Zur Homepage der Dissertation

Struktur und Verhalten von verteilten endlichen Automaten

Dissertation
zur Erlangung des Doktorgrades
des Fachbereichs Informatik
der Carl-von-Ossietzky-Universität
Oldenburg

vorgelegt von
Henning Reineke

Oldenburg
1995

Gutachter: Prof. Dr. Volker Claus
Prof. Dr. Ernst-Rüdiger Olderog
Prof. Dr. Volker Diekert

Tag der mündlichen Prüfung: 23. Juni 1995

Fachbereich Informatik
Carl-von-Ossietzky-Universität
Postfach 2503
26111 Oldenburg

Zusammenfassung

Traces sind Verallgemeinerungen von Wörtern auf nebenläufige Prozesse: Dem Alphabet wird eine symmetrische und antireflexive *Unabhängigkeitsrelation* auf Zeichen zugeordnet. Wörter sind Repräsentanten desselben *Traces*, wenn sie sich nur in der Reihenfolge von aufeinanderfolgenden unabhängigen Zeichen unterscheiden. Viele aus der Theorie der formalen Sprachen bekannte Begriffe und Konzepte lassen sich ebenso für Traces und Trace-Sprachen formulieren und anwenden.

Im Jahre 1987 definierte Zielonka mit dem *endlichen asynchronen Automaten* den Typ eines verteilten Automaten zur Erkennung von Trace-Sprachen. Die Klasse der *erkennbaren Trace-Sprachen* ist daher eine Verallgemeinerung der Klasse der regulären (Wort-)Sprachen. Obwohl die erkennbaren Trace-Sprachen ein vielfältigeres und interessanteres Gebiet darstellen als die Klasse der regulären Wortsprachen, scheinen sie bisher weniger untersucht worden zu sein.

In dieser Arbeit werden Teilklassen der erkennbaren Trace-Sprachen durch Betrachtung der verteilten Struktur und des nebenläufigen Verhaltens des endlichen asynchronen Automaten definiert. In diesem Zusammenhang spielen die Mechanismen eine besondere Rolle, nach denen die Teilautomaten kooperieren. Die definierten Teilklassen werden unabhängig vom Automatentyp charakterisiert, so daß sie allgemeine Eigenschaften (erkennbaren) nebenläufigen Verhaltens beschreiben. Es werden Eigenschaften der Sprachklassen und Beziehungen zwischen den Sprachklassen untersucht. Hierbei dient die Unabhängigkeitsrelation als Parameter, z. B. stimmen die Teilklassen mit der Klasse der erkennbaren Trace-Sprachen bei leerer Unabhängigkeitsrelation überein. Drei Teilklassen sind identisch, wenn nur präfixabgeschlossene Sprachklassen betrachtet werden.

Zielonkas Konstruktion eines endlichen asynchronen Automaten zu einer gegebenen Trace-Sprache ist ebenso kompliziert wie aufwendig. Von daher ergibt sich ein weiteres Motiv für die Betrachtung von Teilklassen der erkennbaren Trace-Sprachen. Es zeigt sich jedoch, daß die hier definierten Sprachklassen keine einfachere Automatenkonstruktion zulassen. Andererseits offenbart die Untersuchung von Zielonkas Konstruktionsverfahen eine entscheidende Vereinfachung für erkennbare Trace-Sprachen, die auf Alphabeten mit einer speziellen Unabhängigkeitsrelation definiert sind.

Im Anhang wird das Trace Modell einer Fertigungszelle vorgestellt. Das Verhalten wird mit Hilfe der hier definierten Begriffe interpretiert.

Abstract

Traces are generalizations of words to concurrent processes. This is done by adjoining a symmetric and antireflexive *independence relation* on letters to the alphabet. Words will be representatives of the same *trace*, if their sequences of letter occurences differ only in the order of neighbouring independent letters. Most of the well-known concepts and notions from formal language theory are also applicable to traces and trace languages.

In 1987 Zielonka defined the *finite asynchronous automaton* as a distributed device for recognizing trace languages. Thus, the *class of recognizable trace languages* is a generalization of the class of ordinary regular languages. Although the recognizable trace languages constitute a more complex and interesting field than the ordinary regular languages, they seem to be much less investigated.

In this paper subclasses of the recognizable trace languages are defined by considering the distributed structure and concurrent behaviour of the finite asynchronous automaton. In this context the mechanisms of cooperation between the subautomata are of special interest. We characterize these subclasses independently of the automaton. So they represent properties of (recognizable) concurrent behaviour in general. In order to establish a theory of these language classes we analyze their properties and mutual relationships. The independence relation serves as a parameter in this examination, e. g. if the independence relation is empty the subclasses will be identical with the original class of recognizable trace languages. Closure properties and set-theoretic relationships are examined with respect to the independence relation. Three subclasses coincide, if we restrict to prefix closed language classes.

Zielonka's construction of a finite asynchronous automaton recognizing a given trace language uses a complicated and expensive algorithm. This reason is another motivation for the restriction to language subclasses. However, it is shown that the defined subclasses do not allow a simpler construction. The analysis of the construction given by Zielonka reveals a significant simplification in case of recognizable trace languages defined on alphabets with a special dependence structure.

In the appendix the trace model of a production cell is given. Its behaviour is interpreted in terms of the introduced notions.

Danksagung

Mein erster Dank gilt Prof. V. Claus, der mir nicht nur die Arbeit an der Universität ermöglichte, sondern mir bei der Wahl des Themas dieser Dissertation freie Hand ließ. Von seinen vielen Anregungen sind einige in die Dissertation eingeflossen und haben deren Richtung wesentlich bestimmt.

Prof. E.-R. Olderog nahm mich nach dem Wechsel von Prof. Claus nach Stuttgart in seiner Abteilung auf. Ihm und Prof. Diekert sei dafür gedankt, die Begutachtung der Dissertation übernommen und die Gutachten sehr zügig angefertigt zu haben.

Von vielen Hinweisen habe ich drei letztlich in die Arbeit aufgenommen: Anca Muscholls Erklärungen zu triangulierten Graphen und asynchronen Abbildungen, Elke Wilkeits Literaturhinweis auf [MM65] und Prof. Olderogs sowie Wil Janssens Hinweis auf *Communication Closed Layers*, in diese Arbeit eingegangen als Lemma 2.3.10.

Am Ende will ich mich bei einer Reihe von Kollegen und Kolleginnen bedanken, die in den letzten paar Jahren meine mit der Dissertation verbundenen Sorgen und Erfolgserlebnisse mit mir geteilt haben: Ulli Bartels, Hans Fleischhack, Ulla Levens, Ulrike Lichtblau, Gerlinde Schreiber, Elke Wilkeit.

Inhaltsverzeichnis

Kapitel 1

Einführung

Petrinetze sind ein bewährtes Beschreibungsmittel für verteilte diskrete Systeme. Mit einem Petrinetz wird ein solches System in doppelter Hinsicht beschrieben: Die Zerlegung in passive und aktive Komponenten als Stellen und Transitionen gibt die Struktur des Systems wieder. Über die Schaltregel ist definiert, wie das Netz den Systemzustand wechselt, so daß durch die Netzbeschreibung auch das Systemverhalten gegeben ist. Ausgangspunkt dieser Arbeit ist die Frage, wie zu einem gegebenen Systemverhalten algorithmisch eine Netzstruktur konstruiert werden kann.

Für die Darstellung von Abläufen in Petrinetzen sind eine Reihe von Konzepten entwickelt und benutzt worden. In [CR89] wurden für sichere Petrinetze mehrere dieser Ansätze verglichen und ihre Äquivalenz bewiesen. D. h. es gibt unterschiedliche Formalisierungen von Abläufen in Petrinetzen, die jedoch ineinander überführt werden können. Wenn wir über das Verhalten von Petrinetzen sprechen, reicht es daher aus, nur einen dieser Ablaufbegriffe heranzuziehen.

Eines der in [CR89] behandelten Konzepte ist der *Trace-Begriff*. Ein Trace ist ein „Wort“ über einem Alphabet, in dem durch Einführung einer „Unabhängigkeitsrelation“ auf Zeichen die Nebenläufigkeit von Zeichenvorkommen ausgedrückt wird. So nah dieser Begriff an der Theorie der formalen Sprachen angesiedelt ist, so einfach lassen sich Begriffe aus den formalen Sprachen auf die Trace-Theorie übertragen. Traces wurden zuerst durch Cartier und Foata (vgl. [CF69]) als Elemente eines partiell-kommutativen Monoids definiert. Mazurkiewicz prägte die Bezeichnung „Trace“ und stellte den Zusammenhang zu den Petrinetzen her (vgl. [Maz77, Maz87]).

In der Literatur finden sich zur Konstruktion von sicheren Petrinetzen bei gegebener Trace-Sprache als Netzverhalten zwei Ansätze: Graubmann ([Grau88]) übernimmt Mazurkiewiczs Definition des Trace-Verhaltens eines Netzes und setzt die Transitionenmenge des Netzes gleich dem Alphabet. Die Klasse der Trace-Sprachen, zu denen nach seiner Methode ein Netz konstruiert werden kann, wird so allerdings allein durch den Netztyp bestimmt, denn nicht einmal zu jeder endlichen Trace-Sprache läßt sich auf diese Weise ein sicheres Petrinetz konstruieren. Beispielsweise läßt sich kein sicheres Netz angeben, dessen einzige Transition genau zweimal schaltet. Also läßt sich das Verhalten „Das einzige bekannte Ereignis tritt zweimal ein“ nach Graubmann nicht durch ein sicheres Netz darstellen.

Einen anderen Weg beschreitet Zielonka in [Ziel87]. Er konstruiert zu jeder erkennbaren Trace-Sprache einen *endlichen asynchronen Automaten*, der als sicheres transitionsbeschriftetes Petrinetz dargestellt werden kann. Dabei sind die erkennbaren Trace-Sprachen netzunabhängig als die Klasse der von endlichen Automaten erkennbaren Trace-Sprachen charakterisierbar. Unglücklicherweise ist Zielonkas Konstruktion aufwendig und relativ kompliziert.

In dieser Arbeit werden durch systematische Einschränkung des Konzepts des endlichen asynchronen Automaten Teilklassen der erkennbaren Trace-Sprachen definiert. Es wird untersucht, ob zu den Trace-Sprachen aus den Teilklassen eine einfachere Automatenkonstruktion möglich ist. Die Betrachtung von Teilklassen ist naheliegend, weil durch die volle Klasse der erkennbaren Trace-Sprachen mehr als das im intuitiven Sinne nebenläufige erkennbare Verhalten abgedeckt wird. So ist jede endliche Trace-Sprache von einem Automaten erkennbar, aber nicht jede ist anschaulich das Verhalten eines verteilten Systems. Wir werden sehen, daß die in der Definition des endlichen asynchronen Automaten verwendeten globalen Konzepte (insbesondere Endzustände) in diesem Zusammenhang von Bedeutung sind.

Um eine über das spezielle Automatenmodell hinausgehende Relevanz der Sprachklassen zu zeigen und sie gleichzeitig in den Eigenschaften ihrer Sprachen voneinander abzugrenzen, werden automatenunabhängige Charakterisierungen der Klassen angegeben. Einmal vom Automatenmodell losgelöst, werden Abschlußeigenschaften der Klassen und Querbeziehungen nachgewiesen. Bei diesen Untersuchungen dient die Unabhängigkeitsrelation des Alphabets als Parameter. Sie ist quasi ein Maß für die Nebenläufigkeit im System. Über einem Alphabet mit leerer Unabhängigkeitsrelation stimmen beispielsweise alle betrachteten Sprachklassen mit der Klasse der regulären (Wort-)Sprachen überein. Das bedeutet umgekehrt, daß durch Einführung von Nebenläufigkeit die Klasse der regulären Wortsprachen zu den definierten Trace-Sprachklassen verallgemeinert wird, wobei jeweils ein anderer Aspekt nebenläufigen Verhaltens Gegenstand der Verallgemeinerung ist.

Erste Ansätze, Teilklassen der erkennbaren Trace-Sprachen so zu definieren, gab es bereits in [Dub86] und [Ziel87]. Dort wurden zwei Teilklassen betrachtet, die sich aus den regulären Wortsprachen durch Synchronisation und Vereinigung ergeben. Die beiden Sprachklassen werden hier im Zusammenhang mit den neudefinierten Sprachklassen weiter untersucht.

Die Arbeit gliedert sich folgendermaßen: Die Kapitel 2 bis 4 enthalten die Grundlagen der Arbeit. Kapitel 2 bringt die Definitionen der Begriffe, die im weiteren gebraucht werden und in der Fachliteratur zu finden sind. Kapitel 3 enthält grundlegende bekannte Ergebnisse aus der Literatur, die den Rahmen des hier behandelten Themas bilden und auf denen die Arbeit aufbaut. Dabei wird den bekannten Konstruktionsverfahren, die erst in Kapitel 4 folgen, ein eigener Abschnitt zugestanden.

Den Hauptteil der Arbeit bildet Kapitel 5. Dort finden sich die Definitionen, Abschlußeigenschaften und Teilmengenbeziehungen der neudefinierten Teilklassen. Das Kapitel wird mit einem Abschnitt beendet, in dem beschrieben wird, wie sich die Ergebnisse nach Einschränkung auf präfixabgeschlossene Trace-Sprachen vereinfachen. Kapitel 6 behandelt das ursprüngliche Ziel der Arbeit, die Konstruktion von endlichen asynchronen Automaten zu Trace-Sprachen aus den definierten Teilklassen. In Kapitel 7 wird eine Einschätzung der Ergebnisse vorgenommen und ein Ausblick gegeben.

Um zu zeigen, wie Traces in der Praxis angewendet werden würden, werden in Anhang A die Abläufe in einer Fertigungszelle mit Traces modelliert und das Ergebnis der Modellierung mit den in dieser Arbeit behandelten Begriffen interpretiert. In Anhang B werden die hier benutzten graphentheoretischen Begriffe zusammengefaßt.

Kapitel 2

Grundlagen

Mit diesem Kapitel wird der Begriff *Trace* eingeführt und die grundlegenden Eigenschaften der Traces erklärt. Es werden Trace-Sprachen und die Klassen der erkennbaren und der rationalen Trace-Sprachen definiert. In den letzten beiden Abschnitten werden die Grundlagen der Petrinetztheorie und der endlichen asynchronen Automaten behandelt.

Die Menge der natürlichen Zahlen $\{1, 2, \ldots\}$ wird mit $\mathbb{N}$ bezeichnet. Es ist $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

2.1 Wörter und Wortsprachen

Zunächst werden in diesem Abschnitt Bezeichnungen und Schreibweisen für Alphabete, Wörter und Wortsprachen geklärt.

Definition 2.1.1 (Alphabet, Zeichen)

Ein Alphabet ist eine nichtleere endliche Menge. Seine Elemente werden Zeichen genannt.

Durch Hintereinanderschreiben („Verketten“) der Zeichen werden Wörter gebildet. Die Menge der Wörter bildet zusammen mit der assoziativen Operation Verkettung und einem neutralen Element die algebraische Struktur „freies Monoid“.

Definition 2.1.2 (Wörter, Verkettung)

Die Grundmenge des freien Monoids über dem Alphabet Σ wird mit Σ^* bezeichnet. Elemente aus Σ^* sind Wörter über dem Alphabet Σ. Die Operation $\circ$ auf Wörtern, die das freie Monoid definiert, heißt Verkettung. (Der Operator $\circ$ wird zur besseren Lesbarkeit meist nicht geschrieben.) Das neutrale Element der Verkettung wird leeres Wort genannt und mit ε bezeichnet.

Da das Monoid der Wörter frei erzeugt ist, läßt sich jedes nichtleere Wort eindeutig als Verkettung von Zeichen darstellen. Ein Wort $w \in \Sigma^*$ ist also $w = a_1 \ldots a_n$ mit eindeutigem $n \in \mathbb{N}_0$ und eindeutigen $a_1, \ldots, a_n \in \Sigma$.

Definition 2.1.3 (Länge, Alphabet und Spiegelwort eines Worts)

Sei $w = a_1 \dots a_n \in \Sigma^*, n \in \mathbb{N}_0$, ein Wort über Σ.

i) Die Länge von w (geschrieben $|w|$) ist n.

ii) Das Alphabet von w ist die Menge Alph(w)$=_{Df}$ $\{a_1, \dots, a_n\} \subseteq \Sigma$.

iii) $w^R =_{Df} a_n \dots a_1$ ist das Spiegelwort zu w.

Durch Numerierung gleicher Zeichen können die Vorkommen eines Zeichens im Wort unterschieden werden. Die Abfolge der Zeichenvorkommen legt nahe, ein Wort als geordnete Menge seiner Zeichenvorkommen aufzufassen. Es läßt sich eine Halbordnung (eine transitive identitive zweistellige Relation) auf den Vorkommen definieren.

Definition 2.1.4 (Zeichenvorkommen, Ordnung eines Worts)

Sei $w = a_1 \dots a_n \in \Sigma^*, n \in \mathbb{N}_0$, ein Wort über Σ.
Die Menge der Zeichenvorkommen von w ist

$$Vor(w) =_{Df} \left\{(a_i, n_i) \in \Sigma \times \mathbb{N} \middle| 1 \leq i \leq n \wedge n_i = |\{j \in \mathbb{N} | a_i = a_j \wedge j \leq i\}|\right\}$$

Die Anzahl der Vorkommen des Zeichens a in w ist $| w |_a =_{Df} |\{(a_i, n_i) \in Vor(w) \mid a_i = a\}|$.
Die Ordnung von w ist das Paar Ord(w)$=_{Df}$(Vor(w),$\leq_w$) mit

$$(a_i, n_i) \leq_w (a_j, n_j) \Longleftrightarrow_{Df} i \leq j \text{ für } i, j \in \{1, \dots, n\}$$

$Ord(w)$ ist eine totale Halbordnung, d. h. für zwei Vorkommen $x, y \in Vor(w)$ gilt $x \leq_w y$ oder $y \leq_w x$.

Beispiel: $Vor(abbab) = \{(a,1), (b,1), (b,2), (a,2), (b,3)\}$
$Ord(abbab)$ ist die Halbordnung, die durch den Graphen in Abbildung 2.1 dargestellt ist.

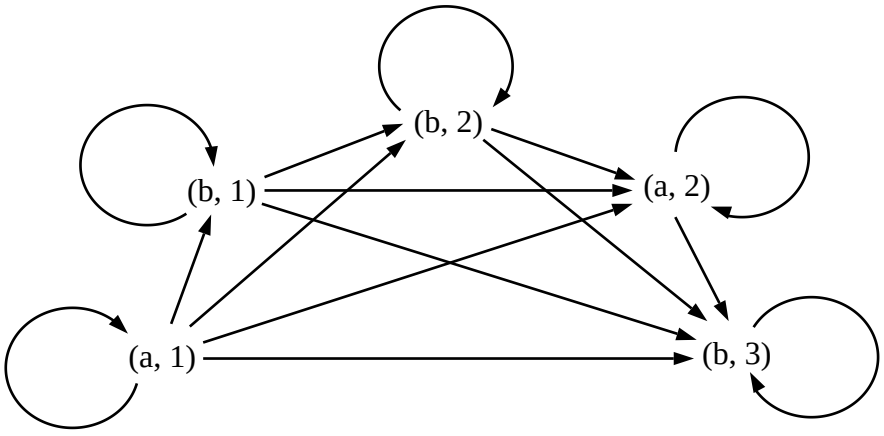

Abbildung 2.1: Die Halbordnung $Ord(abbab)$

Gewöhnlich werden in der Darstellung von Halbordnungen solche Pfeile weggelassen, die Reflexivität ausdrücken oder sich aufgrund der Transitivität aus anderen Pfeilen ergeben. Der gerichtete Graph in Abb. 2.1 wird so reduziert auf $(a,1) \to (b,1) \to (b,2) \to (a,2) \to (b,3)$.

Definition 2.1.5 (Anfangsstück)

Die Menge der Anfangsstücke eines Worts $w \in \Sigma^*$ ist

$$Pref(w) =_{Df} \{u \in \Sigma^* | \bigvee_{v \in \Sigma^*} uv = w\}$$

Definition 2.1.6 (Projektion)

Seien Σ_1 und Σ_2 zwei Alphabete. Die Projektion $w|_{\Sigma_2}$ des Worts $w \in \Sigma_1^*$ auf Σ_2 wird induktiv definiert:

$$w|_{\Sigma_2} =_{Df} \begin{cases} \varepsilon & falls \quad w = \varepsilon \\ v|_{\Sigma_2} a & falls \quad w = va \wedge a \in \Sigma_1 \cap \Sigma_2 \\ v|_{\Sigma_2} & falls \quad w = va \wedge a \notin \Sigma_1 \cap \Sigma_2 \end{cases}$$

Eine zentrale Bedeutung hat der Begriff der Sprache über einem Alphabet.

Definition 2.1.7 (Sprache)

Eine (Wort-)Sprache über einem Alphabet Σ ist eine Menge von Wörtern, also eine Teilmenge von Σ^*.

Die für Wörter definierten Begriffe und Operationen müssen auf Sprachen übertragen werden. Als noch nicht für Wörter definierte Operation kommt die *Iteration* hinzu.

Definition 2.1.8 (Operationen auf Sprachen)

i) Die Verkettung zweier Sprachen $L_1, L_2 \subseteq \Sigma^*$ wird definiert als

$$L_1 \circ L_2 =_{Df} \{vw \in \Sigma^* | v \in L_1 \wedge w \in L_2\}$$

(Für $L_1 \circ L_2$ wird kurz $L_1 L_2$ geschrieben.)

ii) Der Präfixabschluß einer Sprache $L \subseteq \Sigma^*$ ist

$$\begin{aligned} Pref(L) &=_{Df} \{u \in \Sigma^* | \bigvee_{v \in \Sigma^*} uv \in L\} \\ &= \bigcup_{w \in L} Pref(w) \end{aligned}$$

iii) Die Spiegelsprache einer Sprache $L \subseteq \Sigma^*$ ist

$$L^R =_{Df} \{w \in \Sigma^* | w^R \in L\}$$

iv) Die Projektion einer Sprache $L \subseteq \Sigma_1^*$ über Σ_1 auf ein zweites Alphabet Σ_2 ist

$$L|_{\Sigma_2} =_{Df} \{w \in \Sigma_2^* | \bigvee_{v \in L} w = v|_{\Sigma_2}\} = \{v|_{\Sigma_2} | v \in L\}$$

v) Die Iteration einer Sprache $L \subseteq \Sigma^*$ *ist*

$$L^* =_{Df} \bigcup_{n \in \mathbb{N}_0} L^n,$$

wobei $L^0 =_{Df} \{\varepsilon\}$ *und* $L^{n+1} =_{Df} L^n L$ *für* $n \in \mathbb{N}_0$. L^+ *ist die Sprache der positiven Iterationen von* L*:*

$$L^+ =_{Df} \bigcup_{n \in \mathbb{N}} L^n$$

2.2 Traces und Trace-Sprachen

Der Begriff des Traces ist eine Verallgemeinerung von gewöhnlichen Wörtern über einem Alphabet auf nebenläufige Prozesse. Prinzipiell können alle aus der Theorie der formalen Sprachen bekannten Begriffe und Konzepte ebenso für Traces formuliert werden.

Nebenläufigkeit wird ins Spiel gebracht, indem von der Reihenfolge bestimmter Alphabetsymbole abstrahiert wird. Seien beispielsweise a, b und c drei Aktionen, wobei bekannt ist, daß b und c unabhängig voneinander ausgeführt werden können, weil sie auf disjunkte Mengen von Ressourcen zugreifen oder weil sie voneinander isolierten Systemen zugeordnet sind. Betrachtet wird die Folge von Ereignissen *abc*. Während „b nach a“ möglicherweise eine Reihenfolgebedingung bezeichnet, die für alle Abläufe über a, b und c wesentlich ist, wird „c nach b“ aufgrund der Unabhängigkeit von b und c nur eine unwichtige Eigenschaft dieser Folge sein. Die Folge *acb* würde die wesentlichen Reihenfolgebedingungen ebenso einhalten. Man sagt, *abc* und *acb* bezeichnen denselben Trace.

Die Bezeichnung „Trace“ und die Beschreibung des Verhaltens von Petrinetzen durch Traces geht auf Mazurkiewicz ([Maz77]) zurück. Eine umfassende Darstellung der Theorie der Traces findet sich in [AR88], [Maz87] und [Diek90]. Die Menge der Traces über einem festen Alphabet bildet zusammen mit der Verkettungsoperation ein freies partiell kommutatives Monoid. Diese algebraische Struktur wurde bereits 1969 von Cartier und Foata zur Untersuchung von kombinatorischen Problemen auf Wörtern eingeführt (vgl. [CF69]).

Andere Ansätze für die Darstellung nebenläufiger Prozesse sind die Kalküle CCS und CSP von Milner bzw. Hoare ([Mil80], [Hoa85]). Dabei werden Abläufe durch Ausdrücke beschrieben, in denen die Nebenläufigkeit von Aktionen durch die Verwendung von speziellen Operatoren explizit formuliert wird. Im Unterschied zu den Traces resultiert Nebenläufigkeit hier nicht aus einer besonderen Eigenschaft der Aktionen, sondern ist wie die sequentielle Verkettung eine Operation auf Prozessen.

