Teilmenge aller möglichen Maßzahlen zur Auswahl anbieten, definiert werden können. Insgesamt realisiert *VIOLA* hiermit die Zweigleisigkeit zwischen „Big“ und „Little Programmer“, wie sie etwa in [ACSW96] propagiert wird: Ein Datenanalyseexperte definiert über den Modul- und Maßzahleditor Programmbausteine, die dann ein Anwender im *VIOLA*–System zu Analyseprogrammen zusammenfügen kann.

6.2 Optimierung visueller Anfragen und Analysen

Bisher stand in dieser Arbeit als Anwendungsgebiet die Domäne deskriptiver epidemiologischer Untersuchungen im Vordergrund. Da die in diesem Umfeld verarbeiteten Datenräume in der Regel nicht allzu groß werden, konnte darauf verzichtet werden, genauere Betrachtungen zur Effizienz der Datenverarbeitung anzustellen. Natürlich sollte auch hier bereits durch eine adäquate Umsetzung der *VIOLA*–Konzepte darauf geachtet werden, daß Datenmanipulation und -zugriff tatsächlich interaktiv, also ohne allzu große Wartezeiten (etwa bis zu einigen Sekunden), erfolgen können, um den kreativen Prozeß der Datenexploration nicht zu stören. Spätestens

[5] Vgl. in diesem Zusammenhang auch Arbeiten im Rahmen des *MMM*– (*Model Management System*) Projekts zur Nutzung verteilter statistischer Software [GMS+97].

[6] Bis auf weiteres sollen dies die einzig möglichen Erweiterungen sein. Weder sollen neue Kontrollstrukturen oder sonstige wirklich *neue* (und nicht nur speziell parametrisierte) Modulklassen eingeführt noch allgemeine Änderungen an Strukturen und Operatoren von *MADEIRA* vorgenommen werden können.

aber, wenn die in dieser Arbeit entwickelten Konzepte bei der Auswertung typischer, *großer* Data Warehouses zum Einsatz kommen sollen — was *konzeptionell* kein Problem darstellt —, sind umfassende effizienzsteigernde Maßnahmen zur Analyseunterstützung unabdingbar.

Aus diesem Grund werden in diesem Abschnitt einige Überlegungen zur Optimierung der Ausführung von *VIOLA*–Programmen skizziert. Es soll plausibel werden, daß *VIOLA* auch auf größere Datenbestände anwendbar ist, ohne jedoch die Details der Ausführung zu konkretisieren. Dies bleibt Anschlußarbeiten vorbehalten. Außerdem werden auch ausgefeilte Optimierungstechniken nichts daran ändern, daß der Ansatz bzw. Wunsch, alle Einzelschritte eines Analyseablaufs zugreifbar und in (nahezu) Echtzeit interaktiv manipulierbar zu machen, der Datenanalyse gewisse Grenzen hinsichtlich der Größe gleichzeitig zu untersuchender Datenräume auferlegt.

Ein Aspekt zur Steigerung der Verarbeitungsgeschwindigkeit von *VIOLA* ist sicherlich die Parallelisierung, also die Verteilung der Bearbeitung von Modulen auf mehrere Rechnersysteme, wie sie bereits im vorigen Abschnitt skizziert wurde. Hierauf wird im folgenden jedoch nicht näher eingegangen; vgl. zu diesem Thema etwa [UFK+89, LAC+92, GR91]. Vielmehr wollen wir uns hier auf die Übertragung von Konzepten der Anfrageverarbeitung in Standard–Datenbanksystemen (siehe etwa [Mit95, Vos99]) auf die Verarbeitung von Analysegraphen in *VIOLA* konzentrieren. Die Parallelen sind unverkennbar; auch Anfragen an Datenbanksysteme sowie deren Ausführungspläne werden oftmals in ihrer Interndarstellung als Operatorgraphen modelliert, deren Knoten den Basisoperationen der jeweiligen Anfragesprache bzw. deren physischer Umsetzung entsprechen. Die Anfrageoptimierung läßt sich nun in Form von Manipulationen dieser Operatorgraphen beschreiben. Je nachdem, ob Eigenschaften der internen Datenbank–Organisation in die Optimierung einbezogen werden, unterscheidet man zwischen

- *High–Level–Optimierung* (*Anfragerestrukturierung*, oft auch als *algebraische Optimierung* bezeichnet[7]), in relationalen Systemen etwa die Gruppierung von Projektionen und Selektionen über der gleichen Relation oder das Vorziehen von Selektionen und Projektionen (ohne Duplikateliminierung) gegenüber Verbunden, weiterhin die Identifikation gemeinsamer Teilbäume im Operatorgraphen sowie das Zusammenfassen oder Zerlegen von Operationen, und
- *physischer* oder *Low–Level–Optimierung* (auch *Anfragetransformation* oder *nicht–algebraische Optimierung* genannt), also die Auswahl physischer Operatoren zur Realisierung der Anfrageoperatoren, hierbei insbesondere auch die Nutzung von Indexstrukturen und Datenbank–Caches.

Datenflußbasierte Analysegraphen lassen sich auf recht ähnliche Art und Weise optimieren, also zunächst in semantisch äquivalente, aber effizienter auszuführende Analysegraphen umformen (siehe Abschnitt 6.2.1) und anschließend in direkt auszuführende Operatorgraphen übersetzen. In letzterem Schritt können durch Zusammenfassung von *VIOLA*–Modulen neue Operatoren eingeführt und die interne Handhabung der verarbeiteten Datenräume speziell organisiert (siehe Abschnitt 6.2.2) sowie Cache–Zugriffe in das *VIOLA*–Programm eingefügt werden (siehe Abschnitt 6.2.3). Im Gegensatz zur klassischen Anfrageverarbeitung bleiben im Rahmen der Optimierung jedoch auch viele Operationen (*VIOLA*–Module) unverändert und werden anschließend direkt ausgeführt.

Im Extremfall (der aktuell realisiert ist) kann auch ganz sowohl auf High–Level– als auch auf Low–Level–Optimierung verzichtet werden. Im Kontext der Optimierung neu definierte Modulklassen dienen lediglich der gleichzeitigen und somit effizienteren Bearbeitung mehrerer *VIOLA*– bzw. *MADEIRA*–Operationen (ohne Zwischenergebnisse der Einzelschritte). Es wird hier keine prinzipiell neue (niedrigere) Abstraktionsebene betreten, d. h. der optimierte Analysegraph bleibt ein *VIOLA*–Programm (ggf. mit einzelnen neuartigen Modulen und einigen internen Verarbeitungsflags) und kann dem Anwender auf Wunsch auch zur Ansicht und evtl. sogar zur Manipulation angeboten werden.

Ein wesentlicher Unterschied zur Anfrageoptimierung in Standard–Datenbanksystemen besteht hier jedoch darin, daß die Bearbeitung eines Analysegraphen als Teil einer gesamten Analysesitzung zu sehen ist, in de-

[7] Unter den Begriff der High–Level–Optimierung fallen auch Ansätze zur *semantischen Optimierung*, die auf nicht–algebraischer Ebene Eigenschaften der Datenbank*extension* berücksichtigen (vgl. [Tap99]). Dieser Aspekt spielt im weiteren jedoch keine Rolle.

ren Verlauf dieser Graph immer wieder leicht modifiziert und neu ausgeführt wird. Damit nicht jedesmal eine vollständige Neu–Optimierung erfolgen muß, ist der in Abschnitt 5.4 spezifizierte Analysealgorithmus so anzupassen, daß diese Interaktionen jeweils auch auf das bereits optimierte Pendant des Ausgangsgraphen übertragen werden können. Insgesamt ist stets möglichst vorausschauend zu optimieren, d. h. daß Optimierungsschritte daraufhin ausgewählt werden, ob sie auch eine effizientere Bearbeitung zukünftig zu erwartender Benutzerinteraktionen ermöglichen und nicht evtl. sogar im Gegenteil im weiteren Analyseverlauf zu einem höheren Verarbeitungsaufwand führen. Schließlich sollten sich Ausgangs- und optimierter Graph „nicht allzu sehr“ voneinander unterscheiden, um eine Wiederverwendung von bei der Ausführung des optimierten Graphen angefallenen Zwischenergebnissen zu ermöglichen. Eine Metrik hierfür wäre noch zu entwickeln.

Prinzipiell kann der vom Anwender modellierte Analysegraph als Vorgabe eines „guten“ Ausführungsplans angesehen werden, der oftmals bereits auf geplante Analyseschritte ausgerichtet ist. Hier eine adäquate, behutsame Einbeziehung von gängigen Techniken zur Anfrageoptimierung zu finden, ist sicherlich eine anspruchsvolle Aufgabe für zukünftige Arbeiten. In den folgenden Abschnitten werden wir lediglich einige grundsätzliche Ideen anführen — etwas detaillierter sind die Ausführungen in [Hüs99, Tap99].

Eine wichtige Rolle für die Optimierung in *VIOLA*, speziell die Abschätzung zukünftiger Manipulationen einer Analyse, spielt die Wahrscheinlichkeit für bzw. Häufigkeit von Änderungen an Modulparametern. Folgende tendenzielle Unterscheidungen sind hierbei zu machen:

- Im Verlauf einer Analysesitzung werden Zielkategorien einer Ableitungsoperation (vor allem einer Restriktion) oder einer expliziten Aggregierung in der Regel recht häufig modifiziert; das gleiche gilt für alternativ berechnete bzw. ausgewählte Maßzahlen eines Analyse-, Routing- oder Selektionsmoduls. Mit Änderung der Kategorienmengen von Datenräumen wird in der Regel auch die Auswahl der Kategorien einer kategoriellen Datenquelle beeinflußt. Gerade diese Variationen machen das vergleichende und verfeinernde Explorieren aus.

- Demgegenüber sind (nicht–kategorielle) Datenquellenzugriffe, Rollenumbenennungen, Split und Merge meistens sehr spezifisch auf eine bestimmte Situation, ein konkretes Ziel ausgelegt. Ebenso definiert die Variante einer Ableitungs- oder expliziten Aggregierungsoperation eine bestimmte, durch die Analysestruktur festgelegte Operationsart. Dementsprechend wollen wir all diese Parametrisierungen, ebenso wie die Rolle, die eine kategorielle Datenquelle einnimmt, als relativ statisch ansehen.

- Alle übrigen Parameter liegen in ihrer Änderungshäufigkeit zwischen den beiden obigen Extremen.

Tabelle 6.1 gibt auf Basis dieser Einteilungen einen Überblick der Parameter–Variabilitäten von gering ($-$) über mittel ($\circ$) bis hoch ($+$). Nicht abschätzbare Werte sind mit „?“, Änderungsraten von Parametern, die nicht zur Ergebnisberechnung (sondern nur zur Steuerung von Nebeneffekten) nötig sind, mit „/“ bezeichnet. Nicht näher differenzierte Bezüge auf *alle* Parameter eines Moduls sind durch „. . .“ angedeutet. Die beliebigen zusätzlichen Parameter von Mikro- und Makrodatenquellen sowie Analysemodulen, die bzgl. ihrer Änderungshäufigkeit wie die internen Parameter bzw. der Maßzahlparameter einzustufen sind, sind nicht speziell aufgeführt.

Neben der Parameterbetrachtung ist natürlich jeweils auch noch die Wahrscheinlichkeit für das Erzeugen neuer Kanäle und Module relevant. Die Arbeit von Canter [CRS85] gibt im Hinblick hierauf einige Anhaltspunkte für typische Interaktionsfolgen in bestimmten Analysesituationen (angefangen vom ziellosen „Umherwandern“ bis zum sehr zielgerichteten, verfeinernden Suchen — jeweils modelliert als Navigationspfade durch den betrachteten Datenraum). Die von Canter angegebenen, sehr groben Muster wären jedoch sicher noch zu verfeinern und an die Operatoren von *VIOLA* anzupassen. Ein weiterer Ansatz zur Vorhersage von Anfragesequenzen und Interaktionen auf der Basis der Modellierung von Benutzerprofilen findet sich bei Sapia [Sap99]. Konkret wird hier in Leerlaufzeiten ein Prefetching (vgl. [Ger96]) von Datenräumen durchgeführt, die zu den bisher benötigten im Sinne einer speziellen Metrik „ähnlich“ sind (d. h. im wesentlichen: die hinsichtlich beteiligter Dimensionen, summarischer Attribute und gewählter Aggregierungsebenen weitgehend übereinstimmen). Analog zur Belegung des Caches könnte dieser Ansatz auch genutzt werden, um

Modul	**Parameter**	**Änderung**	**Modul**	**Parameter**	**Änderung**	**Modul**	**Parameter**	**Änderung**
$\mathcal{MOD}^{\text{mak}}$	(intern)	–	$\mathcal{MOD}^{\cup}$	$2^{\mathcal{DO}}$	∘	$\mathcal{MOD}^{\text{defpar}}$	...	?
$\mathcal{MOD}^{\text{mik}}$	(intern)	–		$2^{\mathcal{E}\times\mathcal{OR}}$	∘	$\mathcal{MOD}^{\text{iopar}}$	...	?
$\mathcal{MOD}^{\text{in}}$	...	–		$\mathcal{DR}$	∘	$(\mathcal{MOD}^{\text{out}})$	–	/
$\mathcal{MOD}^{\text{attr}}$	$\widetilde{2^{\mathcal{K}}}$	+	$\mathcal{MOD}^{\text{split}}$	...	–	$\mathcal{MOD}^{\text{save}}$	(intern)	/
	$\mathcal{R}$	–	$\mathcal{MOD}^{\text{merge}}$	...	–	$\mathcal{MOD}^{\text{vis}}$	(intern)	/
	$\mathcal{M}$	∘	$\mathcal{MOD}^{\text{role}}$	...	–	$\mathcal{MOD}^{\text{route}}$	$\mathcal{M}$	+
$\mathcal{MOD}^{\text{abl}}$	$\widetilde{2^{\mathcal{K}}}$	+	$\mathcal{MOD}^{\text{cell}}$	$\mathcal{M}$	+		$2^{\mathcal{OR}}$	∘
	$\widetilde{2^{\mathcal{E}\times\mathcal{OR}}}$	∘	$\mathcal{MOD}^{\text{aggr}}$	$\widetilde{2^{\mathcal{K}}}$	+	$\mathcal{MOD}^{\text{if}}$	$\mathbb{B}$	∘
	$\mathbb{N}$	–		$\widetilde{2^{\mathcal{E}\times\mathcal{OR}}}$	∘	$\mathcal{MOD}^{\text{loop}}$	...	∘
$\mathcal{MOD}^{\text{ablv}}$	$2^{\mathcal{E}\times\mathcal{OR}}$	∘		$\mathcal{M}$	∘	$\mathcal{MOD}^{\text{stop}}$	$\mathbb{B}$	∘
$\mathcal{MOD}^{\text{sel}}$	$2^{\mathcal{M}}$	+		$\mathbb{N}$	–	$\mathcal{MOD}^{\text{sub}}$	...	?
	$2^{\mathcal{OR}}$	∘	$\mathcal{MOD}^{\text{join}}$	$\mathcal{M}$	+			

Tabelle 6.1: Grobe Schätzung der Änderungshäufigkeiten von Modulparametern

Analysegraphen so zu verallgemeinern, daß sie zu erwartende ähnliche Anfragen bereits integrieren, bzw. um optimierende Modifikationen nur dann durchzuführen, wenn sie keinen ähnlichen Folgeanfragen (und deren Beantwortung aus dem nicht optimierten Analysegraphen) zuwiderlaufen.

Die im folgenden vorgestellten, teilweise konkurrierenden Alternativen zur Optimierung von Analysegraphen sind somit vor dem Hintergrund der oben angestellten Grundüberlegungen zur „vorausschauenden“ Optimierung zu bewerten und einzusetzen. Die in Abb. 6.1 integrierte, über einen Regeleditor verwaltete Regelbasis enthält entsprechende Heuristiken, auf deren Grundlage der *Optimierer* einen zur Ausführung übergebenen Analysegraphen bzw. eine bereits zu diesem erstellte optimierte Variante umformt und an den Scheduler zur Verarbeitung weiterleitet. Man beachte, daß der Anwender bestimmte Regelgruppen oder sogar die gesamte Optimierung auch deaktivieren kann, um dem System mitzuteilen, daß bestimmte Modulfolgen bewußt zur Unterstützung zukünftiger Interaktionen modelliert wurden.

Die Darstellung in den nächsten Abschnitten folgt im wesentlichen der Chronologie der Anwendung von Optimierungstechniken, wobei jedoch häufig Abhängigkeiten zwischen den einzelnen Optimierungsphasen zu berücksichtigen sind.

6.2.1 Restrukturierung von *VIOLA*–Programmen

Die in diesem Abschnitt vorgestellten Optimierungsschritte erzeugen jeweils aus einem bestehenden *VIOLA*–Programm ein neues, möglicherweise effizienter auszuführendes oder in nachfolgenden Optimierungsphasen besser zu optimierendes Programm.

Zerlegung und Duplizierung von Modulen

Vor allem, um anderen Optimierungsstrategien mehr Alternativen zu bieten, können

- Ableitungsoperationen in jeweils zwei Einzelschritte (eine Restriktion und eine implizite Aggregierung),

- Module mit paralleler Verarbeitung unabhängiger Datenports in mehrere identische Instanzen oder
- Y–förmige Teilgraphen, die — beginnend mit einer Datenquelle — in den Schenkeln weitere Datenmanagementschritte aufweisen, in zwei getrennte lineare Graphen aufgeteilt werden (siehe Abb. 6.2). Weiter unten werden wir sehen, daß es sinnvoll sein kann, die Zugriffe auf die Datenbasen jeweils mit dem nachfolgenden Datenmanagement zusammen in Verbunden zu kapseln.

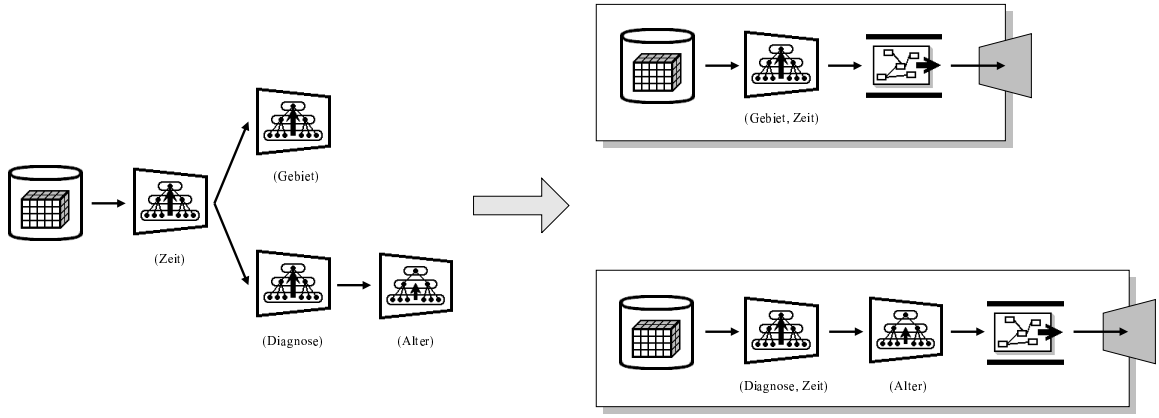

Abbildung 6.2: Duplizierung von Datenquellen und Datenmanagementoperationen

Verallgemeinerung von Datenbankanfragen

Aus mehreren Gründen kann es sinnvoll sein, durch Datenquellen repräsentierte Datenbankanfragen zu verallgemeinern (vgl. [KB96] sowie Arbeiten zur Integration materialisierter Sichten bei der Anfrageverarbeitung [CS94, SDJL96, LA97]):

- wenn im gleichen Programm eine andere Datenquelle existiert, in der ein *allgemeinerer* Datenraum bereitgestellt wird, von dem also der Inhalt der betrachteten Datenquelle durch Datenmanagementoperationen abgeleitet werden kann,
- wenn zukünftig im gleichen Sinne *allgemeinere* Datenbankzugriffe zu erwarten sind,
- auch wenn zwei Datenquellen verschiedene, nicht ineinander enthaltene Teilräume eines nur „unwesentlich“ größeren Datenraums anfordern oder
- wenn ein *allgemeinerer* Cache–Eintrag existiert.

In jedem dieser Fälle kann bzw. sollte die Datenquelle durch eine andere, allgemeinere, oder durch einen Cache–Zugriff (vgl. Abschnitt 6.2.3) sowie daran anschließend eine Folge von Managementoperationen, die den ursprünglich geforderten Datenraum generieren, ersetzt werden.

Vertauschung von Modulen

Die Möglichkeit zur Vertauschung von Analyse- und Datenmanagementmodulen spielt — analog zu Standard–DBMS — eine wichtige Rolle in *VIOLA.* Datenquellen, Datensenken, Parametermodule, Schleifen- und Abbruchmodule sind unter diesem Aspekt per Definition nicht sinnvoll zu betrachten. Bei Verzweigungsmodulen können ggf. identische Module an beiden Ausgängen vor die Verzweigung gezogen werden. Ähnliches gilt für Module als Fan–in der Eingänge von Routingmodulen, sofern ihre Operationen definitionsgemäß die summarischen Attribute der jeweiligen Datenräume nicht verändern (und somit das Routing nicht beeinflussen) können: In diesem Fall können die Module hinter das Routingmodul verschoben werden. Verbundmodule sollen stets

als Einheit betrachtet werden; eine Beurteilung ihrer Kommutativität mit anderen Modulen ist in der Regel schwierig, also werden sie hier nicht näher diskutiert.

Damit zwei Module in einer sequentiellen Abfolge[8] miteinander vertauscht werden können, müssen die jeweiligen Operationen natürlich kommutativ zueinander sein. Ohne auf alle Einzelfälle einzugehen, läßt sich hierzu feststellen, daß in der Regel zumindest dann, wenn die Mengen der jeweils von den Operationen betroffenen Attribute disjunkt sind, eine Kommutativität gegeben ist. Hinzu kommen weitere Spezialfälle, so können z. B. auch zwei Restriktionen über dem gleichen Attribut trivialerweise miteinander vertauscht werden.

Generell muß es das Ziel sein, Operationen, die

1. den betrachteten Datenraum für nachfolgende Verfahren stark eingrenzen bzw. einen deutlich kleineren Ergebnisdatenraum generieren und dazu
2. (auch im Hinblick auf einen Eintrag von Zwischenergebnissen in einen Cache) möglichst allgemein von anderen Modulen verwendbare Datenräume erzeugen; die außerdem
3. einfach zu berechnen sind und
4. deren Parameter eher selten modifiziert werden,

in einem Analysegraphen weit nach vorne zu ziehen. Hierbei ist die Gewichtung der Kriterien diskussionswürdig; gegenüber dem erstgenannten scheinen die anderen jedoch *eher* nebensächlich. Tabelle 6.2 gibt eine grobe Approximation dieser Kriterien für die *VIOLA*–Analyse- und Datenmanagementmodule.

Die Aufwandsabschätzung unterscheidet danach, ob

- mit sehr geringem (konstantem) Aufwand im Prinzip nur eine neue Sicht auf die gleichen Daten definiert wird (Restriktion, Selektion, Rollenumbenennung, Merge),
- ein gewisser, aber bei geeigneter Realisierung deutlich schwächer als linear mit der Größe der betrachteten Datenräume ansteigender Verwaltungsaufwand besteht (Vereinigung, Split) oder
- unbedingt alle Datenraumzellen durchlaufen werden müssen, woraus sich ein mindestens linear mit der Zellenzahl steigender Aufwand ergibt (Aggregierung, Maßzahlberechnung).

Eine zusätzliche Differenzierung für unterschiedlich aufwendig zu berechnende Maßzahlen wäre denkbar — prinzipiell können die jeweiligen Berechnungsfunktionen beliebig komplexe Auswertungen spezifizieren. Bei der zellenweisen und der freien Maßzahlverknüpfung fließt jede Zelle des Quelldatenraums nur in die Belegung jeweils *einer* Zielzelle ein; dies ist bei impliziter und expliziter Aggregierung nicht notwendigerweise der Fall. Auf die für diese Operationen nötige Zuordnung auseinander ableitbarer Kategorien kommen wir am Ende von Abschnitt 6.2.2 noch kurz zu sprechen. Darüber hinaus ist die Belegung von Datenwürfeln durch externe Applikationen als mit Abstand am aufwendigsten anzusehen.

Die Konstanz der Parameterwerte in Tab. 6.2 ergibt sich aus Tab. 6.1; weiterhin ist der Grad der Reduktion des Datenumfangs jeweils (für den Durchschnittsfall) relativ direkt definierbar: Vollständige Ableitung und Maßzahlverknüpfung eliminieren ganze Dimensionen des Datenraums, einige andere Operatoren selektieren Teilbereiche, und der Rest beläßt die Zellenzahl konstant. Die Nützlichkeit der Ergebnisse schließlich bevorzugt Datenräume mit einfachen bzw. wenigen Maßzahlen und Strukturen sowie möglichst umfassenden (etwa ganze Aggregierungsebenen repräsentierenden) kategoriellen Attributen. Die Angabe einer exakten Metrik ist hierbei problematisch. Allerdings erscheint eine Unterscheidung von Operationen, die die „Komplexität“ eines Datenraums

- erhöhen (Maßzahlberechnung, Merge),
- beibehalten (Rollenumbenennung, Ableitung, Vereinigung) bzw.
- verringern (vollständige Ableitung, Selektion, Split),

möglich.

[8] Unter bestimmten Umständen können auch Vertauschungen über Verzweigungen im Analysegraphen erfolgen. Derartige Fälle betrachten wir hier der Einfachheit halber nicht genauer.

Modul	*Daten-reduktion*	*Ergebnis–Nützlichkeit*	*Einfache Berechnung*	*Parameter–Konstanz*	**Modul**	*Daten-reduktion*	*Ergebnis–Nützlichkeit*	*Einfache Berechnung*	*Parameter–Konstanz*
$\mathcal{MOD}^{\text{abl}}$	o	o	–	o	$\mathcal{MOD}^{\text{merge}}$	–	–	+	+
(Restriktion)	o	o	+	–	$\mathcal{MOD}^{\text{role}}$	–	o	+	+
$\mathcal{MOD}^{\text{ablv}}$	+	+	–	o	$\mathcal{MOD}^{\text{cell}}$	–	–	–	–
$\mathcal{MOD}^{\text{sel}}$	o	+	+	–	$\mathcal{MOD}^{\text{aggr}}$	o	–	–	–
$\mathcal{MOD}^{\text{U}}$	–	o	o	o	$\mathcal{MOD}^{\text{join}}$	+	–	–	–
$\mathcal{MOD}^{\text{split}}$	–	+	o	+					

Tabelle 6.2: Für Vertauschungsstrategien relevante Charakteristika von *VIOLA*–Modulen

Insgesamt bieten sich — bei Übergewichtung des Kriteriums der Daten–Reduktion — vollständige Ableitungen, Selektionen und auch Restriktionen zum Verschieben zu den Datenquellen an. Datenanalysefunktionen und evtl. auch Vereinigungen sollten eher spät angewendet werden, wobei die freie Maßzahlverknüpfung aufgrund ihres stark „datenreduzierenden“ Effektes trotz ansonsten negativer Einstufung auch für eine vorgezogene Ausführung in Frage kommen kann.

Gruppierung von Modulen

Zur Vorbereitung auf ein nachfolgendes Zusammenfassen sollten gleichartige Operationen in sequentieller Abfolge gruppiert werden.

Zusammenfassung von Modulen

Durch die „Vereinigung“ von zwei oder mehreren Modulen zu einem neuen können *VIOLA*–Programme vereinfacht werden:

- Die Möglichkeit, gleichartige Module im Fan–in oder Fan–out von Kontrollmodulen zusammenzufassen und mit dem Kontrollmodul zu vertauschen, wurde bereits angesprochen.
- Mehrere aufeinanderfolgende vollständige Ableitungen können zu einem Ableitungsmodul unter Vereinigung der Parametermengen zusammengefaßt werden.[9]
- In bestimmten Fällen, die von der jeweils aktuellen Modulparametrisierung abhängen (und damit oftmals nur bis zur nächsten Benutzerinteraktion bestehen), könnten auch andere gleichartige Module zu einem vereinigt werden. In der Regel entstünden hierbei jedoch Operationen, die nicht mehr durch *VIOLA*–Module abgedeckt werden können — hierauf kommen wir im nächsten Unterabschnitt (Verbundbildung) sowie in Abschnitt 6.2.2 nochmals zurück.
- Läßt sich im Falle zweier aufeinanderfolgender Analysemodule die vom zweiten Modul ermittelte Maßzahl bereits aus den Eingangsdaten des ersten ermitteln, kann dieses eliminiert werden.
- Sofern es nicht einer Kapselung von Teilprogrammen in Verbunden (s. u.) zuwiderläuft, können identische Teilgraphen, speziell auch gleich parametrisierte Datenquellmodule zu einem zusammengefaßt werden.

[9] Unter Umständen kann es hierzu nötig werden, ein zusätzliches Parameterdefinitionsmodul, das allein die Vereinigung der Parameterwerte realisiert, einzuführen.

Theoretisch denkbar wäre es auch, Programme durch die Elimination wirkungsloser oder nicht anwendbarer Module zu vereinfachen. Jedoch spricht in derartigen Fällen vieles dafür, daß die Module in zukünftigen Analyseschritten aufgrund dann geänderter Parameter wieder relevant werden. Somit wird eine automatische Löschung von Modulen nicht vorgesehen.

Bildung von als Ganzes zu verarbeitenden Verbunden

In Vorbereitung auf die physische Optimierung (siehe Abschnitt 6.2.2) können Teilgraphen, die als Einheit und somit ohne die explizite Erzeugung von Zwischenergebnissen verarbeitet werden sollen, in Verbundmodulen gekapselt werden. Diese Möglichkeit ist vor allem relevant für

- Folgen von Ableitungsoperationen,
- Folgen von Selektionen,
- Folgen von Vereinigungsoperationen und
- Datenquellen mit nachfolgenden Folgen von Datenmanagementoperationen.

In den drei erstgenannten Fällen wird durch die Kapselung eine Ersetzung durch eine *MADEIRA*–basierte Operation, die nicht in *VIOLA* spezifizierbar ist, vorbereitet. So erfordert z. B. die Selektionsoperation, die man als Ergebnis der Verknüpfung von zwei *VIOLA*–Selektionen erhält, als Selektionsparameter wieder — wie in *MADEIRA*— Mengen summarischer Attribute und kommt nicht mit Maßzahlen und Objektmengen, wie sie von der *VIOLA*–Selektion genutzt werden, aus. Der vierte Fall bietet einen Ansatz, nicht direkt die im Datenbanksystem abgelegten Datenbasen, sondern — unter Einbeziehung von Datenmanagementoperationen wie Restriktion und Ableitung — nur Ausschnitte von diesen anzufragen. Somit werden die als Ergebnis der „Anfrage“–Verbunde im Hauptspeicher zu haltenden und interaktiv bearbeitbaren Datenwürfel gegenüber den ursprünglichen Basisdatenräumen deutlich kleiner. Gerade für den Zugriff auf Mikrodatenbasen typischen Umfangs ist diese Möglichkeit unverzichtbar, da bereits mit wenigen Basisdaten–Attributen Datenwürfel von einer Größe, die (ohne Komprimierung) keine vollständige Materialisierung mehr zuließe, aufgespannt werden.

Von der Funktionalität der multidimensionalen Datenbankschnittstelle hängt es ab, welche Operationen mit dem Datenquellen–Zugriff zusammengefaßt werden können. Unter Umständen ist es sogar möglich, transparent für den Benutzer (vgl. [Ruf97]) die Ausführung bestimmter Analysefunktionen auf die Datenbankschnittstelle zu verlagern.

Zu beachten ist, daß durch die Zusammenfassung von Modulen zwar eine effizientere Verarbeitung des aktuellen Programms ermöglicht wird, infolge von nachträglichen Parameteränderungen und Zugriffen auf Zwischenergebnisse innerhalb des neu geschaffenen Verbundes jedoch später evtl. mehr (Teil–)Berechnungen wiederholt werden müssen, als wenn alle Schritte separat durchgeführt worden wären. Dies kann im letztgenannten Fall bis zu neuen, aufwendigen Zugriffen auf externe Datenbestände führen. Sinnvoll ist es weiterhin, nur solche Operationen in den Datenbankzugriff zu integrieren, die den erhaltenen Datenraum nicht zu sehr spezialisieren, um ihn etwa über den Cache auch für andere Analysen nützlich zu machen. Vor diesem Hintergrund bieten sich vor allem vollständige Ableitungen zur Zusammenfassung an.

6.2.2 Physische Optimierung

Im Bereich der „Low–Level–Optimierung“ werden im folgenden die Einführung zusätzlicher „physischer“ Operatoren sowie das Handling der Belegung von Datenräumen diskutiert. Abschließend folgt noch eine Bemerkung zur effizienten Ableitung von Datenräumen und zur Verwaltung von Berechnungs- und Aggregierungsfunktionen.

Definition zusammenfassender Operationen

Im vorangegangenen Abschnitt wurde bereits mehrfach die Möglichkeit zur Einführung neuer Typen von Operationen (sozusagen interner „Hilfsmodule“) in *VIOLA*–Programme angesprochen. Diese fassen — weiterhin

auf der Grundlage von *MADEIRA* — Verarbeitungsverfahren von *VIOLA*–Modulen zu insgesamt effizienter zu berechnenden Operationen zusammen. Gekennzeichnet wurden derartig zu vereinigende Teilgraphen durch die Bildung von Verbunden im Rahmen der Programmrestrukturierung. Natürlich kann auch der Anwender explizit entsprechende Verbundmodule definieren und somit dem Optimierer durch den „Verzicht“ auf die Vorhaltung von Zwischenergebnissen eine Hilfestellung leisten.

Einige vergleichbare Ansätze in der Literatur fassen stets das gesamte betrachtete Datenflußprogramm (sofern alle einzelnen Analyseschritte auch vom zugrundeliegenden DBMS unterstützt werden) bzw. die zur Belegung einer Datensenke nötigen Berechnungen in ein an das Datenbanksystem zu richtendes Anfragestatement zusammen [DRR+96, SSW96]. Etwas differenzierter ist die Betrachtungsweise im *Tioga*–System [SCN+93]; Detailinformationen zum Abwägen zwischen Einmalberechnung und interaktiver Manipulation hinsichtlich höherer Effizienz liegen hierzu jedoch leider nicht vor.

Effiziente Parametermanipulation zur Realisierung interaktiver Graphiken

Gerade zur *direkten* Manipulation interaktiver Graphiken sind spezielle Überlegungen zur effizienten Datenpropagierung erforderlich. Im Normalfall erzeugen die Verarbeitungsfunktionen von Modulen in jedem Schritt neue Datenraum–Objekte, die im Cache abgelegt und von Modulen im Fan–out der jeweiligen Ports referenziert werden. Dieses Vorgehen kann bei der Realisierung von Echtzeit–Interaktionen (z. B. der Parametrisierung von Modulen durch Schieberegler oder der Umsetzung des Linking von Graphiken) zu ineffizient sein. Auf Spezifikation des Benutzers oder auch automatisch für die jeweiligen kritischen Module sollten statt dessen in bestimmten Fällen Datenräume vor einer Neuberechnung nicht gelöscht, sondern lediglich ihre Inhalte (neben den Würfelzellen evtl. auch Metadaten) neu belegt werden. Hierzu ist ggf. der genutzte Speicherbereich für Modulergebnisse von vornherein groß genug für die Resultate aller eventuellen Parametrisierungen anzulegen. Als Richtlinie kann hierbei dienen, daß — abgesehen von Analyseschritten, die gleichzeitig mehrere (zusätzliche) Maßzahlen berechnen, sowie von Sonderfällen der Ableitung vieler verschieden granularer Zielkategorien — ein Ergebnisdatenwürfel nicht größer als der jeweilige Quellwürfel ist. Es wird durch die Kommunikation über feste Speicherbereiche also eine höhere Laufzeiteffizienz durch einen größeren Speicherbedarf erkauft. Außerdem können die jeweiligen Datenräume aufgrund ihrer dynamischen Änderbarkeit nicht über den Cache anderen Analysen zugänglich gemacht werden.

Analog ist es in der Regel auch sinnvoller, erstellte Graphiken nicht nach jeder Änderung neu zu generieren, sondern lediglich ihre Parameter und Wertebelegungen zu modifizieren.

Referenzierung unveränderter Datenräume

Falls ein Modul seine Eingangsdaten unverändert läßt, kann der Ergebnisdatenraum als Referenz auf die Quelldaten, jedoch mit der Möglichkeit zur Spezifikation abweichender (textueller) Metadaten, realisiert werden.

Lazy Evaluation und dynamische Sichten

Der Verarbeitungsalgorithmus von *VIOLA* ist bereits darauf ausgerichtet, jeweils nur die für eine Anforderung (etwa eine Visualisierung) nötigen Berechnungen durchzuführen. Theoretisch ließe sich dieser Ansatz noch dahingehend fortführen, daß jeweils auch nur die benötigten Würfel*zellen* auf Anforderung ermittelt werden („Lazy Evaluation“). Nicht praktikabel wäre es, für jede Werteabfrage (etwa in einer Visualisierung) eine entsprechende Anforderung durch den Datenflußgraphen zu propagieren. Statt dessen könnten die entsprechenden Informationen über Restriktionen und Selektionen, die ja gerade zur Folge haben, daß nicht immer alle Würfelinhalte auch gebraucht werden, jeweils zusammen mit der bisher realisierten datenraumbezogenen Anforderung in Richtung der Datenquellen gesendet werden. Somit stünde später in der datengetriebenen Propagierungsphase das Wissen zur Verfügung, um nur die nötigen Teildatenräume zu belegen. Diese Vorgehensweise realisiert praktisch aber auch gerade die Vertauschung von Modulen im Rahmen der Anfragerestrukturierung (Abschnitt 6.2.1), so daß wir hier kein zusätzliches Konzept benötigen. Statt dessen werden lediglich

Restriktionen und Selektionen bzw. die hierdurch erzeugten Datenräume als dynamische (virtuelle, d. h. nicht materialisierte) Sichten auf die Quelldatenräume implementiert.

In ähnlicher Weise wird die identische Duplizierung von Maßzahlwerten über kategorielle Attribute, die für das entsprechende summarische Attribut nicht relevant sind, umgesetzt: Es werden intern lediglich Teilwürfel mit den relevanten Dimensionen verwaltet. Beim Datenzugriff können auch auf andere Dimensionen bezogene Koordinaten angegeben werden, die jedoch zur Identifikation einer Datenraumzelle nicht herangezogen werden. Diese Vorgehensweise spielt eine wichtige Rolle bei der flexiblen und benutzungsfreundlichen, aber zugleich effizienten Realisierung des Vereinigungsoperators. Vor dem gleichen Hintergrund ist schließlich auch die Verwaltung von mit gleichartigen Nullwerten gefüllten Teilwürfeln nur anhand der Beschreibung der jeweiligen kategoriellen Ausprägungen (aber ohne Materialisierung der Nullwert–Würfel) zu sehen. Diese Nullwerte entstehen oftmals ebenfalls bei der Vereinigung von nicht exakt zueinander „passenden“ Datenräumen.

Realisierung von Ableitungs- und Analyseoperationen

Im Rahmen einer explorativen Navigation durch den Datenbestand ist die Ableitung neben der Vereinigung von Datenräumen wahrscheinlich der am häufigsten genutzte *MADEIRA*–Operator und somit möglichst effizient zu realisieren. Problematisch ist hierbei die Erkennung der Ableitbarkeit von Kategorien aus einer gegebenen Kategorienmenge. Es wäre zu evaluieren, ob dies effizient, ohne allzu großen Speicheroverhead umgesetzt werden kann, indem jedes kategorielle Attribut bereits eine Zuordnungs- (Hash-)tabelle mit den aus seinen Kategorien ableitbaren Kategorien mit sich trägt. Andernfalls müßte, z. B. unter Einführung einer mit den Hierarchiestrukturen kompatiblen Schlüsselordnung, eine effiziente Suche in Kategorienhierarchien (mit logarithmischem Aufwand) realisiert werden.

Schließlich soll noch auf einen Aspekt hingewiesen werden, über den in *MADEIRA* bisher weitgehend hinweggesehen wurde: Viele mengenwertige Komponenten der *MADEIRA*–Strukturen können u. U. sehr viele Elemente enthalten. Hierzu zählen vor allem die Spezifikationen der Berechnungs- und Aggregierungsmöglichkeiten von Maßzahlen und summarischen Attributen. Wie bereits in Abschnitt 4.2.5 motiviert, können hier prädikatbasierte Beschreibungen Speicherung und Zugriff deutlich vereinfachen. Zum Beispiel lassen sich Mengen von Tripeln von Eigenschaften zu rollenbehafteten Objektmengen, über die mittels einer bestimmten Funktion aggregiert werden kann, in der Regel durch Angabe einer gemeinsamen Domäne (evtl. mit einzelnen Ausnahme–(e, O, r)–Tripeln) spezifizieren. Ähnlich können Kombinationen von Quellmaßzahlen zur Berechnung einer Maßzahl durch Angaben zu ihren Metadaten (vgl. Abschnitt 4.4.2) charakterisiert werden.

6.2.3 Caching von Datenräumen

Eine erste Möglichkeit, häufig benötigte Daten in *VIOLA* möglichst effizient zugreifbar zu machen, stellt die Definition von Datenbasen als voraggregierte oder einschränkende (materialisierte) Sichten auf andere Datenbasen dar, wie sie bereits in Abschnitt 6.1.1 angesprochen wurde (vgl. hierzu auch Abschnitt 2.3.4). Neben dieser durch den Anwender bzw. Systemadministrator explizit spezifizierten Zugriffsbeschleunigung sichert ein Cache in dynamischer Weise die schnelle Bereitstellung häufig angefragter Datenräume (vgl. Standardliteratur zur Implementierung von Datenbanksystemen, etwa [LS87, Mit95]). Vorarbeiten zu dieser Thematik im Rahmen von *VIOLA* sind in [Ahl98, Gru97] beschrieben; ein ähnlicher Ansatz des Caching in einer datenflußbasierten Datenanalyse–Umgebung findet sich im *IBM Data Explorer* [LAC+92], wo dem Systembenutzer sogar ein expliziter Zugriff auf Cache–Einträge gestattet wird. Folgende konkreten Anforderungen an einen Cache ergeben sich in *VIOLA*:

- Der Cache sollte explizit durch ausgewählte zentrale Voraggregierungen belegt werden können, so daß Teilgraphen zum Zugriff auf bestimmte Datenquellen mit Sequenzen nachfolgender Datenmanagementoperationen automatisch durch effizienter zugreifbare Daten ersetzbar sind. Die Belegung kann durch einen Administrator erfolgen oder auch, indem ein Systemanwender explizit häufig genutzte Daten durch entsprechende *VIOLA*–(Hilfs–)Programme spezifiziert.

- Damit Berechnungen nicht mehrfach durchgeführt werden, sollten bestimmte Zwischenergebnisse innerhalb eines Programms sowie auch programm- und — mittels persistenter Ablage häufig benötigter, evtl. aufwendig zu berechnender Datenräume — sitzungsübergreifend mehrfach nutzbar gemacht werden. Entsprechende Cache–Einträge sind nicht mehr mit dem erzeugenden Programm gekoppelt, d. h. Programm–Manipulationen erzeugen ggf. neue Cache–Elemente.

- Schon für jedes Modul an sich sollen in *VIOLA* prinzipiell alle Portinhalte stets direkt verfügbar sein, so daß nach Parameteränderungen sofort eine abermalige Nutzung von Zwischenergebnissen vorangehender Module möglich und nicht, wie in vielen anderen datenflußbasierten Analyseumgebungen (siehe Abschnitt 3.4 sowie Tab. 5.1), der gesamte Analysegraph neu zu berechnen ist. Dies bedeutet, daß erzeugte Datenräume auch dann noch lokal aufbewahrt werden müssen, wenn der jeweilige Fan–out bedient wurde. Auch individuelle Änderungen der Metadaten eines Datenraums müssen an dieser Stelle verwaltet werden (vgl. Abschnitt 6.1.3).

- Wenn ein *VIOLA*–Programm zu groß wird, können natürlich nicht mehr alle Zwischenergebnisse hauptspeicherresident gehalten werden. Somit müssen ausgewählte Ergebnis–Teilmengen auf Sekundärspeicher in einem Cache zur späteren Nutzung ausgelagert werden können.

- Auch ein hauptspeicherbasierter, dynamischer Cache ist nötig, um etwa nach mehrfachen Parameteränderungen alte Ergebnisse verfügbar zu halten und so eine schnellere Interaktion zu ermöglichen. Man erinnere sich: Die direkte Interaktion steht bei der komfortablen Datenexploration im Vordergrund. Weiterhin können als Alternative oder Ergänzung zur Überschreibung von Datenraumzellen durch Interaktionen (siehe Abschnitt 6.2.2) auch „Belegungshistorien“ für ein Modul verwaltet werden, um einen schnelleren Wechsel zwischen historischen Ergebnissen auf Kosten höheren Speicheraufwandes zu realisieren.

Insgesamt ergibt sich also ein dreistufiges Caching: modulbezogen lokal, global hauptspeicherbasiert und global auf Sekundärspeicher.

Für jeden Berechnungsschritt, d. h. jeden Port, ist jeweils zu entscheiden, *wo* Ergebnisdatenräume abgelegt werden sollen, um einen geeigneten Trade–off zwischen Speicheraufwand und Laufzeiteffizienz (für die aktuelle Analyse, aber auch für zukünftige Analyseschritte) zu realisieren. Auch Kombinationen der drei Alternativen oder die Entscheidung dafür, einen Datenraum nach seiner Nutzung durch Folgemodule sofort zu löschen, sind möglich.

Im Zusammenhang mit dem Entwurf des *Tioga*–Systems werden in [SCN+93, WS95] Situationen bzw. Positionen eines Moduls innerhalb eines Datenflußprogramms charakterisiert, die jeweils generell für ein Caching eines Zwischenresultats sprechen. So sollten Datenräume, die Visualisierungen zugrunde liegen, unbedingt materialisiert werden, um Darstellungen flexibel erzeugen und wieder löschen zu können. Außerdem sind Module vor Verzweigungen und vor Modulen mit häufigen Parameteränderungen Kandidaten zur Materialisierung ihrer Ergebnisse. Darüber hinaus sollten Manipulations- und Zugriffshistorien sowie die weitere Struktur eines Analysegraphen in die Entscheidungsfindung einfließen (auf weitere Details wird in Veröffentlichungen zu *Tioga* leider nicht eingegangen).

Ein wichtiges Kriterium für Cache–Zugangs- und –Verdrängungsstrategien ist die Nützlichkeit eines Datenraums, also die Wahrscheinlichkeit, daß dieser selbst oder von ihm abgeleitete Datenwürfel an anderer Stelle benötigt werden (vgl. z. B. [SSV96, Gru97, Ahl98] sowie die Darstellung in Tab. 6.2). Auch ist es interessant, ob Module, die zur Erzeugung des betrachteten Datenraums beigetragen haben, noch in bestehenden *VIOLA*–Programmen existieren bzw. wie viele existierende Module auf einen Cache–Eintrag zugreifen. Weiterhin spielen natürlich auch hier, analog zum Caching in Standard–Datenbanksystemen, Größe, Berechnungskosten und aktueller Cache–Inhalt eine wichtige Rolle.[10] Zusätzlich sollte jedoch auch der Systembenutzer explizit Einfluß auf das Caching von Analyseergebnissen nehmen können.

[10] Die gleiche Problematik findet sich bei der Auswahl zu materialisierender Sichten in Data Warehouses, siehe Abschnitt 2.3.4.

Konkret kann von jedem Datenausgang in einem *VIOLA*–Programm eine Beziehung zu einem Eintrag in jedem der drei bereitgestellten Caches bestehen [Hüs99]. Ein Cache–Manager verwaltet diese Referenzen in einer auf den Datenraum–Schemata basierenden Zugriffsstruktur (vgl. auch [LA97]). Hierbei definieren die Attribute eines Datenraums eine semantische Cachebeschreibung [KB96, DFJ+96], anhand derer er identifizierbar und mit in einem Programm benötigten Zwischenergebnissen inhaltlich vergleichbar ist.

Bei der (von Datensenken zu Datenquellen propagierten) Anforderung von Daten wird jeweils geprüft, ob die zu berechnenden Datenräume bereits im Cache vorliegen, und ggf. ein Cache–Zugriff in das Programm integriert. Bei einfachen Berechnungen kann auf diese Prüfung verzichtet werden, um keinen unangemessen großen Verwaltungsoverhead zu erzeugen. Unter Umständen können Programme auch umgeformt werden (insbesondere durch das Einfügen einfacher Datenmanagementoperationen), um Cache–Inhalte nutzbar zu machen (siehe Abschnitt 6.2.1). In diesem Zusammenhang sei auch auf die in [SSW96] beschriebenen Arbeiten zur automatischen Erkennung von auseinander ableitbaren Zwischenergebnissen innerhalb eines datenflußbasierten Programms zur multidimensionalen Datenanalyse verwiesen. Von zu großen automatischen Programm–Modifikationen sollte jedoch im Hinblick auf die zukünftige Erweiterung und Manipulation eines Programms durch den Anwender Abstand genommen werden.

6.3 Ein Anwendungsbeispiel: Einsatz von *VIOLA* im Niedersächsischen Krebsregister

Um die Funktionalität von *VIOLA* zu verdeutlichen und die Praxisrelevanz der entwickelten Konzepte zu belegen, wird im folgenden anhand einiger konkreter Anwendungsszenarien ein möglicher Einsatz von *VIOLA* im Epidemiologischen Krebsregister Niedersachsen (kurz EKN) beschrieben. Die Aufgabe dieses Registers besteht (nach [HW97b, EKN00]) in der Erfassung aller Krebs–Neuerkrankungen und –Sterbefälle in Niedersachsen sowie im Aufbau hierzu geeigneter Meldewege mit dem Ziel der

- Schätzung von Inzidenz- und Mortalitätsraten,
- Beobachtung zeitlicher Trends,
- Identifikation auffälliger Teilregionen oder Zeitperioden,
- Generierung von Hypothesen zur Krebsätiologie,
- Bereitstellung einer Datengrundlage für epidemiologische Studien,
- Unterstützung gezielter Untersuchungen, z. B. in Arbeits- und Ernährungsmedizin oder Umwelttoxikologie, sowie der
- Ermittlung von Basisdaten für die Planung von Einrichtungen der Gesundheitsversorgung.

Aus dieser Aufstellung lassen sich die vielschichtigen Anforderungen an ein adäquates Auswertungssystem erahnen. Gerade auch der Bedarf nach unterschiedlichen Benutzungsschnittstellen für die Bearbeitung differenzierter Aufgabenstellungen durch verschiedene Benutzergruppen, wie er auch als Motivation für die in Abschnitt 6.1 vorgeschlagene *VIOLA*–Architektur herangezogen wurde, wird besonders deutlich. Im folgenden zeigen wir, wie in diesem Sinne sowohl die Durchführung von Routineauswertungen (Abschnitt 6.3.2) sowie die automatisierte Berichterstellung (Abschnitt 6.3.3) als auch die interaktive Datenexploration im Registerbestand (Abschnitt 6.3.4) durch *VIOLA* unterstützt werden können. Zunächst aber werden in Abschnitt 6.3.1 die auszuwertenden Datenbestände und Analyseverfahren vorgestellt und mittels *MADEIRA* modelliert.[11] Art und Vielfalt von Daten sowie Maßzahlen sind hierbei durchaus mit denen typischer Data Warehouses vergleichbar, so daß eine Anwendung von *VIOLA* auch in dieser Domäne konzeptionell problemlos erscheint.

[11] Wir beziehen uns jeweils auf den aktuellen Stand der Arbeiten im EKN und integrieren stellenweise bereits einzelne für die Zukunft geplante Erweiterungen.

6.3.1 Ausprägungen der *MADEIRA*–Basisentitätsmengen im EKN

In der Krebsregistrierung werden Personen und Tumoren als grundlegende Objektmengen betrachtet. Im EKN derzeit interessierende Eigenschaften von Personen sind Alter und Geschlecht sowie Geburts-, Wohn- und Sterbeort. Weiterhin repräsentiert ein Registrierungsjahr die Zuordnung einer Person zu einer Teilpopulation in einem bestimmten Erhebungsjahr. Für die Zwecke des EKN ist es ausreichend, das Registrierungsjahr als einwertige Eigenschaft und somit eine reale Person über die Jahre in Form einer Menge von Personenobjekten, deren (gleiche) Identität unerheblich ist, zu modellieren. Den Tumoren sind folgende Eigenschaften zugeordnet:

- Geschlecht, Erkrankungs- und Sterbealter der jeweiligen Person,
- Zeitpunkt von Erkrankung und Tod,
- Wohnort zum Zeitpunkt der Erkrankung sowie des Versterbens,
- Diagnose nach der International Classification of Diseases (ICD), neunte oder zehnte Auflage,
- Meldewege, über die der Tumor registriert wurde,
- bis zum Zeitpunkt der Erkrankung am längsten ausgeübte Tätigkeit[12] sowie
- eine Menge weiterer medizinischer Parameter, die aber zu den meisten Standardauswertungen nicht herangezogen werden und hier nicht näher interessieren sollen.

Wie bereits mehrfach erwähnt, werden als Rollen von Personen und Tumoren Studien- und Standardpopulation unterschieden, wobei letztere nach internem Standard (ganz Niedersachsen), externem Standard (derzeit meist Hamburg oder das Saarland als ein Bundesland mit relativ vollständiger Erhebung der Krebsinzidenz) und künstlichem Standard (eine für die BRD von 1987, Europa bzw. die Welt repräsentative Alters- und Geschlechtsverteilung) differenziert werden. Zusätzlich werden noch Rollen „Vergleich“ und „Simulation“ zur Gegenüberstellung bzw. Simulation von Maßzahlwerten benötigt, wie wir noch weiter unten an einigen Beispielen sehen werden.

Gemäß der obigen Eigenschaftsmenge sind im EKN Dimensionen auf den Domänen

- *Geschlecht* (trivial),
- *Alter* (neben einer Wurzelebene sowie einer Aggregierungsebene mit einem Gesamtwert und einer NULL–Kategorie gibt hier es noch eine Ebene mit jahresbezogenen Altersangaben sowie Fünf–Jahres–Gruppierungen bis 75, 80 bzw. 85, denen jeweils eine zusammenfassende Kategorie für alle älteren Altersgruppen hinzugefügt ist),
- *Zeit* (Monate, Quartale, Halbjahre, Jahre, 2–, 3–, 5– und 10–Jahreszeiträume sowie wiederum eine Wurzelebene und eine Gesamtebene mit NULL–Kategorie),
- *Region* (siehe Abb. 6.3, hiermit werden auch Standardbevölkerungen als spezielle, künstliche Regionen codiert; weiterhin sind neben unterschiedlich granularen Angaben zu Niedersachsen zur Modellierung von Vergleichsdaten auch Bezüge auf das Saarland und auf Teilregionen von Hamburg möglich),
- *Diagnose* (drei- und vierstellige Codes nach verschiedenen ICD–Versionen, siehe Abb. 4.3) und
- *Meldeweg* (eine Dimension mit mengenwertigen Aggregierungsebenen wie in Abb. 4.5)

sowie weitere einfache Dimensionen für Tätigkeitsangaben und medizinische Parameter definiert. Außerdem werden wir zur Modellierung von Fallzahlsimulationen als Grundlage einiger raumbezogener Clusteranalyseverfahren noch eine Domäne benötigen, die die Zuordnung zu Simulationsläufen spezifiziert. Entsprechend erhalten Tumoren und Personen noch eine künstliche „Simulationseigenschaft“, die für nicht simulierte Daten eine spezielle Ausprägung aufweist.

[12] Sicherlich wäre auch die Tätigkeit ein Kandidat für eine mengenwertige Eigenschaft von Personen. Auf eine entsprechende Nutzung von Berufsangaben wird jedoch derzeit im EKN verzichtet.

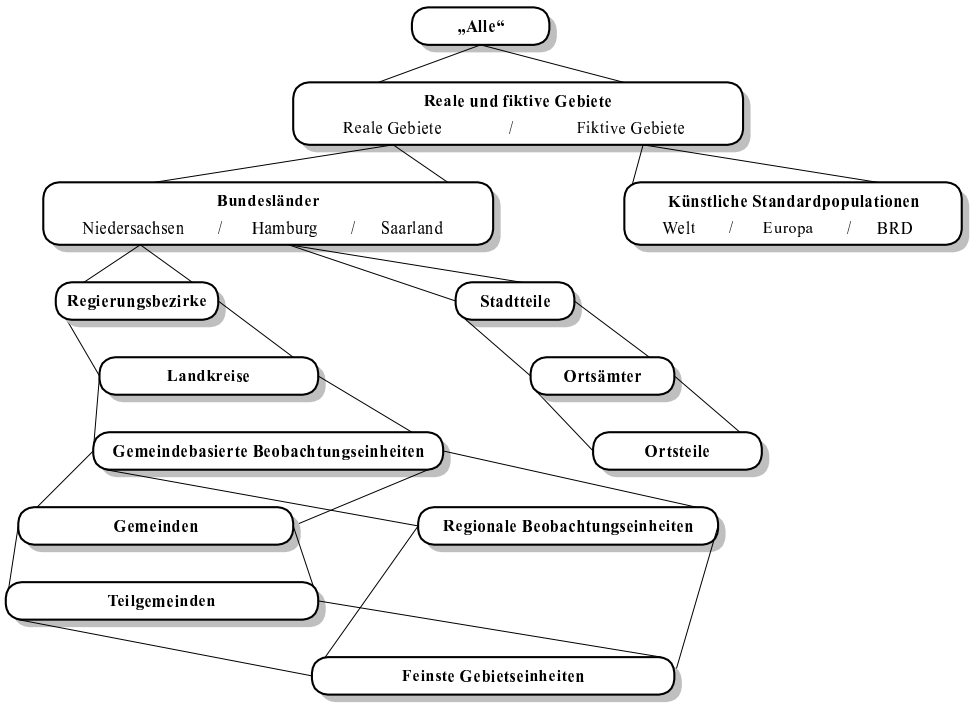

Abbildung 6.3: Die Aggregierungshierarchie auf der Geo–Dimension im Datenbestand des EKN

δ–Kategorien werden verbreitet (vor allem in der Gebietsdimension für Angaben oberhalb der feinsten Ebene sowie zur Kennzeichnung von Jahresangaben ohne Monat), jedoch nicht in allen Dimensionen verwendet. Auch eine NULL–Kategorie ist nicht in jedem Fall erforderlich; so dürfen etwa Fälle mit unbekanntem Wohnort gar nicht im Register gespeichert werden. Die verwendeten Codierungsschemata für Diagnosen (nach ICD) sowie andere medizinische Angaben enthalten zudem in vielen Fällen bereits entsprechende Codes für „*unbekannt*“ bzw. „ . . . , *ohne nähere Angabe*“, die somit nahtlos in das *MADEIRA*–Konzept integrierbar sind.

Der Datenbestand des Registers besteht aus

- Mikrodaten zur niedersächsischen Inzidenz und Mortalität,
- Mikrodaten zur Hamburger Inzidenz und Mortalität,
- Makrodaten zur niedersächsischen und Hamburger Bevölkerung (nach Geschlecht, Region, Alter und Jahr),
- Makrodaten zu Bevölkerung, Inzidenz und Mortalität des Saarlandes (nach Geschlecht, Alter, Jahr und Diagnose) sowie
- Makrodaten zu den künstlichen Standardbevölkerungen (nach Geschlecht, Alter und Zeitpunkt der Definition).

Hieraus ergeben sich fast unmittelbar die durch *VIOLA* angebotenen Datenquellen unter Verwendung der entsprechenden Rollen. Unter den niedersächsischen Falldaten sind lediglich noch verschiedene Erhebungszeitpunkte zu unterscheiden, für die separate Datenbank–Schnappschüsse abgelegt wurden. Darüber hinaus sind einige regionale Hintergrundinformationen (als Metadaten der Gebietskategorien) verfügbar, auf die an dieser Stelle jedoch nicht näher eingegangen werden soll. Weiterhin wird eine Datenquelle definiert, die beliebige simulierte Maßzahlen über Mengen von Gebieten liefern kann. Die Datenräume mit simulierten Daten haben also jeweils zwei kategorielle Attribute (Gebiet und Simulationslauf); die Simulationsrolle ersetzt hier die Studienpopulation.

Die Extensionen $\text{ext}(O,r)$ bzw. $\text{ext}(O)$ von durch Datenräume beschriebenen Objektmengen (vgl. Abschnitt 4.2) lassen sich einfach als Identifikatoren implementieren, die zwischen verschiedenen Erhebungszeitpunkten der Falldaten differenzieren, um eine fälschliche Verknüpfung zu verhindern. Simulierte Daten erhalten eine zwischen allen erzeugten Datenräumen verschiedene Kennung zur Angabe der Objektmengen–Extension. Dies ist erforderlich, da all diese Datenräume unterschiedliche Daten zu den gleichen (durch Gebiets- und Simulationskategorien definierten) Teilpopulationen enthalten.

Neben durch Kategorien und ihre Metadaten definierten Größen werden im wesentlichen folgende Maßzahlen bei der Datenanalyse im EKN verarbeitet bzw. erzeugt, wobei jeweils auch Angaben zu Berechnungsvorschriften und Aggregierungsfunktionen (stets auf disjunkten Teilpopulationen) gemacht werden[13]:

- *Bevölkerungszahlen* geben die mittlere Bevölkerung in einem Zeitintervall an. Durch Multiplikation mit der Länge des jeweiligen Zeitraums erhält man hieraus *Personenjahre n*. Umgekehrt ergibt die Division von Personenjahren durch Zeitintervalle wiederum mittlere Bevölkerungen.[14] Personenjahre sind über alle Domänen mittels Summation aggregierbar, Bevölkerungszahlen dagegen nicht über die Zeit (vgl. Abschnitt 4.2.5). Aus diesem Grund und weil sich die meisten der weiteren mit Bevölkerungsbezug definierten Maßzahlen stets auf ein Jahr beziehen, bietet sich zu deren Berechnung die Verwendung von Personenjahren anstelle von Bevölkerungszahlen an.

- (Tumorbezogene) *Fallzahlen d* werden durch Zählung von Einzelfällen in Mikrodaten bestimmt und sind analog zu Personenjahren beliebig summierbar. (Soweit nicht anders erwähnt, sind die weiteren hier genannten Maßzahlen demgegenüber gar nicht aggregierbar.) Inzidenz- und Mortalitätsfallzahlen sind Spezialisierungen von Fallzahlen.

- *Rohe Raten* sind definiert als $r = d/n \cdot 100.000$. Hier und auch bei allen nachfolgenden auf der Basis von Fallzahlen definierten Größen kann ebenfalls zwischen Inzidenz und Mortalität unterschieden werden, was natürlich auch in den zugehörigen Berechnungsfunktionen bei den Quellmaßzahlen zu berücksichtigen ist.

- *Direkt standardisierte Raten* r_{dir} sind Summen über mit den jeweiligen Populationsgrößen einer Standardpopulation gewichtete Raten zu verschiedenen, disjunkten Teilgruppen der Studienpopulation. Unter Verwendung von Großbuchstaben für Maßzahlen zu einer Standardpopulation ergibt sich also eine Berechnungsformel wie $r_{\text{dir}} = (\sum_i r_i N_i)/(\sum_i N_i)$. Standardisiert wird typischerweise über Alter und/oder Geschlecht. Allgemein ergeben sich verschiedene Spezialisierungen standardisierter Raten je nach den zur Standardisierung verwendeten Domänen, was wiederum auch für einige der nachfolgenden Maßzahlen analog gilt. Kehrt man die Rollen von Standard- und Studienpopulation um, erhält man (für die Studienpopulation) *erwartete Raten*. Die Oberklasse von beiden Varianten soll einfach als *standardisierte Rate* bezeichnet werden.

- Die *kumulative Rate* ist analog zur direkt über das Alter standardisierten Rate definiert, nur daß anstelle des Anteils einer Altersgruppe an der gesamten Standardpopulation mit der Länge des jeweiligen Altersintervalls gewichtet wird.

- Das *relative Risiko* ist der Quotient der rohen Raten von Studien- und Standardpopulation.

- Die *Standardized (Mortality/Incidence) Ratio (SMR/SIR)* ist der Quotient aus roher Rate der Studienpopulation und erwarteter Rate.

- Die *Cumulative (Mortality/Incidence) Figure (CMF/CIF)* ist der Quotient aus direkt standardisierter Rate und roher Rate der Standardpopulation.

[13] Auf Hintergründe zum Sinn und zur Verwendung einzelner Maße gehen wir hier nicht ein. Derartige Informationen sind in der gängigen Literatur zur Epidemiologie, etwa [EBR94, Rot86], oder auch [HW97b, RWW99] nachzulesen.

[14] Entsprechende Umkehrfunktionen lassen sich natürlich auch für die meisten übrigen Maßzahlen definieren. Da diese jedoch dort in der Regel nicht benötigt werden, brauchen diese Zusammenhänge nicht modelliert zu werden.

- Als *Diagnoseanteil* wird der Quotient zweier Fallzahlen mit den Rollen „Studienpopulation“ und „Vergleich“ (einer Gesamtfallzahl entsprechend) bezeichnet. Der Diagnoseanteil ist über Diagnosen zur Studienpopulation, aber *nicht* über Diagnosen zum Vergleich per Summation aggregierbar. Wir kommen weiter unten noch auf die Problematik zurück, inwiefern festgestellt bzw. spezifiziert werden kann, daß die beiden Quellmaße Einzeldiagnosen bzw. übergeordnete Gesamtsummen über mehrere (alle) Diagnosen beschreiben. Entsprechende Maßzahlen zur Beschreibung von Anteilen über andere Domänen sind prinzipiell natürlich analog als Spezialisierungen einer allgemeinen Anteils–Maßzahl definierbar, hier aber nicht relevant.

- Der *Median* interessiert im EKN–Kontext nur als medianes (Erkrankungs– oder Sterbe–)Alter, wird also in diesem Fall durch Feststellung des mittleren Wertes über die Altersdomäne gewonnen.

- Verschiedene *(raumbezogene) Clusterindizes* aggregieren jeweils bestimmte Maßzahlen über die Gebietsdomäne und beschreiben den Grad der Unregelmäßigkeit der Fallverteilung in der Studienpopulation. Als interner Parameter ist oft eine Art der zu verwendenden Nachbarschaftsmatrix[15] auf dem Untersuchungsgebiet zu spezifizieren.

Für Maßzahlen mit nicht direkt auf Fallzahlen und Personenjahren als Basisdaten formulierten Definitionen sind als Berechnungsvorschriften jeweils mehrere Varianten vorzusehen — je nachdem, ob nur die Basisdaten oder bereits daraus abgeleitete Maßzahlen als Quellmaße vorliegen. So kann etwa die SMR neben obiger Definition auch direkt aus Fallzahlen und Personenjahren ermittelt werden.

Wie wir weiter unten sehen werden, kann es weiterhin hilfreich sein, analog zum Fall der direkt standardisierten Rate auch für Bevölkerungen, Personenjahre, Fallzahlen, rohe und kumulative Raten zwischen Maßen zur Studienpopulation (*Beobachtete Fallzahl* etc.) und solchen zur Standardpopulation (*Standardpopulationsfallzahl* etc.) zu unterscheiden. Außerdem definieren wir die Maße *Gesamtpersonenjahre*, *Gesamtbevölkerung* und *Gesamtfallzahl*, die sich einfach als Summe über Werte der jeweiligen einfachen Maßzahl (für alle Altersgruppen und/oder Geschlechter) ergeben, sowie die *Gesamtrate*, die entsprechend der rohen Rate als Quotient von Gesamtfallzahl und 100.000 Gesamtpersonenjahren definiert ist.

Varianten vieler oben angeführter Maßzahlen zur Studienpopulation berechnen zusätzlich zu sich selbst *Konfidenzintervalle* um den beobachteten Wert, die den Grad der rein zufallsbedingten Schwankung ausdrücken. Zu einem Clusterindex wird jeweils ein *Signifikanzniveau* (ein sogenannter *p–Wert*) ermittelt, das beschreibt, ob eine statistisch signifikante Abweichung von einer zufälligen Fallverteilung über das Untersuchungsgebiet vorliegt.[16] Schließlich benötigen wir in diesem Kontext noch eine Maßzahl *Signifikanz*, die aus einem Maß zur Studienpopulation, dessen Konfidenzintervall und einem Referenzwert zur Standardpopulation in fünf möglichen Ausprägungen einstuft, ob ein beobachteter Wert normal oder (unauffällig bzw. signifikant) erhöht oder erniedrigt ist.

Insgesamt ergeben sich bei dieser Art der Modellierung jeweils recht viele Untertypen einzelner Maßzahlen. In einer Implementierung dieser Strukturen läßt sich dies jedoch relativ einfach als Parametrisierung von Maßzahlen (über Fallart, Bezug zur Studien- oder Standardpopulationsrolle, zur Standardisierung/Aggregierung herangezogene Domänen sowie die Einbeziehung von Konfidenzintervall bzw. Signifikanzniveau) umsetzen.

6.3.2 Anbindung einer menübasierten graphischen Benutzungsoberfläche

Das aktuell im EKN eingesetzte epidemiologische Auswertungssystem *CARESS* (vgl. Abschnitt 3.6) bietet dem Anwender eine menübasierte Benutzungsoberfläche, über die Standardauswertungen komfortabel durchgeführt werden können [Wie99, WGG+97, RWW99]. Nach Festlegung eines Parametersatzes, der die interessierenden Dimensionsausprägungen spezifiziert, können thematische Karten, Balken- und Liniendiagramme sowie verschiedene Tabellen generiert werden. Außerdem kann die regionale Verteilung von Erkrankungsraten mit Hilfe

[15] Eine derartige Matrix gibt zu jedem Paar von Teilregionen an, ob diese als Nachbarn anzusehen sind. Eventuell sind hierbei auch (unter Berücksichtigung der Entfernung zwischen zwei Regionen) feinere Differenzierungen als nur „ja“ und „nein“ möglich.

[16] Genaugenommen definiert das Signifikanzniveau die Wahrscheinlichkeit, mit der der beobachtete Wert des Clusterindex unter einer rein zufälligen Fallverteilung aufgetreten wäre.

räumlicher Clusterindizes beurteilt werden. Ein Schwerpunkt von *CARESS* liegt auf der interaktiven Parametrisierbarkeit von Visualisierungen: In jeder Graphik und Tabelle können dynamisch darzustellende Maßzahlen, Fallart (Inzidenz oder Mortalität) und zu nutzende Standardbevölkerungen sowie weitere auswertungsspezifische Parameter ausgewählt werden, worauf sich die jeweiligen Darstellungen umgehend anpassen.

Indem nun die menübasierten Auswertungen von *CARESS* in durch *VIOLA* modellierte Datenanalyseprogramme übersetzt werden und *CARESS* als ein Client des *VIOLA*–Servers realisiert wird, bietet sich die Möglichkeit,

1. die Semantik der angebotenen Analysen genau zu definieren,
2. *CARESS* bei Bedarf auf einfache Weise um neue Verfahren zu ergänzen oder bestehende Auswertungsmöglichkeiten zu modifizieren sowie
3. auf Wunsch jeweils interaktiv die standardisierten Analysen durch einen Wechsel von der menübasierten auf die datenflußbasierte, in einem graphischen Editor vorgehaltene Ebene zu vertiefen, indem spezifische Manipulationen an einer Kopie des unterliegenden Datenflußprogramms vorgenommen werden, ohne die Funktionalität von *CARESS* generell zu modifizieren.

In Abb. 6.4 wird grob angedeutet, wie Masken und Menüs aus *CARESS* jeweils auf der Basis von Teilen eines Datenflußprogramms definiert werden können bzw. wie die Module des Datenflußprogramms durch Interaktionen auf der *CARESS*–Oberfläche parametrisiert werden. Das Fenster zur Parameterwahl entspricht einer Menge von Datenmanagementoperationen, das Maßzahlmenü einer Auswertung steuert den Einsatz von Analysemodulen, und die Spezifikation eines Darstellungsverfahrens wird in die Wahl bzw. Parametrisierung eines Visualisierungsmoduls übersetzt, dessen Ausgabe dem Anwender jeweils präsentiert werden kann.

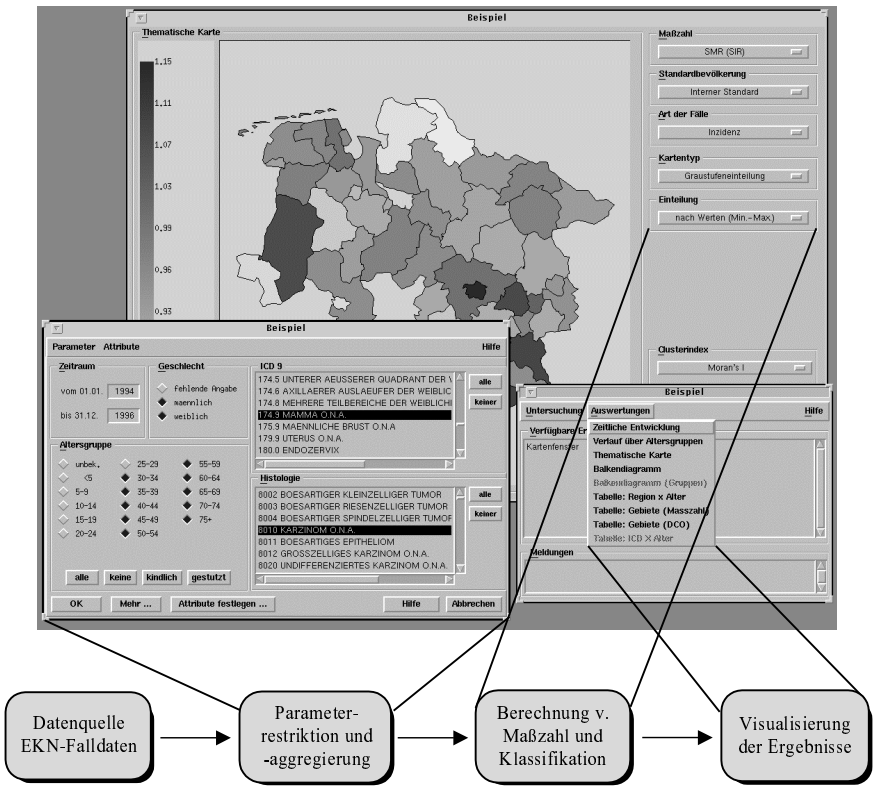

Abbildung 6.4: Übersetzung menübasierter Auswertungen in Datenflußprogramme

Im folgenden stellen wir für repräsentative Ausschnitte der *CARESS*–Funktionalität jeweils entsprechende *VIOLA*–Programme vor und gehen auf deren dynamische Steuerung und Parametrisierung durch *CARESS*

ein. Die gewählten Teilbereiche von *CARESS* werden jeweils weitgehend vollständig modelliert; nur einzelne, konzeptionell leicht zu integrierende Aspekte werden ggf. ignoriert, um die präsentierten Abbildungen nicht zu sehr zu überladen.

Abbildung 6.5 zeigt die Umsetzung des allgemeinen *CARESS*–Parameterfensters zur Eingrenzung von Studien- und Standardpopulation durch ein *VIOLA*–Verbundmodul. (Dessen Abbruchmodul wird hier nicht explizit benötigt, aber der Vollständigkeit halber — jedes Verbundmodul besitzt ein Abbruchmodul — trotzdem rechts unten abgebildet.) Durch Pfeile sind jeweils die Stellen gekennzeichnet, an denen von *CARESS* aus Modulparameter manipuliert werden können. Hier und in allen anderen Abbildungen von *VIOLA*–Programmen in diesem Abschnitt sind Module, die mehrere Eingänge parallel und unabhängig voneinander verarbeiten (etwa unter (2)), durch die jeweils gegenüberliegenden Anknüpfungspunkte klar von solchen Modulen zu unterscheiden, die aus mehreren Eingangsdatenräumen ein gemeinsames Ergebnis berechnen (z. B. unter (4)). Man beachte, daß Vereinigungsoperatoren in beiden Varianten auftreten können, wie wir in späteren Abbildungen noch sehen werden.

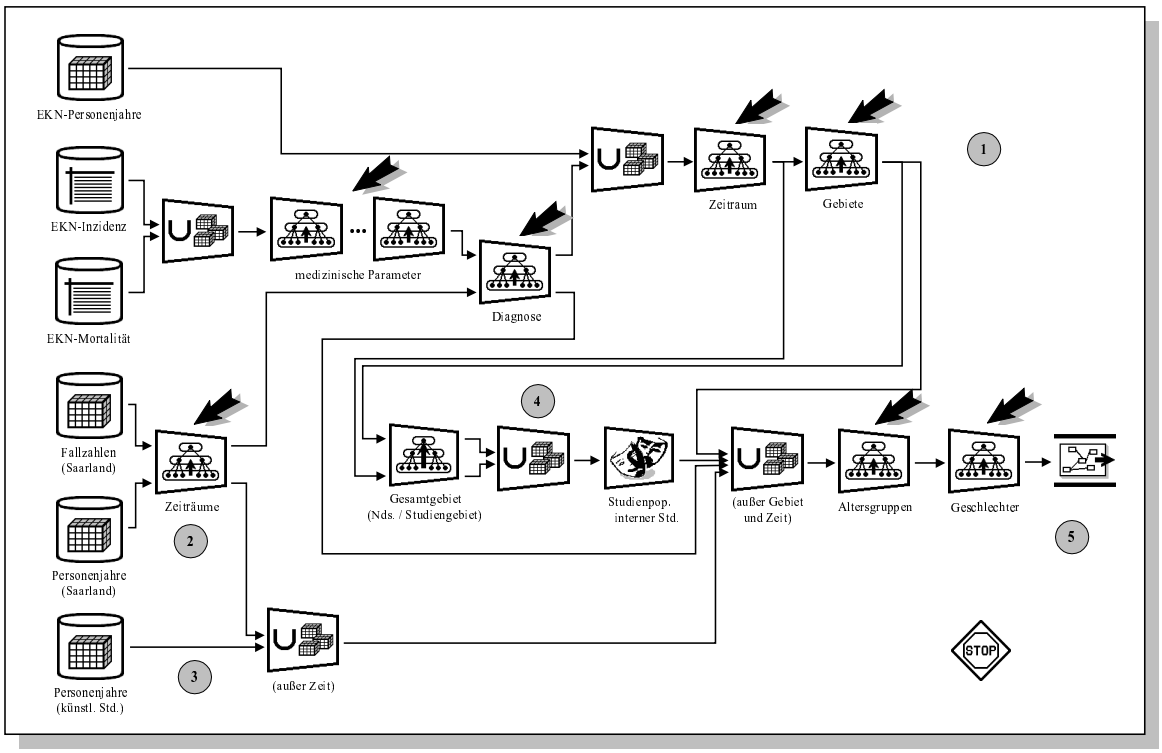

Abbildung 6.5: Modellierung der *CARESS*–Datenbankanfragen mit *VIOLA*

Im oberen Bereich (1) wird die niedersächsische Studienpopulation über verschiedene Parameter eingegrenzt. Da die in der Datenbasis des EKN als Mikrodaten vorliegenden Falldaten etwa zwanzig verschiedene Attribute aufweisen, ist es hierbei unverzichtbar, daß die Restriktionen bzgl. der medizinischen Parameter im Rahmen einer Anfrageoptimierung (vgl. Abschnitt 6.2.1) in den Datenquellenzugriff integriert werden. Andernfalls wäre der als Ausgabe der Mikrodatenquellen erhaltene Datenraum mit zwanzig kategoriellen Attributen, die jeweils zwischen drei und mehreren hundert Kategorien umfassen, sicherlich deutlich zu groß zur hauptspeicherresidenten Bearbeitung. Alternativ könnte als Datenquelle auch eine geringerdimensionale, evtl. parametrisierbare Sicht auf die Basisdaten bereitgestellt werden.