Definition 2.2.1 (Alphabet mit Unabhängigkeitsrelation)

Ein Alphabet mit Unabhängigkeitsrelation ist ein Paar (Σ, I) *bestehend aus einem gewöhnlichen Alphabet* Σ *und einer antireflexiven symmetrischen zweistelligen Relation (genannt Unabhängigkeitsrelation)* $I \subseteq \Sigma^2$*.*
(Es gilt also $(a,a) \notin I$ *und* $(a,b) \in I \iff (b,a) \in I$ *für alle* $a, b \in \Sigma$*.)*

$D =_{Df} \Sigma^2 \setminus I$ ist die reflexive symmetrische Abhängigkeitsrelation des Alphabets.

I heißt quasi-transitiv $\Longleftrightarrow_{Df} \bigwedge_{a,b,c \in \Sigma} (a \neq c \wedge (a,b) \in I \wedge (b,c) \in I \Rightarrow (a,c) \in I)$

Die Unabhängigkeit von Zeichen soll zur Folge haben, daß Wörter als äquivalent angesehen werden, die sich nur in der Reihenfolge unabhängiger Zeichen unterscheiden. In Definition 2.2.2 wird zunächst die Ähnlichkeit von solchen Wörtern erklärt und in Definition 2.2.3 die Ähnlichkeit dann zu einer Äquivalenzrelation fortgesetzt.

Definition 2.2.2 (Ähnlichkeit von Wörtern)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und seien $v, w \in \Sigma^*$ Wörter über Σ. Dann heißen v und w ähnlich ($v \cong_I w$), wenn sie sich nur in der Reihenfolge von zwei unmittelbar aufeinanderfolgenden unabhängigen Zeichen unterscheiden:

$$v \cong_I w \Longleftrightarrow_{Df} \bigvee_{u,u' \in \Sigma^*} \bigvee_{a,b \in \Sigma} v = uabu' \wedge w = ubau' \wedge (a,b) \in I$$

Definition 2.2.3 (Trace-Äquivalenz)

Die Trace-Äquivalenz $\sim_I$ auf den Wörtern über einem Alphabet mit Unabhängigkeitsrelation ist der reflexive und transitive Abschluß der Ähnlichkeitsrelation, d. h.

$$v \sim_I w \Longleftrightarrow_{Df} \bigvee_{n \in \mathbb{N}_0} \bigvee_{u_0,\ldots,u_n \in \Sigma^*} u_0 = v \wedge u_n = w \wedge \bigwedge_{i \in \{1,\ldots,n\}} u_{i-1} \cong_I u_i$$

$\sim_I$ ist eine Äquivalenzrelation.

Definition 2.2.4 (Traces, Trace-Sprache)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation.
Dann ist $E(\Sigma, I) = \Sigma^*|_{\sim_I}$ die Menge der Äquivalenzklassen der Relation $\sim_I$ auf Σ^*.
Die Elemente von $E(\Sigma, I)$ heißen Traces über (Σ, I).
Eine Menge $T \subseteq E(\Sigma, I)$ von Traces wird Trace-Sprache genannt.

In Analogie zum Begriff „Trace-Sprache“ werden in der Theorie der Traces gewöhnliche Sprachen über Σ^* zur besseren Unterscheidung häufig als Wortsprachen bezeichnet.

Definition 2.2.5 (Klammernschreibweise)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und $w \in \Sigma^*$.
Dann bezeichnet $[w]_I$ den Trace $t \in E(\Sigma, I)$ mit $w \in t$.
(Der Index I wird weggelassen, wenn die Unabhängigkeitsrelation aus dem Zusammenhang klar ist.)

Der Trace $[w]$ enthält alle Wörter, die durch fortgesetztes Vertauschen von aufeinanderfolgenden unabhängigen Zeichen aus w gewonnen werden können. Alle Wortrepräsentanten eines Traces haben demnach die gleiche Länge und enthalten dieselben Zeichenvorkommen. Es kann definiert werden:

Definition 2.2.6 (Länge u. Alphabet, Spiegeltrace, Vorkommen, Projektion)

Sei $t = [w] \in E(\Sigma, I)$ ein Trace über (Σ, I).

i) Die Länge von t ist $|t| =_{Df} |w|$.

ii) Das Alphabet von t ist $Alph(t) =_{Df} Alph(w)$.

iii) Die Menge der Zeichenvorkommen von t ist $Vor(t) =_{Df} Vor(w)$.

iv) Der Spiegeltrace von t ist $t^R =_{Df} [w^R]$.

v) Sei (Σ', I') ein zweites Alphabet mit Unabhängigkeitsrelation mit $I|_{\Sigma\cap\Sigma'} \subseteq I'|_{\Sigma\cap\Sigma'}$.
Die Projektion von t auf (Σ', I') ist $t|_{(\Sigma',I')} =_{Df} [w|_{\Sigma'}]_{I'}$.
(Wenn aus dem Zusammenhang dem Alphabet Σ' eindeutig eine Unabhängigkeitsrelation I' zugeordnet werden kann, wird als Abkürzung für $t|_{(\Sigma',I')}$ nur $t|_{\Sigma'}$ geschrieben.)

Beispiel:
Sei $\Sigma = \{a, b, c\}$ mit $I = \{(a, c), (b, c), (c, a), (c, b)\}$
und $\Sigma' = \{a, b, d\}$ mit $I' = \{(a, b), (a, d), (b, a), (b, d), (d, a), (d, b)\}$.

Es ist $[acb]_I|_{(\Sigma',I')} = [ab]_{I'}$. Man beachte, daß a und b in $[acb]_I$ nicht vertauscht werden können, wohl aber in $[ab]_{I'} = [ba]_{I'}$.

Bei soviel Übereinstimmung zwischen den Traces und ihren Wortrepräsentanten unterscheiden sich die Strukturen in der Ordnung ihrer Zeichenvorkommen.

Definition 2.2.7 (Ordnung eines Traces)

Sei $t \in E(\Sigma, I)$ ein Trace über (Σ, I).
Für Vorkommen $(a, i), (b, j) \in Vor(t)$ wird die Relation $\leq_t$ definiert:

$$(a, i) \leq_t (b, j) \Longleftrightarrow_{Df} \bigwedge_{w\in t} (a, i) \leq_w (b, j)$$

(D. h. (a, i) kommt in allen Repräsentanten $w \in t$ vor (b, j).)

$Ord(t) =_{Df} (Vor(t), \leq_t)$ ist die Halbordnung des Traces t.

Die Halbordnung eines Traces, dargestellt als azyklischer Graph, eignet sich gut, um die Abfolge der Zeichen im Trace auf einen Blick zu veranschaulichen.

Beispiel: Sei $\Sigma = \{a, b, c, d\}$ mit $I = \{(a, b), (b, a), (c, d), (d, c)\}$. Sei $t = [abcda]_I$.
Eine Veranschaulichung von $Ord(t)$ ist in Abb. 2.2 dargestellt. Kanten, die sich aus der Reflexivität oder Transitivität von $\leq_t$ ergeben, wurden weggelassen.

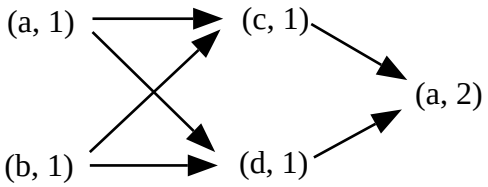

Abbildung 2.2: Abfolge der Vorkommen in $t = [abcda]_I$

Die Darstellung von Traces durch Halbordnungen ist sehr eng verwandt mit den Abhängigkeitsgraphen in [Maz87], [AR88] und [EHR94]. In Abhängigkeitsgraphen werden nur abhängige Zeichenvorkommen durch Kanten verbunden.

Die Verkettung von Wörtern verträgt sich mit der Trace-Äquivalenz, d. h. es gilt

$$w \sim_I w' \wedge v \sim_I v' \Rightarrow wv \sim_I w'v'$$

Somit ist die folgende Definition korrekt:

Definition 2.2.8 (Verkettung von Traces)

Seien $[w], [v] \in E(\Sigma, I)$ *Traces über* (Σ, I).
Die Verkettung von $[w]$ *und* $[v]$ *ist*

$$[w] \circ [v] =_{Df} [wv].$$

Wie bei Wörtern wird auch hier der Operator $\circ$ *meist weggelassen.*

$E(\Sigma, I)$ bildet zusammen mit der Verkettung und dem leeren Trace $[\varepsilon]$ als neutralem Element ein freies partiell kommutatives Monoid. Dieses Monoid ist identisch mit dem Monoid über der Basis Σ und der Eigenschaft $a \circ b = b \circ a \iff (a, b) \in I$ für alle $a, b \in \Sigma$. Das Monoid der Traces ist isomorph zum Monoid der Wörter über dem gleichen Alphabet, wenn $I = \emptyset$. In diesem Fall hat jeder Trace genau einen Wortrepräsentanten.

Mit der Verkettung läßt sich der Begriff des Anfangsstücks auf Traces übertragen.

Definition 2.2.9 (Anfangsstück eines Traces)

Sei $t \in E(\Sigma, I)$ *ein Trace über* (Σ, I).
Die Menge der Anfangsstücke von t *ist*

$$Pref(t) =_{Df} \{t_\alpha \in E(\Sigma, I) | \bigvee_{t_\omega \in E(\Sigma, I)} t_\alpha t_\omega = t\}$$

Offensichtlich gilt $[\varepsilon], t \in Pref(t)$ und $v \in Pref(w) \Rightarrow [v] \in Pref([w])$.

Die Operation $Pref$ definiert eine Halbordnung auf der Menge aller Traces über einem Alphabet mit Unabhängigkeitsrelation.

Definition 2.2.10 (Halbordnung auf $E(\Sigma, I)$)

Für $t_1, t_2 \in E(\Sigma, I)$ wird definiert:

$$t_1 \sqsubseteq t_2 \Longleftrightarrow_{Df} t_1 \in Pref(t_2)$$

Eine wichtige Eigenschaft der Verkettung wird durch das Levi-Lemma (vgl. [Maz87]) beschrieben. Es macht eine Aussage über die Zerlegung eines Traces:

Lemma 2.2.11 (Levi-Lemma für Traces)

Seien $r_1, r_2, t_1, t_2 \in E(\Sigma, I)$ Traces über (Σ, I) mit $r_1 t_1 = r_2 t_2$.
Dann gibt es Traces $r_0, r', t', t_0 \in E(\Sigma, I)$ mit folgenden Eigenschaften:

i) $r_1 = r_0 r', \quad t_1 = t' t_0$

ii) $r_2 = r_0 t', \quad t_2 = r' t_0$

iii) $Alph(r') \times Alph(t') \subseteq I$

Das Levi-Lemma wird durch Abbildung 2.3 veranschaulicht. r_0 ist das längste gemeinsame Anfangsstück von r_1 und r_2, ebenso ist t_0 das längste gemeinsame Schlußstück von t_1 und t_2. r' und t' sind nebenläufig.

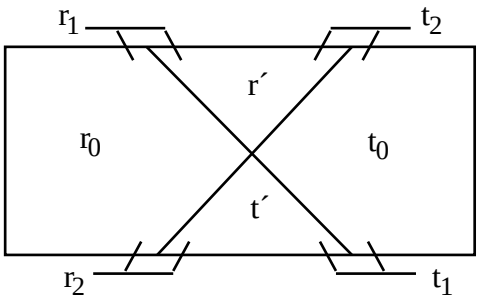

Abbildung 2.3: Zerlegung eines Traces nach dem Levi-Lemma

Definition 2.2.12 (Trace-Sprache zur Wortsprache L)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und $L \subseteq \Sigma^*$ eine Wortsprache. Dann ist

$$[L]_I =_{Df} \{[w] \in E(\Sigma, I) | w \in L\}.$$

die Trace-Sprache zur Wortsprache L.
Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache. Die T zugrundeliegende Wortsprache ist

$$Lin(T) =_{Df} \{w \in \Sigma^* | [w] \in T\}.$$

$[L]$ ist die Menge der Traces, die mindestens einen Repräsentanten in L haben. Von $w \in L$ läßt sich somit nicht auf $[w] \subseteq L$ schließen. In der Regel gilt also nicht $Lin([L]) = L$.

Die einer Trace-Sprache T zugrundeliegende Wortsprache $Lin(T)$ enthält die vollen Äquivalenzklassen der Traces aus T als Teilmengen. Es gilt stets $[Lin(T)] = T$.

2.3 Operationen auf Trace-Sprachen

Projektion, Verkettung und Präfixabschluß lassen sich einfach auf Trace-Sprachen übertragen. Neu definiert werden müssen die Trace-Iteration und die Synchronisation.

Definition 2.3.1 (Projektion einer Trace-Sprache)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I) und (Σ', I') ein zweites Alphabet mit Unabhängigkeitsrelation mit $I|_{\Sigma\cap\Sigma'} \subseteq I'|_{\Sigma\cap\Sigma'}$.
Die Projektion von T auf (Σ', I') ist

$$T|_{(\Sigma',I')} =_{Df} \{t' \in E(\Sigma', I') | \bigvee_{t\in T} t|_{(\Sigma',I')} = t'\}.$$

Für eine Wortsprache $L \subseteq \Sigma^*$ und ein zweites Alphabet mit Unabhängigkeitsrelation und der angegebenen Eigenschaft gilt $[L]_I|_{(\Sigma',I')} = [L|_{\Sigma'}]_{I'}$.

Definition 2.3.2 (Verkettung von Trace-Sprachen)

Seien $T_1, T_2 \subseteq E(\Sigma, I)$ Trace-Sprachen über (Σ, I).
Die Verkettung von T_1 und T_2 wird definiert als

$$T_1 \circ T_2 =_{Df} \{t_1 t_2 \in E(\Sigma, I) | t_1 \in T_1 \wedge t_2 \in T_2\}$$

(Auch bei Verkettung von Trace-Sprachen wird $\circ$ gewöhnlich weggelassen.)

Die Verkettung von Trace-Sprachen hätte ebenso über die Verkettung von Wortsprachen definiert werden können. Es gilt $[L_1][L_2] = [L_1 L_2]$ für alle Wortsprachen $L_1, L_2 \subseteq \Sigma^*$.

Definition 2.3.3 (Trace-Spiegelsprache)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I).
Die Trace-Spiegelsprache zu T ist

$$T^R =_{Df} \{t \in E(\Sigma, I) | t^R \in T\}.$$

Definition 2.3.4 (Iteration von Trace-Sprachen)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I).
Analog zu Wortsprachen wird die Iteration von T definiert als

$$T^* =_{Df} \bigcup_{n\in\mathbb{N}_0} T^n$$

mit $T^0 =_{Df} \{[\varepsilon]\}$ und $T^{n+1} =_{Df} T^n T$ für $n \in \mathbb{N}_0$.

Der Zusammenhang zwischen der Iteration auf Trace-Sprachen und der Iteration auf Wortsprachen ist genauso eng wie im Falle der Verkettung. Es gilt $[L]^* = [L^*]$ für alle Wortsprachen $L \subseteq \Sigma^*$.

Definition 2.3.5 (Präfixabschluß von Trace-Sprachen)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I).
Die Trace-Sprache

$$\begin{aligned} Pref(T) &=_{Df} \{t \in E(\Sigma, I) | \bigvee_{t' \in E(\Sigma,I)} tt' \in T\} \\ &= \bigcup_{t \in T} Pref(t) \end{aligned}$$

ist der Präfixabschluß von T.

Die Synchronisation ist die erste der hier betrachteten Operationen auf Trace-Sprachen, die nicht die Entsprechung einer gebräuchlichen Operation auf Wortsprachen ist, abgesehen von Synchronisationsoperatoren in Kalkülen wie CSP. Bei der Synchronisation zweier Abläufe werden solche Ereignisse identifiziert, die in beiden Abläufen vorkommen, während die nur im ersten bzw. nur im zweiten Ablauf vorkommenden Ereignisse als voneinander unabhängig angesehen werden. In der Theorie der Traces wird die Synchronisation daher als Umkehrung der Projektion auf Teilalphabete definiert. Das setzt voraus, daß die Teilalphabete (mit Unabhängigkeitsrelation) bekannt sind. Für die Synchronisation wird folglich eine Trace-Sprache zusammen mit ihrem Alphabet mit Unabhängigkeitsrelation betrachtet. Ein solches Paar wird als Trace-System bezeichnet.

Definition 2.3.6 (Trace-System)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über diesem Alphabet.
Dann ist $((\Sigma, I), T)$ das Trace-System bestehend aus (Σ, I) und T.

Gegenüber einfachen Trace-Sprachen läßt sich innerhalb von Trace-Systemen nicht nur von Vorkommen eines Zeichens in einem Trace sprechen, sondern auch von seinem Nichtvorkommen.

Mit dem Begriff des Trace-Systems kann jetzt die Synchronisation definiert werden.

Definition 2.3.7 (Synchronisation)

Seien $((\Sigma_1, I_1), T_1)$ und $((\Sigma_2, I_2), T_2)$ zwei Trace-Systeme. Dann ist ihre Synchronisation das Trace-System

$$((\Sigma, I), T) =_{Df} ((\Sigma_1, I_1), T_1) \| ((\Sigma_2, I_2), T_2)$$

mit $\Sigma =_{Df} \Sigma_1 \cup \Sigma_2$, $\Sigma^2 \setminus I = \Sigma_1^2 \setminus I_1 \cup \Sigma_2^2 \setminus I_2$ und

$$T =_{Df} \{t \in E(\Sigma, I) | t|_{(\Sigma_1, I_1)} \in T_1 \wedge t|_{(\Sigma_2, I_2)} \in T_2\}$$

Beispiel:
$\Sigma_1 = \{a, b, c, d\}$ mit $I_1 = \{(a, b), (a, c), (b, a), (b, c), (c, a), (c, b)\}$, $T_1 = \{[ab], [abc]\}$,
$\Sigma_2 = \{a, b, d, e\}$ mit $I_2 = \{(a, e), (b, e), (e, a), (e, b)\}$, $T_2 = \{[dab], [eba]\}$
Man beachte, daß der Trace $[dab] \in T_2$ nicht mit den Traces aus T_1 synchronisiert werden kann, weil das Zeichen d in keinem Trace aus T_1 vorkommt, obwohl $d \in \Sigma_1$ ist. Die Zeichen a und b sind in (Σ_1, I_1) unabhängig, in (Σ_2, I_2) abhängig. Sie sind somit im Synchronisationsalphabet abhängig, und Vorkommen dieser Zeichen in Traces der Synchronisationssprache werden so geordnet wie in den Traces aus T_2. Die Zeichen c und e sind im Synchronisationsalphabet unabhängig, weil sie aus Σ_1 bzw. Σ_2 stammen.
Die Synchronisation ist also $((\Sigma, I), T) = ((\Sigma_1, I_1), T_1) \| ((\Sigma_2, I_2), T_2)$ mit
$\Sigma = \{a, b, c, d, e\}$, $I = \{(a, c), (a, e), (b, c), (b, e), (c, a), (c, b), (c, e), (e, a), (e, b), (e, c)\}$ und $T = \{[eba], [ebac]\}$.

Obwohl Trace-Systeme für die korrekte Definition der Synchronisation eingeführt wurden, wird im folgenden nur noch von der Synchronisation von Trace-Sprachen gesprochen. Diese Ungenauigkeit soll damit entschuldigt werden, daß diese Vereinfachung zur besseren Lesbarkeit der Formeln beiträgt und die fraglichen Trace-Sprachen zusammen mit den Alphabeten, auf denen sie definiert sind und die in der Regel explizit angegeben sind, Trace-Systeme bilden. In Zweifelsfällen muß auf den Begriff des Trace-Systems zurückgegriffen werden. Man beachte, daß durch die Synchronisation zweier Trace-Sprachen ein neues Alphabet mit Abhängigkeitsrelation berechnet wird.

Korollar 2.3.8 (Synchronisation und Wortsprache)

Seien $T_1 \subseteq E(\Sigma_1, I_1)$ und $T_2 \subseteq E(\Sigma_2, I_2)$ zwei Trace-Sprachen. Für die Wortsprachen $Lin(T_1)$, $Lin(T_2)$ und $Lin(T_1 \| T_2)$ gilt:

$$Lin(T_1 \| T_2) = \{w \in (\Sigma_1 \cup \Sigma_2)^* | \; w|_{\Sigma_1} \in Lin(T_1) \wedge w|_{\Sigma_2} \in Lin(T_2)\}$$

In Termen bindet der Operator $\|$ schwächer als Verkettung und Iteration und stärker als mengentheoretische Operationen.

Korollar 2.3.9 (Rechenregeln für die Synchronisation)

Für Trace-Sprachen T_1, T_2 und T_3 über beliebigen Alphabeten mit Unabhängigkeitsrelation gelten die folgenden Gleichungen:

$T_1 \| T_1 = T_1$

$T_1 \| T_2 = T_2 \| T_1$

$T_1 \| (T_2 \| T_3) = (T_1 \| T_2) \| T_3$

$(T_1 \cup T_2) \| T_3 = T_1 \| T_3 \cup T_2 \| T_3$

$T_1 \| \emptyset = \emptyset$

$$\{[\varepsilon]\}\|\{[\varepsilon]\} = \{[\varepsilon]\}$$

Die Synchronisation von zwei Traces (einelementigen Sprachen T_1, T_2) ergibt wieder einen Trace ($|T_1\|T_2| = 1$) oder die leere Trace-Sprache. Im ersten Fall heißen die Traces *synchronisierbar*. Zwei Traces sind genau dann synchronisierbar, wenn ihre Projektionen auf das Schnittalphabet denselben Trace ergibt.

Der folgende Zusammenhang zwischen Verkettung und Synchronisation findet sich für eine Verallgemeinerung der Traces auf hierarchische Prozesse in [Jans94]. Als Beschreibung von Systemverhalten werden dort Mengen von hierarchischen Halbordnungen betrachtet. Die durch die Halbordnungen angeordneten Aktionen sind atomar oder ihrerseits wieder Halbordnungen. Die definierte Verkettungsoperation entspricht der Verkettung auf Traces. Die Synchronisation ist bei [Jans94] etwas anders definiert als hier.

Die Rechenregel wird hier als eine Eigenschaft der Trace-Synchronisation formuliert und bewiesen. Sie zeigt, unter welchen Bedingungen von einer sequentiellen zu einer nebenläufigen Darstellung einer Trace-Sprache übergegangen werden kann und umgekehrt.

Lemma 2.3.10 (Zusammenhang Verkettung – Synchronisation)

Seien $P, R \subseteq E(\Sigma_1, I_1)$ und $Q, S \subseteq E(\Sigma_2, I_2)$ vier Trace-Sprachen mit $Alph(P) \cap Alph(S) = Alph(Q) \cap Alph(R) = \emptyset$. Es gilt:

$$(P\|Q)(R\|S) = PR\|QS$$

Beweis:
$\subseteq$: Sei $t \in (P\|Q)(R\|S)$. Dann ist $t = t_{PQ}t_{RS}$ mit $t_{PQ} \in P\|Q$ und $t_{RS} \in R\|S$. Mit der Definition der Synchronisation ergibt sich $t_{PQ}|_{(\Sigma_1,I_1)} \in P$, $t_{PQ}|_{(\Sigma_2,I_2)} \in Q$, $t_{RS}|_{(\Sigma_1,I_1)} \in R$ und $t_{RS}|_{(\Sigma_2,I_2)} \in S$. Weil die Projektion sich mit der Verkettung verträgt, ist

$$t|_{(\Sigma_1,I_1)} = (t_{PQ}t_{RS})|_{(\Sigma_1,I_1)} = t_{PQ}|_{(\Sigma_1,I_1)}t_{RS}|_{(\Sigma_1,I_1)} \in PR.$$

Ebenso ist $t|_{(\Sigma_2,I_2)} \in QS$. Wiederum die Synchronisation angewendet ergibt $t \in PR\|QS$.