Die Saarland–Daten werden links unten (2) auf bestimmte Jahrgänge eingegrenzt bzw. zu Jahrgangsgruppen aggregiert, die als Standardbevölkerungen dienen. Die künstlichen Standards (3) werden zunächst nicht weiter manipuliert.

Im mittleren Bereich (4) werden Daten zu Gesamt–Niedersachsen sowie zum gesamten gewählten Studiengebiet durch Rollenumbenennung zur Nutzung als Standardpopulationen präpariert. Abschließend werden alle Datensätze auf interessierende Altersgruppen und Geschlechter eingeschränkt bzw. entsprechend aggregiert und über ein Ausgangsmodul zur Weiterverarbeitung bereitgestellt (5).

Soweit nicht anders angegeben, identifizieren Vereinigungsoperationen jeweils alle Attribute zur gleichen Domäne miteinander. Lediglich werden unter dem Zeitaspekt Studienpopulation (Beobachtungsjahr), Saarland–Standard (Vergleichsjahr) und künstlicher Standard (Definitionsjahr) sowie unter dem Gebietsaspekt Studien- und Standardpopulation unterschieden, also durch eigene kategorielle Attribute beschrieben. Hier zeigt sich die Flexibilität der datenflußbasierten Modellierung: Durch leichte Änderungen könnte etwa erzwungen werden, daß der zeitliche Bezug von Saarland– und Niedersachsen–Daten stets übereinstimmt. Analog wäre auch denkbar, durch ein separates Geschlechtsattribut für die Standardpopulation andere Varianten der Standardisierung von Erkrankungsraten vorzubereiten. Zum Beispiel könnten so geschlechtsspezifische Daten der Studienpopulation auf Gesamtwerte des Standards und umgekehrt bezogen werden — eine Thematik, die aktuell im EKN diskutiert wird.

Weiterhin wird die zentrale Rolle der Datenraumvereinigung deutlich. Sie dient nicht nur der Zusammenstellung verschiedener Quellmaßzahlen, die anschließend zu einer neuen Zielmaßzahl verknüpft werden sollen (wie es häufiger in Abb. 6.6 geschieht), sondern auch der Zusammenfassung von Datenräumen, die im weiteren Verlauf auf die gleiche Art und Weise, aber getrennt voneinander verarbeitet werden sollen (dies betrifft praktisch alle Vorkommen des Operators in Abb. 6.5). Somit stellt die Vereinigung eine Alternative zur Nutzung paralleler Datenports dar und hilft, das Programmlayout übersichtlicher zu halten. Als Nachteil ist zu nennen, daß so vereinigte Datenräume nur durch explizite Anwendung eines Selektions- oder Restriktionsmoduls wieder zu trennen sind. Dies stellt einen gewissen Verwaltungsaufwand dar und ist auch nicht in jedem Fall (als exakt inverse Operation) möglich.

In Abb. 6.6 ist die Grundlage der interaktiven Auswahl beliebiger Maßzahlen in den Auswertungen von *CARESS* modelliert. Es kann im Rahmen der Nutzung des gezeigten Verbundmoduls für *CARESS* davon ausgegangen werden, daß die an den Verbundeingängen anliegenden Datenräume zu Studien- und Standardpopulation die gleichen Ausprägungen der bei der Maßzahlberechnung interessierenden Attribute (in der Regel Alter und Geschlecht) aufweisen und somit völlig problemlos vereinigt werden können. Aber auch Abweichungen wären, ebenso wie das Fehlen von (für einige Maßzahlen nötigen) Fallzahlen für bestimmte Standardpopulationen, kein Problem: Die Vereinigungsoperationen erzeugen in diesem Fall einfach entsprechende Nullwerte, die dann durch den Analysegraphen propagiert und jeweils geeignet berücksichtigt werden.

Im Bereich (1) werden die nötigen kategorienbasierten Hilfsinformationen zur Berechnung von Bevölkerungszahlen und kumulativen Raten über Parameterdefinitionsmodule und kategorielle Datenquellen bereitgestellt. Ansonsten folgt das Programm recht unmittelbar den Maßzahldefinitionen aus Abschnitt 6.3.1. Außer zu den rein bevölkerungsbezogenen Maßen werden stets auch Konfidenzintervalle zur jeweiligen Maßzahl berechnet.

Zu beachten ist, wie unter (2) alle Daten zu Studien- und Standardpopulation zusammengefaßt werden und die nachfolgenden beiden Standardisierungsmodule aufgrund der jeweiligen Maßzahldefinition nur die benötigten Quellmaßzahlen in den jeweils richtigen Rollen auswählen können. Analog braucht auch unter (3) vor der CMF/CIF–Berechnung nicht erst die rohe Rate der Standardpopulation von der der Studienpopulation separiert zu werden, wobei sich als Grundlage der automatischen Selektion die explizite Differenzierung zwischen beobachteter Rate und Standardpopulationsrate als unterschiedliche Maßzahlen bezahlt macht. Somit können Programmierung vereinfacht und Programme (unter Nutzung von Kommentierungsmöglichkeiten) übersichtlicher gehalten werden. Allein der Übersichtlichkeit dient auch die redundante Bestimmung der rohen Rate bei (4) und (5) — bei Rückgriff auf die in Abschnitt 6.2 skizzierten Optimierungstechniken ergeben sich hieraus kaum Effizienzeinbußen.

Die Parametrisierung des Verbundes in Abb. 6.6 erfolgt über Parameterschnittstellenmodule.[17] Es können

[17] Soweit es hier modelliert wird, ist es im Prinzip unerheblich, ob *CARESS* Parameter innerhalb eines Verbundes wie in Abb. 6.5 direkt anspricht oder Parameterschnittstellenmodule verwendet werden. Der Einsatz letzterer wird erst dann interessant, wenn der Verbund durch

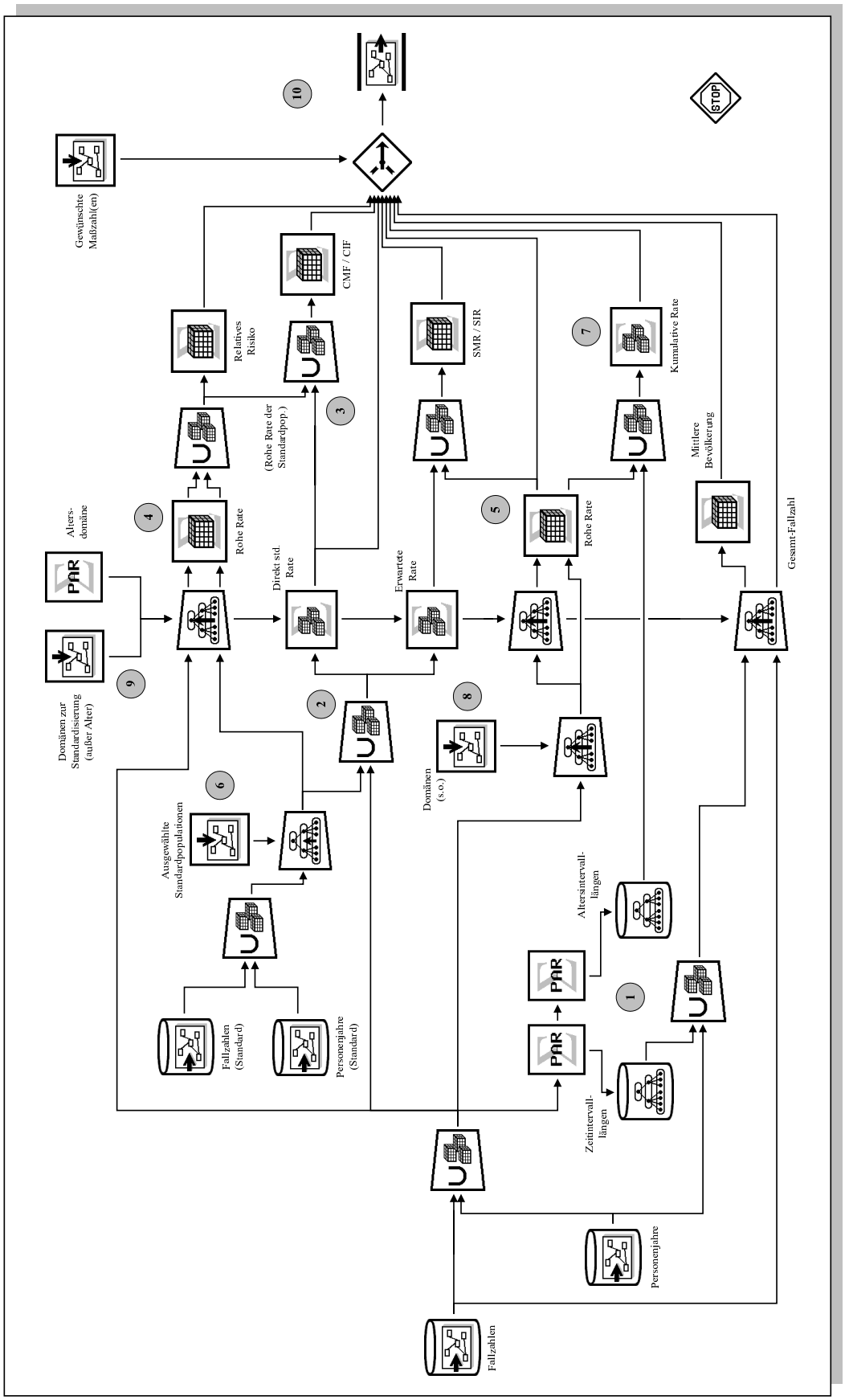

Abbildung 6.6: Ein *VIOLA*–Programm zur flexiblen Maßzahlberechnung in *CARESS*

nochmals bestimmte Standardpopulationen ausgewählt werden, was über Gebiets- oder Zeitbezug geschehen kann (6). Weiterhin ist zu spezifizieren, über welche Domänen standardisiert werden soll. Hierbei ist die Altersdomäne immer explizit vordefiniert, nicht zuletzt, weil sie auch für die kumulative Rate einen Sonderfall darstellt (7). Unter (8) wird von der Möglichkeit der visuellen Duplizierung des Parametereingangs (9) Gebrauch gemacht, um als Vorbereitung auf die Berechnung der kumulativen Rate über alle Domänen *außer* der Altersdomäne zu aggregieren.

Abschließend können über ein Routingmodul die für die weitere Verarbeitung interessierenden Maßzahlen selektiert werden (10). Das Abbruchmodul ist wiederum nur für die interne Verarbeitung von Bedeutung.

Die explizite Modellierung der Abhängigkeiten zwischen den verschiedenen Maßzahlen im Programm aus Abb. 6.6 bietet zum einen die größtmögliche Klarheit der Maßzahl- und Programmsemantik und ermöglicht zum anderen auch ein sehr effizientes Umschalten zwischen zu visualisierenden Maßen, da alle Teilergebnisse explizit in den jeweiligen Modulen zwischengespeichert werden. Auf eine alternative Modellierung, die geeigneter für die einfache direkte Erstellung und Manipulation des *VIOLA*–Programms über einen graphischen Editor ist, kommen wir in Abschnitt 6.3.4 noch zu sprechen.

Man sieht an dem vorgestellten Berechnungsbeispiel, daß das Finden eines geeigneten Programmlayouts, das Überschneidungen von Kanälen verhindert und möglichst oft von der parallelen Nutzung unabhängiger Datenportpaare Gebrauch macht, keine triviale Aufgabe ist. Insbesondere ist es schwierig, auch im Hinblick auf zukünftige Programmerweiterungen gute Raumaufteilungen zu finden (vgl. [GP96]). Dies unterstreicht nochmals die in Abschnitt 5.4.6 bzw. Abschnitt 6.2 formulierten Anforderungen an einen graphischen Editor zur Unterstützung der Modulplazierung sowie an den Optimierer zur effizienten Verarbeitung von Analysegraphen, in denen allein zur Erhöhung der Übersichtlichkeit Berechnungen in getrennten Teilbereichen eines Programms doppelt modelliert werden.

Anhand des Beispiels der Erstellung thematischer Karten inklusive der Bestimmung zugehöriger Clusterindizes zeigen wir nun mit Abb. 6.7, wie eine vollständige Untersuchungen in *CARESS*— von der Datenbankanfrage bis zur Visualisierung — durch *VIOLA* implementiert werden kann. Wieder sind die Stellen der Programm–Parametrisierung durch *CARESS* mit Pfeilen belegt.

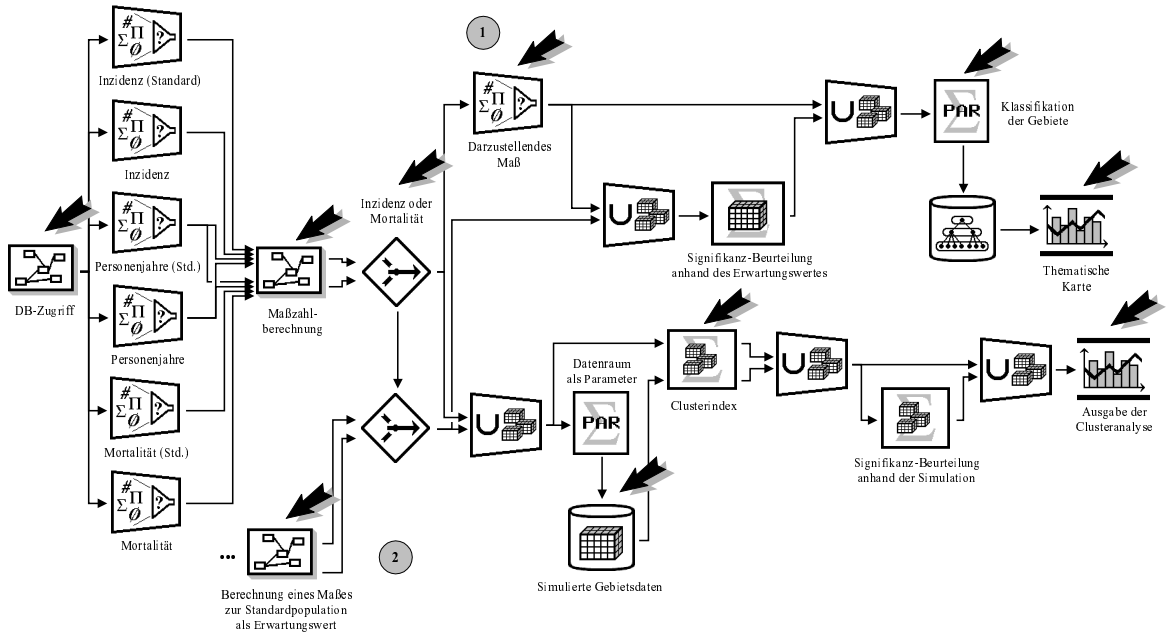

Abbildung 6.7: Umsetzung der Kartenerstellung in *CARESS* mittels *VIOLA*

andere externe Module und deren Parameter automatisch gesteuert werden soll — hier zeigen wir nur die prinzipielle Möglichkeit auf.

Zunächst werden über ein Verbundmodul gemäß Abb. 6.5 die interessierenden Daten aus der Datenbank selektiert. Anschließend wird der Verbund aus Abb. 6.6 getrennt mit Inzidenz- und Mortalitätsdaten aus Studien- und Standardpopulation sowie zugehörigen Bevölkerungsdaten belegt. Über die Modulparameter werden eine Standardpopulation, Standardisierungsdomänen sowie ein oder mehrere Maße spezifiziert. Die Tatsache, daß für Standardbevölkerungen ohne Fallzahlen nicht alle Maßzahlen bestimmt werden können, wird von *CARESS* automatisch in der Benutzungsoberfläche berücksichtigt. Nachfolgend kann dann über ein Verzweigungsmodul dynamisch zwischen Inzidenz- und Mortalitätsmaßen gewählt werden.

Im oberen Teil von Abb. 6.7 wird nun die thematische Karte generiert. Hierzu wird in einem Selektionsmodul zunächst genau eine Maßzahl mit ihrem Konfidenzintervall selektiert (1). Demgegenüber sind für den unteren Bereich, die Bestimmung eines Clusterindex, u. U. mehrere Maße (etwa zusätzlich zu standardisierten Raten jeweils noch Bevölkerungszahlen) vom Verbundmodul zur Maßzahlberechnung vorzuhalten.

In einem zur Maßzahlberechnung ähnlichen Modul (2), das hier aber nicht im Detail vorgestellt werden soll, wird ein auf die Standardpopulation bezogenes Maß als Erwartungswert für die unter (1) gewählte Maßzahl ermittelt. Abhängigkeiten bei dieser Wahl werden durch *CARESS* bestimmt.

Für die thematische Karte wird zunächst noch ein Signifikanzmaß bestimmt, um dann hierüber oder direkt über die gewählte Maßzahl eine Klassifikation der gewählten Gebiete vorzunehmen und diese in einem Visualisierungsmodul zu visualisieren (zur Kartenerstellung vgl. die Diskussion in Abschnitt 5.2.2 im Anschluß an Def. 5.13). Klassifikationsverfahren (Einteilung in gleich große Gruppen, anhand von Werteintervallen oder Signifikanz etc.), deren Parameter (vor allem Anzahl der Gruppen) sowie Visualisierungsparameter (etwa genutzte Farben) können von *CARESS* aus gewählt werden.

Im Rahmen der Clusteranalyse werden zusätzlich zur Bestimmung des Clusterindex und seines p–Wertes für die beobachteten Maße auch noch Simulationen von Maßzahlen im betrachteten Gebiet durchgeführt. Indem auf einer Menge von Simulationsläufen jeweils ebenfalls der Clusterindex bestimmt wird, kann hier eine weitere Einstufung der statistischen Signifikanz des beobachteten Maßzahlwertes vorgenommen werden.[18] Abschließend können die gesammelten Ergebnisse, zusammen mit einigen Metadaten zum jeweiligen Clusterindex (u. a. seiner approximativen Verteilungsfunktion und Dichte), visualisiert werden.

Insgesamt haben wir unterschiedliche Varianten zur Implementierung der interaktiven Parametrisierung von *CARESS*–Visualisierungen in *VIOLA* kennengelernt:

- Die Wahl der Fallart erfolgt als Selektion eines Ausgangs des Verbundmoduls zur Maßzahlberechnung, d. h. Inzidenz und Mortalität werden „gleichzeitig“ und unabhängig voneinander verarbeitet.
- Mehrere Standardpopulationen sind prinzipiell über ein zusätzliches Gebiets- und/oder Zeitattribut in jedem berechneten Datenraum ansprechbar. Das Beispiel der Kartenerstellung macht hiervon keinen Gebrauch und wählt jeweils über den Maßzahlberechnungsverbund genau eine Standardpopulation aus. Diese Auswahl könnte alternativ auch erst direkt vor der Visualisierung erfolgen.
- Eine Vielzahl von Maßzahlen wird über ein Netzwerk von voneinander abhängigen Modulen ständig bereitgehalten. Über ein Routingmodul werden einzelne Maße selektiert.
- Klassifikationsverfahren zur Definition der Gebietsgruppierung und -einfärbung in der Kartendarstellung sind als Berechnungsmethoden eines Parameterdefinitionsmoduls realisiert, das wiederum durch zusätzliche Angaben noch näher parametrisiert werden kann.
- Spezielle Visualisierungsparameter entsprechen internen Parametern des Visualisierungsmoduls.

Gerade im Hinblick auf die Wahl der ersten drei Varianten sind natürlich auch andere Systemkonfigurationen denkbar und relativ einfach realisierbar. Diesbezügliche Entwurfsentscheidungen orientieren sich an einem Trade–off zwischen einem schnellen und einfachen Wechsel zwischen verschieden parametrisierten

[18] Hierzu ist eine weitere allgemein verwendbare Signifikanzmaßzahl zu definieren, die sich aus beliebigen simulierten und beobachteten Clusterindizes ergibt, indem letztere in die geordnete Liste der Simulationsergebnisse einsortiert werden.

Ergebnissen und dem erhöhten Speicheraufwand für viele Berechnungsalternativen und Zwischenergebnisse. Eine geringere Rolle spielt an dieser Stelle die möglichst einfache und übersichtliche Modellierung in *VIOLA*.

Weitere durch *CARESS* angebotene Auswertungen zur Erstellung von Diagrammen und Tabellen sind in ganz ähnlicher Weise wie die Kartenerstellung zu realisieren. Allgemein bieten sich bei der Definition des Gesamtprogramms (zumindest) drei Alternativen hinsichtlich der Integration der Teilprogramme zur Erstellung unterschiedlicher Graphiken, die jeweils Werteverteilungen über verschiedene Dimensionen (Gebiet, Alter, Zeit, Diagnose etc.) darstellen:

- Alle Auswertungen verwenden je ein gemeinsames Verbundmodul zur Datenbankanfrage und zur Maßzahlberechnung. Erst im Anschluß werden spezifische vollständige Aggregierungen und Selektionen vorgenommen. Da viele Maßzahlen nicht aggregierbar sind, müssen bereits vor der Maßzahlberechnung entsprechende *Gesamt*–Kategorien als „Randsummen“ in den betrachteten Datenraum eingefügt werden. Dieses Vorgehen ermöglicht eine einfache Modellierung und vermeidet weitgehend redundante Berechnungen in verschiedenen Teilprogrammen. Dafür bestehen kaum Möglichkeiten, die Größe des initial aus der Datenbank abzufragenden Datenraums durch optimierende Zusammenfassung von Datenquellen mit vollständigen Aggregierungen einzuschränken. Mit steigender Vielfalt der möglichen Auswertungen und wachsender Menge gesondert betrachteter Dimensionen ist dieses Verfahren nicht mehr praktikabel, da die betrachteten Datenräume einfach zu groß für den Hauptspeicher werden.

- Wenn nur die Datenbankanfrage gemeinsam formuliert wird, brauchen keine Randsummen mehr gebildet zu werden, was eine gewisse Speicherersparnis zur Folge hat. Die grundlegende Speicherproblematik bleibt jedoch unverändert.

- Eine jeweils spezifische Modellierung des Datenbankzugriffs und die Verwendung jeweils einer eigenen Instanz des Verbundmoduls zur Maßzahlberechnung (oder separater Ein- und Ausgänge eines gemeinsam genutzten Verbundes) sind zwar in der initialen Beschreibung am aufwendigsten, aber letztendlich — trotz teilweise redundanter Berechnungen — vor allem aufgrund eines im Rahmen der Anfrageoptimierung leicht einzuschränkenden Ausgangsdatenraums in der Regel am effizientesten in der Ausführung. Durch Definition geeigneter, mehrfach nutzbarer Verbunde läßt sich das Gesamtprogramm trotzdem weiterhin gut strukturieren.

Auch hier muß eine konkrete Entscheidung für eine Modellierungsvariante auf der Basis einer Evaluation von Alternativen und in Abstimmung mit der Mächtigkeit der umgesetzten internen Optimierungsverfahren erfolgen.

6.3.3 Erstellung des Krebsregister–Jahresberichts mit *VIOLA*

Eine etwas anders geartete Aufgabenstellung für *VIOLA* bildet die Unterstützung der Erstellung eines Auswertungsberichts, konkret hier am Beispiel des ersten Jahresberichts des EKN [EKN00] vorgestellt. Dieser Jahresbericht bietet eine standardisierte Darstellung der Erkrankungshäufigkeiten des Berichtszeitraums (1996) in Niedersachsen. Einige Überblicksdarstellungen beziehen sich auf alle Formen von Krebserkrankungen und das gesamte Studiengebiet; andere, differenziertere Auswertungen betrachten einzelne Krebsarten und Regionen genauer.

Über *VIOLA* werden Auswertungsmengen spezifiziert, die automatisiert, quasi „auf Knopfdruck“, einfach oder — unter Verwendung parametrisierter Schleifen — mehrfach durchlaufen werden und die erstellten Tabellen und Graphiken in Dateien ablegen. Im folgenden verwenden wir Datenspeichermodule für textuell abgelegte tabellarische Ergebnisse und Visualisierungsmodule zur Repräsentation von Graphiken, die aber ebenfalls nicht angezeigt, sondern als Bilddatei persistent gespeichert werden. Wie schon im vorigen Abschnitt werden auch hier repräsentative Beispiele angeführt — ggf. mit leichten Vereinfachungen hinsichtlich des Umfangs, aber nicht grundlegende Konzepte oder Funktionsmächtigkeit betreffend.

Gegenüber den bereits für *CARESS* modellierten Auswertungen stellen die für Registerberichte üblichen Übersichtsdarstellungen zu Altersverteilung und zeitlicher Entwicklung von Krebsinzidenz und -mortalität keine neue Herausforderung an *VIOLA* dar, so daß hierauf an dieser Stelle nicht näher eingegangen wird.

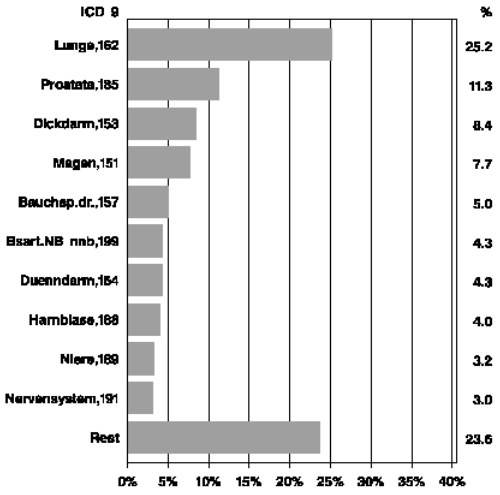

Abbildung 6.8: Diagnosen mit höchster Männer–Sterblichkeit (aus [EKN00])

Einen neuen Aspekt, nämlich die Berechnung von (Diagnose–)Anteilen, bringt allein die Darstellung der Verteilung der häufigsten Diagnosen (im Jahr 1996) gemäß Abb. 6.8.

Ein *VIOLA*–Programm zur Erstellung dieser Graphik zeigt Abb. 6.9: Die Gesamtzahl von Fällen über alle Diagnosen wird den Einzelhäufigkeiten gegenübergestellt, indem für die Gesamtkategorie ein zusätzliches kategorielles Attribut über der Diagnosedomäne und der Vergleichsrolle eingeführt wird, so daß der Gesamtwert über das Attribut der Einzeldiagnosen identisch dupliziert wird. Der Anteil ist nun jeweils als entsprechender Quotient aus Fallzahl zur Einzeldiagnose und Fallzahl zu allen Diagnosen definiert. Mittels eines Parameterdefinitionsmoduls können die zehn häufigsten Diagnosegruppen definiert und alle übrigen Diagnosen zu einem Gesamtwert aggregiert werden. Da Diagnoseanteile über Diagnosen der Studienpopulation problemlos addierbar sind, liefert eine Ableitungsoperation mit der so ermittelten Parametrisierung das für die Umsetzung von Abb. 6.8 gewünschte Ergebnis.

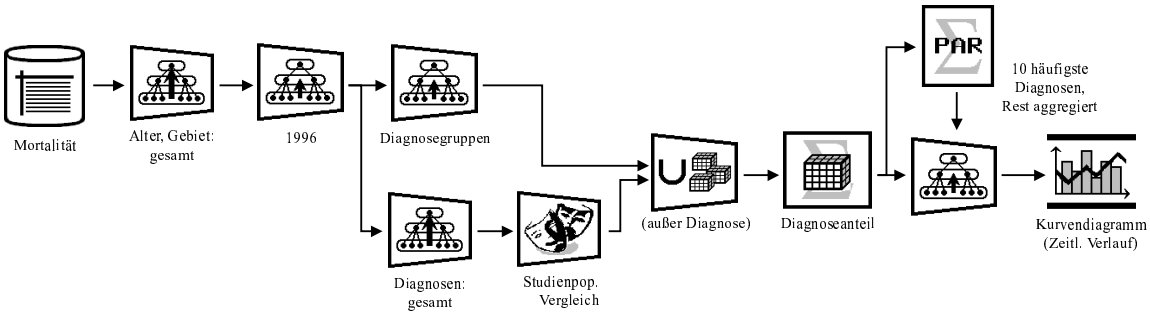

Abbildung 6.9: Erstellung der Graphik aus Abb. 6.8 mit *VIOLA*

Bei der Berechnung von Anteilen zeigt sich eine kleine Schwäche von *VIOLA* bzw. *MADEIRA*: Anteile stellen eigentlich eine Gliederungszahl dar, d. h. ein Teil wird zu einem übergeordneten Ganzen ins Verhältnis gesetzt. Da die Definition von Maßzahlberechnungen in *MADEIRA* jedoch keine zellenbezogenen Betrachtungen zuläßt (in der Form „Verfeinert eine Diagnose eine andere?“ bzw. „Inwiefern ist eine Diagnose ein Gesamtwert?“), können Anteile lediglich als Indexzahlen modelliert werden, d. h. zwei gleichartige Maße zu Teilen einer Grundgesamtheit werden miteinander verglichen. Somit ist strenggenommen die Bezeichnung *Anteil* auch nicht gerechtfertigt, da z. B. auch Brustkrebs- auf Leberkrebsfälle bezogen werden könnten. Die Aggregierbarkeit von Anteilen ist trotz allem aufgrund der Differenzierung der beiden Rollen *Vergleich* und *Studienpopulation* problemlos als Summe definierbar.

Das Kernkapitel des EKN–Jahresberichts zum Berichtsjahr 1996 umfaßt eine Gruppe von Auswertungen, die für eine Reihe von ausgewählten Diagnosegruppen (bestimmte dreistellige ICD–Codes) gleichermaßen durchgeführt und jeweils auf einer Berichtsseite dargestellt werden. Die Auswertungen bestehen jeweils für Inzidenz

und Mortalität aus

- einer Tabelle über Fallzahl, direkt standardisierte und kumulative Rate, Altersmedian und Diagnoseanteil, wie sie in Tab. 6.3 exemplarisch für die Mortalität dargestellt ist,
- einer Verteilung der altersspezifischen Raten des Berichtsjahres für Männer und Frauen in einem Kurvendiagramm,
- einer zeitlichen Entwicklung der kumulativen Raten für Männer und Frauen in der Altersgruppe 35–64 sowie für die über 65jährigen (also vier Kurven in einem Diagramm) über den Gesamtzeitraum 1991–96 und
- einer thematischen Karte, die auf Kreisebene die SIR zum Saarland–Standard für 1996 bzw. die direkt auf den BRD–Standard standardisierte Mortalität zum Zeitraum 1992–96 darstellt.

Mortalität 1996 (Niedersachsen)

	Fall	I_{dir}		I_{kum}		Alter	Diag
	#	BRD87	Welt	0-64	0-74	Med	%
♂	2669	81.9	43.0	2.2	5.4	68.2	25.2
♀	743	15.7	8.6	0.4	1.0	72.1	7.2

Tabelle 6.3: Diagnosenspezifische Maßzahlen im EKN–Jahresbericht

In Abb. 6.10 ist ein durch ein *VIOLA*–Verbundmodul gekapselter Analysegraph zur Erzeugung dieser Tabellen und Graphiken für eine Fallart dargestellt. Über ein Schleifenmodul werden alle interessierenden Diagnosegruppen durchlaufen. Es wird davon ausgegangen, daß am Eingang des Verbundes Inzidenz- *oder* Mortalitätsdaten auf Ebene dreistelliger ICD–Codes sowie Bevölkerungen aus Niedersachsen für die Jahre 1991–96 auf Landkreisebene nach Altersjahrgängen und Geschlecht vorliegen. Ferner werden entsprechende Daten mindestens zu den Standardbevölkerungen BRD, Saarland (Gesamtwert 1991–95) und Welt benötigt. Um die verarbeiteten Datenräume nicht zu groß werden zu lassen, kann es u. U. sinnvoll sein, alle nicht näher interessierenden Diagnosegruppen bereits vor Übergabe an den Verbund zu einer Gruppe aufzuaggregieren. Da in keiner Auswertung sowohl die Zeit- als auch die Gebietsdimension interessiert, ergibt sich eine weitere deutliche Speicher- und damit auch Rechenzeitersparnis aus der Möglichkeit, nur die entsprechenden „Randsummen“ im Datenraum zu materialisieren.[19]

Im linken Bereich (1) von Abb. 6.10 werden unterschiedliche, für die Einzelauswertungen benötigte Teildatenräume anhand der jeweils interessierenden Domänen (Zeit oder Gebiet) vorselektiert sowie Studien- und Standardpopulation zusammengeführt (2). Für die nach Gebieten aufgeschlüsselten Daten zur Kartenerstellung wird zudem je nach Fallart ein anderer Zeitraum gewählt (3).[20] Neben der Auswahl der jeweiligen Diagnosegruppe (4) werden zur Bestimmung von Diagnoseanteilen (analog zu Abb. 6.9) auch Vergleichs–Fallzahlen über alle Diagnosen als Gesamtwert ermittelt (5). Die meisten Auswertungen benötigen zur Altersstandardisierung (über das Geschlecht wird hier nicht standardisiert) 5–Jahres–Altersgruppen (6); lediglich der Altersmedian (7) operiert auf Einzeljahren. Im rechten Bereich der Abbildung sind untereinander die Teilgraphen für die einzelnen Auswertungen angeordnet: Zeitlicher Verlauf (8), Altersverteilung (9), Tabelle (10) und thematische Karte (11). Für letztere ist nochmals eine kleine Schleife über das Geschlecht implementiert, so daß für Männer und Frauen jeweils eine eigene Karte generiert wird. Zur Tabelle (10) wäre noch anzumerken, daß die Anordnung der verschiedenen summarischen und kategoriellen Attribute (wie generell bei allen nicht–trivialen Tabellen und Visualisierungen) wie auch die Sortierung der Ausprägungen in der Regel über interne Parameter des entsprechenden Speicher- (oder Visualisierungs-)moduls durch den Anwender zu konfigurieren ist. In gewissem Rahmen kann das System jedoch zumindest auch Vorschläge zur Darstellung unterbreiten.

Aus den verschiedenen Tabellen im Anhang von [EKN00] sei schließlich noch als Beispiel die Verteilung altersspezifischer Fallzahlen (#) und Mortalitätsraten (M) über alle Diagnosen nach Geschlecht herausgegriffen (siehe Tab. 6.4).

Bemerkenswert ist hier vor allem, daß einzelne Diagnose–Geschlecht–Kombinationen von der Darstellung ausgeschlossen sind (u. a. ICD 174, männlich, sowie ICD 185, weiblich). Weiterhin ist, wie auch schon in

[19] Man erzeugt also im Vorfeld einen Datenraum über Zeit, Alter, Geschlecht und Diagnose zum Gesamtgebiet Niedersachsen sowie einen über Region, Alter, Geschlecht und Diagnose zum Jahr 1996 sowie zum Zeitraum 1991–95 und vereinigt diese. Die Nullwerte für

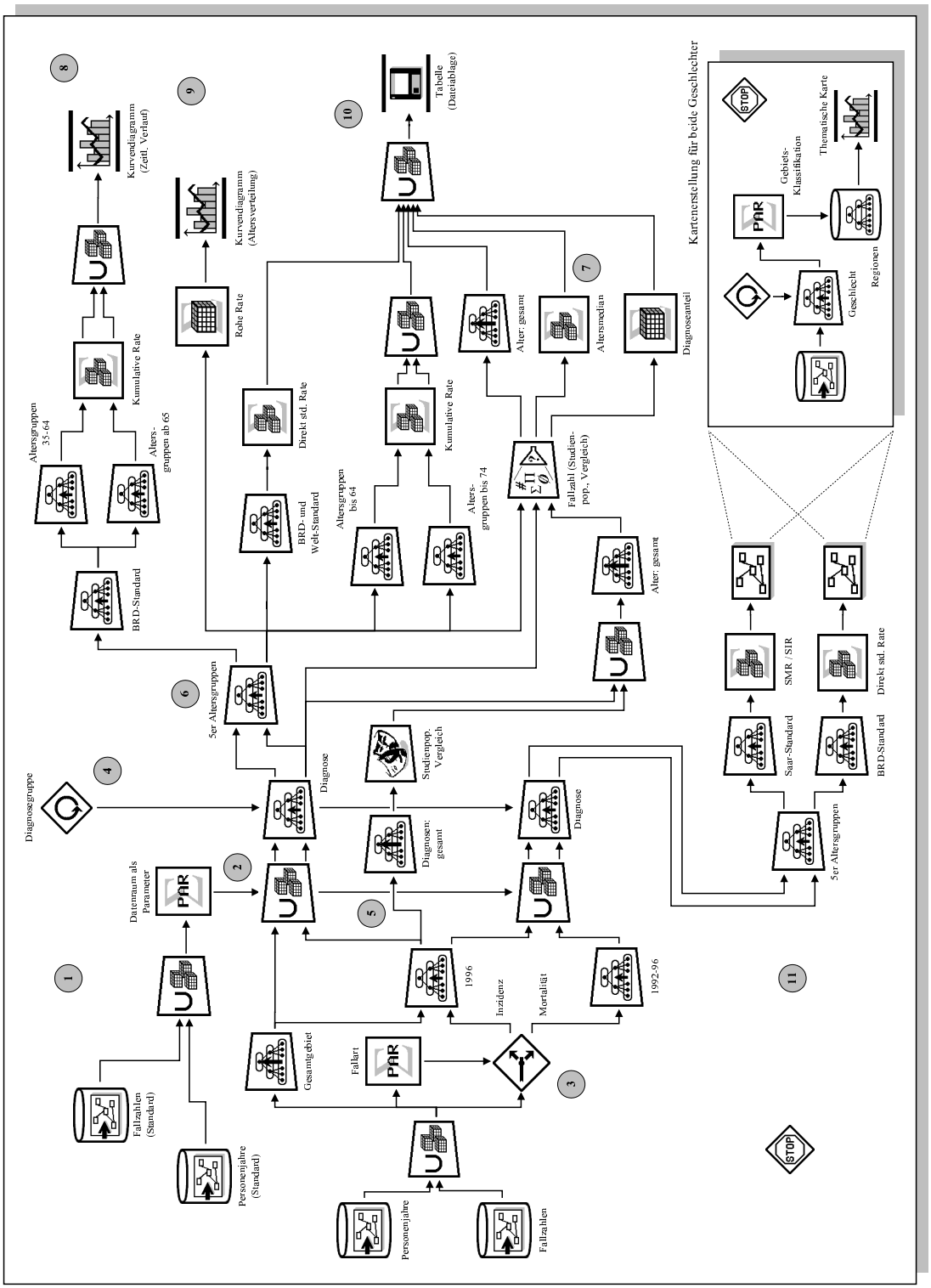

Abbildung 6.10: Ein Verbundmodul mit Iteration zur Generierung diagnosespezifischer Auswertungsgruppen für den EKN–Jahresbericht

ICD-9		Altersklassen														M_{dir}
		0-14		15-19		20-24		...	80-84		85+		gesamt			
		#	M	#	M	#	M		#	M	#	M	#	M		
...																
160 Nasenhöhlen	♂							...			1	2.7	9	0.2		0.3
	♀								1	0.8	1	0.9	6	0.1		0.1
...																
165 Atmungsorg. o.n.A.	♂							...								0.0
	♀															0.0
160-65 Atmungsorgane	♂					2	0.9	...	281	526.1	149	404.2	2922	76.6		89.5
	♀					1	0.4		106	83.6	86	78.7	809	20.2		17.0
...																
174 weibliche Brust	♀							...	199	156.9	223	204.0	1800	45.0		38.9
...																
185 Prostata	♂							...	291	544.9	298	808.4	1197	31.4		
...																
140-208 Gesamt	♂	12	1.8	8	3.8	14	6.0	...	1321	2473.4	1205	3268.9	10624	278.4		340.2
	♀	13	2.1	4	2.0	9	4.0		1717	1353.9	2091	1913.3	10272	256.7		203.6

Tabelle 6.4: Diagnosespezifische Fallzahlen und Raten (aus [EKN00])

Tab. 6.3, die Dimensionalität der dargestellten Maßzahlen unterschiedlich: Fallzahlen und rohe Raten sind für einzelne Altersgruppen verfügbar, die direkt (über das Alter) standardisierte Rate dagegen (per Definition) nicht. Beide Aspekte können elegant in dem *VIOLA*–Programm in Abb. 6.11 berücksichtigt werden: zum einen über den Merge–Operator, der Diagnosen und Geschlechter zu einem zusammengesetzten Attribut zusammenführt und eine anschließende Selektion zuläßt, und zum anderen über die flexible Vereinigung von Datenräumen.

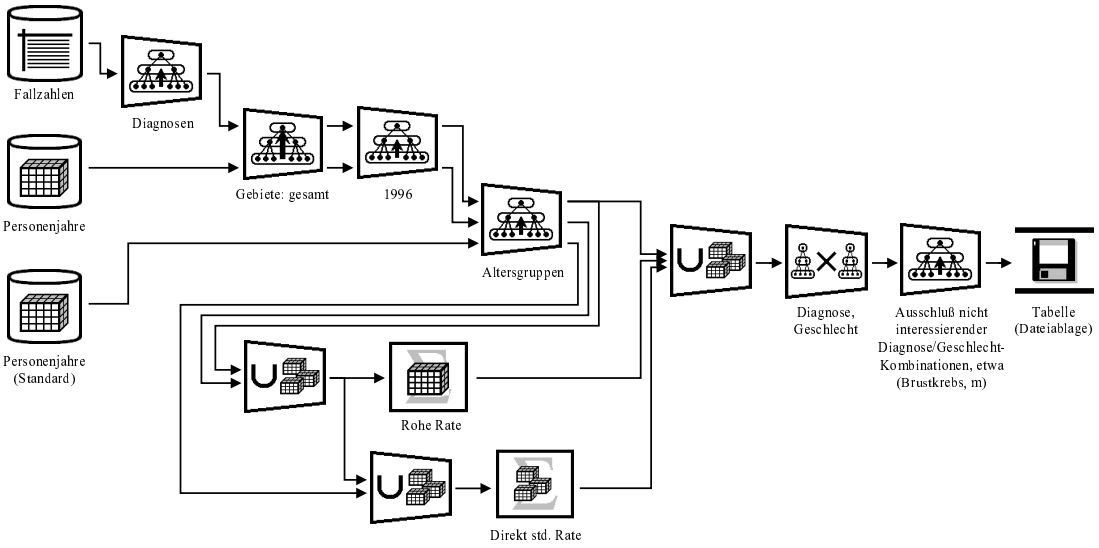

Abbildung 6.11: Ein *VIOLA*–Programm zur Generierung von Tab. 6.4

Kombinationen aus Gebiets- und Zeitangabe im entstehenden Datenraum brauchen nicht explizit materialisiert zu werden.

[20] Ein Aspekt des Jahresberichts [EKN00], der hier nicht berücksichtigt wird, ist, daß Inzidenzdarstellungen sich stets nur auf das Weser–Ems–Gebiet, Mortalitätsbetrachtungen sich dagegen stets auf ganz Niedersachsen beziehen. Diese Unterscheidung wäre in ähnlicher Weise über ein Verzweigungsmodul zu implementieren.

Bei der Verwendung des Merge–Operators zu beachten ist dessen Plazierung im Analysegraphen vor allem hinsichtlich der Bestimmung von Gesamtsummen. In Abb. 6.11 schließen evtl. durch die erste Diagnose–Ableitung oben links gebildete Gesamtsummen über alle Diagnosen auch die durch das Merge ausgeschlossenen Diagnose–Geschlecht–Kombinationen (also eventuelle Fälle von Prostatakrebs bei Frauen — äußerst selten, aber nicht völlig ausgeschlossen — oder Brustkrebs bei Männern) ein. Um dies zu verhindern, dürften entsprechende Summen nicht von vornherein als eine Gesamtkategorie neben den Einzelkategorien vorgesehen sein, sondern sollten ggf. erst nach dem Merge durch eine neuerliche Ableitung generiert werden.

6.3.4 Monitoring und interaktive Analyse

Zwei weitere mögliche Einsatzgebiete von *VIOLA* in der Krebsregistrierung sind die Unterstützung des Inzidenz- und Mortalitätsmonitoring, also die systematische Beobachtung des Falldatenbestandes auf bestimmte Auffälligkeiten, sowie natürlich die interaktive Datenanalyse.

Im Rahmen des Monitoring kann *VIOLA* z. B. dabei helfen, für bestimmte Auswertungen geeignete Aggregierungsebenen bzgl. ausgewählter Domänen zu finden, um aussagekräftige Fallzahlen pro Teilpopulation bei trotzdem angemessener Differenzierung der Gesamtpopulation sicherzustellen. Ein entsprechendes einfaches Schleifenmodul zur Aggregierung auf die betreffende Ebene hat etwa die Form wie in Abb. 6.12. Hier kommt das Abbruchmodul eines Verbundes zum Einsatz, um bei Erreichen einer geeigneten Ebene die Suche zu beenden.

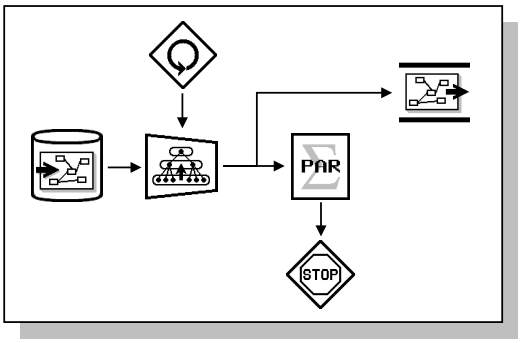

Abbildung 6.12: Suche adäquater Aggregierungsebenen durch *VIOLA*–Schleifen

Ein anderes einfaches Anwendungsbeispiel aus dem Monitoring–Kontext zeigt Abb. 6.13: Die auf einem längeren Beobachtungszeitraum basierende Auswahl „auffälliger“ Regionen kann anhand des Folgejahres überprüft werden, indem dem Benutzer die entsprechenden Maßzahlen (hier Raten) zu diesen Regionen ausgegeben werden.

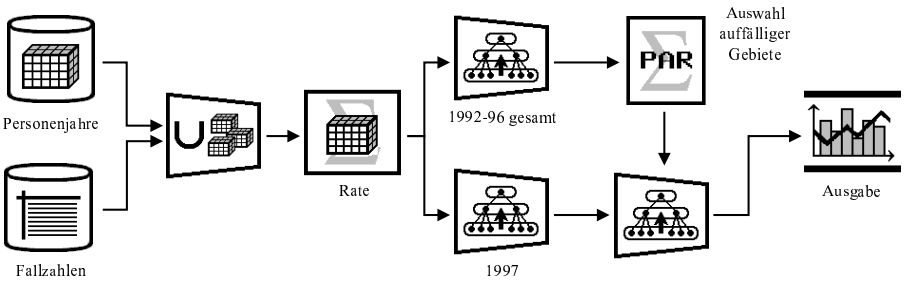

Abbildung 6.13: Überprüfung der Selektion auffälliger Regionen

Es sei an dieser Stelle jedoch nochmals darauf hingewiesen, daß *VIOLA* nicht dafür eingesetzt werden soll und kann, komplexe Algorithmen auf der Basis der Datenflußlogik zu modellieren, die iterativ voneinander abhängige Folgen und Parametrisierungen von Einzelauswertungen definieren (etwa in beliebigen, nicht nur den von *VIOLA* unterstützten „horizontal parallelen“ Schleifen). Derartige Algorithmen sind durch Maßzahlen und deren Berechnungsfunktion in atomaren Modulen zu kapseln. Unter Umständen könnte die von *VIOLA* angebotene Kontrollfunktionalität noch leicht an neue Anforderungen von Monitoringprozessen angepaßt werden, die sich im Laufe einer Systemevaluation ergeben. Derzeit liegen hierzu leider noch kaum konkrete Anwendungsszenarien aus dem EKN vor. Alle Systemerweiterungen sollten jedoch der Einfachheit des Da-

tenflußprinzips nicht wesentlich zuwiderlaufen. Wir kommen hierauf in Kapitel 7 im Rahmen der Diskussion möglicher Anschlußarbeiten zum Ausbau des *VIOLA*–Systems nochmals kurz zurück.

Als zentrale Aufgabe von *VIOLA* bleibt natürlich die Unterstützung der interaktiven, explorativen Datenanalyse: Die Abhängigkeiten bei der Festlegung aufeinanderfolgender Auswertungen sollen in flexibler Weise durch den Anwender selbst definiert werden. Bereits die Darstellungen in den vorangegangenen Abschnitten haben die vielseitigen Möglichkeiten von *VIOLA* zur Parametrisierung und Steuerung von Analyseabläufen, zur Gegenüberstellung und Kombination verschiedener Auswertungsverfahren bzw. (Teil–)Datenräume sowie zur Zusammenstellung auszugebender Daten aufgezeigt. Analysegraphen, die evtl. auch als Implementierung der Anwendungslogik von menübasierten Oberflächen wie *CARESS* oder zur Umsetzung von Berichten definiert wurden, können erweitert und umstrukturiert werden, wobei dem Anwender meist verschiedene Wege zur Berechnung eines bestimmten Ergebnisdatenraums offenstehen. Einen wesentlichen Beitrag zu dieser Flexibilität leisten die Möglichkeiten zur Definition alternativer Berechnungsverfahren von Maßzahlen und deren Abstützung auf das Vorliegen jeweils nötiger Quellmaßzahlen.

Zum Beispiel ist in Abb. 6.14 eine Alternative zum Verbundmodul zur Maßzahlwahl aus Abb. 6.6 dargestellt, die interaktiv wesentlich leichter zu manipulieren ist. Hier werden einfach alle relevanten Daten vereinigt, so daß sich die gewünschte Maßzahl aufgrund der Definition ihrer Berechnungsvorschriften aus dieser Menge von Datenräumen jeweils die benötigten Quellmaßzahlen auswählen kann. Weiterhin findet sich in diesem Zusammenhang eine sinnvolle Anwendung der Maßzahlen *Gesamtfallzahl*, *Gesamtrate* etc. Ohne spezielle Ableitungsoperationen einfügen zu müssen, können auch diese Maße alternativ zu verschiedenen Standardisierungsverfahren über das gleiche Analysemodul selektiert werden.

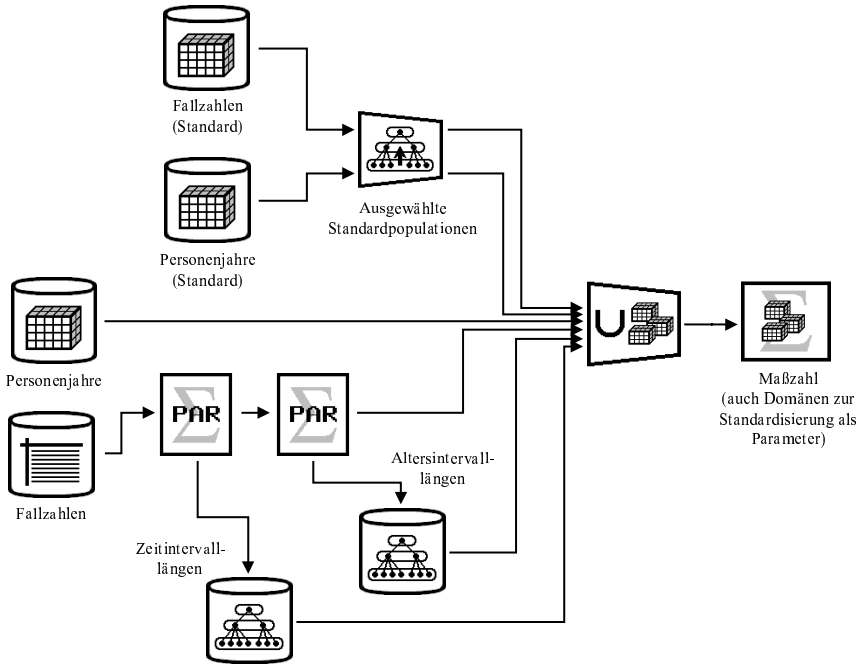

Abbildung 6.14: Alternative Umsetzung des Programms in Abb. 6.6 zur interaktiven Analyse

Empirische Studien zur Extraktion typischer explorativer Analyseabläufe im Krebsregister und zur Evaluation der Anwendbarkeit von *VIOLA* auf diese Szenarien konnten bisher noch nicht durchgeführt werden. Sobald das *VIOLA*–System vollständig implementiert sein wird, ist deren Durchführung jedoch eine vorrangige Aufgabe, um die Zufriedenheit der Anwender bei der Nutzung von *VIOLA* sicherzustellen bzw. durch eventuelle Modifikationen am Sprachumfang zu erhöhen. Schließlich soll der menschliche Anwender bei der von dieser Arbeit propagierten Art der *intelligenten* Datenanalyse stets im Mittelpunkt stehen.

Kapitel 7

Bewertung und Ausblick

In dieser Arbeit wurde mit dem Datenmodell *MADEIRA* und der darauf aufbauenden Analyseumgebung *VIOLA* ein neuartiger Ansatz zum On–line Analytical Processing, also zur interaktiven Analyse multidimensionaler (aggregierter) Datenräume vorgestellt. Nicht die (als „Black Box“ anzusehende) Abfrage interessierender Teildatenräume einer Datenbasis, sondern deren explorative, schrittweise Spezifikation, Manipulation und Analyse stehen im Vordergrund der Datenmodellierung bzw. Systemkonzeption. Die Grundphilosophie des hier verfolgten Ansatzes definiert der Begriff der *intelligenten* Datenanalyse: Nur durch ein enges kooperatives Zusammenwirken zwischen menschlichem Datenanalysten und rechnerbasiertem Analysesystem bzw. ihrer jeweiligen komplementären Analysefähigkeiten wird eine effektive Datenexploration ermöglicht.

Im folgenden fassen wir die wesentlichen Inhalte und Ergebnisse der Arbeit nochmals kurz zusammen und nennen im Rahmen einer umfassenden Systemevaluation möglicherweise kritisch zu hinterfragende Entwurfsentscheidungen sowie zentrale Anknüpfungspunkte für notwendige und mögliche Anschlußarbeiten.

7.1 Die Standbeine dieser Arbeit

Eine Eigenheit dieser Arbeit liegt in der Verbindung von Anforderungen, Ideen und Konzepten einer ganzen Reihe verschiedener Domänen. So beschäftigt sie sich mit der

- explorativen Analyse
- multidimensionaler Datenräume
- mit Mitteln datenflußbasierter visueller Programmierung
- auf der Basis einer umfassenden Datenbeschreibung durch Metadaten
- am Beispiel des Anwendungsgebiets der Krebsepidemiologie.

Nach einer kurzen Vorstellung der Krebsepidemiologie als Anwendungsdomäne in Abschnitt 1.2 wurden die verbleibenden vier Säulen des Entwurfs von *MADEIRA* und *VIOLA* mit ihren zentralen Begriffen und Konzepten in Kapitel 2 näher charakterisiert sowie Bezüge zwischen ihnen geknüpft. Es wurde speziell untersucht,

- welche unterschiedlichen Facetten verschiedene Ansätze zur Datenanalyse (deskriptive, konfirmatorische und explorative Statistik, OLAP und Data Warehousing, KDD, Data Mining sowie Maschinelles Lernen) betonen, aus welchen Teilschritten (Interaktionen) Datenanalysesitzungen aufgebaut sind sowie welche besondere Rolle jeweils die interaktive Datenexploration und -visualisierung spielt,
- welche Vor- und Nachteile die visuelle Programmierung im allgemeinen bietet, welche Arten visueller Programmierung es gibt und inwiefern speziell der Einsatz datenflußbasierter Programmierung in der Datenanalyse sinnvoll ist,

- welche besonderen Anforderungen die Analyse multidimensionaler Datenräume (sogenannter Datenwürfel) definiert, welche Strukturen und Operationen besonders zu unterstützen sind und welche Ansätze bestehende multidimensionale Datenmodelle hierbei jeweils verfolgen sowie
- wie die Vielfalt von Metadaten zur ergänzenden Beschreibung von Daten allgemein und insbesondere auf dem Gebiet der Datenanalyse klassifiziert werden kann.

Als Grundlage der weiteren Ausführungen wurde jeweils ein Basisvokabular definiert.

7.2 Wozu ein weiteres Analysesystem?

Eine Vielzahl von Systemen zur Datenanalyse befindet sich bereits auf dem Markt und deckt die Anforderungen unterschiedlichster Anwendergruppen und Aufgabenstellungen ab; viele Prototypen aus dem akademischen Bereich integrieren neuartige Wege der Analyseunterstützung.

In Kapitel 3 wurden die Charakteristika verschiedener Systemklassen anhand einiger exemplarischer Beispiele vorgestellt. Ein besonderes Augenmerk wurde auf die Klasse der datenflußbasierten Analysesysteme gelegt; auch auf Systeme zur (interaktiven) Datenvisualisierung wurde genauer eingegangen. Schließlich wurde auch über das im Niedersächsischen Krebsregister eingesetzte Auswertungssystem *CARESS* der Bezug zur Krebsepidemiologie hergestellt.

Die verschiedenen Systemklassen wurden einander vor allem anhand der jeweils unterstützten Analyseinteraktionen und Interaktionsmodelle, also der möglichen Abfolgen von Interaktionen auf einem zu untersuchenden Datenbestand, gegenübergestellt. Hieraus ergab sich als Zielsetzung von *VIOLA*, die Ansätze von

- auf einem multidimensionalen Datenmodell basierenden OLAP–Tools,
- bestehenden datenflußbasierten, vor allem auf die (wissenschaftliche) Visualisierung ausgerichteten Analyseumgebungen und
- Systemen zur interaktiven Datenvisualisierung

in neuartiger Weise zu kombinieren und hierbei ein möglichst flexibles Interaktionsmodell zu realisieren. Darüber hinaus besteht der wesentliche Beitrag dieser Arbeit zum *modernen* Data Warehousing in der Integration und sehr engen Kopplung eines exakten, formal definierten Datenmodells mit einer intuitiv nutzbaren, und hinsichtlich bereitgestellter Operatoren und Schnittstellen direkt an den Erfordernissen explorativer Datenanalysen ausgerichteten Analyseplattform.

7.3 Der Kern der Arbeit: *MADEIRA*

Kapitel 4 kann insofern als Kern dieser Arbeit angesehen werden, als mit dem dortigen Entwurf des multidimensionalen Datenmodells *MADEIRA* auch bereits die Funktionalität der Analyseumgebung *VIOLA* in ihren Grundzügen festgelegt wurde.

Zunächst erfolgte eine Einordnung der Ziele von *MADEIRA* in eine Klassifikation bestehender multidimensionaler Datenmodelle. *MADEIRA* wurde als logisches, sehr eng an der „reinen“ multidimensionalen Betrachtungsweise orientiertes Modell motiviert, das seine Schwerpunkte auf die Modellierung und Unterstützung der interaktiven Datenexploration sowie die genaue, metadatenbasierte Beschreibung der Datensemantik legt — auch dies ist (zumindest in der von *MADEIRA* angebotenen Breite und Tiefe) neuartig für aktuell von bestehenden OLAP–Tools umgesetzte Datenmodelle. Die effiziente Umsetzbarkeit des Modells wurde gegenüber diesen Aspekten als zweitrangig deklariert; eine ausreichend effiziente Unterstützung deskriptiver epidemiologischer Analysen wurde diesbezüglich als Mindestanforderung festgelegt.

Anschließend wurden Entitätstypen von *MADEIRA* (insbesondere multidimensionale Datenräume, Maßzahlen und ihre Berechenbarkeit, Kategorien und ihre Aggregierbarkeit sowie hierüber definierte Dimensionen/Kategorienhierarchien) und eine differenzierte Algebra auf Datenräumen in einer formalen Notation im Detail spezifiziert. Weiterhin wurden den Konstrukten von *MADEIRA* zusätzliche Metadaten zugeordnet, die

einer besseren Benutzerinformation, einer einfacheren Implementierung und Anwendung sowie der Integration von Hintergrundinformationen in eine Datenanalyse dienen.

Schließlich erfolgte eine zusammenfassende Bewertung von *MADEIRA*, u. a. auf der Grundlage verschiedener Qualitätsmerkmale „guter“ Datenmodelle, die von Brodie in [Bro84] definiert werden.

Eine grundlegende Eigenschaft von multidimensionalen Datenräumen in *MADEIRA* besteht darin, daß sich jede ihrer Zellen als Wert einer Maßzahl über denjenigen Teilen betrachteter Objektmengen beschreiben läßt, die durch die jeweiligen Zellkoordinaten festgelegt werden. Diese Fokussierung auf die Kernidee der Betrachtung multidimensionaler, *aggregierter* Datenbestände ermöglicht eine relativ einfache Definition der Datensemantik, hat jedoch auch gewisse Einschränkungen der Menge von *MADEIRA* unterstützter Maßzahlen zur Folge. So können etwa heuristische, schätzungsbasierte Disaggregierungen und Konvertierungen, Sortierungen, Ränge, Anteile am Gesamtwert oder auch klassifizierende Gruppierungen, wie sie etwa für die Erstellung thematischer Karten benötigt werden, nicht oder nur „auf Umwegen“ dargestellt werden. Wie umfangreiche Beispiele aus der Krebsepidemiologie in Abschnitt 6.3 zeigen konnten, ist diese Beschränkung nicht gravierend, ja trägt sogar teilweise zur klareren Strukturierung von Analysen bei. Trotzdem könnte eine Evaluation von *VIOLA*, gerade auch in anderen Anwendungsdomänen, den Bedarf nach einer diesbezüglichen Erweiterung von *MADEIRA* aufdecken. Es müßte sich zeigen, inwieweit diese zumindest in Einzelfällen für bestimmte Maßzahlklassen ohne allzu tiefgreifende Neudefinitionen und vor allem unter Beibehaltung einer relativ intuitiv nutzbaren Datensemantik realisierbar wäre. Die Unterstützung *beliebiger* Berechnungsmöglichkeiten wäre in diesem Sinne sicherlich nicht möglich, ohne den grundlegenden Modellierungsansatz von *MADEIRA* zu unterlaufen bzw. weitgehend zu modifizieren.

In ähnlicher Weise wären auch die Angemessenheit des Verzichts auf Dimensionsattribute bzw. der Beschränkung auf flexible Kategorienhierarchien sowie der Bedarf einer weitergehenden Einbeziehung von Mikrodaten in die Datenanalyse noch zu evaluieren. Gerade zur umfassenden Unterstützung typischer Anwendungen des Data Warehousing wäre zumindest letzteres eine sinnvolle und konzeptionell auch nicht allzu schwierige Erweiterung von *MADEIRA*.

Einige Operatordefinitionen in *MADEIRA* gestalteten sich etwas komplizierter, um möglichst genau einer intuitiven Nutzung in typischen Analysesituationen zu entsprechen. Inwiefern dies in jedem Fall gelungen ist, muß ebenfalls die Nutzung von *VIOLA* im Routineeinsatz zeigen.

7.4 Mehr als „nur noch“ eine Benutzungsschnittstelle: *VIOLA*

VIOLA definiert eine datenflußbasierte Programmiersprache, deren wesentliche Bausteine (Module) sich (fast) unmittelbar aus den Operatoren von *MADEIRA* ergeben. Darüber hinaus sind — auf der Grundlage allgemeiner Überlegungen in den Kapiteln 2 und 3 — noch einfache Kontrollstrukturen, Abstraktionen von Teilprogrammen, Visualisierungsverfahren und Module zur Parameterdefinition in die Sprache integriert.

Analog zu Kapitel 4 diente auch in Kapitel 5 eine Klassifikation existierender datenflußbasierter Analyseumgebungen zur Charakterisierung und Einordnung der Entwurfsziele von *VIOLA*. Als Besonderheiten von *VIOLA* wurden vor allem die multidimensionale Datenanalyse als Anwendungsgebiet, die formale Fundierung der Analysesprache und die nahtlose Integration interaktiver Graphiken angeführt.

Die unterschiedlichen Modulklassen von *VIOLA* sowie die Verbindung von Modulen über Kanäle in Analyseprogrammen wurden formal definiert. Großer Wert wurde auf die flexible Parametrisierbarkeit von Operationen über nahezu beliebige *MADEIRA*–Entitäten gelegt. In diesem Zusammenhang wurden die eher elementaren, prozedural auf spezifische Datenräume anwendbaren Operatoren der *MADEIRA*–Algebra leicht spezialisiert bzw. zusammengefaßt, um in einer deklarativen Sichtweise jeweils gewünschte Analyseergebnisse spezifizieren zu können und somit eine gewisse Unabhängigkeit von den konkret zur Verfügung stehenden Basisdaten zu erlangen. Der Auswahl unterstützter Kontrollstrukturen lag ein Kompromiß zwischen Sprachmächtigkeit und intuitiver Nutzbarkeit auf einer abstrakten Analyseebene zugrunde. Komplexe Algorithmen sollen nicht durch *VIOLA*–Programme realisiert werden, sondern sind jeweils durch die Berechnungsfunktionalität einzelner Module zu kapseln.

Weiterhin erfolgte die Spezifikation einer flexiblen Verarbeitung von *VIOLA*–Programmen, die daten- und anforderungsgetriebene Elemente kombiniert, auf der Basis von modul- und kanalbezogenen Programmzuständen. Eine Grundidee bestand darin, zumindest das Schema eines jeweils durch ein Modul berechneten Datenraums zu jeder Zeit vorzuhalten bzw. bei Bedarf sofort zu aktualisieren, um den Datenanalysten bei der interaktiven Programmkonstruktion bestmöglich zu unterstützen.

Schließlich wurde noch auf die Benutzungsoberfläche eines graphischen Editors zur Erstellung von *VIOLA*–Programmen eingegangen, speziell die Repräsentation von Modulen, Kanälen und ihrer Zustände sowie die vielfältigen Möglichkeiten zur Interaktion mit dem Analysesystem und zur Abfrage von Informationen. Eine Bewertung des Sprachentwurfs, u. a. auf Basis der Kriterien von Green und Petre aus [GP96], bildete den Abschluß des Kapitels.

Wie schon für *MADEIRA* sind zukünftig auch einige der *VIOLA* betreffenden Entwurfsentscheidungen nochmals aufgrund umfangreicherer Erfahrungen aus dem praktischen Einsatz von *VIOLA* sowie einer eventuellen Übertragung in weitere Anwendungsdomänen zu bewerten und ggf. zu überdenken. Insbesondere ist zu prüfen, ob bei der Anpassung der *MADEIRA*–Operatoren sowie allgemein der Wahl der Abstraktionsebene, der Auswahl von Kontrollstrukturen und der Modulparametrisierung wirklich in jedem Fall ein adäquater Kompromiß zwischen intuitiver Nutzbarkeit, Mächtigkeit und exakter, einheitlicher Definierbarkeit gefunden wurde. Hierbei spielt natürlich auch die visuelle Repräsentation und Manipulierbarkeit von *VIOLA*–Programmen und ihrer Komponenten eine wesentliche Rolle. Ein weiterer Aspekt könnte die stärkere Berücksichtigung „schöner“ algebraischer Eigenschaften der Sprache sein. So sind derzeit etwa mehrfache, aufeinanderfolgende Anwendungen des gleichen Datenmanagementoperators aufgrund ihrer spezialisierten Parametrisierung in vielen Fällen nicht zu einer einzelnen Instanz dieses Operators zusammenzufassen, obwohl dies auf Basis von *MADEIRA* durchaus möglich wäre. Zum Beispiel sind zwei Selektionen oder Vereinigungen nicht durch ein einzelnes *VIOLA*–Modul zur Selektion bzw. Vereinigung zu beschreiben.

Als weitere Diskussionspunkte bzw. Erweiterungsmöglichkeiten seien genannt:

- die genauere Modellierung von Visualisierungen und ihrer Erzeugung,
- eine differenziertere Anwendbarkeitsprüfung (was natürlich auch Ergänzungen in *MADEIRA* nötig machen würde; wir kommen hierauf in Abschnitt 7.6 noch etwas näher zu sprechen),
- im gleichen Zusammenhang auch die Einführung mächtigerer Prüfungen der Anwendbarkeit von Verbundmodulen bzw. flexiblerer Parametrisierungen der Zuordnung von Verbundeingängen (vgl. etwa Abb. 6.6 in Abschnitt 6.3.2, wo evtl. ein Abgleich der Dimensionalität der eingehenden Datenräume sinnvoll sein könnte),
- eine Aufhebung der Trennung zwischen internen und externen Parametern — allerdings ohne hierdurch die Sprache *zu* mächtig bzw. Programme zu unübersichtlich werden zu lassen,
- etwas mächtigere Schleifen, die z. B. Schleifenergebnisse in den jeweils nächsten Durchlauf mit einfließen lassen,
- die flexiblere Unterstützung kategorieller Datenquellen, insbesondere zur Erzeugung *mehr*dimensionaler Datenräume (bisher können über diesen Operator lediglich *ein*dimensionale Datenräume generiert werden).

Die meisten dieser Aspekte wurden ganz bewußt bisher nicht bei der Gestaltung von *VIOLA* berücksichtigt, um die Sprache möglichst einfach zu halten. Sollte aber deutlich werden, daß typische Anwendungen die entsprechende Funktionalität tatsächlich benötigen, wären die entsprechenden Entwurfsentscheidungen zu revidieren. Grundlegende Umwälzungen des Sprachkonzepts von *VIOLA* würden sich hieraus jeweils nicht ergeben; alle genannten Modifikationen wären relativ „lokal“ zu realisieren.

7.5 Von der Theorie zur Praxis ...

Die vollständige Implementierung eines im Routinebetrieb einsetzbaren Analysesystems war nicht Gegenstand dieser Arbeit. Da bisher erst Erfahrungen mit prototypischen Umsetzungen von Teilkomponenten eines möglichen *VIOLA*–Gesamtsystems gesammelt werden konnten, muß die wesentliche Aufgabe zukünftiger Arbeiten in einer entsprechenden Implementierung und Evaluierung der Konzepte in der täglichen Anwendung bestehen.

In Kapitel 6 wurde eine Client–Server–Architektur zur Umsetzung des *VIOLA*–Entwurfs skizziert. Es wurden Aspekte der Anbindung verschiedener Datenquellen, der Bereitstellung von auf unterschiedliche Nutzergruppen abgestimmten Benutzungsschnittstellen sowie der Erweiterung von *VIOLA* durch Integration neuer Verfahren, aber auch über Schnittstellen zu externen Analysesystemen diskutiert.

Beim Entwurf von *MADEIRA* und *VIOLA* war — im Hinblick auf eine Beschränkung auf Datenbestände „mittlerer Größe“ (wie sie etwa in der Epidemiologie vorliegen) — der effizienten Umsetzbarkeit der vorgeschlagenen Konzepte eine nachrangige Bedeutung beigemessen worden. In Kapitel 6 wurde nun diese Lücke geschlossen, indem einige Ansätze zur Laufzeit- und Speicheroptimierung der Anfrage- bzw. Analyseverarbeitung vorgeschlagen wurden. Eine Berücksichtigung dieser Konzepte sollte auch eine Übertragung von *VIOLA* auf typische betriebswirtschaftliche Anwendungen des Data Warehousing möglich machen.

Konkret wurde gezeigt, wie bestehende Aufgabenstellungen zur Datenanalyse im Niedersächsischen Krebsregister mit Hilfe von *VIOLA* angegangen werden können. Hierbei wurde unterschieden nach Anbindung einer menübasierten Benutzungsoberfläche an *VIOLA* für Routineauswertungen, Erstellung eines standardisierten Jahresberichts, Unterstützung eines Monitoring der Erkrankungshäufigkeiten sowie einer interaktiven Analyse des Registerbestandes.

Bei der Nutzung von *VIOLA* zur Auswertung großer Datenbestände ist — bei aller möglichen Optimierung — darauf zu achten, ob das jeweilige Anwendungsszenario (vor dem Hintergrund aktuell typischerweise verfügbarer Rechnerressourcen) tatsächlich eine interaktive, schrittweise Datenanalyse zuläßt. Sofern nicht jeweils eine initiale Eingrenzung eines im Anschluß näher zu betrachtenden, unter Ermittlung verschiedener Maßzahlen explorativ zu analysierenden Teildatenraums vorgenommen werden kann, ist der von *VIOLA* verfolgte Analyseansatz nicht sinnvoll anwendbar. Hier können statt dessen gängige OLAP–Tools ihre Stärken bei der effizienten Datenbankabfrage voll ausspielen. Ein datenflußbasierter Analyseansatz würde nur unnötigen Overhead generieren, wenn die einzelnen Analyse*schritte* zu großen „Sprüngen“ im Datenraum werden.

Weiterhin darf bei der Diskussion der Skalierbarkeit von *VIOLA* zum Einsatz im Data Warehousing nicht in Vergessenheit geraten, daß längst nicht alle Anwendungen der multidimensionalen Datenanalyse auf Datenbasen im Giga- und Terabyte–Bereich basieren. Gerade diejenigen Applikationen, die bei der Auswertung klar abgegrenzter, eher kleiner Datenbestände bisher etwa auf Tabellenkalkulationen wie *EXCEL* oder Statistikpakete wie *SAS* oder *SPSS* (vor gar nicht allzu langer Zeit noch dateibasiert) zurückgegriffen haben, können sehr gut durch *VIOLA* unterstützt werden.

Das gesamte Kapitel 6 liefert Anknüpfungspunkte für eine Vielzahl interessanter weiterführender Arbeiten. Neben der Konkretisierung noch nicht im Detail betrachteter Architekturkomponenten sind speziell zu nennen

- tiefergehende Überlegungen zur Analyseoptimierung, die der interaktiven Manipulierbarkeit von Analysegraphen Rechnung tragen und das Wissen des Anwenders um *geplante* Analyseschritte einbeziehen, aber trotzdem die Grenze zwischen physischer und konzeptioneller Modellierungsebene nicht zu stark verschwimmen lassen,
- die Nutzung geeigneter DBMS und/oder OLAP–Tools zur effizienten Verwaltung multidimensionaler Daten sowie heterogener Metadaten (vgl. [KSW97]) bzw. die Integration von *VIOLA* in eine Gesamtarchitektur zum Data Warehousing,
- die Anbindung eines externen Metadatenbanksystems zur strukturierten Verwaltung und Bereitstellung noch umfangreicherer führender Metadaten sowie
- die Realisierung vielfältiger Schnittstellen zu externen Systemen als Grundlage verteilter Datenanalysen.

Im Rahmen einer Systemevaluation wäre weiterhin die Erfassung von Interaktionsprofilen repräsentativer Datenanalysesitzungen sehr interessant. Hierauf aufbauend könnte die Diskussion von Interaktionsmodellen aus Abschnitt 2.1.2 und Kapitel 3 vertieft und die Wahl der Analysebausteine in *VIOLA* ggf. überdacht werden.

7.6 Intelligente Datenanalyse mit *VIOLA*

Der Begriff der „intelligenten Datenanalyse“ hat viele Facetten. In dieser Arbeit steht in diesem Kontext im Vordergrund, *sinnvolle*, *differenzierte*, eben *intelligente* Auswertungen durch ein rechnergestütztes Analysesystem zu ermöglichen. Die eigentliche „Intelligenz“ steuert der jeweilige menschliche Systemanwender zur Durchführung einer Analyse bei: Er gibt die Analyseziele und Arbeitshypothesen vor, er spielt seine überragenden kognitiven Fähigkeiten bei der Identifikation interessanter Auffälligkeiten und Zusammenhänge im Datenbestand aus, und er trifft die Entscheidungen über die Auswahl des jeweils nächsten Analyseschrittes. Damit dies alles jedoch weitgehend problemlos möglich ist, muß das Analysesystem (hier *VIOLA*) ihm die jeweilige Analysesituation, also die Semantik betrachteter Datensätze und Verfahren, die Abfolge bereits ausgeführter und die Palette und Möglichkeiten als nächstes auswählbarer Analyseschritte möglichst detailliert, umfassend und intuitiv zugänglich machen. Dies ermöglicht dem Datenanalysten zum einen, sich auf die eigentliche Datenexploration zu konzentrieren, Analyseschritte frei zu parametrisieren und zu kombinieren, und stellt zum anderen eine exakte Spezifikation und Dokumentation einer Analysesitzung sicher.

Ein anderer Aspekt intelligenter Datenanalyse, nämlich die Bereitstellung und Nutzung *an sich* intelligenter Datenanalysesoftware, spielte für *VIOLA* bisher keine zentrale Rolle. Derartige statistische Expertensysteme versuchen (zumindest teilweise), dem Datenanalysten einige der Aufgaben abzunehmen bzw. ihn bei deren Durchführung zu beraten, die oben gerade der menschlichen Intelligenz zugeordnet wurden. Eine interessante, untersuchenswerte Fragestellung wäre es, inwiefern *VIOLA* in diese Richtung gehend sinnvoll erweitert werden könnte. In der in Kapitel 5 beschriebenen Form bietet *VIOLA* (mit der Unterstützung intelligenter Datenanalysen im erstgenannten Sinne) eine solide und breite *Basis* für ein statistisches Expertensystem. Herauszuheben sind hierbei

- die genaue semantische Beschreibung von Daten, Verfahren und Analysesitzungen,
- die Definition eines Typsystems zum Summary Type Management, das der Überprüfung der Aggregierbarkeit bzw. der automatischen (impliziten) Aggregierung von Maßzahlen dient,
- die polymorphe Anwendung von Analyse- und Visualisierungsverfahren (durch die flexible Spezifikation alternativer Berechnungsfunktionen bzw. Symboldefinitionen) sowie
- das Angebot von Vorschlägen zur Methodenwahl auf Grundlage der Definitionen von Berechenbarkeit bzw. Anwendbarkeit/Nutzbarkeit von Maßzahlen und Visualisierungsverfahren.