$\supseteq$: Sei $t \in PR\|QS$. Dann ist $t|_{(\Sigma_1,I_1)} \in PR$ und $t|_{(\Sigma_2,I_2)} \in QS$. Sei nun $t|_{(\Sigma_1,I_1)} = t_Pt_R$ mit $t_P \in P, t_R \in R$ und $t|_{(\Sigma_2,I_2)} = t_Qt_S$ mit $t_Q \in Q, t_S \in S$. Folglich haben wir $t_Pt_R\|t_Qt_S = t$.
Das Schnittalphabet von (Σ_1, I_1) und (Σ_2, I_2) sei $(\Sigma', I') =_{Df} (\Sigma_1 \cap \Sigma_2, I_1|_{\Sigma_2} \cup I_2|_{\Sigma_1})$. Wegen $\Sigma' \subseteq \Sigma_1 \subseteq \Sigma_1 \cup \Sigma_2$ und $\Sigma' \subseteq \Sigma_2 \subseteq \Sigma_1 \cup \Sigma_2$ ist $t|_{(\Sigma',I')} = (t_Pt_R)|_{(\Sigma',I')} = (t_Qt_S)|_{(\Sigma',I')}$. Somit ist $t_P|_{(\Sigma',I')}t_R|_{(\Sigma',I')} = t_Q|_{(\Sigma',I')}t_S|_{(\Sigma',I')}$.
Mit $Alph(P) \cap Alph(S) = Alph(Q) \cap Alph(R) = \emptyset$ gilt diese Beziehung insbesondere für Projektionen der vier Traces t_P, t_Q, t_R und t_S auf (Σ', I'). Wir bekommen also $t_P|_{(\Sigma',I')} = t_Q|_{(\Sigma',I')}$ und $t_R|_{(\Sigma',I')} = t_S|_{(\Sigma',I')}$. Demnach sind t_P und t_Q bzw. t_R und t_S synchronisierbar, d. h. $t_P\|t_Q$ und $t_R\|t_S$ existieren. Sei t' der (einzige) Trace aus der Trace-Sprache $(\{t_P\}\|\{t_Q\})(\{t_R\}\|\{t_S\})$. Nach dem ersten Teil des Beweises ist $t' \in \{t_P\}\{t_R\}\|\{t_Q\}\{t_S\} = \{t\}$, also $t' = t$.
Wegen $(\{t_P\}\|\{t_Q\})(\{t_R\}\|\{t_S\}) \subseteq (P\|Q)(R\|S)$ ist schließlich $t \in (P\|Q)(R\|S)$. □

Korollar 2.3.11 (Disjunkte Synchronisation)

Seien $T_1 \subseteq E(\Sigma_1, I_1), \ldots, T_k \subseteq E(\Sigma_k, I_k)$ *Trace-Sprachen mit* $\Sigma_i \cap \Sigma_j = \emptyset$ *für alle* $i, j \in \{1, \ldots, k\}, i \neq j$. *Es gilt:*

$$T_1 \| \ldots \| T_k = T_1 \ldots T_k$$

Die Verkettung der Trace-Sprachen auf der rechten Seite der Gleichung ist nur dann formal korrekt, wenn die Trace-Sprachen $T_1, \ldots, T_k$ *als über dem Synchronisationsalphabet definiert gesehen werden.*

Die Trace-Iteration ist eine auf Trace-Sprachen definierte Verallgemeinerung der Iteration auf Wortsprachen. Für Alphabete mit Unabhängigkeitsrelation $I = \emptyset$ stimmt die Trace-Iteration mit der gewöhnlichen Iteration überein. Die Motivation für die Einführung der Trace-Iteration ergibt sich aus folgenden Überlegungen:

Gegeben sind zwei isolierte Teilsysteme, deren Verhalten die Trace-Sprachen T_1 und T_2 sind. Beide Systeme zusammen betrachtet haben das Verhalten T_1T_2 $(= T_2T_1)$, weil die Aktionenfolgen in den Teilsystemen unabhängig voneinander ablaufen können. Die Iteration dieser Trace-Sprache ist $(T_1T_2)^*$, das Verhalten des Gesamtsystems wird in der Iteration also beliebig oft wiederholt. Ein Trace aus $(T_1T_2)^*$ liegt insbesondere in einem $(T_1T_2)^n = T_1^nT_2^n, n \in \mathbb{N}_0$. Das bedeutet aber, daß die Teilsysteme im Widerspruch zu ihrer Isoliertheit zusammen iteriert werden, etwa so, als wären sie durch einen zentralen Taktgeber gesteuert.

Im einfachsten Fall ist $\Sigma = \{a, b\}$, $I = \{(a, b), (b, a)\}, T_1 = \{[a]\}$ und $T_2 = \{[b]\}$. Durch gewöhnliche Iteration bekommen wir mit $(T_1T_2)^*$ eine Trace-Sprache, die von einem endlichen Automaten nicht erkannt werden kann. Wortrepräsentanten von Traces aus $(T_1T_2)^*$ enthalten gleich viele Zeichen a wie b. Ein diese Wortsprache erkennender Automat muß in der Lage sein, erst beliebig viele Zeichen a zu zählen, diese Zahl zu speichern und mit der Anzahl der Zeichen b im eingegebenen Wort zu vergleichen. Das ist nur mit unendlich vielen Zuständen möglich.

Die Trace-Iteration muß so definiert werden, daß das Verhalten zunächst in seine unabhängigen Bestandteile zerlegt wird und diese Teilsprachen dann getrennt iteriert werden.

Definition 2.3.12 (Zerlegung eines Traces)

Sei (Σ, I) *ein Alphabet mit Unabhängigkeitsrelation.*

i) Seien $t_1, t_2 \in E(\Sigma, I)$ *zwei Traces über diesem Alphabet.*

$$t_1 \text{ und } t_2 \text{ heißen unabhängig} \iff_{Df} \mathit{Alph}(t_1) \times \mathit{Alph}(t_2) \subseteq I$$

ii) Ein Trace $t \in E(\Sigma, I)$ *heißt zusammenhängend*

$$\iff_{Df} \bigwedge_{t_1, t_2 \in E(\Sigma, I)} t_1 \neq [\varepsilon] \wedge t_2 \neq [\varepsilon] \wedge t = t_1t_2 \Rightarrow t_1 \text{ und } t_2 \text{ sind nicht unabhängig}$$

iii) Die Zerlegung eines Traces $t \in E(\Sigma, I)$ ist die Menge der Traces $\Delta(t) =_{Df} \{t_0, \ldots, t_n\} \subseteq E(\Sigma, I), n \in \mathbb{N}_0$, mit den Eigenschaften

(a) $t = t_0 \ldots t_n$

(b) t_i *ist nicht leer und zusammenhängend für jedes* i, $0 \leq i \leq n$

(c) t_i *und* t_j *sind unabhängig für alle* $i, j \quad 0 \leq i, j \leq n$

$\Delta(t)$ *ist durch (a) – (c) eindeutig bestimmt.*

Der Begriff des Zusammenhangs eines Traces ergibt sich aus der Darstellung als Halbordnung. Ein Trace t ist genau dann zusammenhängend, wenn der Graph seiner Halbordnung $Ord(t)$ schwach zusammenhängend ist.

Definition 2.3.13 (Trace-Iteration, [Och85])

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache.

i) Die Zerlegung von T ist die Vereinigung der Zerlegungen der Traces aus T:

$$\Delta(T) =_{Df} \bigcup_{t \in T} \Delta(t)$$

ii) Die Trace-Iteration von T (geschrieben $T^{\#}$) ist die Iteration der Zerlegung von T:

$$T^{\#} =_{Df} (\Delta(T))^*$$

Beispiel: $\Sigma = \{a, b, c, d\}$
$I = \{(a, b), (a, d), (b, a), (b, c), (c, b), (c, d), (d, a), (d, c)\}$
$T = \{[abcd]\}$
Dann ist $\Delta(T) = \{[ac], [bd]\}$
und $T^{\#} = (\Delta(T))^* = \{[ac], [bd]\}^* = [\{ac, bd\}^*]$

Für Alphabete mit leerer Unabhängigkeitsrelation sind alle Wörter zusammenhängend, so daß für alle Trace-Sprachen $T \subseteq E(\Sigma, \emptyset)$ gilt: $\Delta(T) = T$ und $T^{\#} = T^*$.

2.4 Rationale und erkennbare Trace-Sprachen

Für die Charakterisierung der regulären Wortsprachen über einem Alphabet sind mehrere äquivalente Konzepte bekannt: der endliche Automat, die syntaktische Kongruenz mit endlichem Index, die Typ-3-Grammatik und die Definition über reguläre Ausdrücke. Alle Konzepte lassen sich aufgrund der Verträglichkeit von Trace-Äquivalenz und Verkettungsoperation leicht auf Trace-Sprachen übertragen. Es stellt sich heraus, daß auf Trace-Sprachen nicht alle der genannten Charakterisierungen äquivalent sind.

Reguläre Ausdrücke definieren über einem Alphabet mit nichtleerer Unabhängigkeitsrelation eine größere Sprachklasse als die Klasse der von endlichen Automaten erkennbaren Trace-Sprachen. Die Bezeichnung „reguläre Trace-Sprache“ ist somit verwirrend, wenn nicht hinzugefügt wird, welcher Regularitätsbegriff zugrundegelegt ist. Es sollen deswegen in der Bezeichnung die durch rationale (=reguläre) Ausdrücke definierbaren Trace-Sprachen und die von endlichen Automaten erkennbaren Trace-Sprachen unterschieden werden.

Definition 2.4.1 (Rationale Trace-Sprachen)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation.
Die Klasse der rationalen Trace-Sprachen über (Σ, I) ist die kleinste Klasse von Sprachen über (Σ, I), die

i) alle endlichen Trace-Sprachen über (Σ, I) enthält und

ii) bezüglich Vereinigung der Sprachen, Verkettung und Iteration (Def. 2.3.4) abgeschlossen ist.

Wie oben schon erwähnt wurde, ist die gebräuchlichste Darstellung von Sprachen aus einer so definierten Sprachklasse der reguläre Ausdruck. Es soll hier die Definition der regulären Ausdrücke wiederholt werden.

Definition 2.4.2 (Reguläre Ausdrücke)

*Sei Σ ein Alphabet und seien o. B. d. A. $\cup$, *, ε, $\emptyset$, (,) neue Zeichen.*
Die Menge der regulären Ausdrücke über Σ wird induktiv definiert:

i) Jedes Zeichen aus $\Sigma \cup \{\varepsilon, \emptyset\}$ ist ein regulärer Ausdruck über Σ.

ii) Seien $r_1, r_2 \in (\Sigma \cup \{\cup, *, \varepsilon, \emptyset, (,)\})^*$ zwei reguläre Ausdrücke über Σ.
Dann sind auch $(r_1 r_2)$, $(r_1 \cup r_2)$ und r_1^* reguläre Ausdrücke über Σ.

Jeder reguläre Ausdruck bezeichnet eine Sprache über Σ. Die Interpretierbarkeit setzt voraus, daß der Aufbau eines regulären Ausdrucks eindeutig ist.

Definition 2.4.3 (Interpretation eines regulären Ausdrucks)

Die Wortsprache $L(r)$ eines regulären Ausdrucks $r \in (\Sigma \cup \{\cup, *, \varepsilon, \emptyset, (,)\})^*$ wird wieder induktiv definiert:

i) Für $|r| = 1$ ist

$L(a) =_{Df} \{a\}$ *für* $a \in \Sigma$
$L(\varepsilon) =_{Df} \{\varepsilon\}$
$L(\emptyset) =_{Df} \emptyset$

$$ii)\ L(r) =_{Df} \begin{cases} L(r_1)L(r_2) & falls \quad r = (r_1 r_2) \\ L(r_1) \cup L(r_2) & falls \quad r = (r_1 \cup r_2) \\ (L(r_1))^* & falls \quad r = r_1^* \end{cases}$$

Die Klasse der regulären Wortsprachen über Σ ist die Menge der Sprachen, die durch reguläre Ausdrücke über Σ dargestellt werden können. Eine genaue Charakterisierung der regulären Wortsprachen kann analog zu Definition 2.4.1 erfolgen. Wegen $[L_1 L_2] = [L_1][L_2]$, $[L_1 \cup L_2] = [L_1] \cup [L_2]$ und $[L^*] = [L]^*$ gilt:

Korollar 2.4.4 (Darstellung rationaler Trace-Sprachen)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I).

$$\begin{aligned} & T \text{ ist rational} \\ \iff\ & \text{Es gibt eine reguläre Wortsprache } L \text{ über } \Sigma \text{ mit } [L] = T \\ \iff\ & \text{Es gibt einen regulären Ausdruck } r \text{ über } \Sigma \text{ mit } [L(r)] = T \end{aligned}$$

Zur Notation der regulären Ausdrücke: Die in der Interpretation der Ausdrücke benutzten Operationen Verkettung und Mengenvereinigung sind assoziativ. Eine Klammerung ist also im Sinne einer eindeutigen Interpretierbarkeit innerhalb von mehrgliedrigen Verkettungen und Mengenvereinigungen nicht notwendig.

Um zur besseren Lesbarkeit der Ausdrücke weitere Klammenpaare zu sparen, wird festgelegt: Die Iteration bindet als einstellige Operation am stärksten, gefolgt von der Verkettung, gefolgt von der Mengenvereinigung. Das äußere Klammerpaar eines Ausdrucks wird weggelassen, wenn es nicht von einem Stern gefolgt wird.

Wenn aus dem Zusammenhang ersichtlich ist, daß nicht ein regulärer Ausdruck selber, sondern die ihm zugeordnete Sprache gemeint ist, wird statt $L(r)$ oft nur r geschrieben. Insbesondere ist für einen regulären Ausdruck r die Schreibweise $[r]$ die Abkürzung für den Ausdruck $[L(r)]$, der eine Trace-Sprache bezeichnet.

Die syntaktische Kongruenz ist verglichen mit den regulären Ausdrücken ein abstrakteres Konzept zur Definition von Regularität.

Definition 2.4.5 (Syntaktische Kongruenz)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I).
Zwei Traces $t, t' \in E(\Sigma, I)$ heißen syntaktisch kongruent bezüglich T ($t \sim_T t'$), wenn gilt:

$$\bigwedge_{t_\alpha, t_\omega \in E(\Sigma, I)} t_\alpha t t_\omega \in T \iff t_\alpha t' t_\omega \in T$$

Es ist leicht zu erkennen, daß $\sim_T$ tatsächlich die Eigenschaften einer Kongruenzrelation hat.

Für Wörter wird die syntaktische Kongruenz bezüglich einer Wortsprache vollkommen analog definiert. Weil die Verkettungsoperation sich mit der Trace-Äquivalenz verträgt, gilt

Korollar 2.4.6 (Isomorphie der Kongruenzrelationen)

Sei $L = Lin(T)$ die T zugrundeliegende Wortsprache und $\sim_L$ die syntaktische Kongruenz bezüglich L auf Wörtern. Für beliebige Wörter v und w gilt:

$$v \sim_L w \iff [v] \sim_T [w]$$

Somit sind die syntaktischen Kongruenzrelationen bezüglich T und $Lin(T)$ isomorph. Diese Beobachtung führt zu einem wichtigen Ergebnis im Anschluß an die folgende Definition.

Definition 2.4.7 (Erkennbarkeit einer Trace-Sprache)

Eine Trace-Sprache über (Σ, I) heißt erkennbar, wenn ihre syntaktische Kongruenz endlich viele Kongruenzklassen definiert.
$Erk(\Sigma, I)$ ist die Klasse der erkennbaren Trace-Sprachen über (Σ, I).

Aus der Automatentheorie ist bekannt, daß die regulären Wortsprachen über einem Alphabet genau die Sprachen sind, deren syntaktische Kongruenz endlich viele Klassen definiert. Daher:

Korollar 2.4.8 (Erkennbarkeit und Regularität)

Eine Trace-Sprache $T \subseteq E(\Sigma, I)$ ist genau dann erkennbar, wenn die ihr zugrundeliegende Wortsprache $Lin(T)$ regulär ist.

Dies erklärt die Bezeichnung „erkennbar“ für diese Klasse von Trace-Sprachen. Reguläre Wortsprachen können von endlichen Automaten erkannt werden. Also heißt „T ist erkennbar“ nichts anderes, als daß $Lin(T)$ im herkömmlichen Sinne von einem endlichen Automaten erkannt werden kann.

Bem.: Es gilt T erkennbar $\Rightarrow$ T rational, aber nicht umgekehrt.

2.5 Petrinetze

Der Begriff des Petrinetzes geht auf [Petri62] zurück. Die Theorie der Petrinetze ist umfassend dargestellt in [Rei82] und [Star90].

Petrinetze sind Graphen mit zwei Knotentypen: Es gibt speicherartige Knoten (sog. Stellen), die Marken tragen können, und prozedurartige Knoten (sog. Transitionen), die die Markenzahl der Stellen in ihrer Umgebung verändern. Gerichtete Kanten stellen die Beziehung zwischen Stellen und Transitionen her und laufen immer von Stellen zu Transitionen oder von Transitionen zu Stellen. Die Kanten haben Gewichte, die die Zahl der Marken festlegen, die bei einem Schaltereignis einer Transition über sie fließen. Eine Transition schaltet, indem sie gemäß der Kantengewichte von den vor ihr liegenden Stellen Marken abzieht und auf den hinter ihr liegenden Stellen Marken hinzufügt.

Definition 2.5.1 (Petrinetz)

Ein Petrinetz ist ein Fünftupel $N = (S, T, F, w, m_0)$, wobei

S *die endliche Menge von Stellen*

T *die endliche Menge von Transitionen* $(S \cup T \neq \emptyset = S \cap T)$

$F \subseteq (S \times T) \cup (T \times S)$ *die Flußrelation*

$w : F \longrightarrow \mathbb{N}$ *die Kantengewichtung und*

$m_0 : S \longrightarrow \mathbb{N}_0$ *die Anfangsmarkierung ist.*

Die Anfangsmarkierung gibt an, wieviele Marken zunächst auf den Stellen liegen. Durch Schalten der Transitionen wird die Markierung verändert.

Definition 2.5.2 (Aktiviertheit und Schalten einer Transition)

Sei $N = (S, T, F, w, m_0)$ ein Petrinetz. Eine Transition $t \in T$ ist in einer Markierung $m : S \longrightarrow \mathbb{N}_0$ aktiviert, wenn gilt

$$\bigwedge_{s \in S} (s,t) \in F \Rightarrow m(s) \geq w(s,t)$$

Wenn t in m aktiviert ist, kann t schalten und so die Markierung m in die Folgemarkierung m' mit

$$m'(s) =_{Df} \begin{cases} m(s) & \text{für} \quad (s,t) \notin F \wedge (t,s) \notin F \\ m(s) - w(s,t) & \text{für} \quad (s,t) \in F \wedge (t,s) \notin F \\ m(s) + w(t,s) & \text{für} \quad (s,t) \notin F \wedge (t,s) \in F \\ m(s) - w(s,t) + w(t,s) & \text{für} \quad (s,t) \in F \wedge (t,s) \in F \end{cases}$$

überführen. Schreibweise:

$$m[t > m' \Longleftrightarrow_{Df} \quad t \text{ ist in } m \text{ aktiviert und überführt } m \text{ in } m'$$

Eine Schaltfolge ist eine Folge von nacheinander aktivierten und schaltenden Transitionen.

Definition 2.5.3 (Schaltfolge)

Sei $N = (S, T, F, w, m_0)$ ein Petrinetz. Das Wort $t_1 \ldots t_n \in T^+$ ist eine Schaltfolge und überführt die Markierung m in die Markierung m' (Schreibweise $m[t_1 \ldots t_n > m'$)

$$\begin{aligned} \Longleftrightarrow_{Df} \quad & \text{Es gibt Markierungen } m_1, \ldots, m_n \text{ mit} \\ & m[t_1 > m_1, \quad m_1[t_2 > m_2, \quad \ldots, \quad m_{n-1}[t_n > m_n \text{ und } m_n = m'. \end{aligned}$$

Für die leere Schaltfolge ε und alle Markierungen m gilt stets $m[\varepsilon > m$.

Definition 2.5.4 (Erreichbarkeit, Errreichbarkeitsgraph)

Sei $N = (S, T, F, w, m_0)$ ein Petrinetz und $m, m' : S \longrightarrow \mathbb{N}_0$ zwei Markierungen von N.

i)

$$m' \text{ ist von } m \text{ aus erreichbar} \iff_{Df} \bigvee_{w \in T^*} m[w > m'$$
$$\iff \text{Es gibt eine Schaltfolge, die } m \text{ in } m' \text{ überführt.}$$

ii) $Err_N(m)$ *ist die Menge aller von m aus erreichbaren Markierungen.*

iii) Der Erreichbarkeitsgraph von N ist $G_N = (V, E, l)$ mit

$V =_{Df} Err_N(m_0)$ *Menge der Knoten*
$E \subseteq V \times V \times T$ *mit* $(m, m', t) \in E \iff_{Df} m[t > m'$ *Menge der Kanten*
$l : E \longrightarrow T$ *mit* $l(m, m', t) =_{Df} t$ *Kantenbeschriftung*

Der Erreichbarkeitsgraph ist eine zustandsorientierte globale Darstellung des Netzverhaltens. Anders als im Netz, das letztlich aufgrund der festen Schaltregel sein Verhalten implizit darstellt, sind im Erreichbarkeitsgraphen die Zustände (= erreichbare Markierungen) global repräsentiert.

Eine ablauforientierte Sichtweise des Netzverhaltens ist die Betrachtung der Schaltfolgen eines Netzes. Es gibt mehrere Möglichkeiten, den Begriff der Sprache eines Netzes (Schaltfolgen sind Wörter) zu konkretisieren (vgl. [Jan87]). Naheliegend ist, die Menge aller in m_0 ausführbaren Schaltfolgen als Verhalten des Netzes zu definieren. Diese sog. P-Typ-Sprachen sind stets präfixabgeschlossen. Zweitens kann in Analogie zu Automaten die Menge der Schaltfolgen betrachtet werden, die die Anfangsmarkierung in eine von endlich vielen vorgegebenen Endmarkierungen überführen (sog. L-Typ-Sprachen). Beide Typen sollen hier definiert werden. Weitere Möglichkeiten sind a) alle Schaltfolgen, die m_0 in eine Markierung überführen, die eine der vorgegebenen Endmarkierungen majorisiert (G-Typ-Sprachen), und b) alle Schaltfolgen, die m_0 in eine Markierung überführen, in der keine Transition mehr schalten kann (T-Typ-Sprachen). Für die G- und T-Typ-Sprachen sei auf die angegebene Literatur verwiesen.

In einem ersten Ansatz für die Klasse der P- und L-Typ-Sprachen wird das Alphabet gerade die Menge der Transitionen sein. Einfache Beispiele zeigen jedoch, daß in diesem Fall die Sprachklassen weder bezüglich Vereinigung und Verkettung abgeschlossen sind, noch die Klasse der regulären Sprachen vollständig enthalten. Für die Definition der Sprachklassen wird daher ein Homomorphismus verwendet, der die Schaltfolgen auf Wörter über dem vorgegebenen Alphabet Σ abbildet.

Definition 2.5.5 (P- und L-Typ-Sprachen)

Sei $N = (S, T, F, w, m_0)$ ein Petrinetz, M eine endliche Menge von Markierungen zu N und $h : T \longrightarrow \Sigma$ eine Beschriftung der Transitionen mit Zeichen aus dem Alphabet Σ. Sei der Homomorphismus h^* die Fortsetzung von h auf T^*.

i) Die P-Typ-Sprache von (N,h) ist

$$P(N,h) =_{Df} \{w \in \Sigma^* | \bigvee_{m \in Err_N(m_0)} \bigvee_{v \in T^*} m_0[v > m \wedge h^*(v) = w\}$$

ii) Die L-Typ-Sprache von (N,M,h) ist

$$L(N,M,h) =_{Df} \{w \in \Sigma^* | \bigvee_{m \in M} \bigvee_{v \in T^*} m_0[v > m \wedge h^*(v) = w\}$$

Definition 2.5.6 (Sprachklassen)

$\underline{P_\Sigma}$ und $\underline{L_\Sigma}$ sind die Sprachklassen der P- bzw. L-Typ-Sprachen über dem Alphabet Σ. $\underline{P_\Sigma^f}$ und $\underline{L_\Sigma^f}$ sind die Sprachklassen der P- bzw. L-Typ-Sprachen über dem Alphabet Σ mit bijektiver Abbildung h, also o. B. d. A. zu Netzen mit $T = \Sigma$.

Der Erreichbarkeitsgraph eines Petrinetzes ist in der Regel unendlich groß. Das Problem der Erreichbarkeit einer Markierung ist zwar entscheidbar, jedoch mit sehr hohem Aufwand. Da manche anderen Fragestellungen an das Verhalten eines Netzes sich auf das Erreichbarkeitsproblem reduzieren, ergibt sich die Frage nach Petrinetztypen, die ein einfacheres Verhalten haben und so eine schnellere Analyse erlauben. Eine solche Klasse von Netzen sind die sog. beschränkten Petrinetze, die einen endlichen Erreichbarkeitsgraphen haben. Beschränkte Netze verhalten sich wie endliche Automaten. Ihre Netzsprachen sind regulär. Der Erreichbarkeitsgraph eines beschränkten Netzes kann allerdings wie die Ackermannfunktion in der Anzahl der Stellen des Netzes wachsen. Eine Teilklasse der beschränkten Netze sind die sicheren Netze.

Definition 2.5.7 (Sicheres Petrinetz)

$$\textit{Ein Petrinetz } N = (S,T,F,w,m_0) \textit{ heißt sicher} \Longleftrightarrow_{Df} \bigwedge_{m \in Err_N(m_0)} \bigwedge_{s \in S} m(s) \leq 1$$

Die Stellen der sicheren Petrinetze tragen höchstens eine Marke, verhalten sich also wie Bedingungen, die zwischen den Zuständen „erfüllt“ und „nicht erfüllt“ hin- und herwechseln. Die Gewichtung w der Kanten ist entweder 1 (an aktivierbaren Transitionen) oder beliebig (an nicht aktivierbaren Transitionen) und braucht daher bei sicheren Netzen nicht ausdrücklich definiert zu werden.

Im Gegensatz zu nicht sicheren Netzen ergibt sich bei sicheren Netzen aus der Struktur des Netzes, welche Transitionen unabhängig schalten können.

Beispiel: Wenn die Anfangsmarkierung des Petrinetzes in Abb. 2.4 $m_0(s_1) = m_0(s_4) = 2$ ist, so können die Transitionen t_1 und t_2 in beliebiger Reihenfolge oder gleichzeitig schalten. Schaltet eine der beiden Transitionen, ergibt sich ein Konflikt zwischen t_1 und t_2 um die zweite Marke auf s_1, weil beide Transitionen dann aktiviert sind, aber nur eine Transition

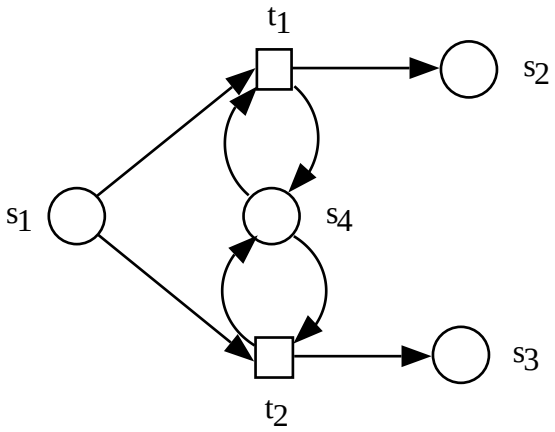

Abbildung 2.4: Zwei abhängig schaltende Transitionen

mit der Marke schalten kann. Liegt auf s_4 in der Anfangsmarkierung nur eine Marke, sind t_1 und t_2 voneinander abhängig, da sie nicht gleichzeitig auf diese Marke zugreifen können.

Ist dagegen die Struktur ein Petrinetz mit $m_0(s_1) \leq 1, m_0(s_4) \leq 1$ und $m_0(s_2) = m_0(s_3) = 0$, so ist es sicher, und die Transitionen t_1 und t_2 sind in jedem Fall voneinander abhängig.

Diekert verallgemeinert in [Diek94] den Trace-Begriff zur Beschreibung von Abläufen in nicht sicheren Petrinetzen. Um die dynamischen Abhängigkeitsbeziehungen zwischen den Transitionen erfassen zu können, wird in einem ersten Ansatz eine asymmetrische Unabhängigkeitsrelation auf den Zeichen zugelassen. Es ergeben sich *Semi-Traces*. Für eine Beschreibung des Netzverhaltens werden *Semiwörter* definiert, das sind spezielle Äquivalenzklassen von Semi-Traces.

Einen anderen Weg beschreiten Hoogers, Kleijn und Thiagarajan in [HKT92]: Sie definieren die Unabhängigkeitsrelation nicht auf Zeichen, sondern auf Schaltfolgen.

Das Verhalten von sicheren Netzen läßt sich aufgrund dieser Überlegungen einfacher beschreiben. Zwei Transitionen sind abhängig, wenn sie über eine Stelle verbunden sind. Andernfalls können sie unabhängig voneinander schalten.

Mazurkiewicz nutzt in [Maz87] diesen Zusammenhang zwischen Netzstruktur und Netzverhalten aus und kommt so zu einer Darstellung des Netzverhaltens durch eine Trace-Sprache. Als Netztyp verwendet Mazurkiewicz sog. Elementary Net Systems (EN-Systeme), die sich nur durch die Schaltregel von sicheren Petrinetzen unterscheiden. Die Definition der Aktiviertheit einer Transitionen ist so modifiziert, daß Transitionen in einer Markierung nicht aktiviert sind, wenn sie durch Schalten eine Stelle doppelt markieren würden. Sichere Petrinetze sind EN-Systeme, und jedes EN-System kann durch Hinzufügen von Komplementstellen zu einem sicheren Petrinetz mit demselben Verhalten ergänzt werden (vgl. [Rei82]).

Beispiel: Im in Abbildung 2.5 dargestellten EN-System müssen t_1 und t_2 abwechselnd schalten. Die Schaltregel für EN-Systeme verhindert, daß t_1 zweimal hintereinander schaltet. Durch Einfügen einer Komplementstelle s' erhält man ein Petrinetz, in dem die gewöhnliche Schaltregel ausreicht, um $m(s) = 1$ sicherzustellen (Abb. 2.6).

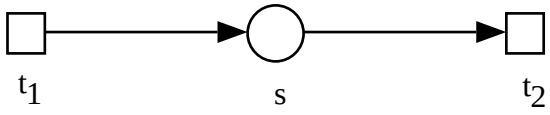

Abbildung 2.5: t_1 und t_2 schalten im EN-System abwechselnd

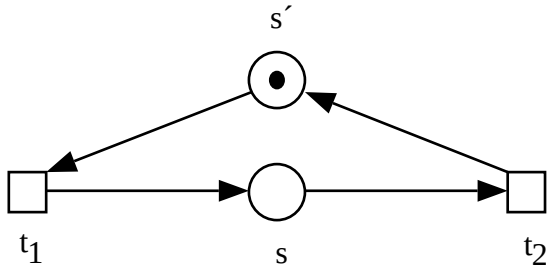

Abbildung 2.6: t_1 und t_2 schalten abwechselnd

Für die Beschreibung des Netzverhaltens durch eine Trace-Sprache wird zunächst das Alphabet und die Unabhängigkeitsrelation definiert:

Definition 2.5.8 (Alphabet mit Unabhängigkeitsrelation zum Netz)

Sei $N = (S, T, F, m_0)$ ein sicheres Petrinetz.

Das Alphabet zu N ist $\Sigma_N =_{Df} T$.

Die Unabhängigkeitsrelation $I_n \subseteq \Sigma_N \times \Sigma_N$ wird definiert als

$$I_N =_{Df} \left\{(t,t') \in \Sigma_N \times \Sigma_N | \bigwedge_{s \in S} ((s,t) \notin F \wedge (t,s) \notin F) \vee ((s,t') \notin F \wedge (t',s) \notin F)\right\}$$

Im Klartext: Zwei Transitionen sind genau dann unabhängig, wenn sie nicht über eine Stelle verbunden sind.

Korollar 2.5.9 (Unabhängigkeit von Transitionen)

Sei $N = (S, T, F, m_0)$ ein sicheres Petrinetz und (Σ_N, I_N) sein Alphabet mit Unabhängigkeitsrelation. Für Markierungen m, m′ von N gilt:

$$(t,t') \in I_N \Rightarrow (m[tt' > m' \iff m[t't > m')$$

$v \sim_{I_N} w$ impliziert somit $m[v > m' \iff m[w > m'$

Definition 2.5.10 (Trace-Verhalten eines Netzes)

Sei $N = (S, T, F, m_0)$ ein sicheres Petrinetz und (Σ_N, I_N) sein Alphabet mit Unabhängigkeitsrelation. Sei h die identische Abbildung auf T. Das Trace-Verhalten von N ist die Trace-Sprache

$$\begin{aligned} T(N) &=_{Df} \{[w]_{I_N} \in E(\Sigma_N, I_N) | \bigvee_{m \in Err(m_0)} m_0[w > m\} \\ &= [P(N,h)]_{I_N} \end{aligned}$$

Das Trace-Verhalten umfaßt die in m_0 aktivierten Abläufe des Netzes, wobei zwischen solchen Schaltfolgen nicht getrennt wird, die sich nur in der Reihenfolge von nebenläufigen Transitionen unterscheiden.

Nicht jede präfixabgeschlossene erkennbare Trace-Sprache ist Verhalten eines sicheren Petrinetzes. Beispielsweise kann die Trace-Sprache $\{[\varepsilon], [a], [aa]\}$ über $(\{a\}, \emptyset)$ nicht durch ein sicheres Petrinetz dargestellt werden. Die Transition a kann nicht genau zweimal in Folge schalten. Dieses Beispiel macht nochmals deutlich, warum in Definition 2.5.5 die Petrinetzsprache als homomorphes Bild der im Netz ausführbaren Schaltfolgen definiert wurde.

2.6 Endliche asynchrone Automaten

Der Begriff des endlichen asynchronen Automaten (EAA) wurde 1987 als ein auf Traces arbeitender Automatentyp von W. Zielonka eingeführt (vgl. [Ziel87]). Ein EAA setzt sich aus parallel arbeitenden Teilautomaten zusammen, die durch Verknüpfung von Zustandsübergängen synchronisiert werden. Die Übergangsrelation kann daher nicht lokal für jeden Teilautomaten angegeben werden, sondern wird global für den ganzen EAA definiert. Ein Teilautomat kann also nicht allein über einen Zustandsübergang entscheiden. Die Anfangs- und Endzustände werden ebenfalls global definiert, so daß sich hier die gleiche Beziehung zwischen der lokalen Repräsentation der Zustände und der globalen Steuerung des Verhaltens des EAA ergibt.

Unabhängig von [Ziel87] definieren Drusinsky und Harel in [DH94] verteilte endliche Automaten mit ähnlicher Struktur, jedoch ohne den Trace-Begriff und eine Unabhängigkeitsrelation auf Zeichen einzuführen.

Die genaue Definition nach [Ziel87]:

Definition 2.6.1 (Endlicher asynchroner Automat)

Ein endlicher asynchroner Automat (EAA) bestehend aus n Teilautomaten ist ein $n+3$-Tupel
$\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit

i) $\mathcal{P}_i = (\Sigma_i, S_i)$ ist der i-te Teilautomat, wobei

Σ_i das Alphabet und

S_i die endliche nichtleere Zustandsmenge des Teilautomaten ist.

Das Alphabet von $\mathcal{A}$ ist $\Sigma_{\mathcal{A}} =_{Df} \bigcup_{i\in\{1,\ldots,n\}} \Sigma_i$.
Die Menge der globalen Zustände von $\mathcal{A}$ ist $S =_{Df} \prod_{i\in\{1,\ldots,n\}} S_i$.
Für ein Zeichen $a \in \Sigma_{\mathcal{A}}$ ist $Dom(a) =_{Df} \{i \in \{1,\ldots,n\} | a \in \Sigma_i\}$ *die Heimat von a.*

ii) $\mathcal{I} \subseteq S$ *ist die Menge der Anfangszustände.*

iii) $\mathcal{D} =_{Df} \{\delta_a | a \in \Sigma_{\mathcal{A}}\}$ *ist eine Schar von Übergangsrelationen*

$$\delta_a : \prod_{i\in Dom(a)} S_i \longrightarrow \wp(\prod_{i\in Dom(a)} S_i)$$

Ein a-Übergang zwischen zwei globalen Zuständen $(s_1,\ldots,s_n),(s'_1,\ldots,s'_n) \in S$ *wird definiert als*
$(s_1,\ldots,s_n) \stackrel{a}{\Longrightarrow} (s'_1,\ldots,s'_n) \Longleftrightarrow_{Df}$

$$\Longleftrightarrow_{Df} \begin{cases} s'_i = s_i & für \quad i \notin Dom(a) \\ (s'_{i_1},\ldots,s'_{i_k}) \in \delta_a(s_{i_1},\ldots,s_{i_k}) & für \quad \{i_1,\ldots,i_k\} = Dom(a) \end{cases}$$

D. h.: Ein a-Übergang bewirkt Zustandsübergänge in den Heimatteilautomaten von a gemäß δ_a und hat auf die Zustände der anderen Teilautomaten keine Auswirkungen.

iv) $\mathcal{F} \subseteq S$ *ist die Menge der Endzustände.*

In [Ziel87] definiert Zielonka den EAA mit $|\mathcal{I}| = 1$, so daß es für jeden Teilautomaten einen lokalen Anfangszustand gibt. Die Verallgemeinerung nutzt das Konzept des nichtdeterministischen Automaten konsequenter aus. Im Zusammenhang mit der Sicherheit des EAAs (Def. 2.6.8) wird dies später eine Rolle spielen.

Die Wortsprache eines EAAs ist die Menge der Zeichenfolgen, die einen Anfangszustand in einen Endzustand überführen.

Definition 2.6.2 (Wortsprache eines EAA)

Sei $\mathcal{A}$ ein EAA wie in Definition 2.6.1 beschrieben.

i) Sei $w \in \Sigma^*_{\mathcal{A}}$ ein Wort über dem Alphabet von $\mathcal{A}$ und seien $s, s' \in S$ globale Zustände von $\mathcal{A}$.
w überführt den Zustand s in den Zustand s' (geschrieben $s \stackrel{w}{\Longrightarrow} s'$)

$$\Longleftrightarrow_{Df} \begin{cases} s = s' & falls \quad w = \varepsilon \\ \bigvee_{s''\in S} s \stackrel{v}{\Longrightarrow} s'' \wedge s'' \stackrel{a}{\Longrightarrow} s' & falls \quad w = va, v \in \Sigma^*_{\mathcal{A}}, a \in \Sigma_{\mathcal{A}} \end{cases}$$

ii) Die Wortsprache von $\mathcal{A}$ ist

$$L(\mathcal{A}) =_{Df} = \{w \in \Sigma^*_{\mathcal{A}} | \bigvee_{s_0\in\mathcal{I}} \bigvee_{s\in\mathcal{F}} s_0 \stackrel{w}{\Longrightarrow} s\}$$

Zwei Zeichen mit disjunkten Heimaten kann ein EAA in beliebiger Reihenfolge akzeptieren, weil bei der Verarbeitung eines Zeichens nur die Zustände der Heimatteilautomaten gelesen und verändert werden. Es gilt das folgende Lemma ([Ziel87]):

Lemma 2.6.3 (Parallele Zeichenverarbeitung)

Sei $\mathcal{A}$ ein EAA und seien $a, b \in \Sigma_\mathcal{A}$ Zeichen seines Alphabets mit $Dom(a) \cap Dom(b) = \emptyset$. Für beliebige Zustände $s, s' \in S$ von $\mathcal{A}$ gilt:

$$s \stackrel{ab}{\Longrightarrow} s' \iff s \stackrel{ba}{\Longrightarrow} s'$$

Die Vertauschbarkeit von Zeichen deutet auf den Trace-Begriff hin. Bevor aber von Traces gesprochen werden kann, muß eine Unabhängigkeitsrelation definiert werden.

Lemma 2.6.4 (Verarbeitung von Traces)

Sei $\mathcal{A}$ ein EAA. Auf seinem Alphabet wird die Unabhängigkeitsrelation

$$I_\mathcal{A} =_{Df} \{(a, b) \in \Sigma_\mathcal{A}^2 | Dom(a) \cap Dom(b) = \emptyset\}$$

definiert.
Für zwei Wörter $v, w \in \Sigma_\mathcal{A}^*$ mit $v \sim_{I_\mathcal{A}} w$, also zwei Repräsentanten desselben Traces, und beliebige Zustände $s, s' \in S$ von $\mathcal{A}$ gilt:

$$s \stackrel{v}{\Longrightarrow} s' \iff s \stackrel{w}{\Longrightarrow} s'$$

Es läßt sich nun definieren, welches die von einem EAA erkannte Trace-Sprache ist.

Definition 2.6.5 (Trace-Sprache eines EAA)

Sei $\mathcal{A}$ ein EAA und $(\Sigma_\mathcal{A}, I_\mathcal{A})$ sein Alphabet mit der entsprechend Lemma 2.6.4 definierten Unabhängigkeitsrelation.

i) Für einen Trace $[w] \in E(\Sigma_\mathcal{A}, I_\mathcal{A})$ und Zustände $s, s' \in S$ von $\mathcal{A}$ wird definiert:

$$s \stackrel{[w]}{\Longrightarrow} s' \iff_{Df} s \stackrel{w}{\Longrightarrow} s'$$

ii) Die Trace-Sprache von $\mathcal{A}$ ist

$$T(\mathcal{A}) =_{Df} \{t \in E(\Sigma_\mathcal{A}, I_\mathcal{A}) | \bigvee_{s_0 \in \mathcal{I}} \bigvee_{s \in \mathcal{F}} s_0 \stackrel{t}{\Longrightarrow} s\}$$

Nach Lemma 2.6.4 bewirken alle Repräsentanten eines Traces in einem EAA dieselben Zustandsübergänge. Folglich akzeptiert ein solcher Automat alle Wörter $w \in \Sigma_\mathcal{A}^*$ mit $[w] = t$ für einen festen Trace $t \in E(\Sigma_\mathcal{A}, I_\mathcal{A})$ oder keins mit dieser Eigenschaft. Die Beziehung zwischen der Wort- und Trace-Sprache eines Automaten $\mathcal{A}$ ist also $T(\mathcal{A}) = [L(\mathcal{A})]$ und $L(\mathcal{A}) = Lin(T(\mathcal{A}))$.

Weil ein EAA $\mathcal{A}$ mit seiner Zustandsmenge S ein gewöhnlicher endlicher Automat ist, ist $L(\mathcal{A})$ regulär. Die Beobachtung $L(\mathcal{A}) = Lin(T(\mathcal{A}))$ ergibt demnach zusammen mit Korollar 2.4.8 (vgl. [Ziel87]):

Lemma 2.6.6 (Erkennbarkeit von $T(\mathcal{A})$)

Sei $\mathcal{A}$ ein EAA und $(\Sigma_\mathcal{A}, I_\mathcal{A})$ sein Alphabet mit Unabhängigkeitsrelation.
Die Trace-Sprache $T(\mathcal{A}) \subseteq E(\Sigma_\mathcal{A}, I_\mathcal{A})$ ist erkennbar.

Die Umkehrung „Jede erkennbare Trace-Sprache wird von einem EAA erkannt“ ist aufgrund der speziellen Struktur dieses Automatentyps nichttrivial und wird in Kapitel 3 behandelt.

Ein EAA $\mathcal{A}$ zeichnet sich gegenüber einem gewöhnlichen endlichen Automaten für die Wortsprache $L(\mathcal{A})$ durch die verteilte Repräsentation seiner Zustände aus. Die Darstellung von EAAs als Petrinetze drängt sich geradezu auf. Die Zustände der Teilautomaten sind die Stellen des Petrinetzes. Übergänge werden durch Transitionen realisiert, die mit den Zeichen des Alphabets beschriftet sind. Die Anfangs- und Endzustandsmengen lassen sich leider nur dann übersichtlich in die Darstellung einbringen, wenn sie Kreuzprodukte von lokalen Anfangs- und Endzuständen sind. Jeder Anfangszustand des EAAs ist eine Anfangsmarkierung des Petrinetzes. Ein Teilautomat befindet sich stets in genau einem Zustand, so daß von den ihn repräsentierenden Stellen immer genau eine einfach markiert ist. Das Netz zu einem EAA ist sicher.

Beispiel:
Sei $\mathcal{A} = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{I}, \mathcal{D}, \mathcal{F})$ ein EAA mit
$\mathcal{P}_1 = (\Sigma_1, S_1), \quad \Sigma_1 = \{a, b\}, \quad S_1 = \{s_1^0, s_1^1\}$
$\mathcal{P}_2 = (\Sigma_2, S_2), \quad \Sigma_2 = \{b, c\}, \quad S_2 = \{s_2^0, s_2^1\}$
$\mathcal{P}_3 = (\Sigma_3, S_3), \quad \Sigma_3 = \{c, d\}, \quad S_3 = \{s_3^0, s_3^1\}$
$\delta_a(s_1^0) = \{s_1^1\}, \quad \delta_b(s_1^1, s_2^0) = \{(s_1^0, s_2^1)\}, \quad \delta_c(s_2^1, s_3^0) = \{(s_2^0, s_3^1)\}, \quad \delta_d(s_3^1) = \{s_3^0\},$
$\delta = \emptyset$ sonst
und $\mathcal{I} = \mathcal{F} = \{(s_1^0, s_2^0, s_3^0)\}$.
Abbildung 2.7 zeigt den Automaten als sicheres transitionsbeschriftetes Petrinetz.

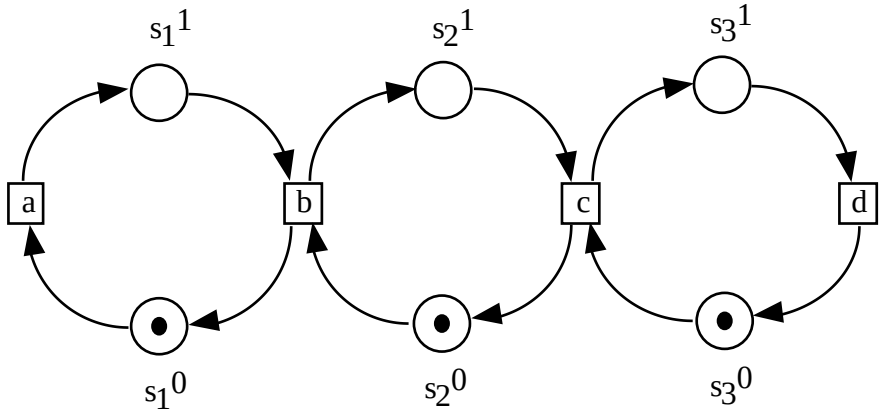

Abbildung 2.7: Der EAA als Petrinetz

Das Alphabet von $\mathcal{A}$ ist $\Sigma_\mathcal{A} = \{a, b, c, d\}$.
Die Unabhängigkeitsrelation ist $I_\mathcal{A} = \{(a, c), (a, d), (b, d), (c, a), (d, a), (d, b)\}$.
Die von $\mathcal{A}$ erkannte Trace-Sprache ist $T(\mathcal{A}) = [(abcd)^*]$.

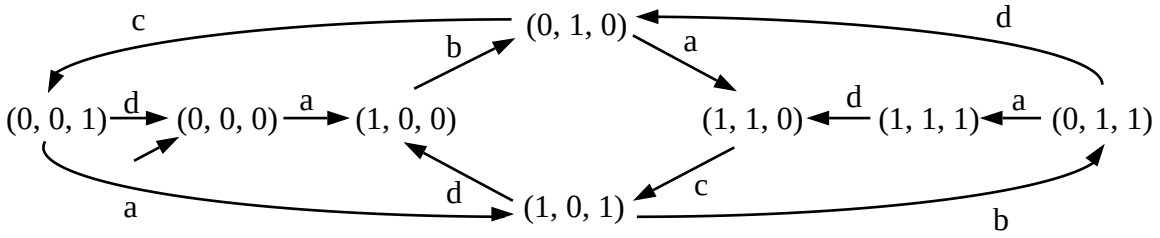

Abbildung 2.8: Übergänge zwischen den Globalzuständen von $\mathcal{A}$

Die Übergänge zwischen den Globalzuständen von $\mathcal{A}$ sind in Abb. 2.8 dargestellt. (x, y, z) ist die Abkürzung für (s_1^x, s_2^y, s_3^z).

Im Beispiel sind alle Globalzustände vom Anfangszustand aus erreichbar. Ein EAA mit der gleichen Struktur verallgemeinert auf n Teilautomaten hat $2n$ lokale Zustände und $n + 1$ Übergänge. Gleichzeitig wächst die Zahl der Globalzustände auf 2^n. Das Beispiel zeigt, daß unter Ausnutzung der Unabhängigkeitsrelation die einer Trace-Sprache zugrundeliegende Wortsprache sehr viel kompakter durch einen EAA dargestellt werden kann.

Komplexitätstheoretische Zusammenhänge in der Größe zwischen äquivalenten deterministischen, nichtdeterministischen, verteilten und sog. $\bigwedge$-Automaten werden in [DH94] behandelt.

Abschließend werden drei Eigenschaften von EAAs definiert.

Definition 2.6.7 (Deterministischer EAA)

Ein EAA $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ heißt deterministisch, wenn

i) es genau einen globalen Anfangszustand gibt: $|\mathcal{I}| = 1$

ii) für alle seine Übergangsrelationen δ_a, $a \in \Sigma_{\mathcal{A}}$, und alle $(s_{i_1}, \ldots, s_{i_k}) \in \prod_{j \in Dom(a)} S_j$ gilt:

$$|\delta_a(s_{i_1}, \ldots, s_{i_k})| \leq 1$$

Definition 2.6.8 (Sicherer EAA)

Ein EAA heißt sicher, wenn für alle globalen Zustände $s \in S$ gilt:

$$\bigvee_{s_0 \in \mathcal{I}} \bigvee_{w \in \Sigma^*} s_0 \stackrel{w}{\Longrightarrow} s \quad \Longrightarrow \quad \bigvee_{s_f \in \mathcal{F}} \bigvee_{v \in \Sigma^*} s \stackrel{v}{\Longrightarrow} s_f$$

In sicheren EAAs gibt es von jedem erreichbaren Globalzustand eine Fortsetzung zu einem Endzustand. Sichere EAAs können außer in Endzuständen nicht verklemmen.

Beispiel (siehe Abb. 2.9): Ein nicht sicherer EAA ist $\mathcal{A} = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit
$\mathcal{P}_1 = (\{a\}, \{s_1^0, s_1^1\}), \quad \mathcal{P}_2 = (\{b\}, \{s_2^0, s_2^1\})$
$\mathcal{I} = \{(s_1^0, s_2^0)\}$
$\delta_a(s_1^0) = \{s_1^1\}, \quad \delta_b(s_2^0) = \{s_2^1\}, \quad \delta_a(s_1^1) = \delta_b(s_2^1) = \emptyset$

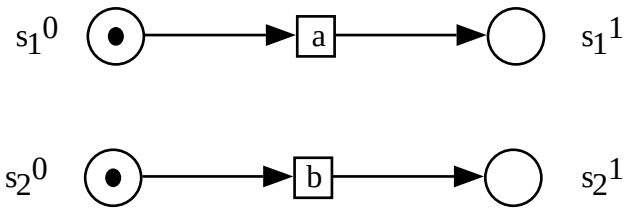

Abbildung 2.9: Ein nicht sicherer Automat

$\mathcal{F} = \{(s_1^0, s_2^1), (s_1^1, s_2^0)\}$

Das Alphabet des Automaten ist $\Sigma_\mathcal{A} = \{a, b\}$ mit der Unabhängigkeitsrelation $I_\mathcal{A} = \{(a, b), (b, a)\}$. $\mathcal{A}$ erkennt die Trace-Sprache $T(\mathcal{A}) = \{[a], [b]\}$.
Nach Verarbeitung des Traces $[ab]$ erreicht $\mathcal{A}$ den Zustand (s_1^1, s_2^1), von dem aus kein Endzustand erreicht werden kann. Offensichtlich kann der Globalzustand (s_1^1, s_2^1) nicht gestrichen werden, ohne die Trace-Sprache des Automaten zu verändern, weil die Zustände s_1^1 und s_2^1 in erreichbaren Endzuständen vorkommen.