Um die „Expertise“ von *VIOLA* in Richtung eines leistungsfähigen statistischen Expertensystems verbessern zu können, wären (als konzeptionelle Grundlage für die Integration des entsprechenden Wissens) u. a. folgende Aspekte zu bearbeiten:

- die Ergänzung der bisher rein maßzahlbezogenen Anwendbarkeitsprüfungen um dimensions- und daten-, also extensionsbezogene Regeln (etwa die Prüfung statistischer Verteilungseigenschaften eines Datenraums),
- vor allem im Hinblick auf datenbezogene Prüfungen die Aufweichung der ja/nein–Beurteilung von Anwendbarkeit zu einer abgestuften Skala von „nicht anwendbar“ bis „sehr gut anwendbar“ (evtl. als zusätzliche Komponente der Berechnungsfunktionen einer Maßzahl),
- die differenziertere Prüfung der Anwendbarkeit von Verbundmodulen sowie allgemein die gewichtete Kombination von Bewertungen aufgrund verschiedener Anwendbarkeitstests,
- die dynamische Einbindung benutzerdefinierter, subjektiver Anwendbarkeitsprüfungen[1] etwa als neue

[1] Gerade die datenbezogenen Tests sind oft nicht objektiv beurteilbar.

Module ähnlich einer bedingten Verzweigung oder über interaktiv manipulierbare Gewichtungsfaktoren von Maßzahl–Berechnungsfunktionen[2],

- die Einbeziehung von benutzerdefinierten Analysezielen bei der Anwendbarkeitsüberprüfung bzw. bei der Beurteilung der Eignung eines Analyseverfahrens in einer gegebenen Analysesituation,
- die automatisierte Ergänzung von *VIOLA*–Programmen um einzelne Module (u. U. auf Anforderung), wenn Vorschläge anwendbarer Verfahren eindeutige Folgeschritte eines Teilgraphen festlegen,
- die Definition eines Kontrollmoduls, das dynamisch das auf den jeweiligen Datenraum am besten anwendbare Verfahren einer gegebenen Verfahrensklasse wählt, sowie
- die Implementierung von „Strategien“ zur Datenanalyse.

Die Definition und Umsetzung von Strategien zur Datenanalyse ist ein schon seit langer Zeit vieldiskutiertes Thema in der Statistik (siehe etwa [Pre86, Hub86, YL95, AC98]). Hand definiert in [Han97b]:

> *Statistical strategy describes steps, decisions, and actions which are taken during the process of analysing data to build a model or answer a question.*

Eine andere Definition findet sich in [YL95]:

> *A statistical strategy is a formal representation of an expert statistician's conceptual structuring of the data–analysis procedures to accomplish a specified data–analysis task, the data analyst's actions [...], and the relationships between the procedures and actions [...].*

Eine statistische Strategie gibt somit (ähnlich zu Entwurfsmustern im Software–Engineering [GHJV96] oder auch — in gewissem Rahmen — Geschäftsprozessen in der betriebswirtschaftlichen Anwendung [VB96]) einen vom Analyseexperten formulierten, mehr oder weniger groben Leitfaden für die Bearbeitung bestimmter Klassen von Analyseaufgaben vor. Zur oftmals geforderten Formalisierung statistischer Strategien kann *VIOLA* einen Beitrag leisten, indem für ausgewählte Problemstellungen Analysegraphen bzw. Verbundmodule unter Verwendung allgemeiner Maßzahlklassen vordefiniert und dem Anwender über eine Bibliothek zur Auswahl angeboten oder sogar automatisch aufgrund ihrer jeweiligen Anwendbarkeit vorgeschlagen werden (vgl. hierzu auch [GR91, AWK94]). Durch Spezialisierung der jeweiligen Maßzahlen, Parametrisierung und ggf. Ergänzung des so hinzugefügten Analysegraphen um zusätzliche Module wird eine Strategie dann anschließend auf die jeweilige konkrete Aufgabenstellung angewendet.

Fazit

Es stehen sicher noch einige Arbeiten hinsichtlich erweiterter Funktionalität, besserer Anwendungsschnittstellen und Übertragung auf verschiedene Anwendungsdomänen aus, bis alle sinnvollen Möglichkeiten ausgeschöpft sind, *VIOLA* zu einem umfassenden System zur Unterstützung intelligenter Datenanalysen zu machen. Jedoch wurde bereits in dieser Arbeit mit der detaillierten Sprachmodellierung und vor allem mit der Befriedigung der Anforderungen einer konkreten Anwendungsdomäne, nämlich der Krebsepidemiologie, eine solide Grundlage sowohl für weitere Arbeiten als auch für die Nutzung von *VIOLA* gelegt. Denn — um mit einem Zitat von Hand aus [Han97b] zu schließen:

> *It is by solving <u>real</u> problems that one changes the world.*

[2] Hierbei ist darauf zu achten, daß *gar nicht* anwendbare Verfahren stets diesen Status behalten, um die Semantik der Maßzahldefinition zu bewahren.

Literaturverzeichnis

[AAD+96] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan und S. Sarawagi. On the computation of multidimensional aggregates. In *22nd International Conference on Very Large Data Bases (VLDB)*, S. 506–521, 1996.

[AB89] A. L. Ambler und M. M. Burnett. Influence of visual technology on the evolution of language environments. *IEEE Computer*, 22(10):9–22, 1989. Auch in [Gli90b].

[AB90] A. L. Ambler und M. M. Burnett. Visual forms of iteration that preserve single assignment. *Journal on Visual Languages and Computing*, 1(2):159–181, 1990.

[AB92] M. Anderson und F. Berman. Removing useless tokens from a dataflow computation. In [Sha92b], Kapitel 11, S. 292–331.

[AB96] F. E. Alexander und P. Boyle. *Methods for Investigating Localized Clustering of Disease*. IARC Scientific Publications No. 135. International Agency for Research on Cancer, Lyon, 1996.

[ABJK94] H.-J. Appelrath, H. Behrends, H. Jasper und V. Kamp. Active database technology supports cancer clustering. In *1st International Conference on Applications of Databases (ADB)*, S. 351–364. Springer Verlag, LNCS 819, 1994.

[AC96a] R. S. Amant und P. R. Cohen. Control representation in an EDA assistant. In D. H. Fisher und H. Lenz, Hrsg., *Learning from Data. 5th International Workshop on Artificial Intelligence and Statistics (1995)*. Springer Verlag, LNS 112, 1996. Siehe auch http://eksl-www.cs.umass.edu/∼stamant/publications.html.

[AC96b] R. S. Amant und P. R. Cohen. A planner for exploratory data analysis. In *3rd International Conference on AI Planning Systems (AIPS)*, S. 205–212, 1996.

[AC98] R. S. Amant und P. R. Cohen. Intelligent support for exploratory data analysis. *Journal of Computational and Graphical Statistics*, 7(4):545–558, 1998. http://www.amstat.org/publications/jcgs/pdf_98/Amant.pdf.

[ACM92] *The Full Computing Reviews Classification System: 1991 Version*. Association for Computing Machinery (ACM), New York, 1992.

[ACSW96] A. Aiken, J. Chen, M. Stonebraker und A. Woodruff. Tioga–2: A direct manipulation database visualization environment. In *12th International Conference on Data Engineering (ICDE)*, S. 208–217, 1996. Siehe auch http://tioga.cs.berkeley.edu:8000/tioga/.

[AD97] M. Auguston und A. Delgado. Iterative constructs in the Visual Data Flow Language. In *13th IEEE Symposium on Visual Languages*, S. 154–161, 1997.

[AGL98] J. Albrecht, H. Günzel und W. Lehner. An architecture for distributed OLAP. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, 1998. Siehe auch http://www6.informatik.uni-erlangen.de/.

[AGS97] R. Agrawal, A. Gupta und S. Sarawagi. Modeling multidimensional databases. In *13th International Conference on Data Engineering (ICDE)*, S. 232–243, 1997.

[Ahl98] R. Ahlborn. Eine ORACLE–Schnittstelle zur Verwaltung multidimensionaler statistischer Objekte für CARESS. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, Mai 1998.

[AKGM96] B. Adelberg, B. Kao und H. Garcia-Molina. Database support for efficiently maintaining derived data. In *Advances in Database Technology — 5th International Conference on Extending Database Technology (EDBT)*, S. 223–240. Springer Verlag, LNCS 1057, 1996.

[Ala97] C. Alalouf. Hybrid OLAP. The Best of Both Worlds. White paper, Speedware Corporation, November 1997. http://www.speedware.com/pubs/misc/holap.pdf.

[Ama97] R. S. Amant. Navigation for data analysis systems. In *Advances in Intelligent Data Analysis — Reasoning about Data. 2nd International Symposium (IDA)*, S. 101–109. Springer Verlag, LNCS 1280, 1997.

[And89] J. R. Anderson. *Kognitive Psychologie: Eine Einführung*. Spektrum der Wissenschaft Verlagsgesellschaft, 1989.

[Arb95] Relational OLAP: Expectations and Reality. White paper, Arbor Software (heute Hyperion), 1995. Siehe auch http://www.hyperion.com/.

[AT95] G. Abram und L. Treinish. An Extended Data–Flow Architecture for Data Analysis and Visualization. Research Report RC 20001 (88338), IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, 1995.

[AVS96] Advanced Visual Systems, Waltham, MA. *AVS/Express User's Guide. Release 3.0*, 1996.

[AWK94] Z. Ahmed, L. Wanger und P. Kochevar. An intelligent visualization system for earth science data analysis. *Journal on Visual Languages and Computing*, 5(4):307–320, 1994.

[BA94] M. M. Burnett und A. L. Ambler. Interactive visual data abstraction in a declarative visual programming language. *Journal on Visual Languages and Computing*, 5(1):29–60, 1994.

[BA96] R. J. Brachman und T. Anand. The process of knowledge discovery in databases: A human–centered approach. In [FPSSU96], S. 37–57.

[Bar92] E. Barszcz. New loop control structures for static data flow. In [Sha92b], Kapitel 15, S. 389–403.

[Bas85] M. A. Bassiouni. Data compression in scientific and statistical databases. *IEEE Transactions on Software Engineering*, SE–11(10):1047–1058, 1985.

[Bau94] P. Baumann. Management of multidimensional discrete data. *VLDB Journal. Special Issue on Spatial Database Systems*, 4(3):401–444, 1994.

[BB94] M. M. Burnett und M. J. Baker. A classification system for visual programming languages. *Journal on Visual Languages and Computing*, 5(3):287–300, 1994.

[BBB+95] M. M. Burnett, M. J. Baker, C. Bohus, P. Carlson, S. Yang und P. van Zee. Scaling up visual programming languages. *IEEE Computer*, 28(3):45–54, 1995.

[BBC+99] P. A. Bernstein, T. Bergstraesser, J. Carlson, S. Pal, P. Sanders und D. Shutt. Microsoft Repository Version 2 and the Open Information Model. White paper, Microsoft Corporation, Redmond, März 1999. Siehe auch http://msdn.microsoft.com/repository/.

[BCW88] R. A. Becker, W. S. Cleveland und A. R. Wilks. Dynamic graphics for data analysis. In [CM88], Kapitel 1, S. 1–72.

[Bec94] R. A. Becker. A brief history of S. In [DO94], S. 81–110.

[Ber74] J. Bertin. *Graphische Semiologie*. Walter de Gruyter Verlag, 1974. Übersetzung der 2. Auflage des französischen Originals (1973).

[Ber96] P. A. Bernstein. Middleware. *Communications of the ACM*, 39(2):86–98, 1996.

[Ber97] P. A. Bernstein. Repositories and object oriented databases. In *7. GI–Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, S. 34–46. Springer Verlag, 1997.

[BG89] R. D. Bergeron und G. Grinstein. A reference model for the visualization of multidimensional data. In *10th European Conference on Computer Graphics (Eurographics)*, S. 393–399, 1989. Entnommen aus [Lee94].

[BG96] T. C. Bailey und A. C. Gatrell. *Interactive Spatial Data Analysis*. Longman Group Ltd., 1996.

[BGL95] M. M. Burnett, A. Goldberg und T. G. Lewis, Hrsg. *Visual Object–oriented Programming*. Manning Publications, 1995.

[BH99] M. Berthold und D. J. Hand, Hrsg. *Intelligent Data Analysis: An Introduction*. Springer Verlag, 1999.

[BJM97] F.-J. Brandenburg, M. Jünger und P. Mutzel. Algorithmen zum automatischen Zeichnen von Graphen. *Informatik Spektrum*, 20(4):199–207, 1997.

[BL93] B. Bell und C. Lewis. ChemTrains: A language for creating behaving pictures. In *9th IEEE Symposium on Visual Languages*, S. 188–195, 1993.

[Bla96] A. F. Blackwell. Metacognitive theories of visual programming: What do we think we are doing? In *12th IEEE Symposium on Visual Languages*, S. 240–246, 1996.

[BM95] M. M. Burnett und D. W. McIntyre, Hrsg. *IEEE Computer. Special Issue on Visual Programming*, 28(3), 1995.

[BMG89] P. Boyle, C. S. Muir und E. Grundmann, Hrsg. *Cancer Mapping*. Recent Results in Cancer Research (Vol. 114). Springer Verlag, 1989.

[BMR94] A. Bezenchek, F. Massari und M. Rafanelli. Storm+: Statistical data storage and manipulation system. In *11th Symposium on Computational Statistics (Compstat)*, S. 351–356, 1994.

[Boc92] H. H. Bock. Grundlegende Methoden der exploratorischen Datenanalyse. In [EGHW92], S. 15–42.

[Bol94] D. Boles. Das IMRA–Modell — Integration von Interaktionen in das Autorensystem FMAD. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, September 1994.

[Bol95] D. Boles. Elektronisches Publizieren — Autorensysteme und Arbeitsumgebungen für Autoren. *Zeitschrift für Informationswissenschaft und -praxis*, 46(5):273–282, 1995.

[Bor81] A. Borning. The programming language aspects of ThingLab, a constraint–oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4):353–387, 1981. Auch in [Gli90b].

[Bor92] M. Borovcnik. Diskussion zu [Boc92]. In [EGHW92], S. 43–46.

[Bra95] R. Braham. Math and visualization: New tools, new frontiers. *IEEE Spectrum*, 32(11):19–36, 1995.

[Bra98a] N. Bradley. *The XML Companion*. Addison Wesley, 1998. Siehe auch http://xml.com/ oder http://www.w3.org/XML/.

[Bra98b] M. Brandt. Einbindung von interaktiven Graphiken in eine visuelle Programmierumgebung für die Datenanalyse. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, September 1998.

[Bre94] F. Bretherton. Reference Model for Metadata: A Strawman. Technischer Bericht, University of Wisconsin, Februar 1994. http://www.llnl.gov/liv_comp/metadata/working_grp.html.

[Bro84] M. L. Brodie. On the development of data models. In M. L. Brodie, J. Mylopoulos und J. W. Schmidt, Hrsg., *On Conceptual Modelling*, Kapitel 2, S. 19–47. Springer Verlag, 1984.

[BRT96] A. Bezenchek, M. Rafanelli und L. Tininini. A data structure for representing aggregate data. In *8th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 22–31, 1996. Siehe auch http://www.iasi.rm.cnr.it/∼antonia/db-papers.html.

[BS94] F. P. Bretherton und P. T. Singley. Metadata: A user's view. In *7th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 166–174, 1994.

[BS98] D. Boles und M. Schlattmann. Multimedia–Autorensysteme: Grafisch–interaktive Werkzeuge zur Erstellung multimedialer Anwendungen. *LOG IN*, 18(1):10–18, 1998.

[BSHD98] M. Blaschka, C. Sapia, G. Höfling und B. Dinter. Finding your way through multidimensional data models. In *International Workshop on Data Warehouse Design and OLAP Technology (DWDOT) im Rahmen der DEXA'98*, S. 198–203, 1998. Auch unter http://www.forwiss.tu-muenchen.de/∼system42/publications.

[BST+93] R. J. Brachman, P. G. Selfridge, L. G. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D. L. McGuinness und L. A. Resnick. Integrated support for data archaeology. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):159–185, 1993.

[BSV+95] F. Berrino, M. Sant, A. Verdecchia, R. Capocaccia, T. Hakulinen und J. Estève. *Survival of Cancer Patients in Europe*. IARC Scientific Publications No. 132. International Agency for Research on Cancer, Lyon, 1995.

[Bur93] M. M. Burnett. Types and type inference in a visual programming language. In *9th IEEE Symposium on Visual Languages*, S. 238–243, 1993.

[Bur95] M. M. Burnett. Seven programming language issues. In [BGL95], Kapitel 8, S. 161–181.

[BW97] N. Becker und J. Wahrendorf. *Krebsatlas der Bundesrepublik Deutschland 1981–1990*. Springer Verlag, 3. Auflage, 1997.

[CCLB97] T. Catarci, M. F. Costabile, S. Levialdi und C. Batini. Visual query systems for databases: A survey. *Journal on Visual Languages and Computing*, 8(2):215–260, 1997.

[CCS93] E. F. Codd, S. B. Codd und C. T. Salley. Providing OLAP (On–Line Analytical Processing) to User Analysts: An IT Mandate. White paper, Arbor Software (jetzt Hyperion), 1993. http://www.hyperion.com/whitepapers.cfm.

[CCS96] A Guide to Statistical Software. George Mason University, Center for Computational Statistics, 1996. http://www.galaxy.gmu.edu/papers/astr1.html.

[CD97] S. Chaudhuri und U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1):65–74, 1997.

[CDGL+99] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi und R. Rosati. A principled approach to data integration and reconciliation in data warehousing. In *International Workshop on Design and Management of Data Warehouses (DMDW) im Rahmen der CAISE'99*, S. 16.1–16.11, 1999. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/.

[CDZ94] W. Citrin, M. Doherty und B. Zorn. Formal semantics of control in a completely visual programming language. In *10th IEEE Symposium on Visual Languages*, S. 208–215, 1994.

[CDZ95] W. Citrin, M. Doherty und B. Zorn. The design of a completely visual OOP language. In [BGL95], Kapitel 4, S. 67–94.

[CED+93] M. Coleman, J. Estève, P. Damiecki, A. Arslan und H. Renard. *Trends in Cancer Incidence and Mortality*. IARC Scientific Publications No. 121. International Agency for Research on Cancer, Lyon, 1993.

[CGN+91] C. Crimi, A. Guercio, G. Nota, G. Pacini, G. Tortora und M. Tucci. Relation grammars and their application to multi–dimensional languages. *Journal on Visual Languages and Computing*, 2(4):333–346, 1991.

[Cha81] J. M. Chambers. Some thoughts on expert software. In *Computer Science and Statistics. 13th Symposium on the Interface*, S. 36–39. Springer Verlag, 1981.

[Cha87] S.-K. Chang. Visual languages: A tutorial and survey. *IEEE Software*, 4(1):29–39, 1987.

[Cha93] J. M. Chambers. Classes and methods in S. *Computational Statistics*, 8(3):167–196, 1993.

[Cha95] J. M. Chambers. Overview of Version 4 of S. Technischer Bericht, AT&T Bell Laboratories, 1995. http://www.research.att.com/areas/stat/doc/95.1.ps.

[Che76] P. P. Chen. The entity relationship model — towards a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[CKLI94] S.-K. Chang, R. R. Korfhage, S. Levialdi und T. Ichikawa. Ten years of visual language research. In *10th IEEE Symposium on Visual Languages*, S. 196–205, 1994.

[Cla98] N. Clausen. *OLAP. Multidimensionale Datenbanken — Produkte, Markt, Funktionsweisen und Implementierung*. Addison Wesley, 1998.

[Cle93] W. S. Cleveland. *Visualizing Data*. Hobart Press, 1993.

[Cle94] W. S. Cleveland. *The Elements of Graphing Data*. Hobart Press, 2. Auflage, 1994.

[CLW+90] S. Chowdhury, R. Linnarsson, A. Wallgren, B. Wallgren und O. Wigertz. Extracting knowledge from a large primary health care database using a knowledge–based statistical approach. *Journal of Medical Systems*, 14(4):213–225, 1990.

[CM84] W. S. Cleveland und R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, September 1984.

[CM88] W. S. Cleveland und M. E. McGill, Hrsg. *Dynamic Graphics for Statistics*. Wadsworth & Brooks / Cole Advanced Books and Software, 1988.

[CM89] M. C. Chen und L. P. McNamee. On the data model and access method of summary data management. *IEEE Transactions on Knowledge and Data Engineering*, 1(4):519–529, 1989.

[CMCI82] R. Chen, N. Mantel, R. R. Connelly und P. Isacson. A monitoring system for chronic diseases. *Methods of Information in Medicine*, 21(2):86–90, 1982.

[CMM89] M. C. Chen, L. McNamee und M. Melkanoff. A model of summary data and its applications in statistical databases. In *Statistical and Scientific Database Management. Proceedings of the Fourth International Working Conference (SSDBM)*, S. 356–387. Springer Verlag, LNCS 339, 1989.

[CN97] F. Cribari-Neto. Econometric programming environments: GAUSS, Ox and S–PLUS. *Journal of Applied Econometrics*, 12(1):77–89, 1997. http://jae.wiley.com/jae.

[Con97] S. Conrad. *Föderierte Datenbanksysteme. Konzepte der Datenintegration*. Springer Verlag, 1997.

[CP88] P. T. Cox und T. Pietrzykowski. Using a pictorial representation to combine dataflow and object–orientation in a language independent programming mechanism. In *International Computer Science Conference*, S. 695–704, 1988. Auch in [Gli90b].

[CRS85] D. Canter, R. Rivers und G. Storrs. Characterizing user navigation through complex data structures. *Behaviour and Information Technology*, 4(2):93–102, 1985.

[CS81] P. Chan und A. Shoshani. SUBJECT: A directory driven system for large statistical databases. In *LBL Workshop on Statistical Database Management (S(S)DBM)*, 1981.

[CS94] S. Chaudhuri und K. Shim. Including group–by in query optimization. In *20th International Conference on Very Large Data Bases (VLDB)*, S. 354–366, 1994.

[CS96] W. Citrin und C. Santiago. Incorporating fisheyeing into a visual programming environment. In *12th IEEE Symposium on Visual Languages*, S. 20–27, 1996.

[CT97] L. Cabibbo und R. Torlone. Querying multidimensional databases. In *6th International Workshop on Database Programming Languages (DBPL)*, S. 319–335. Springer Verlag, LNCS 1369, 1997. Auch unter http://poincare.inf.uniroma3.it:8080/Persone/cabibbo.html/pub/.

[CT98] L. Cabibbo und R. Torlone. A logical approach to multidimensional databases. In *Advances in Database Technology — 6th International Conference on Extending Database Technology (EDBT)*, S. 183–197. Springer Verlag, LNCS 1377, 1998.

[CT99] L. Cabibbo und R. Torlone. A framework for the investigation of aggregate functions in database queries. In *7th International Conference on Database Theory (ICDT)*, S. 383–397, 1999.

[CTODL95] G. Costagliola, G. Tortora, S. Orefice und A. De Lucia. Automatic generation of visual programming environments. *IEEE Computer*, 28(3):56–66, 1995.

[Dad96] P. Dadam. *Verteilte Datenbanken und Client/Server–Systeme*. Springer Verlag, 1996.

[DBB88] G. Di Battista und C. Batini. Design of statistical databases: A methodology for the conceptual step. *Information Systems*, 13(4):407–422, 1988.

[DDI97] On Line Data Mining — Data Distilleries' Vision on Data Mining Technology. White Paper DD-R9704, Data Distilleries BV, Mai 1997. http://www.ddi.nl/pub/wp19971.htm.

[Def89] D. Defays. Statistics and artificial intelligence. In [Did89], S. 381–388.

[DEV97] DEVise Development Group. *DEVise User Manual. Version 1.3.4.* University of Wisconsin–Madison, März 1997. http://www.cs.wisc.edu/~devise/userman.ps.gz.

[DF95] K. M. Decker und S. Focardi. Technology Overview: A Report on Data Mining. Technical Report CSCS TR-95-02, Swiss Center for Scientific Computing, ETH Zürich, Manno, Mai 1995. http://www.cscs.ch/official/TechReports/1995/CSCS-TR-95-02.ps.gz.

[DFJ+96] S. Dar, M. J. Franklin, B. T. Jonsson, D. Srivastava und M. Tan. Semantic data caching and replacement. In *22nd International Conference on Very Large Data Bases (VLDB)*, S. 330–341, 1996.

[DH98] J. B. Dabney und T. L. Harman. *The Student Edition of SIMULINK. Dynamic System Simulation for MATLAB. User's Guide, Version 2*. The MathWorks, 1998.

[Did89] E. Diday, Hrsg. *Data Analysis, Learning Symbolic and Numeric Knowledge*. Nova Science Publishers, 1989.

[DKR97] M. Derthick, J. Kolojejchick und S. F. Roth. An interactive visual query environment for exploring data. In *ACM Symposium on User Interface Software and Technology (UIST)*, S. 189–198, 1997.

[dM94] V. de Mey. *Visual Composition of Software Applications*. Dissertation, Université de Genève, 1994. http://cuiwww.unige.ch/OSG/publications/OO-articles/vicki/THESIS.

[DO94] P. Dirschedl und R. Ostermann, Hrsg. *Computational Statistics*. Physica Verlag, 1994.

[DRR+96] S. Dogru, V. Rajan, K. Rieck, J. R. Slagle, B. S. Tjan und Y. Wang. A graphical data flow language for retrieval, analysis, and visualization of a scientific database. *Journal on Visual Languages and Computing*, 7(3):247–265, 1996.

[DSHB98] B. Dinter, C. Sapia, G. Höfling und M. Blaschka. The OLAP market: State of the art and research issues. In *ACM 1st International Workshop on Data Warehousing and OLAP (DOLAP) im Rahmen der CIKM'98*, 1998. http://www.pages.drexel.edu/faculty/songiy/dolap98.html.

[DT97] A. Datta und H. Thomas. A conceptual model and an algebra for on–line analytical processing. In *7th Workshop on Information Technologies and Systems (WITS) im Rahmen der ICIS'97*, 1997.

[dTSS86] S. H. C. du Toit, A. G. W. Steyn und R. H. Stumpf. *Graphical Exploratory Data Analysis*. Springer Verlay, 1986.

[Dye90] D. S. Dyer. A dataflow toolkit for visualization. *IEEE Computer Graphics and Applications*, 10(4):60–69, 1990.

[EBR94] J. Estève, E. Benhamou und L. Raymond. *Statistical Methods in Cancer Research. Volume IV. Descriptive Epidemiology*. IARC Scientific Publications No. 128. International Agency for Research on Cancer, Lyon, 1994.

[EE98] G. Eddon und H. Eddon. *Inside Distributed COM*. Microsoft Press Deutschland, 1998.

[EGHW92] H. Enke, J. Gölles, R. Haux und K.-D. Wernecke, Hrsg. *Methoden und Werkzeuge für die explo-ratorische Datenanalyse in den Biowissenschaften*. Gustav Fischer Verlag, 1992.

[EKN00] *Krebs in Niedersachsen. Jahresbericht 1996*. Epidemiologisches Krebsregister Niedersachsen. OFFIS / Niedersächsisches Ministerium für Frauen, Arbeit und Soziales, Oldenburg/Hannover, erscheint voraussichtlich im 2. Quartal 2000. Siehe auch http://www.krebsregister-niedersachsen.de.

[EM95] M. Erwig und B. Meyer. Heterogeneous visual languages — integrating visual and textual programming. In *11th IEEE Symposium on Visual Languages*, S. 318–325, 1995.

[EN94] R. Elmasri und S. B. Navathe. *Fundamentals of Database Systems*. The Benjamin Cummings Publishing Company, 2. Auflage, 1994.

[FFH96] K. A. Froeschl, L. Fetz und W. Hennrich. A metadata based client–server approach to the integration of distributed statistical databases. In *Advances in Statistical Software 5 (SoftStat'95)*, S. 259–266. Lucius & Lucius, 1996.

[FL90] J. Fox und J. S. Long, Hrsg. *Modern Methods of Data Analysis*. Sage Publications, 1990.

[FMENR89] G. Falcitelli, L. Meo Evoli, E. Nardelli und F. L. Ricci. The Mefisto* model: An object oriented representation for statistical data management. In [Did89], S. 455–463.

[FMTW98] B. Ford, G. Morgan, A. Trefethen und J. Walton. STABLE: A visual programming environment for statistical computing. In *3rd International Seminar on New Techniques and Technologies for Statistics (NTTS)*. Eurostat, 1998.

[FPSS96a] U. Fayyad, G. Piatetsky-Shapiro und P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[FPSS96b] U. M. Fayyad, G. Piatetsky-Shapiro und P. Smyth. From data mining to knowledge discovery: An overview. In [FPSSU96], S. 1–34.

[FPSSU96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth und R. Uthurusamy, Hrsg. *Advances in Knowledge Discovery and Data Mining*. AAAI Press / The MIT Press, 1996.

[Fre91] J. C. French. Support for scientific database management. In [Mic91b], Kapitel 4, S. 51–82.

[Fro96] K. A. Froeschl. A metadata approach to statistical query processing. *Statistics and Computing*, 6(1):11–29, 1996.

[Fro97] K. A. Froeschl. *Metadata Management in Statistical Information Processing*. Springer Verlag, 1997.

[Frö98] M. Fröhlich. *Inkrementelles Graphlayout im Visualisierungssystem daVinci*. Shaker Verlag, 1998. Siehe auch http://www.informatik.uni-bremen.de/∼inform/forschung/daVinci/daVinci.html.

[FRW+97] P. Furtado, R. Ritsch, N. Widmann, P. Zoller und P. Baumann. Object–oriented design of a database engine for multidimensional discrete data. In *International Conference on Object Oriented Information Systems (OOIS)*, S. 411–421. Springer Verlag, 1997.

[FS98] M. Fowler und K. Scott. *UML konzentriert*. Addison Wesley, 1998. Siehe auch http://www.omg.org oder http://www.rational.com/uml.

[FU+96] U. Fayyad, R. Uthurusamy et al. Data mining and knowledge discorvery in databases. *Communications of the ACM*, 39(11):24–68, 1996.

[GA97] D. W. Gillman und M. V. Appel. Building a statistical metadata repository. In *2nd IEEE Metadata Conference*, 1997.

[Gal86] W. A. Gale, Hrsg. *Artificial Intelligence and Statistics*. Addison Wesley, 1986.

[GAL96] D. W. Gillman, M. V. Appel und W. P. LaPlant, Jr. Design principles for a unified statistical data/metadata system. In *8th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 150–155, 1996. http://www.computer.org/proceedings/meta97/.

[GBLP96] J. Gray, A. Bosworth, A. Layman und H. Pirahesh. Data Cube: A relational operator generalizing group–by, cross–tab, and sub–totals. In *12th International Conference on Data Engineering (ICDE)*, S. 152–159, 1996.

[Ger96] C. A. Gerlhof. *Optimierung von Speicherzugriffskosten in Objektbanken: Clustering und Prefetching*. Dissertation, Universität Passau, Fakultät für Mathematik und Informatik, 1996.

[GG98] R. Gabriel und P. Gluchowski. Grafische Notationen für die semantische Modellierung multidimensionaler Datenstrukturen in Management Support Systemen. *Wirtschaftsinformatik*, 40(6):493–502, 1998.

[GHJV96] E. Gamma, R. Helm, R. Johnson und J. Vlissides. *Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software*. Addison Wesley, 1996. Deutsche Übersetzung von Dirk Riehle.

[GHK+96] M. Ganesh, E.-H. Han, V. Kumar, S. Shekhar und J. Srivastava. Visual Data Mining: Framework and Algorithm Development. Technical Report 96-021, University of Minnesota, Department of Computer and Information Sciences, 1996. http://www.cs.umn.edu/∼njain/mining/papers/datavis.ps.

[Gho86] S. P. Ghosh. Statistical Relational Tables for statistical database management. *IEEE Transactions on Software Engineering*, SE-12(12):1106–1116, 1986.

[Gho88] S. P. Ghosh. Statistics metadata. In S. Kotz und N. L. Johnson, Hrsg., *Encyclopedia of Statistical Sciences, Vol. 8*, S. 743–746. John Wiley & Sons, 1988.

[Gho91] S. P. Ghosh. Statistical relational model. In [Mic91b], Kapitel 10, S. 267–305.

[GHR97] H. Gupta, V. Harinarayan und A. Rajaraman. Index selection for OLAP. In *13th International Conference on Data Engineering (ICDE)*, S. 208–219, 1997.

[GL97] M. Gyssens und L. V. S. Lakshmanan. A foundation for multi–dimensional databases. In *23rd International Conference on Very Large Data Bases (VLDB)*, S. 106–115, 1997.

[Gli90a] E. P. Glinert, Hrsg. *Visual Programming Environments: Applications and Issues*. IEEE Computer Society Press, 1990.

[Gli90b] E. P. Glinert, Hrsg. *Visual Programming Environments: Paradigms and Systems*. IEEE Computer Society Press, 1990.

[GMPS96] C. Glymour, D. Madigan, D. Pregibon und P. Smyth. Statistical inference and data mining. *Communications of the ACM*, 39(11):35–41, 1996.

[GMR98] M. Golfarelli, D. Maio und S. Rizzi. Conceptual design of data warehouses from E/R schemes. In *31st Hawaii International Conference on Science Systems (HICSS)*, S. 334–343, 1998.

[GMS+97] O. Günther, R. Müller, P. Schmidt, H. Bhargava und R. Krishnan. MMM: A WWW–based approach for sharing statistical computing modules. *IEEE Internet Computing*, 1(3), 1997. http://www.computer.org/internet/ic1997/w3toc.htm. Siehe auch http://mmm.wiwi.hu-berlin.de/mmm.

[GN95] W. H. Graf und S. Neurohr. Constraint-based layout in visual program design. In *11th IEEE Symposium on Visual Languages*, S. 116–117, 1995.

[Gol66] S. W. Golomb. Run–length encodings. *IEEE Transactions on Information Theory*, 12(7):399–401, 1966.

[Goo83] I. J. Good. The philosophy of exploratory data analysis. *Philosophy of Science*, 50(2):283–295, 1983.

[GP82] W. A. Gale und D. Pregibon. An expert system for regression analysis. In *Computer Science and Statistics. 14th Symposium on the Interface*, S. 110–117. Springer Verlag, 1982.

[GP96] T. R. G. Green und M. Petre. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal on Visual Languages and Computing*, 7(2):131–174, 1996.

[GQ94] M. Gorlick und A. Quilici. Visual programming–in–the–large versus visual programming–in–the–small. In *10th IEEE Symposium on Visual Languages*, S. 137–144, 1994.

[GR90] E. J. Golin und S. P. Reiss. The specification of visual language syntax. *Journal on Visual Languages and Computing*, 1(2):141–157, 1990.

[GR91] M. M. Gorlick und R. R. Razouk. Using weaves for software construction and analysis. In *13th International Conference on Software Engineering (ICSE)*, S. 23–34, 1991.

[Gri98] F. Griffel. *Componentware. Konzepte und Techniken eines Softwareparadigmas*. dpunkt–Verlag, 1998.

[GRKM94] J. Goldstein, S. F. Roth, J. Kolojejchick und J. Mattis. A framework for knowledge–based interactive data exploration. *Journal on Visual Languages and Computing*, 5(4):339–364, 1994.

[Gro98] W. Grossmann. Use of metadata in the statistical production process. In *3rd International Seminar on New Techniques and Technologies for Statistics (NTTS)*. Eurostat, 1998.

[Gru97] R. Grupe. Cacheabhängige Optimierungsstrategien für eine Client/Server basierte KDD–Architektur. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, Januar 1997.

[GT84] E. P. Glinert und S. L. Tanimoto. Pict: An interactive graphical programming environment. *IEEE Computer*, 17(11):7–25, 1984. Auch in [Gli90b].

[Haa94] U. Haag. Knowledge–based systems in statistics: A tutorial overview with examples. In [DO94], S. 211–236.

[Haa95] V. Haarslev. Formal semantics of visual languages using spatial reasoning. In *11th IEEE Symposium on Visual Languages*, S. 156–163, 1995.

[Hae88] P. E. Haeberli. ConMan: A visual programming language for interactive graphics. *ACM Computer Graphics (Proceedings SIGGRAPH'88)*, 22(4):103–111, 1988.

[Ham95] Hamburgisches Krebsregister. *Hamburger Krebsdokumentation 1989-91*. Freie und Hansestadt Hamburg, Behörde für Arbeit, Gesundheit und Soziales, Fachabteilung Gesundheitsberichterstattung, Dezember 1995.

[Han86] D. J. Hand. Patterns in statistical strategy. In [Gal86], S. 355–387.

[Han92] D. J. Hand. Microdata, macrodata and metadata. In *10th Symposium on Computational Statistics (Compstat)*, S. 325–340, 1992.

[Han94] W. J. Hansen. The 1994 visual languages comparison. In *10th IEEE Symposium on Visual Languages*, S. 90–97, 1994.

[Han97a] J. Han. OLAP mining: An integration of OLAP with data mining. In *IFIP Conference on Data Semantics (DS–7)*, S. 1–11, 1997. Siehe auch http://db.cs.sfu.ca/DBMiner.

[Han97b] D. J. Hand. Intelligent data analysis: Issues and opportunities. In *Advances in Intelligent Data Analysis — Reasoning about Data. 2nd International Symposium (IDA)*, S. 1–14. Springer Verlag, LNCS 1280, 1997.

[Här87] T. Härder. Realisierung von operationalen Schnittstellen. In [LS87], Kapitel 3, S. 163–335.

[Hau86] R. Haux, Hrsg. *Expert Systems in Statistics*. Gustav Fischer Verlag, 1986.

[HBL93] W. Hellmeier, H. Brand und U. Laaser. Epidemiologische Methoden der Gesundheitswissenschaften. In K. Hurrelmann und U. Laaser, Hrsg., *Gesundheitswissenschaften: Handbuch für Lehre, Forschung und Praxis*, S. 91–110. Beltz Verlag, 1993.

[HDP93] W. L. Hibbard, C. R. Dyer und B. E. Paul. The VIS–AD data model: Integrating metadata and polymorphic display with a scientific programming language. In *Database Issues for Data Visualization. IEEE Visualization'93*, S. 37–68, 1993.

[HEK91] J. Hartung, B. Elpelt und K.-H. Klösener. *Statistik — Lehr- und Handbuch der angewandten Statistik*. Oldenbourg Verlag, 8. Auflage, 1991.

[Hel99] C. Helberg. Statistics on the Web, 1999. http://www.execpc.com/∼helberg/statistics.html.

[Heu97] A. Heuer. *Objektorientierte Datenbanken — Konzepte, Modelle, Standards und Systeme.* Addison Wesley, 2. Auflage, 1997.

[HH99] O. Herden und A. Harren. MML und mUML — Sprache und Werkzeug zur Unterstützung des konzeptionellen Data Warehouse–Designs. In *2. GI–Workshop Data Mining und Data Warehousing als Grundlage moderner entscheidungsunterstützender Systeme (DMDW) im Rahmen der Workshoptage LWA'99*, 1999. Auch unter http://www.informatik.uni-oldenburg.de/∼charlii/paper/.

[HHK92] U. Haag, R. Haux und M. Kieser. *Statistische Auswertungssysteme. Eine Einführung in ihre Anwendung, Konstruktion und Bewertung*. Gustav Fischer Verlag, 1992.

[HI82] P. Hájek und J. Iványek. Artificial intelligence and data analysis. In *5th Symposium on Computational Statistics (Compstat)*, S. 54–59. International Association for Statistical Computing, 1982.

[HI94] M. Hirakawa und T. Ichikawa. Visual language studies — a perspective. *Software — Concepts and Tools*, 15(1):61–67, 1994.