Definition 2.6.9 (Normalform eines EAA)

Sei $\mathcal{A}$ ein EAA und $(\Sigma_\mathcal{A}, I_\mathcal{A})$ sein Alphabet mit Unabhängigkeitsrelation. $D_\mathcal{A} = \Sigma_\mathcal{A}^2 \setminus I_\mathcal{A}$ ist die Abhängigkeitsrelation von $\mathcal{A}$ auf Σ.
$\mathcal{A}$ ist in Normalform, wenn für die Alphabete Σ_i der Teilautomaten gilt:

i) $\Sigma_i = \Sigma_j \iff i = j$ *für alle* $i, j \in \{1, \ldots, n\}$

ii) Die Σ_i sind maximale Cliquen der Abhängigkeitsrelation, also

$$\bigwedge_{i\in\{1,\ldots,n\}} \bigwedge_{a\in\Sigma_\mathcal{A}} \Big(\bigwedge_{b\in\Sigma_i} (a, b) \in D \Big) \Longrightarrow a \in \Sigma_i$$

Ein EAA in Normalform hat unter allen EAA mit der gleichen Unabhängigkeitsrelation die minimale Anzahl von Teilautomaten. Nach [Ziel87] gilt

Lemma 2.6.10 (Existenz eines äquivalenten EAAs in Normalform)

Sei $\mathcal{A}$ ein EAA und $(\Sigma_\mathcal{A}, I_\mathcal{A})$ sein Alphabet mit Unabhängigkeitsrelation.
Zu $\mathcal{A}$ gibt es einen EAA $\mathcal{A}'$ in Normalform über $(\Sigma_\mathcal{A}, I_\mathcal{A})$ mit $T(\mathcal{A}) = T(\mathcal{A}')$.

In [Ziel89] modifiziert Zielonka die Definition des EAAs derart, daß der verteilte Automat bei vorgegebenen Alphabet stets die gleiche Zahl von Teilautomaten hat. Bei diesem sog. *zellulären* endlichen asynchronen Automaten wird für jedes Zeichen des Alphabets ein Teilautomat definiert. Der zelluläre EAA verarbeitet ein Zeichen $a \in \Sigma$, indem in dem zugehörigen Teilautomaten $\mathcal{P}_a$ ein Zustandsübergang stattfindet. Dieser Übergang wird durch die Übergangsrelation von den Zuständen der Teilautomaten der mit a abhängigen Zeichen abhängig gemacht. Unabhängige Zeichen $a, b \in \Sigma$ können nebenläufig verarbeitet werden, weil nur Zustandsübergänge in $\mathcal{P}_a$ und $\mathcal{P}_b$ stattfinden und der Zustandsübergang in dem einen Teilautomaten nicht vom Zustand des anderen Teilautomaten abhängt.

Definition 2.6.11 (Zellulärer EAA, [Ziel89])

Ein zellulärer endlicher asynchroner Automat über dem Alphabet mit Unabhängigkeitsrelation (Σ, I) ist ein Viertupel $\mathcal{Z} = (\mathcal{S}, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit

i) $\mathcal{S} = \{S_a | a \in \Sigma\}$ *ist eine Familie von endlichen Mengen.* $S_a, a \in \Sigma$, *ist die Menge der Zustände des a-Teilautomaten.*
Eine feste Ordnung auf Σ *vorausgesetzt, ist* $S =_{Df} \prod_{a \in \Sigma} S_a$ *die Menge der globalen Zustände von* $\mathcal{Z}$.

ii) $\mathcal{I} \subseteq S$ *ist die Menge der globalen Anfangszustände von* $\mathcal{Z}$.

iii) $\mathcal{D} = \{\delta_a | a \in \Sigma\}$ *ist eine Familie von Übergangsfunktionen. Für* $a \in \Sigma$ *sei* $D(a) =_{Df} \{b \in \Sigma | (a, b) \notin I\}$ *die Menge der mit a abhängigen Zeichen.* δ_a *ist eine Abbildung*

$$\delta_a : \prod_{b \in D(a)} S_b \longrightarrow \wp(S_a)$$

Bedeutung: Wenn die b-Teilautomaten, $b \in D(a)$, *sich in den Zuständen* $s_1, \ldots, s_{|D(a)|}$ *befinden, so kann der a-Teilautomat einen Zustandsübergang in einen Zustand aus* $\delta_a(s_1, \ldots, s_{|D(a)|})$ *ausführen.*

iv) $\mathcal{F} \subseteq S$ *ist die Menge der globalen Endzustände von* $\mathcal{Z}$.

Nach Pighizzini (vgl. [Pig94]) läßt sich mit polynomiellem Zeitaufwand zu einem EAA ein äquivalenter zellulärer EAA konstruieren und umgekehrt. Während zelluläre EAAs nur aus $|\Sigma|$ Teilautomaten bestehen, wächst dagegen die Zahl der Teilautomaten eines gewöhnlichen EAAs mit der Zahl der maximalen D-Cliquen im Alphabet Σ, also exponentiell. Ein Alphabet mit n Zeichen (n durch 3 teilbar) kann $3^{\frac{n}{3}}$ maximale Cliquen haben (vgl. [MM65], Beispiel für $n = 6$ in Abb. 2.10).

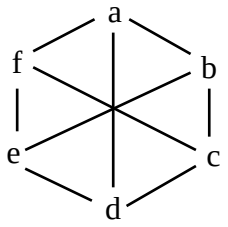

Abbildung 2.10: Alphabet mit sechs Zeichen und neun D-Cliquen

Nach Zielonka sind zelluläre EAAs in der Theorie leichter handhabbar, und es lassen sich theoretische Ergebnisse bezüglich des einen Automatentyps leicht auf den anderen Automatentyp übertragen. Demgegenüber sind zelluläre EAAs schwierig graphisch darzustellen. Während EAAs in sicheren transitionsbeschrifteten Petrinetzen eine anschauliche Repräsentation haben, ist für zelluläre EAAs noch keine befriedigende Darstellungsform gefunden worden. In zellulärens EAA werden Zustände meist nur abgefragt, aber nicht verändert. In

einem Petrinetz müßte dies durch eine Schlinge zwischen der Transition (Zustandsänderung) und der Stelle (abgefragter Zustand) repräsentiert werden.

Beispiel: In Abbildung 2.11 kann der Teilautomat $\mathcal{P}_a$ vom Zustand s_a^1 in den Zustand s_a^2 übergehen, wenn $\mathcal{P}_b$ sich im Zustand s_b^0 befindet ($(a, b) \notin I$).

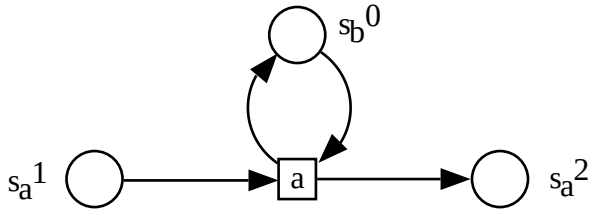

Abbildung 2.11: Zustandsabfrage in zellulären EAAs

Wenn nun ein zweiter Zustandsübergang c mit $(c, b) \notin I$ und $(a, c) \in I$ in gleicher Weise wie a auf die Stelle in $\mathcal{P}_b$ zugreift, sind die Transitionen a und c nicht mehr nebenläufig, sondern stehen im Konflikt um die Marke auf der Stelle s_b^0. a und c können zwar in beliebiger Reihenfolge, jedoch nicht gleichzeitig schalten, im Widerspruch zur Unabhängigkeit von a und c. In der Sprache der Trace-Theorie wird dieses Verhalten mit $\{[ac], [ca]\}$ und $(a, c) \notin I$ beschrieben.

Als Ausweg bietet sich an, das reine Abfragen von Zuständen aus anderen Teilautomaten durch einen neuen Kantentyp darzustellen. Diese sog. aktivierenden Kanten haben nur Einfluß auf die Aktiviertheit der Transition. Wenn sich diese Darstellungsform für zelluläre Automaten noch nicht durchgesetzt hat, mag es daran liegen, daß Graphen mit mehreren Kantentypen schnell unübersichtlich werden.

Kapitel 3

Die Klasse der erkennbaren Trace-Sprachen und endliche asynchrone Automaten

In diesem Kapitel wird ein Überblick über die aus der Literatur bekannten Ergebnisse zum Thema gegeben. Die ersten vier Abschnitte behandeln die Klassen der rationalen bzw. erkennbaren Trace-Sprachen. In den Abschnitten 3.1 und 3.2 werden die Abschlußeigenschaften dieser Sprachklassen untersucht und eine algebraische Charakterisierung der Klasse der erkennbaren Trace-Sprachen gegeben. Für rationale bzw. erkennbare Sprachen von unendlichen bzw. höchstens unendlichen Traces werden Abschlußeigenschaften und Charakterisierungen in [Gast91] behandelt. Diese Ergebnisse werden hier jedoch nicht aufgenommen. In den Abschnitten 3.3 und 3.4 folgen Betrachtungen von Teilklassen der erkennbaren Trace-Sprachen und von Entscheidbarkeitsfragen.

Die Abschnitte 3.5 bis 3.7 sind der Theorie der endlichen asynchronen Automaten gewidmet. Analog zur Theorie der endlichen Automaten wird die Beziehung des Automatenmodells zu den Sprachklassen und die Existenz von eindeutigen minimalen bzw. deterministischen Automaten behandelt.

3.1 Abschlußeigenschaften

Die Klassen der erkennbaren und der rationalen Trace-Sprachen über einem Alphabet mit nichtleerer Unabhängigkeitsrelation unterscheiden sich in ihren Abschlußeigenschaften. Wie im Zusammenhang mit der Trace-Iteration (Seite 16) bereits erwähnt wurde, führt beispielsweise die gewöhnliche Iteration aus der Klasse der erkennbaren Trace-Sprachen in die Menge der nicht erkennbaren rationalen Sprachen, sobald zwei unabhängige Zeichen zusammen iteriert werden.

In diesem und den folgenden Abschnitten über Sprachklassen hat die Unabhängigkeitsrelation den Charakter eines Parameters, d. h. Abschluß- und Entscheidbarkeitsfragen werden

in Abhängigkeit von der Unabhängigkeitsrelation untersucht. Eine besondere Rolle spielen Alphabete mit quasi-transitiver Unabhängigkeits- bzw. transitiver Abhängigkeitsrelation (siehe Def. 2.2.1). Wenn die Unabhängigkeitsrelation quasi-transitiv ist, kann das Alphabet in Klassen von unabhängigen Zeichen zerlegt werden. Ein Trace $t \neq [\varepsilon]$ über einem solchen Alphabet zerfällt zu $t = t_1 \dots t_n$ derart, daß zwischen t_i und t_{i+1} keine Zeichenvorkommen vertauscht werden können, innerhalb der t_i aber abgesehen von mehrfachen Vorkommen desselben Zeichens keine Reihenfolge vorgegeben ist (siehe Abb. 3.1).

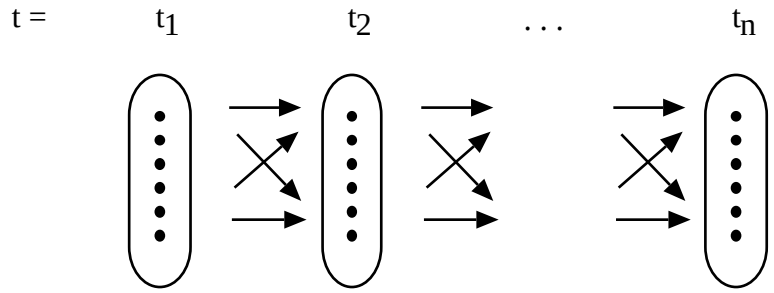

Abbildung 3.1: Trace auf Alphabet mit quasi-transitivem I

Ein Alphabet mit transitiver Abhängigkeitsrelation D ist aus Äquivalenzklassen abhängiger Zeichen zusammengesetzt. Ein Trace ergibt sich als das Produkt von nicht synchronisierten Wörtern über den Äquivalenzklassen (Abb. 3.2).

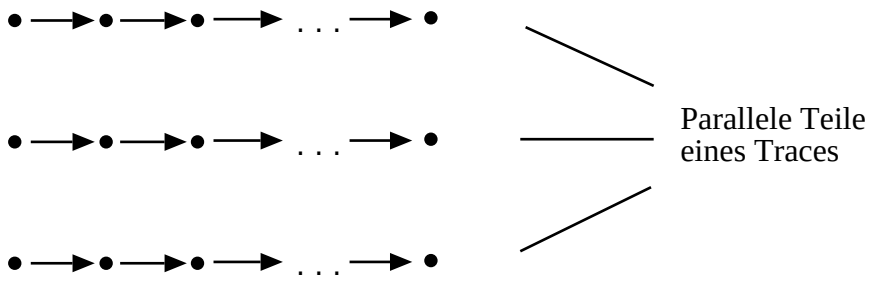

Abbildung 3.2: Trace auf Alphabet mit transitivem D

Ohne größeren Beweisaufwand ergeben sich aus der Definition der rationalen Trace-Sprachen und aus Korollar 2.4.8 für erkennbare Trace-Sprachen folgende Abschlußeigenschaften (vgl. [AW86]):

Lemma 3.1.1 (Einfache Abschlußeigenschaften)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation.

Die Klasse der rationalen Trace-Sprachen über (Σ, I) ist abgeschlossen bezüglich Verkettung, Vereinigung und Iteration.

Die Klasse der erkennbaren Trace-Sprachen über (Σ, I) ist abgeschlossen bezüglich Vereinigung, Durchschnitt, Komplementbildung und Verkettung.

Trace-Sprachen über einem Alphabet mit $I = \emptyset$ entsprechen Wortsprachen. Da die Klasse der regulären Wortsprachen bezüglich Iteration abgeschlossen ist, gilt dies also auch für erkennbare Trace-Sprachen über $(\Sigma, \emptyset)$.

Sei (Σ, I) ein Alphabet mit nichtleerer Unabhängigkeitsrelation. Seien a und b unabhängige Zeichen aus Σ und $T = \{[ab]\}$. T^* ist nicht erkennbar, denn die Wortsprache $Lin(T^*) = \{w \in \{a,b\}^* || w|_a = |w|_b\}$ ist nicht regulär. Folglich gilt:

Lemma 3.1.2 (Abschluß bzgl. Iteration)

Die Klasse der erkennbaren Trace-Sprachen über (Σ, I) ist bezüglich der Iteration abgeschlossen $\iff I = \emptyset$

Dieses Ergebnis steht in [AW86], läßt sich aber auch leicht aus Zielonkas Satz über EAAs und erkennbare Trace-Sprachen folgern (vgl. Abschnitt 3.5). Eine Untersuchung der Frage, für welche (Σ, I) die Erkennbarkeit von T^* (mit erkennbarer Trace-Sprache $T \subseteq E(\Sigma, I)$) entscheidbar ist, findet sich in [GOPR92].

Der Rückgriff auf die Charakterisierung (Korollar 2.4.8) hilft ebenfalls weiter, wenn die Abgeschlossenheit der erkennbaren Trace-Sprachen bezüglich Synchronisation entschieden werde soll.

Seien $T_1 \subseteq E(\Sigma_1, I_1)$ und $T_2 \subseteq E(\Sigma_2, I_2)$ zwei erkennbare Trace-Sprachen. Nach Korollar 2.3.8 gilt für $T_1 \| T_2$:

$$Lin(T_1 \| T_2) = \{w \in (\Sigma_1 \cup \Sigma_2)^* |\ w|_{\Sigma_1} \in Lin(T_1) \wedge w|_{\Sigma_2} \in Lin(T_2)\}$$

Einen endlichen Automaten für $Lin(T_1 \| T_2)$ erhält man somit, indem man die deterministischen endlichen Automaten für $Lin(T_1)$ und $Lin(T_2)$ zu einem Produktautomaten koppelt. Der Produktautomat überprüft die Projektionen auf Σ_1 und Σ_2 und gerät genau dann in einen Endzustand, wenn beide Automaten die Projektionen akzeptieren würden. Folglich gibt es einen endlichen Automaten für $Lin(T_1 \| T_2)$, und $T_1 \| T_2$ ist erkennbar.

Satz 3.1.3 (Abschluß bzgl. Synchronisation für erkennbare Trace-Sprachen)

Die Klasse aller erkennbaren Trace-Sprachen sowie die Klassen der erkennbaren Trace-Sprachen über Alphabeten mit fester Unabhängigkeitsrelation sind bezüglich Synchronisation abgeschlossen.

Die Aussage von Satz 3.1.3 findet sich bereits in [Maz87].

Die Abschlußeigenschaften der erkennbaren Trace-Sprachen ergeben sich recht einfach aus der Regularität der zugrundeliegenden Wortsprachen. Die rationalen Trace-Sprachen stehen nicht in einer so engen Beziehung zu den regulären Wortsprachen, so daß sich hier ein komplizierteres Bild ergibt.

Die Operationen, über die die Klasse der rationalen Trace-Sprachen definiert werden, sind bereits in Lemma 3.1.1 aufgeführt. Für die noch fehlenden mengentheoretischen Operationen gilt nach Aalbersberg/Welzl [AW86] und Sakarovitch [Sak87]:

Satz 3.1.4 (Durchschnitt und Komplement auf rationalen Trace-Sprachen)

Sei (Σ, I) *ein Alphabet mit Unabhängigkeitsrelation.*

$$
\begin{aligned}
& \textit{Die Klasse der rationalen Trace-Sprachen über } (\Sigma, I) \\
& \textit{ist abgeschlossen bezüglich Schnittbildung auf Sprachen} \\
\Longleftrightarrow\ & \textit{Die Klasse ist abgeschlossen bzgl. Komplement} \\
\Longleftrightarrow\ & \textit{I ist quasi-transitiv}
\end{aligned}
$$

Beweisidee: Falls I nicht quasi-transitiv ist, Σ also Zeichen a, b, c mit $\{(a, b), (b, c)\} \subseteq I$ und $(a, c) \notin I$ enthält, lassen sich Trace-Sprachen T, T_1, T_2 über (Σ, I) angeben und beweisen, daß $E(\Sigma, I) \setminus T$ und $T_1 \cap T_2$ nicht rational sind. Der Beweis der Rückrichtung der unteren Äquivalenz erfolgt in drei Schritten:

- Die genannten Abschlußeigenschaften gelten für (Σ, I) mit $I = \Sigma^2 \setminus id_\Sigma$.
- Gelten die Abschlußeigenschaften für (Σ_1, I_1) und (Σ_2, I_2) mit $\Sigma_1 \cap \Sigma_2 = \emptyset$, so auch für $(\Sigma_1 \cup \Sigma_2, I_1 \cup I_2)$
- Ein Alphabet mit quasi-transitiver Unabhängigkeitsrelation ist die disjunkte Vereinigung von Teilalphabeten (Σ, I) mit $I = \Sigma^2 \setminus id_\Sigma$.

3.2 Algebraische Charakterisierung der erkennbaren Trace-Sprachen

Die Ergebnisse diese Abschnitts gehen auf [Och85] zurück. Ochmański betrachtet erkennbare Trace-Sprachen und reguläre Ausdrücke über einem Alphabet Σ. Der Iterationsoperator wird in diesen Ausdrücken als Trace-Iteration (vgl. Abschnitt 2.3) interpretiert. In den Beweisen benutzt Ochmański als Hilfsmittel den lexikalisch kleinsten Wortrepräsentanten eines Traces (Ordnung auf Σ vorausgesetzt). Zu jeder Trace-Sprache läßt sich die Wortsprache der lexikalisch kleinsten Repräsentanten ihrer Elemente definieren.

Lemma 3.2.1 (Abschluß bzgl. Trace-Iteration für erkennbare Trace-Sprachen)

Die Klasse der erkennbaren Trace-Sprachen über einem Alphabet mit Unabhängigkeitsrelation ist bezüglich Trace-Iteration abgeschlossen.

Zusammen mit den Abschlußeigenschaften für erkennbare Trace-Sprachen ergibt sich aus dem Lemma, daß jeder reguläre Ausdruck, in dem der Iterationsoperator durch die Trace-Iteration interpretiert wird, eine erkennbare Trace-Sprache bezeichnet.

Satz 3.2.2 (Reguläre Ausdrücke und erkennbare Trace-Sprachen)

Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache.
T kann durch einen regulären Ausdruck dargestellt werden, wenn die regulären Operatoren durch Vereinigung und Verkettung auf Traces bzw. durch die Trace-Iteration interpretiert werden.

Aus der Folgerung zu Lemma 3.2.1 und Satz 3.2.2 ergibt sich eine neue Charakterisierung der erkennbaren Trace-Sprachen.

Satz 3.2.3 (Algebraische Charakterisierung der erkennbaren Trace-Sprachen)

Die Klasse der erkennbaren Trace-Sprachen über (Σ, I) ist die kleinste Klasse von Sprachen, die

i) alle endlichen Trace-Sprachen über (Σ, I) enthält und

ii) bezüglich Vereinigung, Verkettung und Trace-Iteration abgeschlossen ist.

3.3 Entscheidbarkeitsfragen

Die Ableitung der erkennbaren bzw. rationalen Trace-Sprachen aus den regulären Wortsprachen läßt vermuten, daß die meisten Probleme in diesen Sprachklassen entscheidbar sind. In der Tat gilt dies nur für die erkennbaren Trace-Sprachen und für einige Probleme in der Klasse der rationalen Sprachen. Für die Entscheidbarkeit der übrigen Probleme ist die Quasi-Transitivität der Unabhängigkeitsrelation eine hinreichende (z. T. auch notwendige) Bedingung.

Entscheidbarkeitsprobleme auf erkennbaren Trace-Sprachen lassen sich auf die zugrundeliegenden regulären Wortsprachen einfach zurückführen. Folglich gilt (vgl. [AW86]):

Satz 3.3.1 (Entscheidbarkeitsprobleme auf erkennbaren Trace-Sprachen)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und seien $T_1, T_2 \subseteq E(\Sigma, I)$ zwei beliebige erkennbare Trace-Sprachen über (Σ, I), gegeben durch endliche Automaten für die Wortsprachen $Lin(T_1)$ und $Lin(T_2)$. Folgende Probleme sind entscheidbar:

$$\begin{aligned} T_1 &= \emptyset \\ T_1 &= E(\Sigma, I) \\ T_1 &= T_2 \\ T_1 &\subseteq T_2 \\ T_1 \cap T_2 &= \emptyset \end{aligned}$$

Das Wortproblem „$t \in T$?“ für erkennbare Trace-Sprachen ist ein Spezialfall des Wortproblems für rationale Trace-Sprachen (vgl. [AR88]):

Satz 3.3.2 (Wortproblem)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation, sei $T \subseteq E(\Sigma, I)$ eine beliebige rationale Trace-Sprachen über (Σ, I) und sei $t \in E(\Sigma, I)$.
Das Wortproblem „$t \in T$?“ ist entscheidbar.

Das Entscheidungsverfahren für das Wortproblem hat die Relation I, einen Repräsentanten $w \in \Sigma^*$ von t und eine reguläre Wortsprache $L \subseteq \Sigma^*$ mit $T = [L]$ als Eingabe, wobei L durch einen regulären Ausdruck oder einen endlichen Automaten gegeben ist. Mit Hilfe von I können aus w alle Repräsentanten von t erzeugt werden. Für jeden Repräsentanten wird die Zugehörigkeit zu L überprüft. $t \in T$ gilt genau dann, wenn ein Repräsentant w' mit $w' \in L$ gefunden wird.

Das Leerheitsproblem wird in [AW86] untersucht:

Satz 3.3.3 (Leerheitsproblem für rationale Trace-Sprachen)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und $T \subseteq E(\Sigma, I)$ eine beliebige rationale Trace-Sprachen über (Σ, I).
Das Leerheitsproblem „$T = \emptyset$?“ ist entscheidbar.

Begründung: Eine rationale Trace-Sprache T läßt sich als $[L]$ mit regulärer Wortsprache L darstellen. Es gilt $[L] = \emptyset \iff L = \emptyset$. Das Leerheitsproblem ist für reguläre Wortsprachen entscheidbar, also auch für rationale Trace-Sprachen.

Eine Untersuchung der übrigen Entscheidbarkeitsfragen für rationale Trace-Sprachen findet sich in [AH87]. Hinreichende Bedingungen für Entscheidbarkeitsaussagen wurden bereits in [AW86] formuliert:

Satz 3.3.4 (Hinreichende Bedingungen für Entscheidbarkeitsaussagen)

Sei (Σ, I) ein Alphabet Σ mit quasi-transitiver Unabhängigkeitsrelation I und seien $T_1, T_2 \subseteq E(\Sigma, I)$ zwei beliebige rationale Trace-Sprachen über (Σ, I). Folgende Probleme sind entscheidbar:

$$\begin{aligned} T_1 &= E(\Sigma, I) \\ T_1 &= T_2 \\ T_1 &\subseteq T_2 \\ T_1 \cap T_2 &= \emptyset \end{aligned}$$

Der Beweis in [AW86] setzt auf die in 3.1.4 formulierten Abschlußeigenschaften für rationale Trace-Sprachen über (Σ, I) mit quasi-transitivem I auf. Die Klasse ist abgeschlossen bezüglich Komplement, und für jede rationale Trace-Sprache $T \subseteq E(\Sigma, I)$ kann eine reguläre Wortsprache $K \subseteq \Sigma^*$ mit $[K] = E(\Sigma, I) \setminus T$ effektiv konstruiert werden. Wegen $K = \emptyset \iff T = E(\Sigma, I)$ ist das erste der im Satz angegebenen Entscheidungsprobleme so

entscheidbar. Die übrigen drei Entscheidbarkeitsaussagen lassen sich mit Hilfe mengentheoretischer Operationen auf das erste Problem reduzieren.

Für den Beweis der Nichtentscheidbarkeit gewisser Fragen wird eine Sprache über einem Alphabet mit nicht quasi-transitiver Unabhängigkeitsrelation betrachtet (vgl. [AH87]):

Lemma 3.3.5 (Unentscheidbarkeit bei Nichtquasitransitivität)

Sei $\Sigma = \{a, b, c\}$ ein Alphabet mit der Unabhängigkeitsrelation $I = \{(a, b), (b, a), (b, c), (c, b)\}$. Sei $L \subseteq \Sigma^*$ eine beliebige durch einen endlichen Automaten gegebene reguläre Wortsprache. Das Problem „$[L] = E(\Sigma, I)$?“ ist unentscheidbar.

Der Beweis geht letztlich auf Ibarra (vgl. [Iba78]) zurück. Ibarra konstruiert einen endlichen Automaten mit zwei Eingabebändern über $\{a, c\}^* \times \{b\}^*$, der genau dann eine Eingabe auf dem ersten Band akzeptiert, wenn sie nicht die Konfigurationenfolge einer terminierenden Rechnung einer beliebigen vorgegebenen Einbandturingmaschine in $\{a, c\}^*$ verschlüsselt. Der Zweibandautomat akzeptiert folglich genau dann alle Eingaben, wenn die Einbandturingmaschine nicht terminiert. Von der Unentscheidbarkeit des Halteproblems für Turingmaschinen läßt sich somit auf die Unentscheidbarkeit des angegebenen Problems schließen.

Aus dem Lemma ergibt sich, daß die Quasi-Transitivität nicht nur hinreichende, sondern auch notwendige Bedingung der Entscheidbarkeitsaussagen ist (vgl. [AH87]).

Satz 3.3.6 (Äquivalente Bedingungen für Entscheidbarkeitsaussagen)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und seien $T_1, T_2 \subseteq E(\Sigma, I)$ zwei beliebige rationale Trace-Sprachen über (Σ, I). Folgende Probleme sind genau dann entscheidbar, wenn I quasi-transitiv ist:

$$\begin{aligned} T_1 &= E(\Sigma, I) \\ T_1 &= T_2 \\ T_1 &\subseteq T_2 \end{aligned}$$

Eine notwendige Bedingung für Alphabete mit Unabhängigkeitsrelation, auf denen $T_1 \cap T_2 = \emptyset$ entscheidbar ist, findet sich in [AH87].

Aus Lemma 3.3.5 ergibt sich ein weiteres wichtiges Ergebnis.

Satz 3.3.7 (Erkennbarkeit unentscheidbar, [Sak92])

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation.
Für eine beliebige rationale Trace-Sprache $T \subseteq E(\Sigma, I)$ ist entscheidbar, ob T erkennbar ist $\iff$ I ist quasi-transitiv

Wäre die Erkennbarkeit einer beliebigen rationalen Trace-Sprache über (Σ, I) entscheidbar, wenn I nicht quasi-transitiv ist, könnte auch das in Lemma 3.3.5 formulierte Problem entschieden werden. Eine rationale Sprache $T \subseteq E(\Sigma, I)$, die als nicht erkennbar klassifiziert würde, könnte nicht $E(\Sigma, I)$ sein, denn $E(\Sigma, I)$ ist selber erkennbar. Würde der Algorithmus T als erkennbar einstufen, so könnte nach Satz 3.3.1 das Problem $T = E(\Sigma, I)$ entschieden werden.

Satz 3.3.7 hat entscheidende Konsequenzen für die Arbeit mit erkennbaren bzw. rationalen Trace-Sprachen. Aus Gründen der Anschaulichkeit der möglichen Abläufe werden erkennbare Trace-Sprachen oft in der Form $[L(r)]_I$ mit regulärem Ausdruck r geschrieben. Es stellt sich nun heraus, daß diese Form der Darstellung nicht nur mächtiger ist als die Klasse der erkennbaren Trace-Sprachen, sondern daß es im allgemeinen Fall keinen Algorithmus gibt, der für den Ausdruck r entscheiden kann, ob $[L(r)]_I$ erkennbar ist oder nicht.

Zu wünschen wäre ein Algorithmus, der zu einem regulären Ausdruck r über Σ mit der Unabhängigkeitsrelation I einen regulären Ausdruck r' mit $L(r') = Lin([L(r)]_I)$ berechnet, also die in I festgehaltenen Vertauschungsmöglichkeiten in den Ausdruck r einbaut. Mit Satz 3.3.7 steht fest, daß ein solches Berechnungsverfahren nicht bei allen Eingaben r und I terminiert, weil nicht entschieden werden kann, ob r' existiert.

3.4 Teilklassen der erkennbaren Trace-Sprachen

Ein naheliegender Ansatz für die Definition von Teilklassen der erkennbaren Trace-Sprachen besteht darin, spezielle Typen von EAAs zu betrachten. Nichts anderes machen Duboc ([Dub86]) und Zielonka ([Ziel87]). Duboc untersucht EAAs, deren Teilautomaten unabhängig voneinander arbeiten. Die Teilalphabete müssen nicht disjunkt sein. Die Teilautomaten sind gewöhnliche endliche Automaten, die das Eingabewort unabhängig voneinander lesen, wobei ein Teilautomat die ihm unbekannten Zeichen ignoriert. Die Anfangs- und Endzustandsmengen des Gesamtautomaten sind Kreuzprodukte der Anfangs- bzw. Endzustandsmengen der Teilautomaten. Zielonkas Ansatz ist in dem Sinne allgemeiner, als er im Gegensatz zu Duboc die Endzustandsmenge als beliebige Teilmenge der Zustände des Gesamtautomaten definiert.

Die Untersuchung der Sprachklassen zu diesen Automatentypen hat zum Ergebnis, daß jede erkennbare Trace-Sprache unter Verwendung der Operationen Vereinigung und Synchronisation sowie eines speziellen Homomorphismus aus regulären Wortsprachen gewonnen werden kann. Das Ergebnis nutzt Zielonka für eine algebraische Charakterisierung der Klasse der erkennbaren Trace-Sprachen über beliebigen Alphabeten.

Für die Definition der aus unabhängig arbeitenden Teilautomaten bestehenden EAAs wird zunächst eine Synchronisationsoperation eingeführt, die zwei EAAs zu einem größeren zusammensetzt.

Definition 3.4.1 (Synchronisation von EAAs)

Seien $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ *mit* $\mathcal{P}_i = (\Sigma_i, S_i)$ *und* $\mathcal{A}' = (\mathcal{P}'_{n+1}, \ldots, \mathcal{P}'_{n+m}, \mathcal{I}', \mathcal{D}', \mathcal{F}')$ *mit* $\mathcal{P}'_{n+i} = (\Sigma'_{n+i}, S'_{n+i})$ *zwei EAAs.*
Die Synchronisation von $\mathcal{A}$ *und* $\mathcal{A}'$ *ist*

$$\mathcal{A}'' = \mathcal{A} \| \mathcal{A}' =_{Df} (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{P}'_{n+1}, \ldots, \mathcal{P}'_{n+m}, \mathcal{I}'', \mathcal{D}'', \mathcal{F}''),$$

wobei $\mathcal{I}''$, $\mathcal{D}''$ *und* $\mathcal{F}''$ *folgendermaßen konstruiert werden:*

i) $\mathcal{I}'' =_{Df} \mathcal{I} \times \mathcal{I}'$

ii) Sei $a \in \Sigma'' =_{Df} \Sigma \cup \Sigma'$ *ein Zeichen aus dem Alphabet von* $\mathcal{A}''$.
Für $a \in \Sigma \setminus \Sigma'$ *(bzw.* $a \in \Sigma' \setminus \Sigma$*) wird* $\delta''_a =_{Df} \delta_a$ *(bzw.* $\delta''_a =_{Df} \delta'_a$*) gesetzt.*
Sei in $\mathcal{A}''$ *die Heimat von* $a \in \Sigma \cap \Sigma'$ *die Menge von Indizes* $\{i_1, \ldots, i_k, \ldots, i_l\} \subseteq \{1, \ldots, n+m\}$, $k, l \in I\!N$, *wobei* $i_1, \ldots, i_k$ *die Indizes der Heimatteilautomaten von* a *in* $\mathcal{A}$ *und* $i_{k+1}, \ldots, i_l$ *die Indizes der Heimatteilautomaten von* a *in* $\mathcal{A}'$ *sind. Die Überführungsrelation für* a *in* $\mathcal{A}''$ *ist*

$$\delta''_a(s_{i_1}, \ldots, s_{i_l}) =_{Df} \delta_a(s_{i_1}, \ldots, s_{i_k}) \times \delta'_a(s_{i_{k+1}}, \ldots, s_{i_l})$$

iii) $\mathcal{F}'' =_{Df} \mathcal{F} \times \mathcal{F}'$

Anm.: Das in i), ii) und iii) verwendete kartesische Produkt ist als assoziative Operation zu lesen.

Satz 3.4.2 (Zusammenhang mit Synchronisation von Trace-Sprachen)

Seien $\mathcal{A}$ *und* $\mathcal{A}'$ *zwei EAAs mit den Trace-Systemen* $((\Sigma, I), T(\mathcal{A}))$ *und* $((\Sigma', I'), T(\mathcal{A}'))$.
Sei wie in der Definition $\mathcal{A}'' = \mathcal{A} \| \mathcal{A}'$ *mit* $\Sigma'' = \Sigma_{\mathcal{A}\|\mathcal{A}'}$ *und* $I'' = I_{\mathcal{A}\|\mathcal{A}'}$ *Es gilt:*

$$((\Sigma'', I''), T(\mathcal{A}'')) = ((\Sigma, I), T(\mathcal{A})) \| ((\Sigma', I'), T(\mathcal{A}'))$$

Da die Darstellung dieses Zusammenhangs in [Dub86] von der hier gegebenen stark abweicht, soll der Beweis skizziert werden.

Beweis:

i) $\Sigma''^2 \setminus I'' = \Sigma^2 \setminus I \cup \Sigma'^2 \setminus I'$ ergibt sich ganz einfach aus folgender Überlegung: $a, b \in \Sigma''$ sind in $\mathcal{A}''$ genau dann abhängig, wenn es einen Teilautomaten $\mathcal{P}_i = (\Sigma_i, S_i)$ gibt mit $a, b \in \Sigma_i$. Dann und nur dann gibt es auch einen Teilautomaten mit der gleichen Eigenschaft in $\mathcal{A}$ oder $\mathcal{A}'$. Diese Aussage ist äquivalent zu $(a, b) \in \Sigma^2 \setminus I \vee (a, b) \in \Sigma'^2 \setminus I'$.

ii) Zu zeigen: $T(\mathcal{A}'') = T(\mathcal{A}) \| T(\mathcal{A}')$
$\subseteq$: Sei $[w] \in T(\mathcal{A}'')$. Zu $w = a_1 \ldots a_k \in \Sigma''^*$ gibt es in $\mathcal{A}''$ eine Folge von Zustandsübergängen $s_0 \stackrel{a_1}{\Longrightarrow} s_1, s_1 \stackrel{a_2}{\Longrightarrow} s_2, \ldots, s_{k-1} \stackrel{a_k}{\Longrightarrow} s_k$ mit $s_0 \in \mathcal{I}''$ und $s_k \in \mathcal{F}''$. Für $a_i \notin \Sigma$ ist gemäß Definition 3.4.1 die Relation δ''_{a_i} als δ'_{a_i} definiert, so daß nur Übergänge

in den aus $\mathcal{A}'$ stammenden Teilautomaten von $\mathcal{A}''$ stattfinden. Falls $a_i \in \Sigma \cap \Sigma'$ ist, gibt es Übergänge sowohl in $\mathcal{A}$ als auch in $\mathcal{A}'$, die nach Definition von $\mathcal{D}''$ in $\mathcal{A}''$ unabhängig voneinander ausgeführt werden. Somit ergeben sich in $\mathcal{A}$ globale Zustände $\tilde{s}_{i-1}$ und $\tilde{s}_i$ mit $\tilde{s}_{i-1} \stackrel{a_i}{\Longrightarrow} \tilde{s}_i$ durch Projektion der Zustände s_{i-1} und s_i auf $\mathcal{A}$. Für $a_i \in \Sigma \setminus \Sigma'$ ist derselbe Zusammenhang sofort ersichtlich, weil der a_i-Übergang die aus $\mathcal{A}'$ stammenden Teilautomaten nicht berührt. Die Projektionen der Zustände s_0 und s_k auf $\mathcal{A}$ ergeben nach Definition einen Anfangs- bzw. Endzustand von $\mathcal{A}$, so daß die Folge $s_0, \ldots, s_k$ projiziert auf $\mathcal{A}$ eine akzeptierende Zustandsfolge für $w|_\Sigma$ darstellt, wenn auch eventuell mit einigen leeren Zwischenschritten. Also gilt $w|_\Sigma \in L(\mathcal{A})$ und $[w]|_{(\Sigma,I)} \in T(\mathcal{A})$. Ebenso wird $[w]|_{(\Sigma',I')} \in T(\mathcal{A}')$ gezeigt.
$\supseteq$: Sei $[w] \in T(\mathcal{A}) \| T(\mathcal{A}')$. Aus der Definition der Synchronisation ergibt sich $[w]|_{(\Sigma,I)} \in T(\mathcal{A})$ und $[w]|_{(\Sigma',I')} \in T(\mathcal{A}')$. Somit gilt auch $w|_\Sigma \in L(\mathcal{A})$ und $w|_{\Sigma'} \in L(\mathcal{A}')$. In $\mathcal{A}$ gibt es eine Folge von Zustandsübergängen $s_0, \ldots, s_k$ mit $s_0 \in \mathcal{I}$ und $s_k \in \mathcal{F}$ für $w|_\Sigma$ (analog $s'_0, \ldots, s'_l$ in $\mathcal{A}'$ für $w|_{\Sigma'}$). Aus beiden Folgen kann eine Folge in $\mathcal{A}''$ konstruiert werden, die $w = a_1 \ldots a_n$ akzeptiert: $a_i \in \Sigma \setminus \Sigma'$ bzw. $a_i \in \Sigma' \setminus \Sigma$ ist problemlos, weil ein solches Zeichen nur in einem der beiden Automaten Zustandsübergänge bewirkt. Sei $a_i \in \Sigma \cap \Sigma'$, $s_{j-1} \stackrel{a_i}{\Longrightarrow} s_j$ der Übergang in $\mathcal{A}$ und $s'_{r-1} \stackrel{a_i}{\Longrightarrow} s'_r$ der Übergang in $\mathcal{A}'$. Der a_i-Übergang wirkt sich in $\mathcal{A}$ und $\mathcal{A}'$ nur auf die Teilautomaten in $Dom(a_i)$ aus. Die Definition von $\mathcal{D}''$ stellt sicher, daß die Übergänge in $Dom(a_i)$ von s_{j-1} nach s_j und von s'_{r-1} nach s'_r zu einem Übergang in $\mathcal{A}''$ kombiniert werden können. Somit ist $(s_{j-1}, s'_{r-1}) \stackrel{a_i}{\Longrightarrow} (s_j, s'_r)$ ein globaler Übergang in $\mathcal{A}''$. Weil schließlich (s_0, s'_0) ein Anfangs- und (s_k, s'_l) ein Endzustand von $\mathcal{A}''$ ist, ist die konstruierte Folge eine akzeptierende Rechnung in $\mathcal{A}''$ und $w \in L(\mathcal{A}'')$, also $[w] \in T(\mathcal{A}'')$. □

Satz 3.4.2 soll nun dazu benutzt werden, Dubocs Automatentyp und eine Teilklasse der erkennbaren Trace-Sprachen zu definieren.

Definition 3.4.3 (Modularisierter EAA)

Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ ein EAA mit $\mathcal{P}_i = (\Sigma_i, S_i)$.
$\mathcal{A}$ heißt modularisiert

$$\Longleftrightarrow_{Df} \quad \begin{array}{l} \textit{Es gibt EAAs } \mathcal{A}_1, \ldots, \mathcal{A}_n \textit{ in Normalform mit den Alphabeten} \\ (\Sigma_1, \emptyset), \ldots, (\Sigma_n, \emptyset) \textit{ und der Eigenschaft } \mathcal{A} = \mathcal{A}_1 \| \ldots \| \mathcal{A}_n \end{array}$$

Ein EAA in Normalform mit dem Alphabet $(\Sigma_i, \emptyset)$ besteht aus nur einem Teilautomaten, ist also ein gewöhnlicher endlicher Automat.

Definition 3.4.4 (Synchronisierte Trace-Sprache, Klasse $SYN(\Sigma, I)$)

Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache über dem Alphabet Σ mit der Unabhängigkeitsrelation I. Seien Σ_i, $i \in \{1, \ldots, n\}$, die maximalen Cliquen der Abhängigkeitsrelation D.

$$T \textit{ heißt synchronisiert} \Longleftrightarrow_{Df} \; T = T|_{(\Sigma_1,\emptyset)} \| \ldots \| T|_{(\Sigma_n,\emptyset)}$$

Die Klasse der erkennbaren synchronisierten Trace-Sprachen über (Σ, I) wird mit $SYN(\Sigma, I)$ bezeichnet.

Korollar 3.4.5 (Synchronisierte Sprache u. modularisierter Automat, [Dub86])

Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache.

$$T \text{ ist synchronisiert} \iff \text{Es gibt einen modularisierten EAA } \mathcal{A} \text{ mit } T(\mathcal{A}) = T.$$

Folgerung: Da im modularisierten Automaten mit Hilfe der Potenzautomatenkonstruktion zu den Teilautomaten getrennt deterministische Automaten angegeben werden können, gibt es zu jeder synchronisierten Trace-Sprache T einen deterministischen modularisierten EAA $\mathcal{A}$ mit $T(\mathcal{A}) = T$.

Die synchronisierten Trace-Sprachen sind bezüglich Mengenvereinigung nicht abgeschlossen, wie das folgende einfache **Beispiel** zeigt:
Sei $\Sigma = \{a, b\}$ und $I = \{(a, b), (b, a)\}$.
Die Trace-Sprachen $T_a = \{[a]\}$ und $T_b = \{[b]\}$ sind synchronisiert, denn es ist

$$T_a = \{[a]\}|_{\{a\}} \| \{[a]\}|_{\{b\}} = \{[a]\} \| \{[\varepsilon]\} = \{[a]\}.$$

Andererseits gilt

$$(T_a \cup T_b)|_{\{a\}} \| (T_a \cup T_b)|_{\{b\}} = \{[a], [\varepsilon]\} \| \{[\varepsilon], [b]\} = \{[\varepsilon], [a], [b], [ab]\} \neq T_a \cup T_b$$

Der Abschluß gegen Vereinigung wird erreicht, wenn in den modularisierten EAAs globale Anfangs- und Endzustandsmengen zugelassen werden. Zielonka nennt diesen Automatentyp „schwach kooperierend“.

Definition 3.4.6 (Schwach kooperierender EAA)

Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ ein EAA.
$\mathcal{A}$ heißt schwach kooperierend, wenn es eine Menge $\{\delta_i | i \in \{1, \ldots, n\}\}$ von Übergangsrelationen $\delta_i : S_i \times \Sigma_i \longrightarrow \wp(S_i)$ gibt, mit der sich die $\delta_a, a \in \Sigma$, folgendermaßen darstellen lassen ($Dom(a) = \{i_1, \ldots, i_k\}$):

$$\bigwedge_{(s_{i_1}, \ldots, s_{i_k}) \in S_{i_1} \times \ldots \times S_{i_k}} \delta_a(s_{i_1}, \ldots, s_{i_k}) = \prod_{j \in \{i_1, \ldots, i_k\}} \delta_j(s_j, a)$$

Da die Synchronisation von EAAs über das kartesische Produkt der Anfangs- und Endzustandsmengen sowie der Übergangsrelationen definiert ist (Def. 3.4.1) und modularisierte EAAs durch Synchronisation von gewöhnlichen endlichen Automaten entstehen (Def. 3.4.3), läßt sich für modularisierte EAAs eine analoge Charakterisierung angeben.

In Analogie zu Satz 2.6.10 für allgemeine EAAs kann für schwach kooperierende EAAs formuliert werden (vgl. [Ziel87]):

Satz 3.4.7 (Deterministischer schwach kooperierender EAA in Normalform)

Sei $\mathcal{A}$ ein schwach kooperierender EAA mit Alphabet $\Sigma_{\mathcal{A}}$ und Unabhängigkeitsrelation $I_{\mathcal{A}}$.
Zu $\mathcal{A}$ gibt es einen deterministischen schwach kooperierenden EAA $\mathcal{A}'$ in Normalform mit $\Sigma_{\mathcal{A}'} = \Sigma_{\mathcal{A}}$, $I_{\mathcal{A}'} = I_{\mathcal{A}}$ und $T(\mathcal{A}') = T(\mathcal{A})$.

Im Unterschied zu allgemeinen EAAs kann die Konstruktion hier direkt auf dem Automaten ausgeführt werden. Im ersten Schritt wird der Automat in Normalform bestimmt, indem Teilautomaten zusammengefaßt werden, die auf der gleichen maximalen Clique von abhängigen Zeichen arbeiten. Anschließend wird zu jedem Teilautomaten des EAAs in Normalform über die Potenzautomatenkonstruktion ein deterministischer Automat konstruiert.

Definition 3.4.8 (Klasse $SKA(\Sigma, I)$)

$SKA(\Sigma, I)$ ist die Klasse der erkennbaren Trace-Sprachen über (Σ, I), die von schwach kooperierenden EAAs mit Alphabet Σ und Unabhängigkeitsrelation I erkannt werden können.

Satz 3.4.9 (Abschlußeigenschaften von $SKA(\Sigma, I)$, [Ziel87])

i) Sei (Σ, I) ein festes Alphabet mit Unabhängigkeitsrelation. Die Klasse $SKA(\Sigma, I)$ ist abgeschlossen bezüglich Durchschnitt und Vereinigung von Trace-Sprachen.

ii) Seien $T_1 \in SKA(\Sigma_1, I_1)$ und $T_2 \in SKA(\Sigma_2, I_2)$ zwei Trace-Sprachen. Die Synchronisation $T_1 \| T_2$ ist Trace-Sprache eines schwach kooperierenden EAAs.

Aussage i) ergibt sich, indem auf die Teilautomaten der schwach kooperierenden EAAs in Normalform die Konstruktionsverfahren für Schnitt und Vereinigung von endlichen Automaten getrennt angewendet werden. Aussage ii) folgt aus dem Zusammenhang zwischen Synchronisation von EAAs und Synchronisation von Trace-Sprachen (Satz 3.4.2) sowie der leicht einsehbaren Tatsache, daß die Synchronisation von schwach kooperierenden EAAs wieder einen schwach kooperierenden EAA ergibt.

Lemma 3.4.10 (Beziehung modularisierter und schwach kooperierender EAA)

i) Jeder modularisierte EAA ist schwach kooperierend.

ii) Ein schwach kooperierender EAA $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\mathcal{I} = \prod_{j \in \{1,\ldots,n\}} \mathcal{I}_j$, $\mathcal{F} = \prod_{j \in \{1,\ldots,n\}} \mathcal{F}_j$ und $\mathcal{I}_j, \mathcal{F}_j \subseteq S_j$ lokalen Anfangs- und Endzustandsmengen ist modularisiert.

Weil jede einelementige Trace-Sprache synchronisiert ist, gibt es zu jedem Trace t einen modularisierten EAA, der $\{t\}$ erkennt. Endlich viele modularisierte EAAs, die durch einen gemeinsamen Anfangszustand verbunden sind, ergeben einen schwach kooperierenden EAA. Für endliche Trace-Sprachen gilt nach [Ziel87] also

Lemma 3.4.11 (Alle endlichen Trace-Sprachen in $SKA(\Sigma, I)$)

Sei $T \subseteq E(\Sigma, I)$ eine endliche Trace-Sprache über (Σ, I).
Es gilt: $T \in SKA(\Sigma, I)$.