[Hil93] D. D. Hils. Visual languages and computing survey: Data flow visual programming languages. *Journal on Visual Languages and Computing*, 3(1):69–101, 1993.

[Hin87] H. Hinterberger. *Data Density — A Powerful Abstraction to Manage and Analyse Multivariate Data*. Dissertation, Verlag der Fachvereine an den Schweizerischen Hochschulen und Techniken, Zürich, 1987.

[Hin99] H. Hinrichs. Metadata–based quality management of warehouse data. In *19th Annual Conference on Current Trends in Databases and Information Systems (DATASEM)*, S. 239–248, 1999.

[HLC91] R. B. Haber, B. Lucas und N. Collins. A data model for scientific visualization with provisions for regular and irregular grids. In *IEEE Visualization*, S. 298–305, 1991.

[HO91] C. B. Hurley und R. W. Oldford. A software model for statistical graphics. In A. Buja und P. A. Tukey, Hrsg., *Computing and Graphics in Statistics*, S. 77–94. Springer Verlag, 1991.

[Hös97] H.-P. Höschel. *Datenanalyse und Data Mining mit dem SAS System. Release 6.12. The Orlando II Release.* SAS Institute, Heidelberg, September 1997. Siehe auch http://www.sas.de/.

[HRU96] V. Harinarayan, A. Rajaraman und J. Ullman. Implementing data cubes efficiently. In *ACM SIGMOD International Conference on Management of Data*, S. 205–216, 1996.

[HS95] A. Heuer und G. Saake. *Datenbanken. Konzepte und Sprachen.* Thomson Publishing, 1995.

[HTI90] M. Hirakawa, M. Tanaka und T. Ichikawa. Hi–Visual iconic programming environment. In [IJK90], Kapitel 6, S. 121–145.

[HTU97] H. Hofmann, M. Theus und A. Unwin. MANET Overview. On–line Dokumentation, Universität Augsburg, 1997. http://www1.math.uni-augsburg.de/Manet.

[HU90] J. E. Hopcroft und J. D. Ullman. *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. Addison Wesley, 1990.

[Hub86] P. J. Huber. Environments for supporting statistical strategy. In [Gal86], S. 285–294.

[Hub94] P. J. Huber. Languages for statistics and data analysis. In [DO94], S. 53–80.

[Hüs99] R. Hüsing. Verarbeitungsmodell einer datenflußorientierten Anfragesprache zur Datenanalyse. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, März 1999.

[HV95] M. Horn und R. Vollandt. *Multiple Tests und Auswahlverfahren.* Gustav Fischer Verlag, 1995.

[HW97a] W. R. Hendee und P. N. T. Wells, Hrsg. *The Perception of Visual Information*. Springer Verlag, 2. Auflage, 1997.

[HW97b] I. Hoting und G. Windus. EKN (Epidemiologisches Krebsregister Niedersachsen) — Rahmenkonzept für einen Jahresbericht. Technischer Bericht, Niedersächsisches Sozialministerium und OFFIS, Hannover/Oldenburg, Dezember 1997.

[HYM+92] S. Herath, Y. Yamaguchi, R. Mattingley, J. Herath, N. Saito und T. Yuba. Functional and logic languages in dataflow computing. In [Sha92b], Kapitel 4, S. 78–100.

[HYT+92] J. Herath, Y. Yamaguchi, K. Toda, R. Mattingley, N. Saito und T. Yuba. Comparison of data flow computing models. In [Sha92b], Kapitel 2, S. 16–34.

[IBM97] IBM, T. J. Watson Research Center, Yorktown Heights, NY. *Visualization Data Explorer Documentation. Version 3.1.4*, 1997. On–line Handbuch unter http:/www.ibm.com/dx/documentation.html.

[IJK90] T. Ichikawa, E. Jungert und R. R. Korfhage, Hrsg. *Visual Languages and Applications*. Plenum Press, 1990.

[IM96] T. Imielinski und H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.

[Inm92] W. H. Inmon. *Building the Data Warehouse.* John Wiley & Sons, 1992.

[ISF94] E. Ignatius, H. Senay und J. Favre. An intelligent system for task–specific visualization assistance. *Journal on Visual Languages and Computing*, 5(4):321–338, 1994.

[JJQV99] M. Jarke, M. A. Jeusfeld, C. Quix und P. Vassiliadis. Architecture and quality in data warehouses: An extended repository approach. *Information Systems. Special Issue on Advanced Information Engineering*, 24(3):229–253, 1999.

[JS82] G. Jäschke und H.-J. Scheck. Remarks on the algebra of non first normal form relations. In *1st ACM Symposium on Principles of Database Systems (PODS)*, S. 124–138, 1982.

[JT98] R. Jaworski und A. Totok. Modellierung von multidimensionalen Datenstrukturen mit ADAPT. Technischer Bericht, Technische Universität Braunschweig, Institut für Wirtschaftswissenschaften, Juli 1998. http://www.tu-bs.de/institute/wirtschaftswi/controlling/staff/totok/.

[KAW+93] P. Kochevar, Z. Ahmed, L. Wanger, C. Shade und J. Sharp. A Visualization Architecture for the Sequoia 2000 Project. Sequoia 2000 Technical Report 93/35, University of California, Berkeley, September 1993. http://s2k-ftp.cs.berkeley.edu:8000/sequoia/tech-reports/.

[KB96] A. M. Keller und J. Basu. A predicate–based caching scheme for client–server database architectures. *VLDB Journal*, 5(1):35–47, 1996.

[KBS94] D. T. Kao, R. D. Bergeron und T. M. Sparr. An extended schema model for scientific data. In *Database Issues for Data Visualization. IEEE Visualization'93*, S. 69–82. Springer Verlag, LNCS 871, 1994.

[KDN99] KDNuggets. Tools for Data Mining and Knowledge Discovery, 1999. http://www.kdnuggets.com.

[Kel98] C. Keltsch. Ein visuelles Programmiersystem zur Modellierung deskriptiver Untersuchungen in der Datenanalyse. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, September 1998.

[Ken95] An Introduction to Multidimensional Database Technology. White paper, Kenan Technologies, 1995. Nicht mehr zugreifbar unter http://www.kenan.com/.

[KHA97] J. D. Kiper, E. Howard und C. Ames. Criteria for evaluation of visual programming languages. *Journal on Visual Languages and Computing*, 8(2):175–192, 1997.

[Kim96] R. Kimball. *The Data Warehouse Toolkit*. John Wiley & Sons, 1996.

[KK95] D. A. Keim und H.-P. Kriegel. Visualisierungstechniken zur Exploration und Analyse sehr großer Datenbanken. In *6. GI–Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, S. 262–281. Springer Verlag, 1995.

[KKS94] D. A. Keim, H.-P. Kriegel und T. Seidl. Supporting data mining of large databases by visual feedback queries. In *10th International Conference on Data Engineering (ICDE)*, S. 302–313, 1994. Siehe auch http://www.dbs.informatik.uni-muenchen.de/dbs/projekt/visdb/visdb.html.

[KMK95] Y. Koike, Y. Maeda und Y. Koseki. Improving readability of iconic programs with multiple view object representation. In *11th IEEE Symposium on Visual Languages*, 1995.

[KMR91] J. Kodosky, J. MacCrisken und G. Rymar. Visual programming using structured data flow. In *7th IEEE Workshop on Visual Languages*, S. 34–39, 1991. Siehe auch http://www.natinst.com/labview.

[Knu73] D. E. Knuth. *The Art of Computer Programming. Fundamental Algorithms*. Addison Wesley, 1973.

[Kor93] K. Kornbluh. Active data analysis: Advanced software for the 90's. *IEEE Spectrum*, 30(11):57–83, 1993.

[KS95] L. Kreienbrock und S. Schach. *Epidemiologische Methoden*. Gustav Fischer Verlag, 1995.

[KS97] J.-P. Kent und M. Schuerhoff. Some thoughts about a metadata management system. In *9th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 174–185, 1997.

[KSHK97] M. L. Kersten, A. P. J. M. Siebes, M. Holsheimer und F. Kwakkel. Research and business challenges in data mining technology. In *7. GI–Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, S. 1–16. Springer Verlag, 1997.

[KSW97] V. Kamp, L. Sitzmann und F. Wietek. A spatial data cube concept to support data analysis in environmental epidemiology. In *9th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 100–103, 1997.

[LA97] W. Lehner und J. Albrecht. Anfrageverarbeitung in multidimensionalen Datenbanksystemen. In *42. Internationales Wissenschaftliches Kolloquium Technische Universität Ilmenau*, 1997. Auch unter http://www6.informatik.uni-erlangen.de/.

[LAC+92] B. Lucas, G. D. Abram, N. S. Collins, D. A. Epstein, D. L. Gresh und K. P. McAuliffe. An architecture for a scientific visualization system. In *IEEE Visualization*, S. 107–114, 1992. Siehe auch http://www.ibm.com/dx/.

[Lar86] J. A. Larson. A visual approach to browsing in a database environment. *IEEE Computer*, 19(6):62–71, 1986.

[LAW98] W. Lehner, J. Albrecht und H. Wedekind. Normal forms for multidimensional databases. In *10th International Conference on Scientific and Statistical Database Management (SSDBM)*, S. 63–72, 1998.

[Lee94] J. P. Lee. Data exploration interactions and the ExBase system. In *Database Issues for Data Visualization. IEEE Visualization'93*, S. 118–137. Springer Verlag, LNCS 871, 1994. Siehe auch http://www.cs.uml.edu/∼jlee/exbase.html.

[Leh98] W. Lehner. Modeling large scale OLAP scenarios. In *Advances in Database Technology — 6th International Conference on Extending Database Technology (EDBT)*, S. 153–167. Springer Verlag, LNCS 1377, 1998.

[Leh99] W. Lehner. *Multidimensionale Datenbanksysteme*. Teubner Verlag, 1999.

[Len94a] H.-J. Lenz. M3–database design, micro-, macro- and metadatabase modelling. In *Advances in Statistical Software 4 (SoftStat'93)*, S. 441–452. Gustav Fischer Verlag, 1994.

[Len94b] H.-J. Lenz. The conceptual schema and external schemata of metadatabases. In *7th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 160–165, 1994.

[Leo82] A. N. Leontjew. *Tätigkeit, Bewußtsein, Persönlichkeit*. Pahl–Rugenstein Verlag, 1982.

[LG96] J. P. Lee und G. G. Grinstein. Describing visual interactions to the database: Closing the loop between user and data. In *Visual Data Exploration and Analysis III (SPIE'96)*, S. 93–103, 1996. Auch unter http://www.cs.uml.edu/∼jlee/exbase.html.

[Liu96] X. Liu. Intelligent data analysis: Issues and challenges. *The Knowledge Engineering Review*, 11(4):365–372, 1996.

[Lod83] K. N. Lodding. Iconic interfacing. *IEEE Computer Graphics and Applications*, 3(2):11–20, 1983. Auch in [Gli90a].

[LP88] D. Lubinsky und D. Pregibon. Data analysis as search. *Journal of Econometrics*, 38:247–268, 1988.

[LP98] P. G. Lever und J. S. Perin. AVS/Express Module Writing. An Introductory Course. Technischer Bericht, International AVS Centre, University of Manchester, 1998. http://www.iavsc.org/express/courses/modwrite-xp.

[LRB+97] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, s. Lawande, J. Myllymaki und K. Wenger. DEVise: Integrated querying and visual exploration of large datasets. In *ACM SIGMOD International Conference on Management of Data*, S. 301–312, 1997. Siehe auch http://www.cs.wisc.edu/∼devise.

[LS87] P. C. Lockemann und J. W. Schmidt, Hrsg. *Datenbank–Handbuch*. Springer Verlag, 1987.

[LS97] H.-J. Lenz und A. Shoshani. Summarizability in OLAP and statistical data bases. In *9th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 132–143, 1997.

[LW92] J. Lehn und H. Wegmann. *Einführung in die Statistik*. Teubner Verlag, 1992.

[LW96] C. Li und X. S. Wang. A data model for supporting on–line analytical processing. In *5th International Conference on Information and Knowledge Management (CIKM)*, S. 81–88, 1996.

[Mac86] J. MacKinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[Man96] H. Mannila. Data mining: Machine learning, statistics, and databases. In *8th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 2–9, 1996.

[Mar97] S. Marx. *Datenmanagement in wissensbasierten Statistiksystemen*. Dissertation, Universität Mannhein, Fakultät für Betriebswirtschaftslehre, 1997.

[McC82] J. L. McCarthy. Metadata management for large statistical databases. In *8th International Conference on Very Large Databases (VLDB)*, S. 234–243, 1982.

[MDB87] B. H. McCormick, T. A. DeFanti und M. D. Brown, Hrsg. Visualization in Scientific Computing. *ACM SIGGRAPH. Computer Graphics Special Issue*, 21(6), 1987.

[MDC99] The Open Information Model. On–line Dokumentation, Meta Data Coalition, August 1999. http://www.mdcinfo.com.

[MDL87] H. C. Mayr, K. R. Dittrich und P. C. Lockemann. Datenbankentwurf. In [LS87], Kapitel 5, S. 481–557.

[Mei99] J. Meister. Entwurf einer verteilten Architektur zur Unterstützung der datenflußorientierten Anfrageverarbeitung in der explorativen Datenanalyse. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, Mai 1999.

[Men96] T. Menzies. Visual programming, knowledge engineering, and software engineering. In *8th ACM International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 1996. Langfassung als Technical Report TR96-5 unter http://www.sd.monash.edu.au/research/publications/.

[MERS92] L. Meo-Evoli, F. Ricci und A. Shoshani. On the semantic completeness of macro–data operators for statistical aggregation. In *6th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 239–258, 1992.

[Mic91a] Z. Michalewicz. Security of a statistical database. In [Mic91b], Kapitel 13, S. 357–387.

[Mic91b] Z. Michalewicz, Hrsg. *Statistical and Scientific Databases*. Ellis Horwood Ltd., 1991.

[Mic95] The Case for Relational OLAP. White paper, MicroStrategy, 1995. http://www.microstrategy.com/Publications/WhitePapers/.

[Mil56] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–96, 1956. Auch in [Gli90a].

[Mit95] B. Mitschang. *Anfrageverarbeitung in Datenbanksystemen*. Vieweg Verlag, 1995.

[Mit97] T. Mitchell. *Machine Learning*. McGraw–Hill, 1997.

[MJPM+91] O. Møller Jensen, D. M. Parkin, R. MacLennan, C. Muir und R. G. Skeet, Hrsg. *Cancer Registration: Principles and Methods*. IARC Scientific Publications No. 95. International Agency for Research on Cancer, Lyon, 1991.

[MKHI95] E. Miller, M. Kado, M. Hirakawa und T. Ichikawa. HI–VISUAL as a user–customizable visual programming environment. In *11th IEEE Symposium on Visual Languages*, S. 107–113, 1995.

[MMS96] D. Maier, M. E. Meredith und L. Shapiro. Bringing knowledge to bear: Challenges for decision support databases. In *Multimedia, Knowledge–Based and Object–Oriented Databases. 7th Hong Kong Computer Society Conference*, S. 1–15, 1996.

[MP70] B. MacMahon und T. F. Pugh. *Epidemiology — Principles and Methods*. Little, Brown & Company, 1970.

[MSR99] R. Müller, T. Stöhr und E. Rahm. An integrative and uniform model for metadata management in data warehousing environments. In *International Workshop on Design and Management of Data Warehouses (DMDW) im Rahmen der CAISE'99*, S. 12.1–12.16, 1999. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/.

[MT89] S. Morgenthaler und J. W. Tukey. The next future of data analysis. In [Did89], S. 1–14.

[MT90] K. Matsumura und S. Tayama. Visual man–machine interface for program design and production. In [IJK90], S. 99–119.

[MWS+91] M. Michalewicz, S. T. Wierzchon, E. Syropiatko, A. Pacan, A. Matuszewski und M. Klopotek. Statistical expert systems for data analysis and knowledge acquisition. In [Mic91b], Kapitel 17, S. 431–463.

[Mye86] B. A. Myers. Visual programming, programming by example, and program visualization: A taxonomy. In *Human Factors in Computing Systems. ACM Conference on Computer–Human Interaction (CHI)*, S. 59–66, 1986. Auch in [Gli90b].

[Mye90] B. A. Myers. Taxonomies of visual programming and program visualization. *Journal on Visual Languages and Computing*, 1(1):97–123, 1990.

[NAG96] Numerical Algorithms Group (NAG), Oxford. *IRIS Explorer Documentation. Version 3.5*, 1996. On–line Handbuch unter http://www.iec.co.uk/visual/IE/iecbb/DOC/Index.html.

[Naj94] M. A. Najork. *Programming in Three Dimensions*. Dissertation, University of Illinois at Urbana-Champaign, 1994. http://www.research.digital.com/SRC/personal/Marc_Najork/thesis.

[NBOH96] M. Nagel, A. Benner, R. Ostermann und K. Henschke. *Grafische Datenanalyse*. Gustav Fischer Verlag, 1996.

[Nin97] U. Nink. Using the STEP standard and databases in science. In *9th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 196–207, 1997.

[NS73] I. Nassi und B. Shneiderman. Flowchart techniques for structured programming. *ACM SIGPLAN Notices*, S. 12–26, August 1973.

[Obe94] H. Oberquelle. Formen der Mensch–Computer–Interaktion. In E. Eberleh, H. Oberquelle und R. Oppermann, Hrsg., *Einführung in die Software–Ergonomie*, S. 95–143. Walter de Gruyter Verlag, 2. Auflage, 1994.

[OG95] P. O'Neil und G. Graefe. Multi–table joins through bitmapped join indices. *SIGMOD Record*, 24(3):8–11, 1995.

[OHE98] R. Orfali, D. Harkey und J. Edwards. *Instant CORBA*. Addison Wesley, 1998. Siehe auch http://www.omg.org.

[Ola98] MDAPI — the OLAP Application Program Interface Version 2.0. Technischer Bericht, The OLAP Council, Januar 1998. http://www.olapcouncil.org/research/apico.htm.

[Ole96] J. Olenski. Practical problems of implementing metadata standards in official statistics. In *8th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 130–148, 1996.

[OMG99] Meta Object Facility (MOF) Specification. OMG document ad/99-09-04, Object Management Group, 1999. Siehe auch http://www.omg.org/store/publications.html.

[ÖÖ85] G. Özsoyoglu und Z. M. Özsoyoglu. Statistical database query languages. *IEEE Transactions on Software Engineering*, SE-11(10):1071–1081, 1985.

[ÖÖM87] G. Özsoyoglu, Z. M. Özsoyoglu und V. Matos. Extending relational algebra and relational calculus with set–valued attributes and aggregate functions. *ACM Transactions on Database Systems*, 12(4):566–592, 1987.

[Ora97] Oracle Corporation. *Oracle Express Analyzer Briefing Designer's Guide. Release 2.1*, 1997. Siehe auch http://www.oracle.com.

[P+94] J. Preece et al. *Human–Computer Interaction*. Addison Wesley, 1994.

[Par97] K. Parsaye. OLAP and data mining: Bridging the gap. *Database Programming and Design*, September 1997. http://www.dbpd.com/vault/parsfeb.htm.

[PC99] N. Pendse und R. Creeth. *The OLAP Report*. Business Intelligence Ltd., 1999. http://www.olapreport.com.

[PCF+94] D. M. Parkin, V. W. Chen, J. Ferlay, J. Galceran, H. H. Storm und S. L. Whelan. *Comparability and Quality Control in Cancer Registration*. IARC Technical Report No. 19. International Agency for Research on Cancer, Lyon, 1994.

[Pfl73] M. Pflanz. *Allgemeine Epidemiologie — Aufgaben, Technik, Methoden*. Georg Thieme Verlag, 1973.

[PH91] D. M. Parkin und T. Hakulinen. Analysis of survival. In [MJPM+91], S. 159–176.

[PHR+94] B. Pesch, U. Halekoh, U. Ranft, M. Richter und F. Pott. *Atlas zur Krebssterblichkeit in Nordrhein-Westfalen*. Medizinisches Institut für Umwelthygiene an der Heinrich-Heine-Universität Düsseldorf im Auftrag des Ministeriums für Arbeit, Gesundheit und Soziales des Landes Nordrhein-Westfalen, 1994.

[PJ94] C. Plaisant und V. Jain. Dynamaps: Dynamic queries on a health statistics atlas. In *Human Factors in Computing Systems. ACM Conference on Computer–Human Interaction (CHI)*, S. 439–440, 1994.

[PJ99] T. B. Pedersen und C. S. Jensen. Multidimensional data modeling for complex data. In *15th International Conference on Data Engineering (ICDE)*, S. 336–345, 1999.

[Pos94] J. Poswig. *Visuelle Programmiersprachen. Die Realisierung und konzeptionelle Weiterentwicklung eines Prototypen*. Dissertation, Universität Dortmund, Fachbereich Informatik, 1994.

[Pre86] D. Pregibon. A DIY guide to statistical strategy. In [Gal86], S. 389–399.

[PS94] B. Pagel und H. Six. *Software Engineering, Band 1: Die Phasen der Softwareentwicklung*. Addison Wesley, 1994.

[Rad97] N. Raden. Worlds in collision. *DBMS Online*, August 1997. http://www.dbms-mag.com/9708d14.html.

[Raf91] M. Rafanelli. Data models. In [Mic91b], Kapitel 6, S. 109–166.

[RBM+92] W. Ribarsky, B. Brown, T. Myerson, R. Feldmann, S. Smith und L. Treinish. Object–oriented dataflow visualization systems — a paradigm shift? In *IEEE Visualization*, S. 384–388, 1992.

[RCK+97] S. F. Roth, M. C. Chuah, S. Kerpedjiev, J. Kolojejchick und P. Lucas. Towards an information visualization workspace: Combining multiple means of expression. *Human–Computer Interaction*, 12(1+2):131–185, 1997. Siehe auch http://www.cs.cmu.edu/∼sage.

[Rei91] W. Reisig. *Petrinetze — Eine Einführung*. Springer Verlag, 1991.

[RF92] M. Rafanelli und F. Ferri. VIDDEL: An object–oriented visual data definition language for statistical data. In *6th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 18–28, 1992.

[RM90] S. F. Roth und J. Mattis. Data characterization for intelligent graphics presentation. In *Human Factors in Computing Systems. ACM Conference on Computer–Human Interaction (CHI)*, S. 193–200, 1990.

[Rog97] P. A. Rogerson. Surveillance systems for monitoring the development of spatial patterns. *Statistics in Medicine*, 16(18):2081–2093, 1997.

[Rot86] K. J. Rothman. *Modern Epidemiology*. Little, Brown & Company, 1986.

[Rot90] K. J. Rothman. A sobering start for the cluster–buster's conference. *American Journal of Epidemiology*, 132, Supplement:S6–S13, 1990.

[RR83] M. Rafanelli und F. L. Ricci. Proposal of a logical model for statistical data base. In *2nd International Workshop on Statistical Database Management (S(S)DBM)*, S. 264–272, 1983.

[RR93] M. Rafanelli und F. L. Ricci. Mefisto: A functional model for statistical entities. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):670–681, 1993.

[RS90] M. Rafanelli und A. Shoshani. STORM: A statistical object representation model. In *Statistical and Scientific Database Management. Proceedings of the Fifth International Conference (SSDBM)*, S. 14–29. Springer Verlag, LNCS 420, 1990.

[RS97] J. Rekers und A. Schürr. Defining and parsing visual languages with layered graph grammars. *Journal on Visual Languages and Computing*, 8(1):27–55, 1997.

[Ruf97] T. Ruf. *Scientific & Statistical Databases — Datenbankeinsatz in der multidimensionalen Datenanalyse*. Vieweg Verlag, 1997.

[Rum99] F. J. Rump. *Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten*. Teubner Verlag, 1999.

[RW91] J. R. Rasure und C. S. Williams. An integrated data flow visual language and software development environment. *Journal on Visual Languages and Computing*, 2(3):217–246, 1991. Siehe auch http://www.khoral.com/.

[RWW99] M. Rohde, F. Wietek und I. Wellmann. *Benutzerdokumentation CARESS, Version 4.0*. OFFIS, November 1999. http://www.krebsregister-niedersachsen.de/ekn_de_publikationen.html.

[RZ96] D. Rotem und J. L. Zhao. Extendible arrays for statistical databases and OLAP applications. In *8th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 108–117, 1996.

[Saa96] *Morbidität und Mortalität an bösartigen Neubildungen im Saarland 1993 — Jahresbericht des Saarländischen Krebsregisters*. Statistisches Landesamt Saarland, Mai 1996.

[SAM98] S. Sarawagi, R. Agrawal und N. Megiddo. Discovery–driven exploration of OLAP data cubes. In *Advances in Database Technology — 6th International Conference on Extending Database Technology (EDBT)*, S. 168–182. Springer Verlag, LNCS 1377, 1998.

[San96] G. Sander. Layout of Compound Directed Graphs. Technischer Bericht, Universität des Saarlandes, Saarbrücken, Juni 1996. Siehe auch http://www.cs.uni-sb.de/RW/users/sander/html/gspapers.html.

[Sap99] C. Sapia. On modeling and predicting user behavior in OLAP systems. In *International Workshop on Design and Management of Data Warehouses (DMDW) im Rahmen der CAISE'99*, S. 2.1–2.10, 1999. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-19/.

[Sar97] S. Sarawagi. Indexing OLAP data. *Bulletin of the Technical Committee on Data Engineering*, 20(1):36–43, 1997.

[SAS93] SAS Institute, Heidelberg. *SAS/ASSIST Software, Deutsche Version*, 1993. Siehe auch http://www.sas.de.

[Sat81] H. Sato. Handling summary information in a database: Derivability. In *ACM SIGMOD International Conference on Management of Data*, S. 98–107, 1981.

[SBHD98] C. Sapia, M. Blaschka, G. Höfling und B. Dinter. An Overview of Multidimensional Data Models for OLAP. Technischer Bericht, Bayerisches Forschungszentrum für wissensbasierte Syteme (FORWISS), Technische Universität München, August 1998. http://www.forwiss.tu-muenchen.de/∼system42/publications.

[SBHD99] C. Sapia, M. Blaschka, G. Höfling und B. Dinter. Extending the E/R model for the multidimensional paradigm. In *Advances in Database Technologies. International Workshop on Data Warehousing and Data Mining (DWDM'98) im Rahmen der ER'98*, S. 105–116. Springer Verlag, LNCS 1552, 1999.

[SBM92] R. R. Springmeyer, M. M. Blattner und N. L. Max. A characterization of the scientific data analysis process. In *IEEE Visualization*, S. 235–242, 1992.

[Sch98] S. Schiffer. *Visuelle Programmierung. Grundlagen und Einsatzmöglichkeiten*. Addison Wesley, 1998.

[Sch99] M. Schrandt. Automatisierte Steuerung von Applikationen durch CARESS bei der Visualisierung von Daten. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, Juni 1999.

[SCN+93] M. Stonebraker, J. Chen, N. Nathan, C. Paxson und J. Wu. Tioga: Providing data management support for scientific visualization applications. In *19th International Conference on Very Large Data Bases (VLDB)*, S. 25–38, 1993.

[SDJL96] D. Srivastava, S. Dar, H. V. Jagadish und A. Y. Levy. Answering queries with aggregation using views. In *22nd International Conference on Very Large Data Bases (VLDB)*, S. 318–329, 1996.

[Sha92a] J. A. Sharp. A brief introduction to data flow. In [Sha92b], Kapitel 1, S. 1–15.

[Sha92b] J. A. Sharp, Hrsg. *Data flow computing: Theory and Practice*. Ablex Publishing Corporation, 1992.

[SHDA90] R. R. Sitter, L. P. Hanrahan, D. Demets und H. A. Anderson. A monitoring system to detect increased rates of cancer incidence. *American Journal of Epidemiology*, 132(1):123–130, 1990.

[Shn83] B. Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69, 1983.

[Shn87] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human–Computer Interaction*. Addison Wesley, 1987.

[Shn94] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, 1994.

[Shn96] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *12th IEEE Symposium on Visual Languages*, S. 336–343, 1996.

[Sho82] A. Shoshani. Statistical databases: Characteristics, problems, and some solutions. In *8th International Conference on Very Large Databases (VLDB)*, S. 208–222, 1982.

[Sho97] A. Shoshani. OLAP and statistical databases: Similarities and differences. In *16th ACM Symposium on Principles of Database Systems (PODS)*, S. 185–196, 1997.

[Shu86] N. C. Shu. Visual programming languages. A perspective and a dimensional analysis. In S. K. Chang, T. Ichikawa und P. A. Ligomenides, Hrsg., *Visual Languages*, S. 11–34. Plenum Press, 1986. Auch in [Gli90b].

[Shu88] N. C. Shu. *Visual Programming*. Van Nostrand Reinhold, 1988. Zitiert in [Sch98].

[SKA94] S. Sengupta, T. D. Kimura und A. Apte. An artist's studio: A metaphor for modularity and abstraction in a graphical diagramming environment. In *10th IEEE Symposium on Visual Languages*, S. 128–136, 1994.

[SL90] I. Spence und S. Lewandowsky. Graphical perception. In [FL90], Kapitel 1, S. 13–57.

[SNB83] S. Y. W. Su, S. B. Navathe und D. S. Batory. Logical and physical modelling of statistical/scientific databases. In *2nd International Workshop on Statistical Database Management (S(S)DBM)*, S. 251–263, 1983.

[SNFH86] K. Sato, T. Nakano, Y. Fukasawa und R. Hotaka. Conceptual schema for a wide–scope statistical database and ist application. In *3rd International Workshop on Statistical and Scientific Database Management (SSDBM)*, S. 165–172, 1986.

[Sri92] V. P. Srini. An architectural comparison of dataflow systems. In [Sha92b], Kapitel 5, S. 101–139.

[SS94] S. Sarawagi und M. Stonebraker. Efficient organization of large multidimensional arrays. In *10th International Conference on Data Engineering (ICDE)*, S. 328–336, 1994.

[SSV96] P. Scheuermann, J. Shim und R. Vingralek. WATCHMAN: A data warehouse intelligent cache manager. In *22nd International Conference on Very Large Data Bases (VLDB)*, S. 51–62, 1996.

[SSW96] P. G. Selfridge, D. Srivastava und L. O. Wilson. IDEA: Interactive data exploration and analysis. In *ACM SIGMOD International Conference on Management of Data*, S. 24–35, 1996.

[Sto94] M. Stonebraker. SEQUOIA 2000 — A reflection on the first three years. In *7th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 108–116, 1994.

[SW89] T. Schäfer und H.-W. Wachtel. *Umweltbezogene Gesundheitsberichterstattung: Planungsstudie*. Asgard Verlag, 1989.

[SWZ95] A. Schürr, A. Winter und A. Zündorf. Visual programming with graph rewriting systems. In *11th IEEE Symposium on Visual Languages*, S. 326–333, 1995.

[Tan90] S. L. Tanimoto. VIVA: A visual language for image processing. *Journal on Visual Languages and Computing*, 1(2):127–139, 1990. Entnommen aus [Hil93].

[Tap99] H. Tapken. Anfrageverarbeitung und -optimierung im MOA–Kontext. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, März 1999.

[Ter93] L. G. Terveen. Intelligent systems as cooperative systems. *International Journal of Intelligent Systems. Special Issue on the Social Context of Intelligent Systems*, 3(2–4):217–250, 1993. Auch unter ftp://ftp.research.att.com/dist/terveen/ijis-93.ps.

[TG86] S. L. Tanimoto und E. P. Glinert. Designing iconic programming systems: Representation and learnability. In *2nd IEEE Workshop on Visual Languages*, S. 54–60, 1986.

[The95] M. Theus. Analysing Storm Data Using Highly Interactive Tools. Unveröffentlichter Preprint, Universität Augsburg, 1995. http://www1.math.uni-augsburg.de/∼theus/Storms/Main.

[The96a] M. Theus. Software review: Data Desk 5.0. *Computational Statistics*, 11(2), 1996.

[The96b] M. Theus. *Theorie und Anwendung Interaktiver Statistischer Graphik*. Dissertation, Universität Augsburg, Institut für Mathematik, Lehrstuhl für Rechnerorientierte Statistik und Datenanalyse, 1996.

[Tho97] E. Thomsen. *OLAP Solutions. Building Multidimensional Information Systems*. John Wiley & Sons, 1997.

[THSU95] M. Theus, H. Hofmann, B. Siegl und A. Unwin. MANET — Extensions to interactive statistical graphics for missing values. In *2nd International Conference on New Techniques and Technologies for Statistics (NTTS)*. Eurostat, 1995. http://www1.math.uni-augsburg.de/∼theus/papers.html.

[Tio97] Tioga Project. *DataSplash User Manual. Version 0.1*. University of California, Berkeley, 1997. On–line Handbuch unter http://s2k-ftp.cs.berkeley.edu:8000/tioga/datasplash_user_manual.html.

[TK78] D. C. Tsichritzis und A. C. Klug. The ANSI/X3/SPARC DBMS framework report of the study group on database management systems. *Information Systems*, 3(3):173–191, 1978.

[TLS99] D. Theodoratos, S. Ligoudistianos und T. Sellis. Designing the global data warehouse with SPJ views. In *11th International Conference on Advanced Information Systems Engineering (CAISE)*, S. 180–194. Springer Verlag, LNCS 1626, 1999.

[TSMB95] S. Teufel, C. Sauter, T. Mühlherr und K. Bauknecht. *Computerunterstützung für die Gruppenarbeit*. Addison Wesley, 1995.

[Tuf83] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[Tuf91] E. R. Tufte. *Envisioning Information*. Graphics Press, 2. Auflage, 1991.

[Tuk77] J. W. Tukey. *Exploratory Data Analysis*. Addison Wesley, 1977.

[UFK+89] C. Upson, J. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, V. Jeffrey, R. Gurwitz und A. v. Dam. The Application Visualization System: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, 1989. Siehe auch http://www.avs.com.

[Uli92] E. Ulich. *Arbeitspsychologie*. Schäffer-Poeschel Verlag, 1992.

[Unw94] A. Unwin. REGARDing geographic data. In [DO94], S. 315–326.

[Vas98] P. Vassiliadis. Modeling multidimensional databases, cubes and cube operations. In *10th International Conference on Scientific and Statistical Database Management (SSDBM)*, S. 53–62, 1998. Langfassung unter http://www.dblab.ece.ntua.gr/∼pvassil/publications/publications.html.

[VB96] G. Vossen und J. Becker, Hrsg. *Geschäftsprozeßmodellierung und Workflow–Management. Modelle, Methoden, Werkzeuge.* International Thomson Publishing, 1996.

[vdBdF92] G. M. van den Berg und E. de Feber. Definition and use of meta–data in statistical data processing. In *6th International Working Conference on Scientific and Statistical Database Management (SSDBM)*, S. 290–306, 1992.

[Vel99] P. F. Velleman. Data Desk 6.1. Systemspezifikation und Demo–Version, Data Description, Ithaca, NY, 1999. http://www.datadesk.com/DataDesk.

[Vos99] G. Vossen. *Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme*. Oldenbourg Verlag, 3. Auflage, 1999.

[VR94] W. N. Venables und B. D. Ripley. *Modern Applied Statistics with S–PLUS*. Springer Verlag, 1994. Siehe auch http://www.mathsoft.com.

[WA94] G. Wang und A. Ambler. Applicability checking in visual programming languages. In *10th IEEE Symposium on Visual Languages*, S. 31–38, 1994.

[Wal96] J. Walton. Data visualization with IRIS Explorer. In *2nd Workshop on Visualization in High–Energy Physics (HEPVIS)*, 1996. http://www.cern.ch/Physics/Workshops/hepvis/hepvis96/hepvis96.html.

[Wal98] J. Walton. Data visualization using IRIS Explorer — an introduction. In *3rd Workshop on Visualization in High–Energy Physics (HEPVIS)*, 1998. http://www-sldut.slac.stanford.edu/hepvis/. Siehe auch http://www.iec.co.uk/Welcome_IEC.html.

[WB97] M.-C. Wu und A. P. Buchmann. Research issues in data warehousing. In *7. GI–Fachtagung Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, S. 61–82. Springer Verlag, 1997.

[Wei98] A. Weichert. *Zeitreihenanalyse natürlicher Systeme mit neuronalen Netzen und Methoden der statistischen Physik sowie der nichtlinearen Dynamik*. Dissertation, Universität Oldenburg, Fachbereich Physik, Februar 1998. http://www.physik.uni-oldenburg.de/docserver/diss/docs/weichert_diss.pdf.

[WGG+97] F. Wietek, M. Grawunder, R. Grupe, I. Hoting, S. Koch, M. Lettrari und A. Scharnofske. Benutzerdokumentation CARESS, Version 3.11. Technischer Bericht, OFFIS, August 1997. http://www.krebsregister-niedersachsen.de/ekn_de_publikationen.html.

[Whi97] K. N. Whitley. Visual programming languages and the empirical evidence for and against. *Journal on Visual Languages and Computing*, 8(1):109–142, 1997.

[Wid95] J. Widom. Research problems in data warehousing. In *4th International Conference on Information and Knowledge Management (CIKM)*, S. 25–30, 1995.

[Wie99] F. Wietek. Spatial statistics for cancer epidemiology — the cancer registry's epidemiological and statistical data analysis system (CARESS). In R. Fehr, J. Berger und U. Ranft, Hrsg., *Environmental Health Surveillance. Results of an International Workshop*, S. 157–171. ecomed-Verlag, Fortschritte in der Umweltmedizin, 1999.

[Wil97] A. F. X. Wilhelm. Generalised linking as a means for analysing complex data sets. In *29th Symposium on the Interface: Computing Science and Statistics*, S. 456–461, 1997.

[Wit85] K. M. Wittkowski. *Ein Expertensystem zur Datenhaltung und Methodenauswahl für statistische Anwendungen*. Dissertation, Institut für Informatik der Universität Stuttgart, 1985.

[Wit98] K. Wittenburg. Visual language parsing: If i had a hammer ... In *Multimodal Human–Computer Communication. 1st International Conference on Cooperative Multimodal Communication*, S. 231–249. Springer Verlag, LNAI 1374, 1998.

[WK97] E. Woods und E. Kyral. *OVUM evaluates Data Mining*. OVUM Ltd., 1997.

[WK98] F. Wietek und V. Kamp. Spatial data analysis support for cancer epidemiology in CARESS. In *International Workshop on Geomedical Systems (GEOMED'97)*, S. 183–193. Teubner Verlag, 1998.

[WKC96] E. Woods, E. Kyral und P. Carnelley. *OVUM evaluates OLAP*. OVUM Ltd., 1996.

[Wol92] H. P. Wolf. Zehn Punkte zur Standortbeschreibung von EDA. In [EGHW92], S. 317–328.

[WRH92] C. Williams, J. Rasure und C. Hansen. The state of the art of visual languages for visualization. In *IEEE Visualization*, S. 202–209, 1992.

[Wro98] S. Wrobel. Data Mining und Wissensentdeckung in Datenbanken. *KI–Journal*, 1/98:6–10, März 1998.

[WS95] A. Woodruff und M. Stonebraker. Buffering of intermediate results in dataflow diagrams. In *11th IEEE Symposium on Visual Languages*, S. 187–194, 1995.

[WUT95] A. F. X. Wilhelm, A. Unwin und M. Theus. Software for interactive statistical graphics — a review. In *Advances in Statistical Software 5 (SoftStat'95)*, S. 3–12. Lucius & Lucius, 1995.

[YAK95] M. Young, D. Argiro und S. Kubica. Cantata: Visual programming environment for the Khoros system. *ACM Computer Graphics*, 29(2):22–24, 1995.