Ein schwach kooperierender EAA mit einem Anfangs- und einem Endzustand ist modularisiert. Bei mehreren Anfangs- und Endzuständen definiert jedes Paar aus $\mathcal{I} \times \mathcal{F}$ einen eigenen modularisierten EAA. Weil $\mathcal{I}$ und $\mathcal{F}$ wie alle Zustandsmengen endlich sind und modularisierte EAAs synchronisierte Trace-Sprachen erkennen, ist jede Trace-Sprache $T \in SKA(\Sigma, I)$ eine Vereinigung von endlich vielen synchronisierten Trace-Sprachen über (Σ, I). Es gilt (vgl. [Dub86],[Ziel87]):

Satz 3.4.12 (Darstellung von $T \in SKA(\Sigma, I)$)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation, seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen Cliquen der Abhängigkeitsrelation und sei $T \in SKA(\Sigma, I)$.
Die Trace-Sprache T läßt sich schreiben als

$$T = \bigcup_{i=1}^{k} (L_{i,1} \| \ldots \| L_{i,n})$$

mit regulären Wortsprachen $L_{i,j} \subseteq \Sigma_j^*$ und $k \in \mathbb{N}$.
(Die Synchronisation von Wortsprachen soll als Synchronisation von Trace-Sprachen über Alphabeten mit leerer Unabhängigkeitsrelation verstanden werden.)

Diese Ergebnis von Duboc und Zielonka kann auf die volle Klasse der erkennbaren Trace-Sprachen über (Σ, I) übertragen werden, wenn ein besonderer Homomorphismus ins Spiel gebracht wird.

Definition 3.4.13 (I-erhaltender elementarer Homomorphismus)

Seien (Σ, I) und (Σ', I') zwei Alphabete mit Unabhängigkeitsrelation.
Sei $h : E(\Sigma', I') \longrightarrow E(\Sigma, I)$ ein Homomorphismus.
h heißt elementar und I-erhaltend, wenn gilt:

i) h bildet Zeichen auf Zeichen ab und ist surjektiv: $h(\Sigma') = \Sigma$

ii) h respektiert die Unabhängigkeitsrelation:

$$\bigwedge_{a,b \in \Sigma'} (a, b) \in I' \iff (h(a), h(b)) \in I$$

Ein I-erhaltender elementarer Homomorphismus stellt eine enge Beziehung zwischen den Alphabeten mit Unabhängigkeitsrelation (Σ, I) und (Σ', I') her. Anschaulich entsteht (Σ', I'), indem in (Σ, I) Zeichen vervielfältigt werden. Zeichen in Σ', die h auf das gleiche Original in Σ abbildet, werden als so eng verwandt angesehen, daß sie als abhängig definiert werden. Ansonsten wird die Unabhängigkeitsrelation auf Σ' wie in (Σ, I) definiert. Die maximalen D-Cliquen in (Σ, I) und (Σ', I') entsprechen sich direkt.

In [Dub86] und [Ziel87] wird mit Hilfe dieses Homomorphismus der folgende Satz formuliert:

Satz 3.4.14 (Darstellung erkennbarer Trace-Sprachen)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ seine maximalen D-Cliquen. Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache. Zu T gibt es

i) ein Alphabet mit Unabhängigkeitsrelation (Σ', I') mit den maximalen D-Cliquen $\Sigma'_1, \ldots, \Sigma'_n$,

ii) einen I-erhaltenden elementaren Homomorphismus $h : E(\Sigma', I') \longrightarrow E(\Sigma, I)$,

iii) eine Zahl $k \in \mathbb{N}$ und

iv) eine Familie von regulären Wortsprachen $L_{i,j} \subseteq {\Sigma'_j}^*, \;\; j = 1, \ldots, n, \;\; i = 1, \ldots, k$,

so daß T dargestellt werden kann als

$$T = h\left(\bigcup_{i=1}^{k} (L_{i,1} \| \ldots \| L_{i,n})\right)$$

Zum Beweis des Satzes betrachtet man einen EAA $\mathcal{A}_T$ für die erkennbare Trace-Sprache T. Zu $\mathcal{A}_T$ wird ein neuer EAA $\mathcal{A}'$ konstruiert, der die Zustandsmengen von $\mathcal{A}_T$ übernimmt, aber auf einem anderen Alphabet arbeitet. Dieses neue Alphabet Σ' ist die Menge aller Zustandsübergänge in $\mathcal{A}_T$. Der Kontext, in dem ein Zeichen einen Übergang in $\mathcal{A}_T$ bewirkt, wird so in das neue Alphabet hineincodiert. Umgekehrt ordnet die Abbildung h jedem a-Übergang im neuen Automaten das Zeichen $a \in \Sigma$ zu. Im Automaten $\mathcal{A}'$ steht jedes Zeichen für den eigenen Zustandsübergang, so daß die Übergangsrelation für jeden Automaten getrennt definiert werden kann. Der konstruierte EAA $\mathcal{A}'$ ist demnach schwach kooperierend. Die Trace-Sprache $T(\mathcal{A}')$ kann nach Satz 3.4.12 als $\bigcup_{i=1}^{k} (L_{i,1} \| \ldots \| L_{i,n})$ dargestellt werden.

Weil die Klasse der erkennbaren Trace-Sprachen abgeschlossen ist bezüglich der Anwendung von I-erhaltenden elementaren Homomorphismen, kann nun eine neue Charakterisierung dieser Sprachklasse gegeben werden:

Satz 3.4.15 (Charakterisierung der erkennbaren Trace-Sprachen, [Ziel87])

Die Klasse der erkennbaren Trace-Sprachen ist die kleinste Klasse $\mathcal{R}$ von Sprachen mit folgenden Eigenschaften:

i) Jede reguläre Wortsprache gehört zu $\mathcal{R}$.

ii) $\mathcal{R}$ ist abgeschlossen bezüglich Synchronisation von Trace-Sprachen.

iii) $\mathcal{R}$ ist abgeschlossen bezüglich Vereinigung von Trace-Sprachen, die über dem gleichen (Σ, I) definiert sind.

iv) Für jeden I-erhaltenden elementaren Homomorphismus $h : E(\Sigma', I') \longrightarrow E(\Sigma, I)$ und für jede Trace-Sprache $T \subseteq E(\Sigma', I')$ gilt:

$$T \in \mathcal{R} \Rightarrow h(T) \in \mathcal{R}$$

3.5 EAAs und erkennbare Trace-Sprachen

Der Begriff der Erkennbarkeit bezieht sich wie im Anschluß an Korollar 2.4.8 (T erkennbar gdw. Lin(T) regulär) schon erklärt wurde auf beliebige endliche Automaten. Ein Vergleich von EAAs mit gewöhnlichen endlichen Automaten ergibt die Entsprechung für $I = \emptyset$, während sie für $I \neq \emptyset$ nur reguläre Sprachen $L \subseteq \Sigma^*$ mit der Eigenschaft $L = Lin([L]_I)$ erkennen können.

Zielonka weist in [Ziel87] nach, daß jede reguläre Wortsprache mit dieser Eigenschaft von einem EAA erkannt werden kann. Somit gilt für erkennbare Trace-Sprachen der folgende Satz:

Satz 3.5.1 (EAA-Charakterisierung der erkennbaren Trace-Sprachen)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über dem Alphabet Σ mit der Unabhängigkeitsrelation I.

$$T \text{ ist erkennbar} \iff \text{Es gibt einen deterministischen EAA in Normalform mit } I_{\mathcal{A}} = I \text{ und } T(\mathcal{A}) = T$$

Zielonka führt den Beweis konstruktiv. Von den endlich vielen Klassen der syntaktischen Kongruenz von T gelangt er durch Verfeinerung der Kongruenzklassen zu den globalen Zuständen des EAA. Die Übergänge zwischen den globalen Zuständen sind vollständig definiert und deterministisch. Im letzten Schritt der Konstruktion werden die globalen Zustände in lokale Zustände zerlegt (siehe Abschnitt 4.1).

In [CMZ93] wird die Konstruktion eines deterministischen zellulären EAAs zu einer erkennbaren Trace-Sprache angegeben. Dabei wird der Begriff der asynchronen Abbildung benutzt (vgl. [CM88]):
Eine Abbildung $\rho : E(\Sigma, I) \to M$ (M beliebig) heißt *asynchron*, wenn

i) für jedes $t \in E(\Sigma, I)$ und alle Anfangsstücke $P_\alpha(t)$ und $P_\beta(t)$, $\alpha, \beta \subseteq \Sigma$, der Wert $\rho(P_{\alpha\cup\beta}(t))$ durch die Werte $\rho(P_\alpha(t))$ und $\rho(P_\beta(t))$ bestimmt ist,
($P_\alpha(t)$, $\alpha \subseteq \Sigma$, ist das kürzeste Anfangsstück von t, das alle Vorkommen von Zeichen aus α in t enthält.)

ii) für jedes $t \in E(\Sigma, I)$ und jedes Zeichen $a \in \Sigma$ der Wert $\rho(P_{\{a\}}(t[a]))$ durch den Wert $\rho(P_{D(a)}(t))$ eindeutig bestimmt ist.
($D(a) =_{Df} \{b \in \Sigma | (a, b) \in D\}$)

In der Konstruktion wird eine Abbildung $\rho : E(\Sigma, I) \to M$ zu einer erkennbaren Trace-Sprache definiert. Die genannten Eigenschaften werden nachgewiesen. Die Menge der Globalzustände des zellulären EAAs ergeben sich dann als $\rho(E(\Sigma, I))$. Aus den Eigenschaften der asynchronen Abbildung läßt sich die Zerlegbarkeit der globalen in lokale Zustände und die Korrektheit der Konstruktion schnell ableiten.

Korollar 3.5.2 (Existenz eines deterministischen EAAs)

Sei $\mathcal{A}$ ein endlicher asynchroner Automat. Zu $\mathcal{A}$ gibt es einen deterministischen EAA $\mathcal{A}'$ mit $I_{\mathcal{A}'} = I_{\mathcal{A}}$ und $T(\mathcal{A}') = T(\mathcal{A})$.

Für die effektive Konstruktion eines deterministischen zu einem nichtdeterministischen EAA kann Zielonkas Beweis herangezogen werden. Direktere Verfahren werden in [KMS94] und [Mus94] präsentiert. In [Mus94] wird für die Determinisierung eines zellulären EAAs eine Verallgemeinerung der von gewöhnlichen endlichen Automaten bekannten Potenzautomatenkonstruktion (vgl. [Brau84]) benutzt. Es werden zwei Abbildungen definiert:

i) $\nu : E(\Sigma, I) \to \{0, \ldots, |\Sigma|\}^{\Sigma\times\Sigma}$ ordnet jedem Trace t eine ganzzahlige $\Sigma \times \Sigma$-Matrix zu, mit deren Hilfe aus Anfangsstücken $P_\alpha(t)$ und $P_\beta(t)$, $\alpha, \beta \subseteq \Sigma$, das Anfangsstück $P_{\alpha\cup\beta}(t)$ bestimmt werden kann. Die Matrix gibt die relative Abfolge der „letzten“ Zeichenvorkommen in t an, so daß $P_\alpha(t)$ und $P_\beta(t)$ in Beziehung gesetzt werden können. Eine ähnliche Abbildung definiert Zielonka mit $label_t$ in [Ziel87] (vgl. Abschnitt 4.1).

ii) Die zweite Abbildung $\rho : E(\Sigma, I) \to \wp(S^\Sigma)$ (S: Menge der Globalzustände des gegebenen zellulären EAAs) wird durch den nichtdeterministischen Automaten definiert. Statt nur festzuhalten, welche Globalzustände mit einem Trace t nichtdeterministisch erreicht werden können, wird mit ρ für jede mit einem Anfangszustand beginnende Zustandsfolge zu t protokolliert, welcher Globalzustand mit $P_{\{a\}}(t)$ für alle $a \in \Sigma$ in dieser Zustandsfolge erreicht wird.

Muscholl zeigt: Die Abbildung (ν, ρ) ist asynchron, definiert also einen einen deterministischen zellulären EAA, der zum gegebenen nichtdeterministischen Automaten äquivalent ist.

In Korollar 3.5.2 ergibt sich für $I_{\mathcal{A}} = \emptyset$ das aus der Theorie der gewöhnlichen endlichen Automaten bekannte Resultat. Im folgenden wird gezeigt, daß sich andere Ergebnisse nicht auf EAAs mit $I_{\mathcal{A}} \neq \emptyset$ verallgemeinern lassen.

3.6 Deterministische und sichere EAAs

Für gewöhnliche endliche Automaten kann einfach bewiesen werden, daß ein deterministischer Automat reduziert um die Zustände, von denen aus kein Endzustand erreichbar ist, noch dieselbe Sprache erkennt. Es ist also möglich, Sackgassenzustände im unvollständigen Automaten zu vermeiden bzw. im vollständigen Automaten in einem einzigen Zustand zusammenzufassen. Weil die Zustände in asynchronen Automaten verteilt sind, lassen sie sich auf diese Weise nicht reduzieren.

Beispiel: Sei $\Sigma = \{a, b, c\}$ und $I = \{(b, c), (c, b)\}$.
Ein EAA für die erkennbare Trace-Sprache $T = \{[aba], [aca]\}$ ist $\mathcal{A} = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{a, c\}$, $\mathcal{I} = \{(s_1^0, s_2^0)\}$, $\mathcal{F} = \{(s_1^3, s_2^3)\}$ und S_1, S_2, $\mathcal{D}$ wie im Petrinetz in Abbildung 3.3 dargestellt.

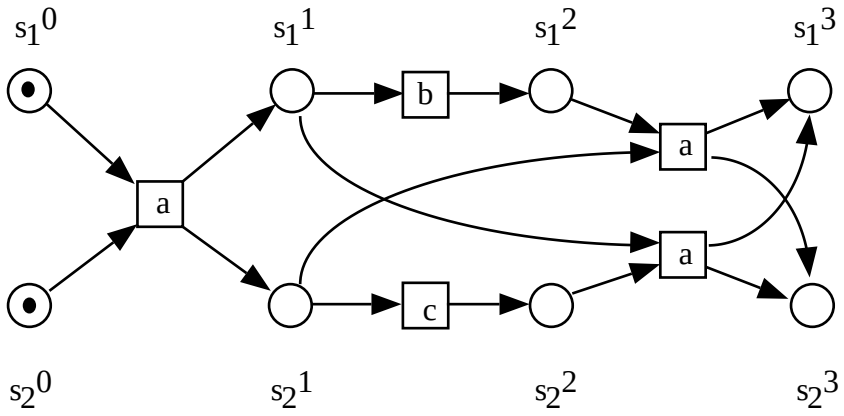

Abbildung 3.3: EAA für $T = \{[aba], [aca]\}$

Der Automat $\mathcal{A}$ ist deterministisch, aber nicht sicher. Mit $[abc]$ wird der Zustand (s_1^2, s_2^2) erreicht, von dem aus keine Fortsetzung möglich ist.

Die Konstruktion eines deterministischen EAA zur Sprache $T = \{[aba], [aca]\}$ führt zwangsläufig zu einem unsicheren Automaten. Nach Verarbeitung von $[a]$ ist der deterministische Automat in einem eindeutig bestimmten Zustand, hier (s_1^1, s_2^1). Von diesem Zustand aus müssen voneinander unabhängige Übergänge $[b]$ und $[c]$ möglich sein. Ein mit $[abc]$ erreichbarer Zustand läßt sich so nicht vermeiden.

Nach [Ziel89] gibt es zu jeder erkennbaren Trace-Sprache einen sicheren EAA, der, wie das Beispiel zeigt, nicht immer deterministisch sein kann:

Satz 3.6.1 (Existenz eines sicheren EAAs zu $T \subseteq E(\Sigma, I)$)

Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache.
Zu T gibt es einen sicheren EAA $\mathcal{A}$ mit $I_{\mathcal{A}} = I$ und $T(\mathcal{A}) = T$.

(Zielonka beweist den Satz für zelluläre EAAs, fügt aber hinzu, daß das Ergebnis auf EAAs übertragbar ist.)

Ein sicherer nichtdeterministischer Automat für die Trace-Sprache $T = \{[aba], [aca]\}$ aus dem obigen Beispiel ergibt sich, wenn in dem angegebenen Petrinetz die Pfeile umgedreht werden und der Anfangszustand mit dem Endzustand vertauscht wird.

3.7 Minimale EAAs

Mit dem Problem der Eindeutigkeit eines minimalen EAAs zu einer erkennbaren Trace-Sprache haben sich Bruschi, Pighizzini und Sabadini in [BPS88] auseinandergesetzt. Eine überarbeitete und erweiterte Fassung dieses Artikels ist [BPS94]. Um EAAs in ihrer Größe vergleichen zu können, wird dort zunächst ein Homomorphismus zwischen EAAs definiert.

Definition 3.7.1 (Homomorphismus zwischen EAAs)

Seien $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ *mit* $\mathcal{P}_i = (\Sigma_i, S_i),\ i \in \{1, \ldots, n\}$
und $\mathcal{A}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, \mathcal{I}', \mathcal{D}', \mathcal{F}')$ *mit* $\mathcal{P}'_i = (\Sigma_i, S'_i),\ i \in \{1, \ldots, n\}$
zwei EAAs über demselben Alphabet mit Unabhängigkeitsrelation.
Ein Homomorphismus $\phi : \mathcal{A} \longrightarrow \mathcal{A}'$ *ist eine Familie von Funktionen* $\phi_i : S_i \longrightarrow S'_i, i \in \{1, \ldots, n\}$, *mit den folgenden Eigenschaften:*

i)

$$\bigwedge_{(s_1,\ldots,s_n)\in\mathcal{I}} \bigvee_{(s'_1,\ldots,s'_n)\in\mathcal{I}'} \bigwedge_{i\in\{1,\ldots,n\}} \phi_i(s_i) = s'_i$$

ii) Für alle $a \in \Sigma$ *mit* $Dom(a) = \{i_1, \ldots, i_k\}$ *gilt:*

$$\bigwedge_{(s_{i_1},\ldots,s_{i_k})\in S_{i_1}\times\ldots\times S_{i_k}} \bigwedge_{j\in\{1,\ldots,k\}} \phi_{i_j}\Big(\pi_{i_j}(\delta_a(s_{i_1}, \ldots, s_{i_k}))\Big) = \pi_{i_j}(\delta'_a(\phi_{i_1}(s_{i_1}), \ldots, \phi_{i_k}(s_{i_k}))\Big)$$

iii)

$$\bigwedge_{(s_1,\ldots,s_n)\in\mathcal{F}} \bigvee_{(s'_1,\ldots,s'_n)\in\mathcal{F}'} \bigwedge_{i\in\{1,\ldots,n\}} \phi_i(s_i) = s'_i$$

Anm.: $\pi_i(x_1, \ldots, x_m) =_{Df}\ x_i$ *für* $i \in \{1, \ldots, m\}$

Definition 3.7.2 (Vergleich von EAAs)

Seien $\mathcal{A}$ *und* $\mathcal{A}'$ *zwei EAAs wie in Definition 3.7.1 beschrieben.*

i) $\mathcal{A}'$ *ist höchstens so groß wie* $\mathcal{A}$ *(geschrieben* $\mathcal{A}' \leq \mathcal{A}$*)*
$\iff_{Df}$ *Es gibt einen Homomorphismus* $\phi : \mathcal{A} \longrightarrow \mathcal{A}'$

ii) $\mathcal{A}$ *ist minimal*
$\iff_{Df}$ *Für jeden EAA* $\mathcal{A}'$ *mit* $\mathcal{A}' \leq \mathcal{A}$ *gilt* $\mathcal{A} \leq \mathcal{A}'$ *und in* $\mathcal{A}$ *ist jeder lokale Zustand* s_i *erreichbar:*

$$\bigwedge_{i\in\{1,\ldots,n\}} \bigwedge_{s_i\in S_i} \bigvee_{s^0\in\mathcal{I}} \bigvee_{t\in E(\Sigma,I)} s^0 \stackrel{t}{\Longrightarrow} (s_1, \ldots, s_i, \ldots, s_n)$$

Satz 3.7.3 (Eindeutige Existenz eines minimalen EAA, [BPS88])

Sei (Σ, I) *ein Alphabet mit Unabhängigkeitsrelation. Es gilt:*

Für jede erkennbare Trace-Sprache $T \subseteq E(\Sigma, I)$ *gibt es einen bis auf Isomorphie eindeutigen minimalen EAA* $\mathcal{A}$ *mit* $I_{\mathcal{A}} = I$ *und* $T(\mathcal{A}) = T$
$\iff$ $D = \Sigma^2 \setminus I$ *ist transitiv*

Im Beweis geben Bruschi, Pighizzini und Sabadini zwei nichtisomorphe EAAs zu einer Trace-Sprache über einem Alphabet mit nichttransitiver Abhängigkeitsrelation an, die nicht minimiert werden können.

In der Rückrichtung zeigen sie, daß EAAs zu Trace-Sprachen über Alphabeten mit transitiver Abhängigkeitsrelation eine besonders einfache Struktur haben. Ihre Teilautomaten sind gewöhnliche endliche Automaten, die nicht durch die Überführungsrelation synchronisiert werden, auf denen die Endzustände aber noch global definiert sind. Die Teilautomaten können jeweils mit Hilfe der Nerode-Kongruenz als Minimalautomaten konstruiert werden. Der entstehende EAA ist minimal im Sinne der Definition 3.7.2.

Kapitel 4

Die Konstruktion von EAAs

In diesem Kapitel werden die aus der Literatur bekannten Konstruktionsverfahren für endliche asynchrone Automaten vorgestellt. Den Anfang macht Zielonkas Konstruktion, die im Mittelpunkt des Beweises des Satzes 3.5.1 steht (vgl. [Ziel87]). Die Konstruktion ist für beliebige erkennbare Trace-Sprachen aufwendig und vereinfacht sich, wenn erkennbare Trace-Sprachen über *triangulierten Alphabeten* betrachtet werden. Diese einfachere Konstruktion wird in Abschnitt 4.2 erklärt. In Abschnitt 4.3 wird Pighizzinis Verfahren behandelt, mit dem zu einer durch einen regulären Ausdruck gegebenen Trace-Sprache ein nichtdeterministischer EAA konstruiert werden kann.

4.1 Zielonkas Konstruktion

Die Konstruktion eines Automaten zu einer Sprache bedeutet einen Wechsel in ihrer Repräsentation. Entscheidend ist folglich, auf welcher Darstellung der Sprache das Konstruktionsverfahren aufsetzt. Zielonka geht in seiner Konstruktion von einer Repräsentation der Trace-Sprache durch die Klassen ihrer syntaktischen Kongruenz aus. Eine Verfeinerung der Kongruenzklassen ergibt die Zustände eines gewöhnlichen endlichen Automaten für die zugrundeliegende Wortsprache. Die Verfeinerung ist so gewählt, daß der endliche Automat gemäß Definition 2.6.1 (EAA) in Teilautomaten zerlegt werden kann.

Für die Definition der verfeinerten Äquivalenzrelation müssen eine Reihe von Begriffen eingeführt werden. Sei in den folgenden Definitionen (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und seien $\Sigma_i \subseteq \Sigma$, $i = 1, \ldots, n$, die maximalen Cliquen abhängiger Zeichen in (Σ, I).

Definition 4.1.1 ($last_i^i(t)$)

Sei $t \in E(\Sigma, I)$ ein Trace. $last_i^i(t)$, $i \in \{1, \ldots, n\}$, ist das letzte Vorkommen eines Zeichens aus Σ_i in t, d. h. $last_i^i(t)$ hat folgende Eigenschaften:

i) $last_i^i(t) = (a, n)$ mit $(a, n) \in Vor(t)$ und $a \in \Sigma_i$

ii) $\bigwedge_{(b,m)\in Vor(t)} b \in \Sigma_i \Rightarrow (b,m) \leq_t last_i^i(t)$

$last_i^i(t)$ existiert nur dann, wenn im Trace t ein Zeichen aus Σ_i vorkommt. In diesem Fall ist es eindeutig bestimmt, weil alle Vorkommen von Zeichen aus Σ_i aufgrund ihrer Abhängigkeit in t total geordnet sind.

Beispiel:
$\Sigma = \{a,b,c,d,e\}$ mit $I = \Sigma^2 \setminus \{(a,b),(b,a),(b,c),(c,b),(c,d),(d,c),(d,e),(e,d)\}$
Die maximalen Cliquen der Abhängigkeitsrelation sind $\Sigma_1 = \{a,b\}$, $\Sigma_2 = \{b,c\}$, $\Sigma_3 = \{c,d\}$, $\Sigma_4 = \{d,e\}$.
Sei $t = [cbac]$. Die Menge der Vorkommen ist $Vor(t) = \{(c,1),(b,1),(a,1),(c,2)\}$. Das Hasse-Diagramm zu $Ord(t)$ zeigt Abb. 4.1.

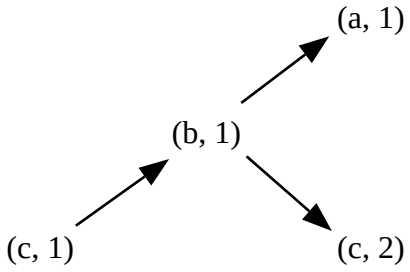

Abbildung 4.1: $Ord([cbac])$

Die $last_i^i(t)$ sind $last_1^1(t) = (a,1)$, $last_2^2(t) = last_3^3(t) = (c,2)$, $last_4^4(t)$ undefiniert.

Definition 4.1.2 (i-Anfangsstück)

Sei $t \in E(\Sigma, I)$ ein Trace und $i \in \{1,\ldots,n\}$.
$P_i(t) \in Pref(t)$ ist das Anfangsstück von t, das $last_i^i(t)$ und alle vor $last_i^i(t)$ auftretenden Vorkommen enthält. Es ist charakterisiert durch die Eigenschaft

$$Vor(P_i(t)) = \{(a,n) \in Vor(t) | (a,n) \leq_t last_i^i(t)\}.$$

Die Menge der Vorkommen auf der rechten Seite der Gleichung ist bezüglich $\leq_t$ nach unten abgeschlossen und daher die Menge der Vorkommen eines Anfangsstücks von t. Falls $last_i^i(t)$ nicht existiert, steht auf der rechten Seite der Gleichung die leere Menge und $P_i(t)$ ist der leere Trace.