[YAW95] M. Young, D. Argiro und J. Worley. An object oriented visual programming language toolkit. *ACM Computer Graphics*, 29(2):25–28, 1995.

[YFM93] F. W. Young, R. A. Faldowski und M. M. McFarlane. Multivariate statistical visualization. In C. R. Rao, Hrsg., *Computational Statistics. Handbook of Statistics, Vol. 9*, S. 959–998. Elsevier Science Publishers, 1993.

[YL95] F. W. Young und D. J. Lubinsky. Guiding data analysis with visual statistical strategies. *Journal of Computational and Graphical Statistics*, 4(4):229–250, 1995.

[You96] F. W. Young. ViSta: The Visual Statistics System. Research memorandum 94-1(b) (2nd ed.), University of North Carolina, 1996. Siehe auch http://sunsite/univie.ac.at/ViSta.

[YS91] F. W. Young und J. B. Smith. Towards a structured data analysis environment: A cognition–based design. In A. Buja und P. A. Tukey, Hrsg., *Computing and Graphics in Statistics*, S. 253–279. Springer Verlag, 1991.

[Zac98] L. Zachewitz. Persistenz und Maintenance im MOA–Kontext. Diplomarbeit, Universität Oldenburg, Fachbereich Informatik, Juli 1998.

[Zlo80] M. M. Zloof. A language for office and business automation. In *AFIPS Office Automation Conference Digest*, S. 249–260, 1980. Auch in [Gli90a].

Über die angeführten URLs zu On–line Journals, technischen Berichten, Workshop–Beiträgen und Systemdokumentationen hinaus sind auch viele weitere der genannten Papiere im WWW zugreifbar. An dieser Stelle sei hierzu lediglich auf drei nützliche Einstiegsseiten verwiesen:

- für Datenbankliteratur allgemein auf die sehr umfangreiche *Computer Science Bibliography* aus dem *Digital Bibliography & Library Project* an der Universität Trier (http://www.informatik.uni-trier.de/∼ley/db/),
- für den Bereich von OLAP, multidimensionaler Datenmodellierung und Data Warehousing auf die *Reserach Oriented Bibliography on Data Warehousing and OLAP* von Alberto Mendelzon an der Universtität Toronto (http://www.cs.toronto.edu/∼mendel/dwbib.html) und
- für Beiträge zur visuellen Programmierung auf die *Virtual Library on Visual Languages and Visual Programming* (http://cuiwww.unige.ch/Visual/).

Glossar

Dieses Glossar gibt kurze Definitionen der wichtigsten Begriffe aus den Bereichen Epidemiologie, Datenanalyse, visuelle Programmierung, multidimensionale Datenmodellierung und Metadaten, die im Laufe der Arbeit eingeführt wurden. Weiterhin werden die zentralen im Rahmen von *MADEIRA* oder *VIOLA* definierten Termini kurz informell erläutert bzw. ihre besondere Semantik in Abgrenzung zur allgemein üblichen Verwendung hervorgehoben. Um das Glossar kompakt zu halten, sind teilweise Spezialisierungen eines Oberbegriffs gleich unter diesem beschrieben.

Alle Glossareinträge finden sich auch im Index. Somit können dort Synonyme und Referenzen auf detailliertere Definitionen dieser und anderer Begriffe im Text der Arbeit entnommen werden. Folgende Notationen werden verwendet:

- **(...)** enthält eine gängige Abkürzung zum jeweiligen Begriff.
- $\sim$ steht als Kürzel für den erklärten Begriff.
- $\uparrow \ldots$ verweist auf einen anderen Begriff im Glossar.

Ableitung Bei der $\sim$ von ↑Datenwürfeln in *MADEIRA* werden die ↑Kategorien eines ↑kategoriellen Attributs durch gröbere oder gleich feine ↑Kategorien ersetzt, wobei die betreffenden ↑Zellen jeweils gemäß der ↑Aggregierungsfunktionen der jeweiligen ↑Maßzahlen zusammengefaßt werden. (Hierbei sprechen wir auch von $\sim$ der Ziel–↑Kategorien.) Spezialfälle der $\sim$ sind die implizite ↑Aggregierung und die ↑Restriktion. In *VIOLA* wird die Zusammenfassung aller ↑Kategorien eines ↑kategoriellen Attributs zu einem Gesamtwert *vollständige* $\sim$ genannt.

Ad–hoc–Anfrage Vor allem im Rahmen einer ↑explorativen Datenanalyse „spontan“ formulierte und interaktiv gestellte Anfrage an ein ↑Datenbanksystem.

Ätiologie Die $\sim$ ist die Lehre von der Entstehung von Krankheiten, ihren Ursachen bzw. Einflußfaktoren (↑Expositionen), die ihre Entwicklung begünstigen.

Aggregierbarkeit Die $\sim$ einer ↑Maßzahl bzgl. eines ↑kategoriellen Attributs bzw. einer ↑Eigenschaft gibt an, ob die Werte der ↑Maßzahl mittels ↑Aggregierung über das bzw. ein entsprechendes ↑kategorielles Attribut (unter Verwendung einer ↑Aggregierungsfunktion) zu „gröberen“ Angaben der gleichen ↑Maßzahl zusammengefaßt werden können.

Aggregierung Unter Aggregierung wird allgemein die Zusammenfassung von Datenwerten eines ↑Datenraums (auf einer jeweils gröberen ↑Aggregierungsebene) verstanden. Hierunter fallen

- die Berechnung von ↑Makrodaten aus ↑Mikrodaten, indem die betrachteten ↑Objekte über ausgewählte ↑Kategorien gemäß ihrer ↑Eigenschaften klassifiziert und auf den erhaltenen Gruppen ↑Maßzahlen berechnet werden,
- die einfache Vergröberung von Angaben oder die Zusammenfassung zu einem Gesamtwert bzgl. eines oder mehrerer ↑kategorieller Attribute gemäß der Subsumtionsbeziehungen auf ↑Kategorien und unter Verwendung der der jeweiligen ↑Maßzahl zugeordneten ↑Aggregierungsfunktionen ebenso wie

– die Berechnung neuer ↑Maßzahlen, die ein oder mehrere ↑kategorielle Attribute gemäß ihrer ↑Berechnungsfunktion „eliminieren“.

In *MADEIRA* werden unter (*impliziter*) ∼ im speziellen diejenigen Anwendungen der ↑Ableitung subsumiert, die keine bloße ↑Restriktion realisieren. Unter *expliziter* ∼ wird die Zusammenfassung von Werten einer ↑Maßzahl unter Anwendung einer explizit (nicht implizit durch die Quell–↑Maßzahl) gegebenen ↑Aggregierungsfunktion verstanden. Schließlich wird als *vollständige* ∼ auch die Zusammenfassung der ↑Maßzahl–Werte zu *allen* ↑Kategorien eines ↑kategoriellen Attributs zu einem Gesamtwert bezeichnet.

Aggregierungsebene Zusammenfassung mehrerer, in gewissem (nicht–strengen) Sinne als gleich „fein“ angesehener ↑Kategorien einer ↑Domäne. Eine ∼ wird in *MADEIRA* als *wohlgeformt* bezeichnet, wenn keine ihrer ↑Kategorien spezieller als eine andere ist.

Aggregierungsfunktion Funktion, die zur Zusammenfassung von Werten einer ↑Maßzahl bei ↑Aggregierung über ein ↑kategorielles Attribut eines ↑Datenwürfels angewendet wird.

Analysegraph Speziell in *VIOLA* zur Modellierung und Durchführung einer ↑Datenanalyse erstelltes ↑Datenflußprogramm oder ein Teil davon.

Anforderungsgetriebene Verarbeitung Variante der Verarbeitung von ↑Datenflußprogrammen, bei der die Ausführung von ↑Modulen transitiv dadurch gesteuert wird, ob und wann ihre Ergebnisse in ↑Datensenken benötigt werden.

Anfrageoptimierung Algebraische Umformung einer Anfrage an ein ↑Datenbanksystem sowie geeignete Auswahl physischer Operatoren zur Umsetzung der Anfrage mit dem Ziel ihrer möglichst effizienten Ausführung.

ANSI/SPARC–Architektur Architektur für ↑Datenbankmanagementsysteme, deren Ziel es ist, (Schnittstellen zu) Benutzer–Applikationen und physikalische Realisierung von ↑Datenbanken klar zu trennen. Sie wird auch als Drei–Schichten–Architektur bezeichnet, da an ihr ausgerichtete ↑Datenbanksysteme ihre jeweiligen ↑Datenbanken über entsprechende ↑Datenschemata auf drei verschiedenen Ebenen, der internen, der konzeptionellen und der externen Ebene modellieren.

Attribut Allgemein sind ∼e Datenstrukturen zur Spezifikation bestimmter Aspekte von Datenbeständen oder einzelnen Datenobjekten: der jeweils definierten ↑Maßzahlen, der beschriebenen ↑Eigenschaften bzw. deren jeweiliger Ausprägungen. In *MADEIRA* und *VIOLA* sind speziell ↑summarische und ↑kategorielle ∼e von ↑Makrodaten, ↑relationale ∼e von ↑Mikrodaten sowie ↑graphische ∼e von ↑Darstellungssymbolen einer ↑Graphik von Interesse.

Basisdaten Als ∼ im engeren Sinne werden die in einer ↑Datenbank materialisierten Ausgangsdaten einer ↑Datenanalyse bezeichnet.

Berechnungsfunktion In *MADEIRA* einer ↑Maßzahl zugeordnete Funktion, die angibt, wie diese aus gegebenen Werten anderer ↑Maßzahlen berechnet werden kann.

Berechnungsvorschrift Eine ∼ einer ↑Maßzahl ergänzt eine ↑Berechnungsfunktion um Angaben zur Art von Quell– und zusätzlich (quasi als Nebenprodukte) berechneten Ziel–↑Maßzahlen sowie — für (nicht ↑zellenbezogene) ↑Maßzahlverknüpfungen — zur Art der ↑kategoriellen Attribute, über die bei der Berechnung eine ↑Aggregierung erfolgen kann.

Cache Temporärer Speicher mit begrenzter Kapazität, auf den effizient zugegriffen werden kann.

Clusteranalyse Oberbegriff für statistische Verfahren zur Gruppierung untersuchter Daten bzw. ↑Objekte anhand ihrer ↑Attribute bzw. ↑Eigenschaften. In der ↑Epidemiologie werden unter ∼ jedoch meist Verfahren verstanden, die Häufungen von ↑Fällen in Zeit und/oder Raum untersuchen bzw. Unregelmäßigkeiten

in ihrem Auftreten beurteilen. Hierunter fallen etwa die Berechnung von ↑Clusterindizes oder die Erstellung und Bewertung thematischer Karten.

Clusterindex Statistische ↑Maßzahl für die räumliche (evtl. auch zeitliche) Verteilung bzw. Clusterung von ↑Fällen innerhalb der betrachteten ↑Studienpopulation und Zeitperiode.

Cumulative Mortality/Incidence Figure (CMF/CIF) Quotient aus ↑direkt standardisierter Rate der ↑Studienpopulation und ↑roher Rate der ↑Standardpopulation.

Darstellungssymbol Atomares Element einer ↑Graphik (z. B. ein Balken, eine Linie oder ein Punkt), das bestimmte ↑graphische Attribute aufweist, über deren Ausprägungen Datenwerte repräsentiert werden.

Data Mart Ein im Hinblick auf Datenumfang, Anwendungsbereich und/oder Funktionalität „kleines“ ↑Data Warehouse.

Data Mining Der eigentliche Analyseschritt im Rahmen des ↑Knowledge Discovery in Databases. In automatisierter Weise werden aus den betrachteten Daten nach vorgegebenen Mustern neuartige Hypothesen zur Beschreibung von Auffälligkeiten und Zusammenhängen generiert und bewertet. Im Gegensatz zum ↑On–line Analytical Processing bzw. zu der ↑explorativen Datenanalyse spielt die Interaktion mit dem Systemanwender bzw. die Steuerung des Analyseablaufs durch diesen eine untergeordnete Rolle.

Data Warehouse ↑Datenbanksystem, das Daten aus verschiedenen operativ im Sinne des ↑On–line Transactional Processing (typischerweise in betriebswirtschaftlichen Anwendungen) genutzten ↑Datenbanksystemen integriert und zum Zwecke der ↑Datenanalyse (etwa des ↑On–line Analytical Processing oder des ↑Data Mining) aufbereitet. Mitunter wird unter einem ∼ auch lediglich die entsprechende ↑Datenbank verstanden.

Data Warehousing Oberbegriff für alle Themen, die sich mit Konzeption, Aufbau und Betrieb eines ↑Data Warehouses beschäftigen.

Datenanalyse Ableitung von Informationen bzw. Wissen aus Daten durch Anwendung von Auswertungsverfahren der ↑Statistik und der Informatik.

Datenanalyseumgebung Software zur Unterstützung der ↑Datenanalyse.

Datenbank (DB) Strukturierte Sammlung von Daten, die mit Hilfe eines ↑Datenbankmanagementsystems verwaltet werden.

Datenbankmanagementsystem (DBMS) Programm zur Verwaltung von Daten in einer ↑Datenbank.

Datenbanksystem (DBS) ↑Datenbankmanagementsystem zusammen mit einer oder mehreren von ihm verwalteten ↑Datenbanken.

Datenbasis In *MADEIRA* Bezeichnung für einen als ↑Basisdaten(raum) materialisierten ↑Datenwürfel.

Datenflußprogramm Mit Mitteln der ↑Datenflußprogrammierung erstelltes Programm.

Datenflußprogrammierung Beschreibung der durch das Vorliegen bzw. die Anforderung von Daten (siehe ↑datengetriebene und ↑anforderungsgetriebene Verarbeitung) gesteuerten Abfolge von Instruktionen in textueller oder Diagrammform. Die ∼ mit (Datenfluß–)Diagrammen, in denen Verarbeitungseinheiten (↑Module) über ↑Kanäle zur Datenpropagierung verbunden werden, ist eine Form der ↑visuellen Programmierung.

Datengetriebene Verarbeitung Variante der Verarbeitung von ↑Datenflußprogrammen, bei der — beginnend bei den ↑Datenquellen — die Reihenfolge der Ausführung von ↑Modulen allein durch das Vorliegen valider Eingangsdaten gesteuert wird.

Datenmanagement Unter ∼ sollen in *MADEIRA* und *VIOLA* alle Operationen auf ↑Datenräumen subsumiert werden, die keine neuen ↑Maßzahlen berechnen.

Datenmodell Formale Sprache zur strukturierten Beschreibung von Daten und ihrer Beziehungen sowie von Operationen auf diesen Daten. Je nach Abstraktionsgrad unterscheidet man an ganz konkrete ↑Datenbankmanagementsysteme und deren interne Strukturen gebundene *physische* ∼e, von Klassen von ↑Datenbankmanagementsystemen unterstützte *logische* ∼e (etwa das relationale ∼) und völlig von einer bestimmten Implementierung unabhängige *konzeptionelle* ∼e (etwa das ER–Modell).

Datenquelle ↑Modul in einem ↑Datenflußprogramm, das der Bereitstellung von ↑Basisdaten und somit als Ausgangspunkt für die Datenpropagierung im Rahmen einer ↑datengetriebenen Verarbeitung dient. Im weiteren Sinne bezeichnet eine (*physische*) ∼ jede Art von Programm, Programmkomponente, Datei oder ↑Datenbank, die Daten zur Verarbeitung bereitstellt.

Datenraum Der Begriff des ∼s wird in *MADEIRA* als Oberbegriff für ↑Datenwürfel und ↑Datenbasen, bisweilen auch allgemein für Bestände einheitlich strukturierter Daten (also sowohl für ↑Makrodaten als auch für ↑Mikrodaten–Relationen) verwendet.

Datenreihe Menge von hinsichtlich ihrer ↑graphischen Attribute gleichartigen ↑Darstellungssymbolen, die eine Menge von Datenobjekten (etwa ausgewählte Tupel einer Relation oder ↑Zellen eines ↑Datenwürfels) gemäß einer ↑Symboldefinition in einheitlicher Weise visualisieren.

Datenschema Ein ∼ beschreibt die Struktur einer ↑Datenbank mit den Mitteln eines ↑Datenmodells. Je nachdem, welche Aspekte der ↑Datenbank (interne Realisierung, allgemeine Konzeption oder externe Sichten für bestimmte Benutzer) betrachtet werden, unterscheidet man gemäß der ↑ANSI/SPARC–Architektur *interne*, *konzeptionelle* und *externe* ∼ta.

Datensenke ↑Modul in einem ↑Datenflußprogramm, das der ↑Datenvisualisierung sowie der Ablage von Berechnungsergebnissen und somit als Ausgangspunkt einer ↑anforderungsgetriebenen Verarbeitung dient.

Datentyp In *MADEIRA* der einer ↑Maßzahl zugeordnete Wertebereich inklusive ↑neutraler Elemente und ↑Nullwerte.

Datenvisualisierung Präsentation von Daten mit Hilfe von ↑Graphiken; im weiteren Sinne jede Art visuell wahrnehmbarer Datenpräsentation (also etwa auch in Form von Tabellen).

Datenwürfel Basisdatenstruktur ↑multidimensionaler Datenmodelle. Ein ∼ ist ein mehrdimensionales Feld, dessen Dimensionen durch ↑kategorielle und dessen Feldinhalte durch ↑summarische Attribute (quasi als zugeordnete ↑Metadaten) beschrieben werden. Die Elemente des Feldes werden auch ↑Zellen genannt. In *MADEIRA* werden als ∼ im speziellen aus ↑Datenbasen und/oder anderen ∼n abgeleitete ↑Datenräume bezeichnet.

Diagnose Zuordnung einer gesundheitlichen Störung zu einem Krankheitsbegriff. Die Abgrenzung verschiedener ∼n zur statistischen Auswertung erfolgt oftmals mittels der ↑International Classification of Diseases.

Dimension Der Begriff der ∼ wird in *MADEIRA* synonym für eine anwendungsspezifische ↑Kategorienhierarchie auf einer ↑Domäne verwendet. Allgemein wird unter einer ∼ (eines ↑Datenwürfels) oft auch ein ↑kategorielles Attribut verstanden.

Dimensionsattribut Allen oder einigen ↑Kategorien einer ↑Domäne zugeordnetes ↑Attribut, das ein ↑Metadatum zur näheren Beschreibung spezifiziert. ∼e können zusätzlich zu einer ↑Kategorienhierarchie zur ↑Klassifikation von ↑Kategorien genutzt werden, wovon jedoch in *MADEIRA* kein Gebrauch gemacht wird. Weiterhin können — wie in *MADEIRA* geschehen — ∼e auch als ↑Basisdaten für einen eindimensionalen, durch die jeweils betrachteten ↑Kategorien aufgespannten ↑Datenwürfel angesehen werden.

Dimensionstabelle In einem ↑Stern- oder ↑Schneeflockenschema diejenigen Relationen, die die ↑Kategorien der ↑kategoriellen Attribute des jeweiligen ↑Datenwürfels inklusive der auf ihnen definierten ↑Kategorienhierarchien und weiterer zu ↑Kategorien definierter ↑Metadaten (insbesondere Namen und Beschreibungen oder spezielle ↑Dimensionsattribute) enthalten.

Direkt standardisierte Rate Gewichtete Summe der ↑rohen Raten verschiedener ↑Teilpopulationen der nach einer oder mehreren ↑Eigenschaften (etwa Alter und/oder Geschlecht) klassifizierten ↑Studienpopulation. Die jeweilige Gewichtung richtet sich nach der Struktur einer ↑Standardpopulation bzgl. der zur ↑Standardisierung verwendeten Merkmale.

Disaggregierung Inverse Funktion zur ↑Aggregierung, die in der Regel nur unter Anwendung von Schätzverfahren oder über einen Zugriff auf die jeweils zugrundeliegenden ↑Basisdaten durchgeführt werden kann.

Domäne Eine (kategorielle) ∼ definiert in *MADEIRA* eine Menge von ↑Kategorien zur Beschreibung von Ausprägungen bestimmter ↑Eigenschaften von ↑Objekten.

Drill–through Abfrage der „hinter“ einer ↑Zelle eines ↑Datenwürfels als ↑Basisdaten liegenden ↑Mikrodaten.

Dynamic Query Eine Form der interaktiven Parametrisierung von (↑interaktiven) ↑Graphiken, bei der die Menge darzustellender Datenobjekte über Schieberegler oder ähnliche Komponenten dynamisch selektiert wird.

Dynamische Graphik ↑Graphik, deren ↑Visualisierungsparameter sich (in Bezug auf ihre Ausprägungen) zur Betrachtungszeit automatisch ändern (etwa rotierende Punktwolken) bzw. verändern lassen (siehe ↑interaktive Graphik).

Eigenschaft ∼en (Merkmale) von ↑Objekten definieren in einer ↑Metadaten–basierten Betrachtungsweise den jeweiligen Anwendungskontext der in *MADEIRA* modellierten Daten. Zum einen können aus ihren Ausprägungen, den ↑Kategorien, Werte von ↑Maßzahlen ermittelt werden, und zum anderen können ↑Kategorien zur ↑Klassifikation der betrachteten ↑Objektmengen und damit zur Definition der Granularität der ↑Aggregierung genutzt werden. Es werden *ein-* und *mengenwertige* ∼en unterschieden.

Einschränkendes Attribut ↑Kategorielles Attribut mit nur einer ↑Kategorie als Ausprägung. Ein ∼ dient somit lediglich zur Eingrenzung der betrachteten ↑Objektmengen, nicht aber — wie ein ↑klassifizierendes Attribut — zu deren Untergliederung.

Epidemiologie Wissenschaft von der Verteilung von Krankheitshäufigkeiten in menschlichen ↑Populationen und von Faktoren, die diese Verteilung beeinflussen. Man unterscheidet allgemein zwischen deskriptiver und analytischer ∼. Die *deskriptive* ∼ beschäftigt sich mit der Beschreibung der Verteilung und Häufigkeit von Erkrankungen und möglichen ↑Risikofaktoren innerhalb einer ↑Studienpopulation. Aufgrund ihrer Betrachtungen aufgestellte konkrete Hypothesen werden im Rahmen der *analytischen* ∼ anhand von ausgewählten Patientengruppen überprüft.

Erwartete Rate Die ∼ gibt an, welche ↑Rate in der ↑Studienpopulation bei Zugrundelegen der alters- (bzw. allgemein ↑Teilpopulations–)spezifischen ↑Raten einer ↑Standardpopulation aufgetreten wäre. Wie bei der ↑Standardisierung werden hier also wiederum unterschiedliche ↑Populations–Strukturen bzgl. einer oder mehrerer ↑Eigenschaften berücksichtigt. Es erfolgt quasi eine ↑direkte Standardisierung der ↑Rate der ↑Standardpopulation unter Verwendung der ↑Studienpopulation als Standard.

Exposition Möglicher Einflußfaktor auf Entstehung oder Verlauf einer Krankheit bzw. Zustand, in dem derartige Faktoren wirksam sind.

Extension Allgemein versteht man unter der ∼ einer ↑Datenbank — in Abgrenzung zur Strukturbeschreibung durch ein ↑Datenschema — einfach die ↑Datenbank selbst (also die enthaltenen Daten). Als ∼ eines ↑Datenwürfels wird die Menge seiner ↑Zellen mit ihren jeweiligen ↑Koordinaten bezeichnet. Die ∼ eines ↑Mikrodatenraums ist die Menge seiner Datentupel. In beiden Fällen wird die ∼ in *MADEIRA* noch um die Identifikation der Menge jeweils repräsentierter ↑Objekte ergänzt.

Explorative Datenanalyse (EDA) Die ∼ erweitert die Verfahrenspalette der deskriptiven ↑Statistik um Methoden zur interaktiven, die kognitiven Fähigkeiten des Menschen stärker einbeziehenden Extraktion von Informationen aus oft großen Datenmengen. Sie propagiert eine durch die Daten geleitete, iterative Vorgehensweise sowie den verstärkten Einsatz graphischer Verfahren zur ↑Datenvisualisierung.

Faktentabelle In einem ↑Stern- oder ↑Schneeflockenschema diejenige Relation, die die ↑Zellen eines ↑Datenwürfels mit ihren ↑Koordinaten (in Form von Fremdschlüsseln auf die verschiedenen ↑Dimensionstabellen) als Tupel enthält.

Fall Von einer Krankheit betroffene Person bzw. die Erkrankung selbst. Es werden ↑Inzidenz-, ↑Prävalenz- und ↑Mortalitäts-∼e unterschieden.

Fallzahl Anzahl von ↑Fällen innerhalb einer ↑Studienpopulation. Meist werden hierbei Erkrankungen (evtl. auch mehrere pro Person), gelegentlich aber auch lediglich verschiedene Patienten gezählt. Die (absolute) ∼ bezieht sich im Gegensatz zur ↑rohen Rate nicht auf die Größe der zugrundeliegenden Bevölkerung.

Fan–in Der ∼ eines ↑Ports spezifiziert die Menge aller Quell–↑Ports eingehender ↑Kanäle (*externer* ∼) bzw. die Menge aller als Berechnungsgrundlage dienender ↑Ports im gleichen ↑Modul (*interner* ∼).

Fan–out Der ∼ eines ↑Ports spezifiziert die Menge aller Ziel–↑Ports ausgehender ↑Kanäle (*externer* ∼) bzw. die Menge aller ↑Ports im gleichen ↑Modul, für die der Inhalt des betrachteten ↑Ports als Berechnungsgrundlage dient (*interner* ∼).

Gesundheitsberichterstattung (GBE) Beschreibende, kommentierende und interpretierende Zusammenstellung von aktuellen Daten und Informationen des Gesundheitswesens.

Graphik Bildhafte Darstellung zur ↑Datenvisualisierung, z. B. in Form von Diagrammen, Netzen, Karten, Piktogrammen etc. Eine ∼ ordnet den betrachteten Daten gemäß eines ↑Visualisierungsverfahrens sowie weiterer ↑Visualisierungsparameter attributierte ↑Darstellungssymbole zu und enthält ferner Titel, Skalen und Legenden.

Graphisches Attribut Parameter eines ↑Darstellungssymbols (etwa Höhe, Breite, Form oder Farbe), dem Vorschriften zur visuellen Umsetzung von Datenwerten im Sinne einer Skalierung zugeordnet sind.

Gruppierung Spezialfall der ↑Ableitung in *VIOLA*, bei dem ↑Kategorien gemäß ihrer Zugehörigkeit zu vorgegebenen ↑Kategorien–Gruppen zu einem jeweiligen Gesamtwert zusammengefaßt werden.

Highlighting Das visuelle Hervorheben von ↑Darstellungssymbolen innerhalb einer ↑Graphik.

Homogenisierte Sicht Von einer ∼ auf einen ↑Datenwürfel wird in *MADEIRA* gesprochen, wenn dessen ↑summarische Attribute in einem ↑Kennzahlattribut zusammengefaßt wurden.

Interaktion Im Kontext der ↑Datenanalyse ein in Kooperation zwischen Analysesystem und Datenanalyst durchgeführter Verarbeitungsschritt innerhalb einer Analysesitzung. Eine ∼ kann als Einheit der Strukturierung von ↑Datenanalysen angesehen werden. Wir unterscheiden grob ∼en zur Datenwahl, zur Anwendung statistischer Verfahren, zur ↑Datenvisualisierung und zur ↑Navigation.

Interaktionsmodell Spezifikation möglicher Abfragen oder Kombinationen von ↑Interaktionen zur ↑Datenanalyse, die zur Bearbeitung einer Fragestellung möglich und sinnvoll sind bzw. durch eine konkrete ↑Datenanalyseumgebung unterstützt werden.

Interaktive Graphik ↑Dynamische Graphik, deren ↑Visualisierungsparameter sich (in Bezug auf ihre Ausprägungen) interaktiv verändern lassen. Typische Beispiele sind das ↑Linking von ↑Graphiken sowie das interaktive ↑Highlighting.

International Classification of Diseases (ICD) Allgemeines ↑Klassifikationsschema zur Codierung von ↑Diagnosen. Aktuell sind die 9. und 10. Revision in Gebrauch. Die 9. Revision ist numerisch (drei Ziffern und bis zu zwei Nachkommastellen), die ∼–10 dagegen alphanumerisch (ein Buchstabe gefolgt von zwei Ziffern und bis zu zwei Nachkommastellen). Der hierarchische Aufbau der ∼ ermöglicht eine Spezifikation von Krankheitsgruppen über Präfixe von ∼–Codes.

Inzidenz Das Erkranken (an einer Krankheit) bzw. die Anzahl von Neuerkrankungen bzgl. einer Krankheit in einem definierten Zeitraum und einer definierten ↑Population.

Kanal „Kante“ zwischen zwei ↑Modulen (genauer: zwei ↑Ports) in einem ↑Datenflußprogramm, über die Daten propagiert werden. *VIOLA* unterscheidet zwischen *Daten-* und *Parameter-*∼en, je nachdem, ob ein ∼ der Weiterleitung von ↑Datenwürfeln oder Werten von (externen) ↑Parametern dient.

Kategorie ∼n beschreiben Ausprägungen von ↑Eigenschaften von ↑Objekten. Sie dienen insbesondere zur ↑Klassifikation der jeweils betrachteten ↑Objektmengen in verschieden grobe Teilgruppen und somit als Grundlage der ↑Aggregierung von ↑Mikro- und ↑Makrodaten. Zwischen ∼n einer ↑Domäne bestehen Subsumtions- (Verfeinerungs-)beziehungen, die feinere ∼n gröberen unterordnen. In *MADEIRA* werden ∼n als logisch verknüpfbare Aussagen bzw. Prädikate über ↑Eigenschaften und ↑Objekten modelliert. Als δ–∼ wird hierbei eine ∼ bezeichnet, die eine im jeweiligen Anwendungskontext nicht genauer zu spezialisierende Aussage der Form „ . . . , *ohne nähere Angabe*“ repräsentiert.

Kategorielles Attribut Ein ∼ eines ↑Datenwürfels spezifiziert über eine Menge von ↑Kategorien einer ↑Domäne eine Einschränkung und ↑Klassifikation des jeweils betrachteten ↑Basisdaten–Bestandes bzw. der zugrundeliegenden ↑Objektmengen bzgl. einer oder mehrerer ↑Eigenschaften. Somit definieren die ∼e eines ↑Datenwürfels über ihre ↑Kategorien–Mengen das „Gitter“ aus einzelnen ↑Zellen des jeweiligen multidimensionalen Feldes.

Kategorienhierarchie Menge von ↑Aggregierungsebenen einer ↑Domäne mit einer partiellen Ordnung, die auf Basis der Subsumtionsbeziehung auf den ↑Kategorien der jeweiligen ↑Domäne definiert wird.

Kennzahlattribut Enthält ein ↑Datenwürfel mehrere ↑summarische Attribute, so können diese auch als ↑Kategorien eines zusätzlichen ↑klassifizierenden ↑kategoriellen Attributs in Form eines sogenannten ∼s (oft auch als *Kennzahldimension* bezeichnet) modelliert werden. Somit läßt sich eine evtl. auferlegte Einschränkung auf *ein* ↑summarisches Attribut pro ↑Datenwürfel (jetzt quasi als „Verteiler“–Attribut) erfüllen und auch die ↑Visualisierung von ↑Datenwürfeln einfacher spezifizieren.

Klassifikation Der Begriff der ∼ wird in dieser Arbeit (wie auch meist — mehr oder weniger explizit — allgemein im Kontext ↑multidimensionaler Datenmodelle) im Sinne der objektorientierten Modellierung zur Bezeichnung einer (oftmals, aber nicht notwendigerweise disjunkten und vollständigen) Einteilung einer Menge (z. B. von ↑Kategorien oder ↑Objekten) in mehrere Teilmengen verwendet. Eine ∼ im strengen mathematischen Sinne einer disjunkten und vollständigen Mengenzerlegung wird hier als ↑Partitionierung bezeichnet.

Klassifizierendes Attribut Im Gegensatz zu einem ↑einschränkenden Attribut ist ein ∼ ein ↑kategorielles Attribut, das *mehr* als eine ↑Kategorie enthält und somit die durch den jeweiligen ↑Datenwürfel betrachteten ↑Objektmengen in verschiedene (oftmals, aber nicht immer disjunkte) Teilgruppen einteilt (klassifiziert). Ein ∼ definiert also eine „Dimension“ des durch den ↑Datenwürfel gegebenen multidimensionalen Feldes.

Knowledge Discovery in Databases (KDD) Der Prozeß der Identifikation valider, neuartiger, möglicherweise nützlicher und verständlich darstellbarer Muster in Daten. Das ∼ umfaßt alle Schritte von der Formulierung von Fragestellungen über Datensammlung, -bereinigung, -integration, -exploration, -transformation, ↑Data Mining und Interpretation bis zur Nutzung und Umsetzung des gefundenen Wissens. Hiermit gibt es eine umfassende Definition der Teilprozesse der ↑Datenanalyse.

Komposition Die ∼ von ↑Kategorien generiert zusammengesetzte ↑Kategorien–Tupel, die nun gleichzeitig zwei oder mehrere ↑Eigenschaften konjunktiv beschreiben. Entsprechend spricht man auch von der ∼ von ↑Dimensionen, ↑Domänen, ↑Aggregierungsebenen, ↑Eigenschaften oder ↑kategoriellen Attributen.

Konfidenzintervall ∼e beschreiben auf der Basis eines unterliegenden statistischen Modells und einer gegebenen Irrtumswahrscheinlichkeit den Bereich rein „zufälliger“ Schwankungen um den Wert einer beobachteten ↑Maßzahl.

Konsolidierung Der Begriff der ∼ wird allgemein oft synonym zur ↑Aggregierung verwendet; in *MADEIRA* wird hierunter speziell die ↑Aggregierung von ↑Makro- aus ↑Mikrodaten verstanden.

Koordinate Die ↑Kategorien der ↑kategoriellen Attribute eines ↑Datenwürfels indizieren als ∼n die ↑Zellen des durch den ↑Datenwürfel repräsentierten multidimensionalen Feldes.

Krebsregister Einrichtung zur Sammlung, Aufbereitung, Verwaltung und Auswertung von Daten zu Krebsfällen einer Klinik oder ähnlichen Institution (*klinisches* ∼) oder einer definierten Region (*bevölkerungsbezogenes* oder *epidemiologisches* ∼). Mitunter wird auch der entsprechende Datenbestand selbst als ∼ bezeichnet.

Kumulative Rate ↑Direkt standardisierte Rate, in der alle ↑Raten zu den betrachteten ↑Teilpopulationen unabhängig von einer ↑Standardpopulation das gleiche Gewicht erhalten.

Linking Beim ∼ von zwei oder mehreren ↑interaktiven Graphiken werden diese in der Art miteinander verbunden, daß eine interaktive Manipulation an einer ↑Graphik eine Änderung der ↑Visualisierungsparameter der anderen ↑Graphik(en) hervorruft. Oft besteht das ∼ in einem *gelinkten* ↑*Highlighting*, bei dem eine Selektion von ↑Darstellungssymbolen zum ↑Highlighting entsprechender Symbole der anderen ↑Graphik(en) führt. Im Gegensatz zu einem derartigen, auf übereinstimmende Datenobjekte bezogenen *empirischen* ∼ definiert ein *algebraisches* ∼ komplexere Abbildungsfunktionen, die die abhängigen ↑Graphiken aus der manipulierten ↑Graphik berechnen.

Makrodaten Aus ↑Mikrodaten abgeleitete, gruppierte oder aggregierte Daten, die nach verschiedenen ↑Attributen klassifiziert sind. ∼ sind Gegenstand der Betrachtung in ↑multidimensionalen Datenmodellen sowie im ↑On–line Analytical Processing.

Maßzahl Statistische Größe, die einen bestimmten ↑Datentyp hat und in Erhebungen oder Experimenten festgestellt oder aus anderen ∼en mittels bestimmter ↑Berechnungsfunktionen ermittelt werden kann. Weiterhin definiert eine ∼, ob ihre Werte gemäß einer Zusammenfassung von Teilen der zugrundeliegenden, durch ↑Kategorien klassifizierten ↑Objektmengen aggregiert werden können (vgl. ↑Aggregierung und ↑Aggregierbarkeit) und welche ↑Aggregierungsfunktion ggf. zu verwenden ist.

Maßzahlverknüpfung Oberbegriff für Operationen auf einem oder mehreren ↑Datenwürfeln, die — im Gegensatz zu oder gelegentlich auch als Verallgemeinerung von ↑zellenbezogenen Operationen — die Werte der ↑Maßzahlen aus jeweils *mehreren* ↑Zellen der Quell–↑Datenwürfel im (weiteren) Sinne einer ↑Aggregierung miteinander verknüpfen.

Merge *MADEIRA*–Operator zur ↑Komposition zweier ↑kategorieller Attribute.

Metadaten $\sim$ beschreiben ↑Mikro- und ↑Makrodaten sowie auf diesen durchführbare oder durchgeführte Operationen auf semantischer, struktureller, statistischer und physikalischer Ebene, um eine sinnvolle Speicherung, Transformation, Abfrage und Verbreitung zu ermöglichen.

Mikrodaten ↑Basisdaten aus Erhebungen oder Experimenten über Einheiten einer Gesamtmenge, also Individuen, einzelne ↑Objekte oder Ereignisse. $\sim$ haben typischerweise relationale Form.

Mikrodatenraum In *MADEIRA* wird der Begriff des $\sim$s synonym für eine Relation mit (als ↑Basisdaten dienenden) ↑Mikrodaten verwendet.

Modul Basisbaustein („Knoten“) eines ↑Datenflußprogramms, der der Bereitstellung (in ↑Datenquellen), Verarbeitung oder ↑Visualisierung/Ablage (in ↑Datensenken) zu betrachtender bzw. betrachteter Daten dient. Ferner können $\sim$e auch Kontrollstrukturen realisieren oder als ↑Verbundmodule Teilprogramme kapseln. Über ↑Ports werden einem $\sim$ Eingabedaten bereitgestellt und Berechnungsergebnisse zu anderen $\sim$en weitergeleitet.

Monitoring Routinemäßige Auswertung der ↑Fälle, insbesondere der ↑Inzidenz–Daten eines ↑Krebsregisters zum frühzeitigen Erkennen von Auffälligkeiten wie Trends oder räumlich–zeitlichen Clusterungen.

Mortalität Das Versterben (an einer Krankheit) bzw. die Sterblichkeit, also die Anzahl an Todesfällen (bzgl. bestimmter Erkrankungen) in einem definierten Zeitraum und einer definierten ↑Population.

Multidimensionale Daten Synonym für ↑Makrodaten.

Multidimensionale Datenbank Auf Grundlage eines ↑multidimensionalen Datenmodells aufgebaute ↑Datenbank, insbesondere im Kontext des ↑Data Warehousing. Analog sind *multidimensionale* ↑*Datenbanksysteme* und ↑*Datenbankmanagementsysteme* definiert.

Multidimensionales Datenmodell ↑Datenmodell, das als Basisdatenstruktur (multidimensionale) ↑Datenwürfel zur Repräsentation von ↑Makrodaten nutzt.

Navigation Die $\sim$ ist eine (wesentliche) Art der ↑Interaktion im Rahmen einer ↑Datenanalyse. Dieser Begriff beschreibt vor allem das „Umherwandern“ (Manövrieren) durch die Einzelschritte des bisherigen Ablaufs der jeweiligen Analysesitzung sowie deren Inspektion, Manipulation und Ergänzung. Im weiteren Sinne werden hierunter gelegentlich auch aufeinanderfolgende Operationen des ↑Datenmanagements verstanden, mit denen durch Teilmengen der betrachteten ↑Datenräume „navigiert“ wird.

Neutrales Element Allgemein erfüllt ein $\sim$ e einer Menge M bzgl. einer Verknüpfung $\circ$ gerade $\forall x \in M: x \circ e = x = e \circ x$. In Anlehnung an diese algebraische Definition dient ein $\sim$ eines ↑Datentyps zur Repräsentation von Werten einer ↑Maßzahl, die sich auf eine leere ↑Objektmenge beziehen. Auch ↑Domänen enthalten $\sim$e bzgl. Konjunktion und Disjunktion von ↑Kategorien, die für maximal allgemeine (↑Wurzelkategorien) bzw. nie erfüllte Aussagen stehen.

Nullwert Element eines ↑Datentyps zur Repräsentation nicht vorhandener Informationen in ↑Zellen eines ↑Datenwürfels. In *MADEIRA* wird durch verschiedene $\sim$e insbesondere unterschieden, ob nur der gerade betrachtete ↑Datenwürfel „Lücken“ aufweist, die nötigen ↑Basisdaten prinzipiell aber vorliegen, ob die jeweiligen Angaben bereits in den ↑Basisdaten unbekannt sind oder ob der jeweils betrachtete Teil einer ↑Objektmenge konzeptionell undefiniert bzw. leer ist. Letzterer Fall definiert einen *strukturellen* $\sim$.

Objekt Basisentität der „realen Welt“, aufgrund deren ↑Eigenschaften alle in *MADEIRA* betrachteten Datenwerte definiert sind.

Objektmenge In *MADEIRA* eine Klasse von ↑Objekten mit gleichen ↑Eigenschaften. Jeder ↑Datenraum beschreibt eine oder mehrere betrachtete $\sim$n (in bestimmten ↑Rollen).

On–line Analytical Processing (OLAP) Oberbegriff für die interaktive Analyse von ↑Data Warehouses auf der Grundlage eines ↑multidimensionalen Datenmodells. Je nachdem, ob das jeweils verwendete ∼–Tool die analysierten Daten physisch in Form von Relationen, multidimensionalen Feldern oder einer Kombination von beidem ablegt, spricht man von *relationalem*, *multidimensionalem* oder *hybridem* ∼ (*ROLAP*, *MOLAP* bzw. *HOLAP*).

On–line Transactional Processing (OLTP) ∼ beschreibt das Verarbeitungsparadigma, das — in Abgrenzung zum ↑On–line Analytical Processing — von „klassischen“ operationalen, transaktionsorientierten, auf die Verwaltung von ↑Mikrodaten ausgerichteten ↑Datenbank–Anwendungen verfolgt wird.

Parameter In *VIOLA* sind einem ↑Modul in der Regel ein oder mehrere ∼ zugeordnet, die die jeweilige Funktionalität näher spezifizieren. Es werden *multidimensionale* (deren Werte Entitäten und Entitätsmengen aus *MADEIRA* sind), *boolesche* und *Basis-*∼ (allgemein übliche Datentypen wie z. B. *Integer* oder *Float*) unterschieden. Multidimensionale ∼ können als *externe* ∼ über ∼-↑kanäle ihre Werte auch zu anderen ↑Modulen (bzw. deren entsprechenden ∼n) propagieren. Basis-∼ sind demgegenüber als *interne* ∼ stets lokal nur für ein ↑Modul definiert. Boolesche ∼ können schließlich sowohl die Rolle eines internen als auch eines externen ∼s annehmen.

Parametermodul ∼e dienen in *VIOLA* als eine Art von ↑Datenquellen zur initialen Bereitstellung von Werten externer ↑Parameter.

Partitionierung Disjunkte und vollständige ↑Klassifikation einer Menge.

Personenjahre Summe der Jahre, die Personen einer ↑Studienpopulation betrachtet wurden (also meist einfach das Produkt aus Bevölkerungszahl und gewähltem Zeitraum).

Population Menge von Personen, oftmals speziell die Einwohner einer bestimmten Region. In der ↑Epidemiologie werden ↑Studien- und ↑Standard-∼en unterschieden sowie oftmals durch Ausprägungen bestimmter ↑Eigenschaften definierte ↑Teil-∼en einer Gesamt-∼ betrachtet.

Port Ein- und/oder Ausgang eines ↑Moduls, also Anknüpfungspunkt für ↑Kanäle. Je nach Art des akzeptierten ↑Kanals werden in *VIOLA Daten-* und ↑*Parameter-*∼s unterschieden.

Prävalenz Der Zustand, erkrankt zu sein, bzw. die Anzahl bereits (an einer bestimmten Krankheit) erkrankter Personen in einer definierten ↑Population (in der Regel zu einem bestimmten Zeitpunkt).

Programmanimation Dynamische Darstellung der Abarbeitung eines Programms.

Programmvisualisierung Oberbegriff für ↑Programmanimation und (statische) visuelle Darstellung von Programmdaten oder -strukturen.

Rate Siehe ↑rohe oder ↑direkt standardisierte ∼.

Relationales Attribut ↑Attribut eines ↑Mikrodatenraums (im Sinne des relationalen ↑Datenmodells).

Relatives Risiko Quotient der ↑rohen Raten von ↑Studien- und ↑Standardpopulation.

Restriktion Die ∼ schränkt die Menge der ↑Kategorien eines oder mehrerer ↑kategorieller Attribute eines ↑Datenwürfels ein, d. h. es werden quasi „Scheiben“ aus dem ↑Datenwürfel herausgeschnitten.

Risikofaktor ↑Exposition bzw. biologische oder körperliche Disposition, die die Wahrscheinlichkeit des Auftretens einer Erkrankung erhöht.

Rohe Rate ↑Fallzahl bezogen auf die betrachtete Bevölkerung und den Zeitraum eines Jahres. Die ∼ wird i. a. in ↑Fällen pro Jahr (oder für die ↑Prävalenz zu einem Zeitpunkt) auf 100.000 Personen angegeben, wobei als Bevölkerung der Durchschnitt über das betrachtete Zeitintervall verwendet wird.

Rolle Jeder ↑Objektmenge kann in *MADEIRA* eine von verschiedenen ∼n zugeordnet werden, die sie bei der Berechnung bestimmter ↑Maßzahlen spielt.

Routing Ein ∼–↑Modul leitet auf der Basis eines selektierenden ↑Parameters die Daten *eines* ↑Kanals von mehreren eingehenden ↑Kanälen an seinen einzigen Ausgangs–↑Port weiter.

Screen–Space–Problem Das ∼ ist eines der Hauptprobleme ↑visueller Programmierung und besteht darin, daß der Anspruch, dem Anwender einen umfassenden Überblick über das erstellte Programm zu gewähren, oft an der beschränkten Größe eines Computerbildschirms bzw. der mangelnden Kompaktheit visueller Notationen scheitert.

Schneeflockenschema Normalisiertes ↑Sternschema, d. h. ↑Dimensionstabellen erhalten — im wesentlichen gemäß der jeweiligen ↑Kategorienhierarchien — noch diverse „Untertabellen“.

Sekundäre Notation Über die formalen, bedeutungstragenden Bestandteile einer (↑visuellen) Programmiersprache hinausgehenden Sprachelemente, die insbesondere der Verständlichkeit eines Programms dienen (etwa Kommentare, Layoutaspekte, Hervorhebungen etc.).

Selektion Der ∼s–Operator von *MADEIRA* beschränkt einen ↑Datenwürfel auf ausgewählte ↑summarische Attribute.

Signifikanzniveau Irrtumswahrscheinlichkeit eines ↑statistischen Tests, mit der eine Beobachtung als signifikante (nicht–zufällige) Auffälligkeit (etwa Abweichung von einem Erwartungswert) eingestuft werden kann.

Split In *MADEIRA* (weitgehend) inverse Operation zum ↑Merge, d. h. Auflösung der ↑Komposition von ↑Kategorien eines ↑kategoriellen Attributs.

Standardisierung Im Kontext der ↑Epidemiologie werden unter ∼ Verfahren subsumiert, die ↑Raten zu durch ein oder mehrere ↑Attribute (etwa Alter oder Geschlecht) definierten ↑Teilpopulationen unter Bezugnahme auf eine ↑Standardpopulation zu einer ↑Maßzahl für die gesamte betrachtete ↑Studienpopulation zusammenfassen. Hierdurch soll die Abhängigkeit der Gesamt–↑Rate von der Verteilung der jeweiligen ∼s–↑Attribute in der ↑Studienpopulation eliminiert werden, wodurch die erhaltenen ↑Maßzahlen eine bessere Vergleichbarkeit zwischen verschiedenen ↑Studienpopulationen gewährleisten.

Standardized Mortality/Incidence Ratio (SMR/SIR) Quotient aus ↑roher, also in der ↑Studienpopulation beobachteter ↑Rate und ↑erwarteter Rate.

Standardpopulation Bezugsbevölkerung zur Durchführung einer ↑Standardisierung. Man unterscheidet allgemein zwischen interner und externer ∼. Eine *interne* ∼ ist jeweils als die gesamte ↑Studienpopulation definiert — sie kann verwendet werden, wenn für ↑Teilpopulationen der ↑Studienpopulation ↑standardisierte Raten berechnet werden sollen. Eine *externe* ∼ ist dagegen unabhängig von der ↑Studienpopulation vorhanden. Dies kann etwa die ↑Population einer externen Vergleichsregion oder ein in seiner Bevölkerungsstruktur *künstlich* definierter Standard (Weltstandard, Europastandard etc.) sein.

Statistik Die ∼ befaßt sich mit der Analyse von Beobachtungen, die unter dem Einfluß des Zufalls oder in derartig komplexen Situationen, daß die Angabe kausaler Abhängigkeiten praktisch nicht möglich ist, entstanden sind. Man unterscheidet grob zwischen *beschreibender* (*deskriptiver*) ∼ zur Aufbereitung des Datenmaterials und *schließender* (*konfirmatorischer*, *induktiver*) ∼ zur Analyse und Beurteilung der Daten anhand konkreter Fragestellungen.

Statistische Datenbank Im wesentlich gleichbedeutend mit dem Begriff der ↑multidimensionalen Datenbank, jedoch weniger im Kontext des ↑Data Warehousing, sondern mehr im Rahmen der Modellierung, Verwaltung und Analyse sozioökonomischer Datenbestände verwendet. Analog sind *statistische* ↑*Datenbanksysteme* und ↑*Datenbankmanagementsysteme* definiert.

Statistischer Test Verfahren der ↑Statistik, das auf der Basis eines statistischen Modells dazu dient, eine gegebene Hypothese (oftmals die Signifikanz der Abweichung eines beobachteten von einem erwarteten Wert einer ↑Maßzahl) im Rahmen einer vorgegebenen Irrtumswahrscheinlichkeit (eines ↑Signifikanzniveaus) zu stützen oder zu verwerfen.

Statistische Tabelle Repräsentation von ↑Makrodaten in Form von Tabellen mit (gemäß der jeweiligen ↑kategoriellen Attribute) geschachtelten Zeilen- und Spaltenköpfen.

Sternschema Relationale Repräsentation von ↑Makrodaten in Form einer ↑Faktentabelle und mehrerer zugehöriger ↑Dimensionstabellen, die über Fremdschlüssel direkt mit der ↑Faktentabelle verknüpft sind.

Studienpopulation In einer Studie der ↑Epidemiologie untersuchte Personengruppe.

Summarisches Attribut Die ∼e eines ↑Datenwürfels spezifizieren die ↑Maßzahlen, deren Werte in dessen ↑Zellen enthalten sind. Weiterhin liefern sie über ihre ↑Summierungsfunktionen Informationen zur genauen Herleitung dieser Werte aus den ↑Basisdaten.

Summierungsfunktion In *MADEIRA* eine einem ↑summarischen Attribut eines ↑Datenwürfels zugeordnete Funktion, die angibt, wie Werte der jeweiligen ↑Maßzahl aus den durch den ↑Datenwürfel betrachteten ↑Objektmengen und deren ↑Eigenschaften (und damit aus den ↑Basisdaten) hergeleitet werden können. Allgemein wird der Begriff der ∼ gelegentlich auch synonym zur ↑Aggregierungsfunktion verwendet.

Symboldefinition Eine ∼ spezifiziert eine Menge ↑graphischer Attribute, die im Rahmen eines ↑Visualisierungsverfahrens zur Parametrisierung von ↑Darstellungssymbolen genutzt werden können, um eine ↑Datenreihe bestimmter maximaler Größe zu visualisieren.

Teilpopulation Teilmenge einer ↑Population, die in der Regel durch gemeinsame Ausprägungen ausgewählter ↑Eigenschaften definiert ist.

Verbale Programmierung Programmierung, die — im Gegensatz zur ↑visuellen Programmierung — auf den Einsatz visuell informativer Elemente verzichtet.

Verbund(modul) Ein ∼ kapselt ein Unterprogramm eines ↑Datenflußprogramms in einem eigenen ↑Modul und dient somit der prozeduralen Abstraktion bzw. der Bewältigung des ↑Screen–Space–Problems.

Vereinigung Die ∼ von ↑Datenwürfeln basiert auf einer an den jeweiligen ↑Koordinaten orientierten ∼ der Mengen ihrer ↑Zellen unter Berücksichtigung verschiedener ↑summarischer Attribute. Hierbei werden die — unter Identifikation und „Verschmelzung“ bestimmter ↑kategorieller Attribute — entstehenden Felder ggf. durch ↑Nullwerte aufgefüllt. In *MADEIRA* bzw. *VIOLA* bildet die ∼ eine zentrale Operation, da die meisten anderen Operatoren nur jeweils *einen* einzelnen ↑Datenwürfel als Eingabe akzeptieren.

Verzweigung Ein ∼s–↑Modul leitet die Daten seines einzigen Eingangs–↑Kanals auf der Basis eines selektierenden ↑Parameters an einen oder mehrere Ausgangs–↑Ports weiter.

Visualisierung Siehe ↑Datenvisualisierung.

Visualisierungsparameter Unter dem Begriff der ∼ werden alle Variablen zusammengefaßt, die über ihre Ausprägungen bzw. Belegungsvorschriften definieren, wie aus einer gegebenen Menge von Datenwerten eine konkrete ↑Graphik zu erzeugen ist. Hierunter fallen die durch ein ↑Visualisierungsverfahren genutzten ↑graphischen Attribute sowie weitere spezifische Angaben zum Layout.

Visualisierungsverfahren Ein ∼ beschreibt über ↑Symboldefinitionen auf Basis ↑graphischer Attribute von ↑Darstellungssymbolen eine allgemeine Vorgehensweise zur Erzeugung einer ↑Graphik eines bestimmten Typs.

Visuelle Programmiersprache Programmiersprache mit *visuell informativen*, also ausschließlich visuell wahrnehmbaren Elementen, die syntaktisch oder semantisch bedeutungstragend sind.

Visuelle Programmierumgebung Programmierumgebung zur Verwendung einer ↑visuellen Programmiersprache, wobei die Trennung zwischen Sprache und Umgebung oftmals stark verschwimmt.

Visuelle Programmierung Programmierung unter Verwendung einer ↑visuellen Programmiersprache. An Paradigmen zur ∼ werden u. a. steuerfluß-, funktions-, objekt-, constraint-, regel-, formular- und beispielorientierte Ansätze sowie die ↑Datenflußprogrammierung unterschieden.

Wurzelebene Gröbste ↑Aggregierungsebene einer ↑Kategorienhierarchie, die nur die ↑Wurzelkategorie enthält.

Wurzelkategorie Einzige ↑Kategorie der ↑Wurzelebene einer ↑Kategorienhierarchie, repräsentiert einen Gesamtwert.

Zelle Ein (evtl. aus Werten zu mehreren ↑Maßzahlen bestehender) Eintrag in dem durch einen ↑Datenwürfel definierten mehrdimensionalen Feld.

Zellenbezogene Operation Operation auf einem oder mehreren ↑Datenwürfeln, die zur Bestimmung einer ↑Zelle des Ergebnis–↑Datenwürfels jeweils nur die (im Hinblick auf ihre ↑Koordinaten) entsprechende ↑Zelle eines jeden Quell–↑Datenwürfels heranzieht. Es erfolgt also keine ↑Aggregierung.

Index

Die nachfolgende Aufstellung gibt zentrale (definierende oder erläuternde) Vorkommen wichtiger Begriffe inkl. ihrer Synonyme sowie von Namen und Abkürzungen in dieser Arbeit an. Unterstrichen sind Seitenzahlen zu Glossareinträgen, **fettgedruckte** Referenzen beziehen sich auf Definitionen zur Spezifikation des jeweiligen Terms in *MADEIRA* oder *VIOLA*, und in *Kursivschrift* erscheinen Seiten mit sonstigen expliziten Begriffs- bzw. Namenseinführungen. Wiederum steht „∼“ für den jeweiligen Begriff.

Symbolverzeichnis

Im folgenden sind im Laufe dieser Arbeit eingeführte Symbole in alphabetischer Reihenfolge (unter Berücksichtigung von Symbol–Spezialisierungen) mit Referenzen auf ihre wichtigsten Vorkommen aufgelistet. **Fettgedruckt** sind Seitenangaben von *MADEIRA*–/*VIOLA*–Definitionen global verwendeter Entitäten, Entitätsmengen und Funktionen; *kursiv* sind Seiten, auf denen weitere Bezeichner im Sinne einer Schreibkonvention erstmals verwendet werden. Einfache Indizierungen (a', a_1, ...) sind nicht gesondert angeführt, ebenso fettgedruckte Symbole, die jeweils Mengen von Mengen repräsentieren. Namen für spezielle Metadaten, boolesche Flags oder Kardinalitäten finden sich im Index.

M^*	Menge von Tupeln über einer Menge *M*, *98*
M^+	ohne das nullelementige (leere) Tupel, *98*
2^M	Potenzmenge einer Menge *M*, *98*
$\widetilde{2^{\mathcal{E}}}$	$2^{\mathcal{E}}$, beschränkt auf Eigenschaftsmengen zu jeweils gleicher Domäne, *172*
$\widetilde{2^{\mathcal{E}\times O\mathcal{R}}}$	$2^{\mathcal{E}\times O\mathcal{R}}$, gleichermaßen beschränkt, *172*, 182, 185, 188, 189
$\widetilde{2^{\mathcal{K}}}$	$2^{\mathcal{K}}$, beschränkt auf Teilmengen jeweils einer Domäne, *172*, 182, 185, 186, 188, 189
$\widetilde{2^{\mathcal{L}}}$	$2^{\mathcal{L}}$, beschränkt auf Ebenenmengen über jeweils gleicher Domäne, *172*
a.*b*	Komponente *b* einer Struktur *a*, *98*
b: *dom*	Kurzform für Port zu Parameter mit Namen *b* und Wertebereich *dom*, *174*
$\circ$	Komposition, 131
$\circ^d$	von Dimensionen, **131**
$\circ^D$	von Domänen, **131**
$\circ^E$	von Eigenschaften, **131**
$\circ^L$	von Aggregierungsebenen, **131**
$\lhd$	Aggregierbarkeit von Aggregierungsebenen, **107**, 108
$\preceq$	Subsumtion (Verfeinerung), 103
$\preceq^{\mathrm{L}}$	von Aggregierungsebenen, **107**
$\preceq^{\mathrm{K}}, \preceq_{e,O}$	(eingeschränkte) Subsumtion von Kategorien, **103**, **132**
$\preceq^{\mathrm{M}}$	von Maßzahlen, **158**
$\preceq^{\mathrm{feat}}$	von Merkmalswerten zu Maßzahlen, **156**
$\equiv^{\mathrm{K}}$	Äquivalenz von Kategorien, **103**
$\perp_{e,O}$	Disjunktheit von Kategorien, **103**
$\|\|, \|\|_{e,O}$	(eingeschränkte) Verwandtschaft von Kategorien, **103**, **132**
$\vee, \wedge, \neg$	Disjunktion, Konjunktion und Negation von Kategorien (Präfixnotation: $\bigvee, \bigwedge$), **103**
$\vdash_e$	Ausprägungen einer Objekteigenschaft, **103**
a, A	relationales Attribut bzw. Menge relationaler Attribute, *105*
$\mathcal{A}$	Menge aller relationalen Attribute, **105**, 139
ag	Analysegraph, *203*, 205, 211
ag_i	*i*–te Instanz des Analysegraphen eines Verbundmoduls, *211*, 224
$\mathcal{AG}$	Menge aller Analysegraphen, **203**, 204
b	Bezeichner, 171, 174, 175, 186, 188, 189, 194, 199, 201, 205

$\mathcal{B}$ Menge aller Bezeichner, *155*
$\mathbb{B}$ Menge der booleschen Werte, *98*
card maximale Kardinalität eines Datenports, *174*
d Dimension, *108*
d_D zu einer Domäne D, **108**
d_e zu einer Eigenschaft e, **108**
d^{kz} Kennzahl-, **128**
D Domäne, *103*, 106, 108, 116, 123, 159
D_e zu einer Eigenschaft, **103**
D_k zu einer Kategorie, **103**
D^{kz} Kennzahl-, **128**
dat, DAT Datenreihe bzw. Menge von Datenreihen, *193*
$\mathcal{DAT}$ Menge aller Datenreihen, **193**
$data(po^{\mathrm{dat}})$ Belegung eines Datenports po^{dat}, **174**, 177, **213**
$data(po^{\mathrm{out}}, po^{\mathrm{in}})$ eines Datenkanals $(po^{\mathrm{out}}, po^{\mathrm{in}})$, **213**
$\mathcal{DI}$ Menge aller Dimensionen, **108**, 131
$\mathcal{DO}$ Menge aller Domänen, **103**, 115, 128, 131, 159, 173, 184, 188
dom Wertebereich eines Parameters, *171*
dr, DR (Makro–)Datenraum bzw. Menge von (Makro–)Datenräumen, *124*
$\mathcal{DR}^{\mathrm{ma}}, \mathcal{DR}$ Menge aller Datenräume, **124**, 139–141, 143–145, 148–153, 173–175, 182–185, 188
dr^{mi} Mikrodatenraum, *105*
$\mathcal{DR}^{\mathrm{mi}}$ Menge aller Mikrodatenräume, **105**, 139, 151
e, E Eigenschaft bzw. Eigenschaftsmenge, *102*, 103, 105, 106, 117, 119, 120, 132
e_D einer als Objekt betrachteten Kategorie aus der Domäne D, **159**
e^{kz} Kennzahl-, **128**
$\mathcal{E}$ Menge aller Eigenschaften, **102**, 103, 131, 151, 173, 182, 184–186, 188
$\mathcal{E}_D$ zu einer Domäne, **103**
$\mathcal{E}_O^{\mu}$ aller mengenwertigen Eigenschaften, 103, **109**
$\mathcal{E}_O^{\sigma}$ aller einwertigen Eigenschaften, 103, **109**
eor, EOR Eigenschaft einer rollenbehafteten Objektmenge bzw. Menge davon, 117, 123, 179
$\mathrm{ext}(O)$ Extension einer Objektmenge O, *102*, 105, 266
$\mathrm{ext}(O, r)$ in Rolle r, *124*, 266
F Menge von Berechnungsvorschriften einer Maßzahl, *117*
$\mathcal{F}^{\mathrm{feat}}$ Menge aller Merkmale von Maßzahlen, **156**
$\mathcal{F}^{X,Y,\ldots}_{A,B,\ldots}$ Menge aller Funktionen von $A \times B \times \ldots$ nach $X \times Y \times \ldots$, *98*
$\mathcal{F}^{\mathrm{aggr}}_{T,D}$ Menge aller (kategorienbezogenen) Aggregierungsfunktionen, **116**
$f^{\cup}$ Vereinigung von Datenräumen, **145**, **184**
f^{abl} Ableitung eines Datenraums, **141**, **182**
f^{aggr} Aggregierungsfunktion, *116*, 117, 120, 121
f^{aggr}_{ka} eines summarischen bzgl. eines kategoriellen Attributs *ka*, *126*
f^{cell} zellenweise Maßzahlberechnung auf Datenräumen, **148**, 180
f^{cube} Erzeugung von Datenräumen aus Kategorienmengen, **160**
f^{dat} Berechnungsfunktion eines Moduls, *175*, 186, 188, 189, 201, 205, 208
f^{eaggr} explizite Aggregierung von Datenräumen, **149**, **185**
f^{eig} Eigenschaften eines Objekts, **102**
f^{ext} Extension, 105
f^{ext}_{a} eines Mikrodatenraums bzgl. eines relationalen Attributs *a*, *105*
f^{ext}_{sa} eines Makrodatenraums bzgl. eines summarischen Attributs *sa*, *124*, 125
f^{feat} Menge der Merkmale einer Maßzahl, **156**

f^{hom} homogenisierte Sicht auf einen Datenraum, **129**, 194
f^{id} identisch aggregierende Aggregierungsfunktion oder Identitätsfunktion, *116*
f^{in} Fan–in eines Ports, **212**
$f^{\text{in(e)}}$ externer, **203**
$f^{\text{in(i)}}$ interner, **212**
f^{join} allgemeine Maßzahlberechnung auf Datenräumen, **150**, 180
f^{ka} Zuordnung eines kategoriellen zu einem graphischen Attribut, *195*
f^{kons} Konsolidierung von Makro- aus Mikrodaten, **139**
f^{mass} Zuordnung von Maßzahlen zu Domänen als Dimensionsattribute, **159**
f^{merge} Merge von kategoriellen Attributen, **152**, **185**
f^{mik} Abfrage von Mikrodaten zu Zellen eines Datenraums, **151**
f^{mval} Maßzahlwerte zu Kategorien im Rahmen von Dimensionsattributen, **159**
f^{out} Fan–out eines Ports, **212**
$f^{\text{out(e)}}$ externer, **203**
$f^{\text{out(i)}}$ interner, **212**
f^{par} Parameterfunktion eines Moduls, *175*, 199, 205, 208
f^{parjoin} Kompositionsfunktion eines Parameterports, *214*, 241
f^{pp} Zuordnung interner Parameter in einem Verbundmodul, *205*, 238
f^{role} Umbenennung der Rolle einer Objektmenge, **153**, 180
f^{sel} Selektion summarischer Attribute, **143**, **183**
f^{split} Split zusammengesetzter kategorieller Attribute, **153**, **185**
f^{sum} Summierungsfunktion eines summarischen Attributs, *120*, 121, 125
f^{T} Zuordnung zu visualisierender Werte zu Kategorien, *195*
f^{val} genaueste Angabe zu einer Eigenschaft eines Objekts, **103**, 112, 151
f^{vis} Visualisierungsfunktion eines graphischen Attributs, *192*
g Graphik, *193*
$\mathcal{G}$ Menge aller Graphiken, **193**
ga, GA graphisches Attribut bzw. Menge graphischer Attribute, *192*
$\mathcal{GA}$ Menge aller graphischen Attribute, **192**
$graph(sym)$ Ausprägungen graphischer Attribute für ein Symbol *sym*, *193*
k, K Kategorie bzw. Kategorienmenge, *103*, 105, 106, 123, 179, 181
k_0 Wurzelkategorie einer Dimension, 103, *108*
k^{δ} δ–Kategorie zur Kategorie *k*, *111*
k^{δ}_{le} δ–Kategorie zur Kategorie *k* auf Ebene *le*, *111*
K_k aus einer Menge *K*, die zu *k* äquivalent sind, *140*
k^{σ} aus le_b, *112*
$\mathcal{K}$ Menge aller Kategorien, **103**, 115, 128, 131, 151, 159, 160, 173, 186
$\mathcal{K}^{\emptyset}$ die nie erfüllt werden, **103**, 106
$\mathcal{K}^{\delta}$ aller δ–Kategorien, **111**
$\mathcal{K}^{\text{NULL}}$ aller NULL–Kategorien, **111**
ka, KA kategorielles Attribut bzw. Menge kategorieller Attribute, *123*, 124, 126
KA^{dat} deren Kategorienkombinationen verschiedene Datenreihen identifizieren, *194*
KA^{k} klassifizierende Attribute eines Datenraums, *124*
ka^{kz} Kennzahlattribut, *129*
KA^{r} einschränkende Attribute eines Datenraums, *124*
KA_{sa} für ein summarisches Attribut *sa* relevante, **125**
KA^{val} deren Kategorien die Werte einer Datenreihe indizieren, *194*
$\mathcal{KA}$ Menge aller kategoriellen Attribute, **123**, 139, 141, 145, 149, 152, 153
$\mathcal{KA}_{EOR,K}$ zu Eigenschafts- und Kategorienmenge, **179**

$\mathcal{L}$ Menge aller Aggregierungsebenen, **106**, 107, 131, 173
$\mathcal{L}_D$ einer Domäne D, **106**
$\mathcal{L}^{\mu}_{e,O}$ aller mengenwertigen Aggregierungsebenen, **106**
$\mathcal{L}^{\sigma}_{e,O}$ aller einwertigen Aggregierungsebenen, **106**
$\mathcal{L}^{\omega}$, $\mathcal{L}^{\omega}_{e,O}$ aller (eingeschränkt) wohlgeformten Aggregierungsebenen, **106**, **132**
le, L Aggregierungsebene bzw. Menge von Aggregierungsebenen, *106*, 108, 111
le_0 Wurzelebene einer Dimension, *108*
le_{b} Basisebene zur Beschreibung mengenwertiger Eigenschaften, *112*
$le^{(i)}$ mit Kategorien, die i–elementige Mengen repräsentieren, *112*
le^{kz} Kennzahl-, **128**
$\mathcal{M}$ Menge aller Maßzahlen, **117**, 148–150, 156, 158–160, 172, 173, 183, 185–189, 201
mod, MOD *VIOLA*–Modul bzw. Menge von Modulen, 171, *174*, 175, 186–189, 194, 199–203, 211, 224
mod^{in} Eingangs-, *186*
mod^{iopar} Parameterschnittstellen-, *199*
mod^{loop} Schleifen-, *201*, 232
mod^{out} Ausgangs-, *194*
mod^{stop} Abbruch-, *201*, 234
mod^{sub} Verbund-, 186, 194, 199, 201, *205*, 232
$\mathcal{MOD}$ Menge aller *VIOLA*–Module, 171, **174**, 175
$\mathcal{MOD}^{\cup}$ aller Vereinigungsmodule, **188**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{abl}}$ aller (allgemeinen) Ableitungsmodule, **188**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{ablv}}$ aller Module zur vollständigen Ableitung, **188**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{aggr}}$ aller Module zur expliziten Aggregierung, **189**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{attr}}$ aller kategoriellen Datenquellen, **186**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{calc}}$ aller Module zur Maßzahlberechnung, **189**
$\mathcal{MOD}^{\mathrm{cell}}$ aller Module zur zellenweisen Maßzahlberechnung, **189**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{ctrl}}$ aller Kontrollmodule, **201**
$\mathcal{MOD}^{\mathrm{defpar}}$ aller Parameterdefinitionsmodule, **199**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{dm}}$ aller Datenmanagementmodule, **188**
$\mathcal{MOD}^{\mathrm{if}}$ aller Verzweigungsmodule, **201**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{in}}$ aller Eingangsmodule von Verbundmodulen, **186**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{iopar}}$ aller Parameterschnittstellenmodule, **199**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{join}}$ aller Module zur freien Maßzahlverknüpfung, **189**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{loop}}$ aller Schleifenmodule, **201**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{mak}}$ aller Makrodatenquellen, **186**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{merge}}$ aller Mergemodule, **188**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{mik}}$ aller Mikrodatenquellen, **186**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{out}}$ aller Ausgangsmodule von Verbundmodulen, **194**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{par}}$ aller Parametermodule, **199**
$\mathcal{MOD}^{\mathrm{role}}$ aller Module zur Rollenumbenennung, **188**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{route}}$ aller Routingmodule, **201**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{save}}$ aller Speichermodule, **194**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{sel}}$ aller Selektionsmodule, **188**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{sink}}$ aller Datensenken, **194**
$\mathcal{MOD}^{\mathrm{split}}$ aller Splitmodule, **188**, 208, 239, 255, 258
$\mathcal{MOD}^{\mathrm{src}}$ aller Datenquellmodule, **186**
$\mathcal{MOD}^{\mathrm{stop}}$ aller Abbruchmodule, **201**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{sub}}$ aller Verbundmodule, **205**, 208, 239, 255
$\mathcal{MOD}^{\mathrm{vis}}$ aller Visualisierungsmodule, **194**, 208, 239, 255

mz, MZ Maßzahl bzw. Menge von Maßzahlen, *117*, 120, 179
$mz_{e,O}$ Eigenschaft als Maßzahl, **119**
mz^{kz} Kennzahl-, **128**
mz^{q} Quell-, *117*
mz^{z} Ziel-, *117*
$\mathbb{N}$ Menge der natürlichen Zahlen, *98*
$\mathbb{N}_0$ mit 0, *98*
$\mathbb{N}^M$ Multimenge über einer Menge M, *98*
o, O Objekt bzw. Objektmenge, *102*, 103, 105, 106, 109, 119, 132
$\mathcal{O}$ Menge aller Objekte, **102**, 103, 159, 173
O_e durch eine Eigenschaft beschreibbare, **102**
$O_{e,k}$ mit Ausprägung k zur Eigenschaft e, **103**
O^{kz} die summarische Attribute repräsentieren (Kennzahlobjekte), **128**
or, OR rollenbehaftete Objektmenge bzw. Menge davon, *114*, 117, 120, 123, 124, 179
$\mathcal{OR}$ Menge aller rollenbehafteten Objektmengen, **114**, 151, 173, 182–186, 188, 201
par, PAR (Modul–)Parameter bzw. Menge von (Modul–)Parametern, *171*
par^{ext} externer, *174*, 208
par^{int} interner, *175*, 186, 188, 189, 194, 199, 205, 208
$\mathcal{PAR}$ Menge aller (Modul–)Parameter, **171**
$\mathcal{PAR}^{b}$ aller Basisparameter, **171**
$\mathcal{PAR}^{bool}$ aller booleschen Parameter, **171**
$\mathcal{PAR}^{ext}$ aller externen Parameter, **171**
$\mathcal{PAR}^{int}$ aller internen Parameter, **171**
$\mathcal{PAR}^{m}$ aller multidimensionalen Parameter, **171**
po, PO Port bzw. Menge von Ports, *174*, 211, 224
po^{dat} Daten-, *174*
po^{in} Eingangs-, *175*, 194, 199, 201, 205, 208, 211, 224
po^{loop} eines Schleifenmoduls, *232*
po^{out} Ausgangs-, *175*, 186, 194, 199, 201, 205, 208, 211, 226
po^{par} Parameter-, *174*, 175, 186–189, 194, 199, 201, 205, 211, 214, 228
po^{sub} eines Verbundes, von dem ein Parameter-/Schleifenmodul abhängt, *199*, 201
$\mathcal{PO}$ Menge aller Ports, **174**
$\mathcal{PO}^{dat}$ aller Datenports, **174**
$\mathcal{PO}^{par}$ aller Parameterports, **174**
PPO Menge von Paaren von Datenein- und -ausgangsports, *177*, 188, 189, 194, 201, 205
r, R Rolle bzw. Menge von Rollen, *102*
r^{kz} Kennzahl-, **128**
$\mathcal{R}$ Menge aller Rollen, **102**, 139, 153, 160, 173, 186, 188
$\mathbb{R}$ Menge der reellen Zahlen, *98*
REL Verbindungsrelation eines Analysegraphen, *203*, 211
REL^{dat} Datenkanäle, *203*, 211
REL^{par} Parameterkanäle, *203*, 211
$\mathcal{S}$ Menge aller endlichen Zeichenketten, *155*
sa, SA summarisches Attribut bzw. Menge summarischer Attribute, *120*, 124
sa^{kz} „Verteiler“–Attribut in homogenisierter Datenraumsicht, *129*
$\mathcal{SA}$ Menge aller summarischen Attribute, **120**, 128, 139, 143
$\mathcal{SA}_{MZ,OR}$ zu Maßzahlmenge und Menge rollenbehafteter Objektmengen, **179**
sd, SD Symboldefinition bzw. Menge von Symboldefinitionen, *192*, 193
$\mathcal{SD}$ Menge aller Symboldefinitionen, **192**

sym, SYM	(Darstellungs–)Symbol bzw. Menge von Symbolen, *193*
$\mathcal{SYM}$	Menge aller Darstellungssymbole, **193**
T	Datentyp, *115*, 116, 117, 192
T^{kz}	Kennzahl-, **128**
t_0, T_0	ein oder mehrere neutrale Elemente eines Datentyps T, *115*, 117, 125
$\mathcal{T}$	Menge aller Datentypen, **115**
T^N	Nullwerte eines Datentyps T, *115*, 117
t_{navail}	Nullwert: nicht verfügbar, *115*, 121, 146
t_{nexist}	struktureller Nullwert, *115*, 121, 125
$t_{unknown}$	Nullwert: unbekannt, *115*
$val(par)$	Belegung eines Parameters *par*, **171**, 177, 213, 214
$val(po^{par})$	eines Parameterports po^{par}, **214**
$val(po^{par}, po^{par\prime})$	eines Parameterkanals $(po^{par}, po^{par\prime})$, **214**
vv, VV	Visualisierungsverfahren bzw. Menge von Visualisierungsverfahren, *192*, 193, 194
$\mathcal{VV}$	Menge aller Visualisierungsverfahren, **192**
W	Wertebereich (eines graphischen Attributs), *192*
$\mathcal{W}$	Menge aller Maßzahlwerte, **115**, 159, 160, 171
x, X	Ausprägung einer Maßzahl bzw. Menge von Ausprägungen, *116*
Ω	Menge aller Werte von Modulparametern, **171**
$\mathbb{Z}$	Menge der ganzen Zahlen, *98*

Verzeichnis der Definitionen

Verzeichnis der Algorithmen

Lebenslauf

Persönliche Daten

Name:	Frank Wietek
Geburtsdatum und -ort:	7. Juli 1969, Hamburg
Wohnort:	Kleinfeld 82a, 21149 Hamburg
Familienstand:	verheiratet

Schulbesuch

8/75 – 7/79	Grundschule in Neu Wulmstorf
8/79 – 5/88	Gymnasium in Neu Wulmstorf Abschluß: Allgemeine Hochschulreife

Wehrdienst

10/88 – 12/89	Grundwehrdienst in Daun und Lüchow–Dannenberg

Studium

10/89 – 8/94	Informatik an der Carl von Ossietzky Universität Oldenburg Vertiefungsfach Praktische Informatik, Nebenfach Mathematik Abschluß: Diplom–Informatiker Thema der Diplomarbeit: „Eine wissensbasierte generische Statistikkomponente für die deskriptive Epidemiologie“

Berufstätigkeit

10/92 – 6/93	Wissenschaftliche Hilfskraft in den Abteilungen Programmiersprachen und -systeme bzw. Informationssysteme des Fachbereichs Informatik der Carl von Ossietzky Universität Oldenburg
7/93 – 7/94	Wissenschaftliche Hilfskraft ohne Abschluß im Forschungsbereich Informationssysteme und Wissensverarbeitung am Oldenburger Forschungs- und Entwicklungsinstitut für Informatik–Werkzeuge und –Systeme (OFFIS)
9/94 – 9/99	Wissenschaftlicher Mitarbeiter in der Abteilung Informationssysteme des Fachbereichs Informatik der Carl von Ossietzky Universität Oldenburg
10/99 – 2/00	Wissenschaftlicher Mitarbeiter im Bereich Informations- und Kommunikationssysteme im Gesundheitswesen am OFFIS
seit 3/00	Senior–Software–Ingenieur bei der sd&m AG, Hamburg