Beispiel: Seien (Σ, I) und t wie oben gesetzt. Dann gilt:

$P_1(t) = [cba]$, $P_2(t) = P_3(t) = [cbc]$, $P_4(t) = [\varepsilon]$.

Definition 4.1.3 (α-Anfangsstück $P_\alpha(t)$)

Sei $t \in E(\Sigma, I)$ ein Trace und $\alpha \subseteq \{1,\ldots,n\}$.
$P_\alpha(t)$ ist das Anfangsstück von t, das durch Vereinigung der Zeichenvorkommen der $P_i(t)$, $i \in \alpha$, entsteht:

$$Vor(P_\alpha(t)) = \bigcup_{i\in\alpha} Vor(P_i(t))$$

Es gilt stets $P_\emptyset(t) = [\varepsilon]$ und $P_{\{1,\ldots,n\}}(t) = t$.

Im Beispiel ist $P_{\{1,4\}}(t) = [cba]$ und $P_{\{1,3\}}(t) = t$.

Definition 4.1.4 ($last^i_j(t)$)

Sei $t \in E(\Sigma, I)$ ein Trace und $i, j \in \{1, \ldots, n\}$.

i) $last^i_j(t)$ wird definiert als $last^i_j(t) =_{Df} last^j_j(P_i(t))$.

ii) $LAST(t)$ ist die Menge aller $last^i_j(t)$:

$$LAST(t) =_{Df} \{last^i_j(t) | i, j \in \{1, \ldots, n\}\}$$

Im Beispiel:
$last^1_2(t) = (b, 1)$, $last^1_3(t) = (c, 1)$, $last^1_4(t)$ nicht definiert
$last^2_1(t) = (b, 1)$, $last^2_3(t) = (c, 2)$, $last^2_4(t)$ nicht definiert
$last^3_1(t) = (b, 1)$, $last^3_2(t) = (c, 2)$, $last^3_4(t)$ nicht definiert
$last^4_1(t)$, $last^4_2(t)$, $last^4_3(t)$ nicht definiert

Es wird nun eine Beschriftung der Zeichenvorkommen und darauf aufbauend eine Äquivalenzrelation auf den Traces definiert.

Definition 4.1.5 (Beschriftung der Zeichenvorkommen)

Sei $t \in E(\Sigma, I)$ ein Trace. Die Funktion $label_t : Vor(t) \to \Sigma \times \mathbb{N}$ ordnet nach dem folgenden Algorithmus jedem Zeichenvorkommen in t eine Beschriftung zu:

i) Dem ersten Vorkommen eines Zeichens $a \in \Sigma$ in t wird die Beschriftung $(a, 1)$ gegeben: $label_t(a, 1) =_{Df} (a, 1)$

ii) Seien die ersten $k-1$ Vorkommen des Zeichens a in t bereits beschriftet ($k > 1$). Sei $P_{(a,k)}(t)$ das Anfangsstück von t mit $Vor(P_{(a,k)}(t)) = \{(b, m) \in Vor(t) | (b, m) \leq_t (a, k)\}$. $P_{(a,k)}(t)$ ist das kürzeste Anfangsstück von t, das (a, k) enthält. Sei

$$G_{(a,k)} =_{Df} LAST(P_{(a,k)}(t)) \cap \{(a, 1), \ldots, (a, k-1)\}$$

und sei $m \in \mathbb{N}$ die kleinste natürliche Zahl mit

$$\bigwedge_{(a,l) \in G_{(a,k)}} label_t(a, l) \neq (a, m).$$

Setze $label_t(a, k) =_{Df} (a, m)$.

Schritt ii) bedeutet anschaulich folgendes: Die Zeichenvorkommen werden in der Reihenfolge der Halbordnung $Ord(t)$ beschriftet, also etwa indem ein Repräsentant von t durchlaufen wird. Zu jedem Vorkommen (a,k) von a wird die Menge $LAST(P_{(a,k)}(t))$ bestimmt, das sind Vorkommen $last^i_j(P_{(a,k)}(t))$, die in t vor (a,k) liegen und bereits beschriftet sind. Es interessieren nur die Beschriftungen der Vorkommen des Zeichens a in dieser Menge. Die Beschriftung des Vorkommens (a,k) wird so gewählt, daß sie sich von den Beschriftungen der anderen a-Vorkommen in $LAST(P_{(a,k)}(t))$ unterscheidet. Um mit einer endlichen Menge von Beschriftungen auszukommen (und um verschieden lange Traces später in einer Äquivalenzklasse zusammenfassen zu können), wird (a,k) mit dem kleinsten Index beschriftet, der für a-Vorkommen gerade frei ist.

Beispiel: Sei $\Sigma = \{a,b,c,d\}$ mit $\Sigma_1 = \{a,b\}$, $\Sigma_2 = \{b,c\}$, $\Sigma_3 = \{c,d\}$. Sei $t = [(abcd)^3ab]$. Die Beschriftungen der Zeichenvorkommen sind in Abb. 4.2 wie die Vorkommen selber angeordnet.

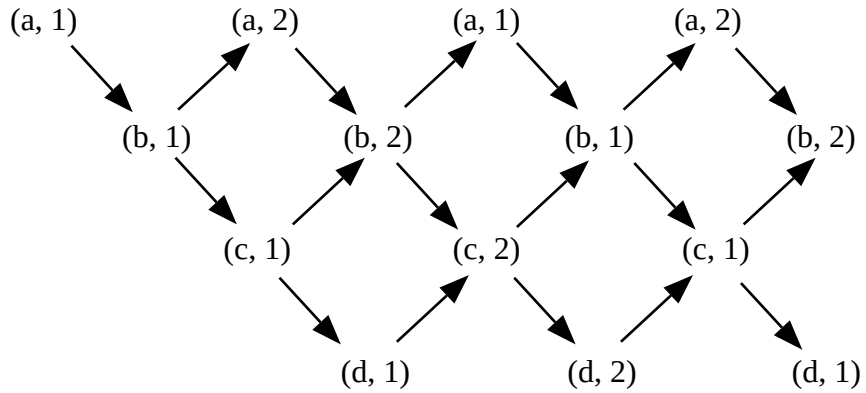

Abbildung 4.2: Beschriftungen der Vorkommen in $[(abcd)^3ab]$

Zum Vorkommen $(b,3)$ wird das Anfangsstück $P_{(b,3)}(t) = [abcdabcab]$ gebildet.
In $LAST(P_{(b,3)}(t))$ gibt es nur zwei Vorkommen von b: $(b,3)$ selber ist $last^1_1(P_{(b,3)}(t))$, und $(b,2)$ ist $last^3_1(P_{(b,3)}(t))$. Das Vorkommen $(b,2)$ hat bereits die Beschriftung $(b,2)$. Für $(b,3)$ ist die Beschriftung $(b,1)$ frei, weil das erste b-Vorkommen nicht zu $LAST(P_{(b,3)}(t))$ gehört.

Im folgenden Lemma werden die Eigenschaften der Beschriftungsfunktion zusammengefaßt (vgl. [Ziel87]):

Lemma 4.1.6 (Eigenschaften von $label_t$)

i) $(a,k) \in Vor(t) \Rightarrow \bigvee_{m \in \{1,\ldots,n\}} label_t(a,k) = (a,m)$
(Die endliche Menge $\Sigma \times \{1,\ldots,n\}$ reicht für die Beschriftungen aus.)

ii) $t' \in Pref(t) \Rightarrow label_{t'} = label_t|_{Vor(t')}$
(Die Beschriftung eines Vorkommens ist unabhängig von im Trace t später kommenden Vorkommen.)

iii) $label_t$ *ist auf* $LAST(t)$ *injektiv.*

Mit $label_t$ wird eine Äquivalenzrelation auf Traces definiert.

Definition 4.1.7 (Relation $\approx_E$)

Seien $t, r \in E(\Sigma, I)$ zwei Traces über (Σ, I). Es gilt $t \approx_E r$ genau dann, wenn die folgenden Bedingungen erfüllt sind:

i) Für alle $i, j \in \{1, \ldots, n\}$ gilt:

$$last_j^i(t) \text{ ist definiert} \iff last_j^i(r) \text{ ist definiert}$$

ii)

$$\bigwedge_{i,j,k,l \in \{1,\ldots,n\}} last_j^i(t) = last_l^k(t) \Rightarrow last_j^i(r) = last_l^k(r)$$

Sei $C : LAST(t) \to LAST(r)$ die Abbildung mit $C(last_j^i(t)) = last_j^i(r)$ für alle $i, j \in \{1, \ldots, n\}$.

iii) Die Abbildung C bildet die Halbordnung $\leq_t |_{LAST(t)}$ isomorph auf die Halbordnung $\leq_r |_{LAST(r)}$ ab.

iv) C respektiert die Beschriftung der Vorkommen, also

$$label_t(last_j^i(t)) = label_r(last_j^i(r)) \text{ für alle } i, j \in \{1, \ldots, n\}.$$

Die Bedingungen i) und ii) stellen sicher, daß der kanonische Isomorphismus C wohldefiniert ist. Die Relation $\approx_E$ ist offensichtlich eine Äquivalenzrelation und unabhängig von einer gegebenen Trace-Sprache T. $\approx_E$ hat endlich viele Äquivalenzklassen.

Definition 4.1.8 (Schlußstück $S_\alpha(t)$)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen Cliquen der Abhängigkeitsrelation von (Σ, I). Sei $\alpha \subseteq \{1, \ldots, n\}$ und sei $\overline{\alpha} =_{Df} \{1, \ldots, n\} \backslash \alpha$. Sei $t \in E(\Sigma, I)$ ein Trace.
Es wird definiert:
$S_\alpha(t)$ ist der Trace, der das Anfangsstück $P_{\overline{\alpha}}(t)$ zu t ergänzt:

$$P_{\overline{\alpha}}(t) S_\alpha(t) = t$$

Definition 4.1.9 (Relation $\approx_T$)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I).
Die Relation $\approx_T$ auf $E(\Sigma, I)$ wird definiert als

$$t \approx_T t' \iff_{Df} \bigwedge_{\alpha \subseteq \{1,\ldots,n\}} S_\alpha(t) \sim_T S_\alpha(t')$$

$\approx_T$ ist eine Verfeinerung der syntaktischen Kongruenz zu T, denn für $\alpha = \{1, \ldots, n\}$ ist $S_\alpha(t) = t$ und somit $t \approx_T t' \Rightarrow t \sim_T t'$. Die Äquivalenzrelation $\approx_T$ zerlegt $E(\Sigma, I)$ genau dann in endlich viele Äquivalenzklassen, wenn T erkennbar ist.

Definition 4.1.10 (Relation $\approx$)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I).

i) Die Relation $\approx$ auf $E(\Sigma, I)$ entsteht durch Schnitt der Äquivalenzrelationen $\approx_E$ und $\approx_T$:

$$t \approx t' \Longleftrightarrow_{Df} t \approx_E t' \wedge t \approx_T t'$$

ii) $\langle t \rangle$ ist die Äquivalenzklasse der Relation $\approx$ zum Trace $t \in E(\Sigma, I)$.

Wie $\approx_T$ definiert $\approx$ genau dann endlich viele Äquivalenzklassen, wenn T erkennbar ist. $\approx$ ist eine Verfeinerung von $\approx_T$ und somit auch eine Verfeinerung der syntaktischen Kongruenz von T. Die Klassen der Relation von i-Anfangsstücken definieren die Zustände des Teilautomaten $\mathcal{P}_i$ eines EAAs, der T erkennt. Für den Beweis der Korrektheit der Konstruktion formuliert Zielonka die folgenden beiden Sätze (vgl. [Ziel87]).

Satz 4.1.11 (Verträglichkeit von $\approx$ mit P_α-Anfangsstückkonzept)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und seien $t, t' \in E(\Sigma, I)$ zwei Traces. Seien $\alpha, \beta \subseteq \{1, \ldots, n\}$ Mengen von Indizes von maximalen Cliquen der Abhängigkeitsrelation von (Σ, I).
Es gilt:

$$P_\alpha(t) \approx P_\alpha(t') \wedge P_\beta(t) \approx P_\beta(t') \Rightarrow P_{\alpha \cup \beta}(t) \approx P_{\alpha \cup \beta}(t')$$

Satz 4.1.12 (Verträglichkeit von $\approx$ mit Verkettung)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation, seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen Cliquen der Abhängigkeitsrelation von (Σ, I), sei $a \in \Sigma$ ein Zeichen und seien $t, r \in E(\Sigma, I)$ zwei Traces.
Es gilt:

$$\bigwedge_{i \in Dom(a)} P_i(r) \approx P_i(t) \Rightarrow \bigwedge_{i \in Dom(a)} P_i(r[a]) \approx P_i(t[a])$$

Für $i \in \{1, \ldots, n\}$ trifft die Behauptung von Satz 4.1.12 ebenfalls zu, weil für $i \notin Dom(a)$ die i-Anfangsstücke $P_i(t)$ und $P_i(t[a])$ gleich sind.

Die Sätze 4.1.11 und 4.1.12 besagen in etwa: Wenn die Abbildung ρ die Menge $E(\Sigma, I)$ auf die Äquivalenzklassen der Relation $\approx$ abbildet, dann ist ρ asynchron (vgl. Anmerkung unter Satz 3.5.1).

Definition 4.1.13 (Konstruktion des EAA zu $T \subseteq E(\Sigma, I)$)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen Cliquen der Abhängigkeitsrelation. Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache.
Der EAA zu T wird definiert als $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit

i) $\mathcal{P}_i =_{Df} (\Sigma_i, S_i)$, *wobei* $S_i =_{Df} \{\langle P_i(t)\rangle | t \in E(\Sigma, I)\}$

ii) $\mathcal{I} =_{Df} \{(s_1^0, \ldots, s_n^0)\}$ *mit* $s_i^0 =_{Df} \langle [\varepsilon]\rangle$ *für* $i = 1, \ldots, n$

iii) $\mathcal{D} =_{Df} \{\delta_a | a \in \Sigma\}$ *mit*

$$\delta_a\Big(\langle P_{i_1}(t)\rangle, \ldots, \langle P_{i_k}(t)\rangle\Big) =_{Df} \Big\{\Big(\langle P_{i_1}(t[a])\rangle, \ldots, \langle P_{i_k}(t[a])\rangle\Big)\Big\}$$

wobei $t \in E(\Sigma, I)$ ein Trace und $Dom(a) = \{i_1, \ldots, i_k\}$ ist.

iv) $\mathcal{F} =_{Df} \{(\langle P_1(t)\rangle, \ldots, \langle P_n(t)\rangle) | t \in T\}$

Die Definition der Übergänge in iii) ist nach Satz 4.1.12 unabhängig vom gewählten $t \in E(\Sigma, I)$ und somit korrekt. Der so konstruierte EAA ist deterministisch und vollständig, d. h. alle Übergänge sind definiert.

Satz 4.1.14 (EAA zu $T \subseteq E(\Sigma, I)$)

Für den in Definition 4.1.13 konstruierten EAA gilt $T(\mathcal{A}) = T$.

Beweisskizze: Für einen Trace $t \in E(\Sigma, I)$ läßt sich durch Induktion über seine Länge leicht zeigen, daß $\mathcal{A}$ mit t den Übergang

$$(\langle [\varepsilon]\rangle, \ldots, \langle [\varepsilon]\rangle) \stackrel{t}{\Longrightarrow} (\langle P_1(t)\rangle, \ldots, \langle P_n(t)\rangle)$$

ausführt. Aus der Definition der Mengen $\mathcal{I}$ und $\mathcal{F}$ ergibt sich somit $t \in T \Rightarrow t \in T(\mathcal{A})$.
Sei umgekehrt $t \in T(\mathcal{A})$. t überführt $\mathcal{A}$ in einen Endzustand aus $\mathcal{F}$, also gibt es einen Trace $t' \in T$ mit $\langle P_i(t)\rangle = \langle P_i(t')\rangle$ für alle $i \in \{1, \ldots, n\}$. Es gilt $P_i(t) \approx P_i(t')$ und folglich nach Satz 4.1.11 $t = P_{\{1,\ldots,n\}}(t) \approx P_{\{1,\ldots,n\}}(t') = t'$. Insbesondere sind t und t' äquivalent bezüglich $\approx_T$ und somit auch syntaktisch kongruent bezüglich T. Mit $t' \in T$ ist demnach $t \in T$. □

Die Konstruktion kann effektiv ausgeführt werden, wenn ein Entscheidungsverfahren für die syntaktische Kongruenz zu T vorgegeben ist. Mit Hilfe dieses Entscheidungsverfahrens kann entschieden werden, ob zwei Traces $\approx$-äquivalent sind. Es kann ein Graph konstruiert werden, der die Übergänge zwischen den globalen Zuständen des EAAs zeigt, denn zwei Traces $t, r \in E(\Sigma, I)$ führen genau dann in denselben globalen Zustand, wenn für alle $i \in \{1, \ldots, n\}$ $\langle P_i(t)\rangle = \langle P_i(r)\rangle$ gilt. Die Übergänge zwischen den globalen Zuständen sind nach den Sätzen 4.1.11 und 4.1.12 unabhängig vom betrachteten Trace t. Die Darstellung der globalen Zuständen als n-Tupel von $\approx$-Äquivalenzklassen führt schließlich zur Definition der Teilautomaten und der Relationen δ_a.

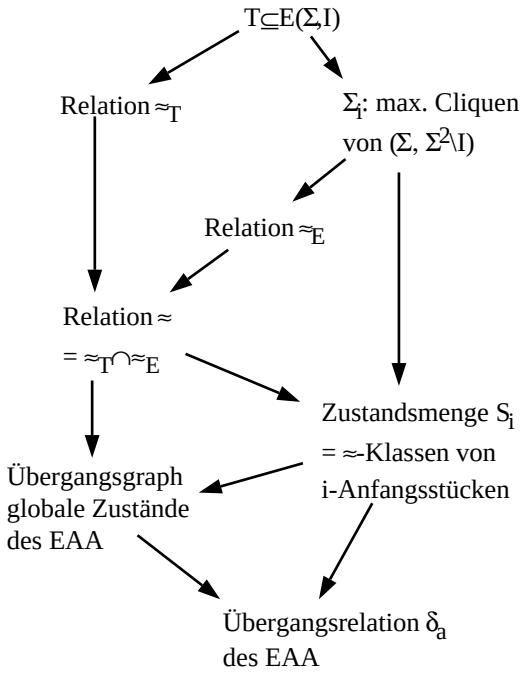

Abbildung 4.3: Schema des Konstruktionsverfahrens

Einen Überblick über das Konstruktionsverfahren gibt Abb. 4.3.

Beispiel 1:
Gesucht ist ein EAA für die Trace-Sprache $T = [(abc)^*]$ über dem Alphabet (Σ, I) mit $\Sigma = \{a, b, c\}$ und $I = \{(b, c), (c, b)\}$. Die maximalen Cliquen der Abhängigkeitsrelation sind $\Sigma_1 = \{a, b\}$ und $\Sigma_2 = \{a, c\}$.
Die Berechnung der Klassen zu den Äquivalenzrelationen ergibt:

Relation	Anzahl der Klassen
$\approx_E$	8
$\sim_T$	20
$\approx_T$	28
$\approx$	34

In Abb. 4.4 ist ein Ausschnitt des Übergangsgraphen des EAAs dargestellt. Für jeden globalen Zustand ist das Paar $(\langle P_1(t)\rangle, \langle P_2(t)\rangle)$ durch den lexikalisch kleinsten kürzesten Repräsentanten der $\approx$-Klasse angegeben.

Der gesuchte EAA, reduziert auf die Globalzustände, von denen aus ein Endzustand (in Abb. 4.4 q_0 und q_4) erreicht werden kann, hat fünf Zustände (q_0 bis q_4). Werden die globalen Zustände verteilt dargestellt, ergibt sich der EAA in Abb. 4.5.

Beispiel 2:
Über dem Alphabet aus Beispiel 1 wird die Trace-Sprache $T = [abc \cup aabbc \cup aabcc]$ zusammen

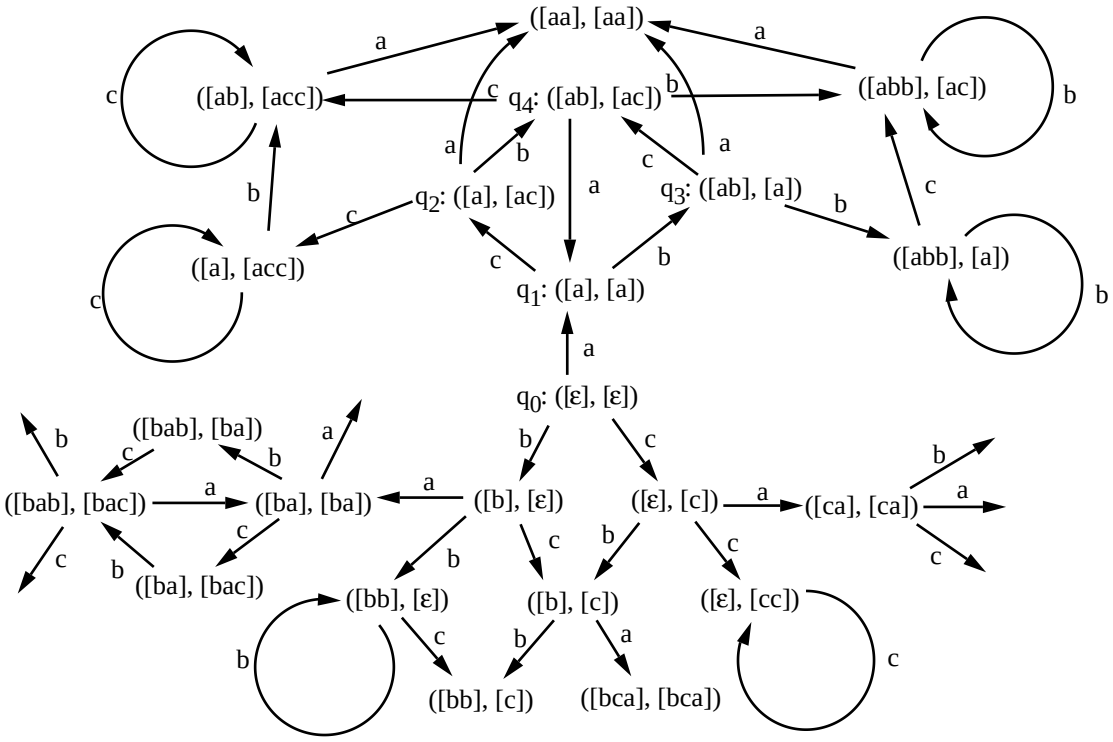

Abbildung 4.4: Übergangsgraph (unvollständig)

mit den Traces $t_1 = [abc]$ und $t_2 = [aabbcc]$ betrachtet. Es gelten die Äquivalenzen

$$P_1(t_1) = [ab] \sim_T [aabb] = P_1(t_2)$$

$$P_1(t_1) \approx_E P_1(t_2)$$

$$P_2(t_1) = [ac] \sim_T [aacc] = P_2(t_2)$$

$$P_2(t_1) \approx_E P_2(t_2)$$

Ein EAA nur mit $\approx_E$ und $\sim_T$ statt $\approx_T$ konstruiert würde also mit t_1 und t_2 dieselben lokalen Zustände in den Teilautomaten für Σ_1 und Σ_2 erreichen. t_1 und t_2 sind jedoch nicht syntaktisch kongruent bezüglich T, denn $t_1 \in T$ und $t_2 \notin T$. Die korrekte Unterscheidung zwischen den lokalen Zuständen übernimmt die Relation $\approx_T$. Beispielsweise gilt nicht $P_1(t_1) \approx_T P_1(t_2)$, weil $S_{\{1\}}(P_1(t_1)) = [b]$ und $S_{\{1\}}(P_1(t_2)) = [bb]$ nicht syntaktisch kongruent bezüglich T sind.

Beispiel 3:
Sei $T = [(abc(d \cup \varepsilon))^*] \subseteq E(\Sigma, I)$ mit $\Sigma = \{a, b, c, d\}$ und $I = \{(a, c), (b, d), (c, a), (d, b)\}$.
Die maximalen Cliquen sind $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$, $\Sigma_3 = \{c, d\}$ und $\Sigma_4 = \{a, d\}$.
Nach Abarbeitung von $[(abc)^3]$ ist $\mathcal{P}_4$ im Zustand $\langle P_4([(abc)^3])\rangle = \langle [abcaba]\rangle$ und $\mathcal{P}_3$ im Zustand $\langle P_3([(abc)^3])\rangle = \langle [(abc)^3]\rangle$. Beide Teilautomaten können von diesen Zuständen aus gemeinsam mit einem d-Übergang in einen globalen Endzustand gelangen und so $[(abc)^3 d]$ akzeptieren.
Nach Abarbeitung von $[(abc)^3 a]$ muß wegen $[(abc)^3 ad] \notin Pref(T)$ ein d-Übergang in einen globalen Zustand führen, von dem aus kein Endzustand erreichbar ist. Die Kontrolle über die

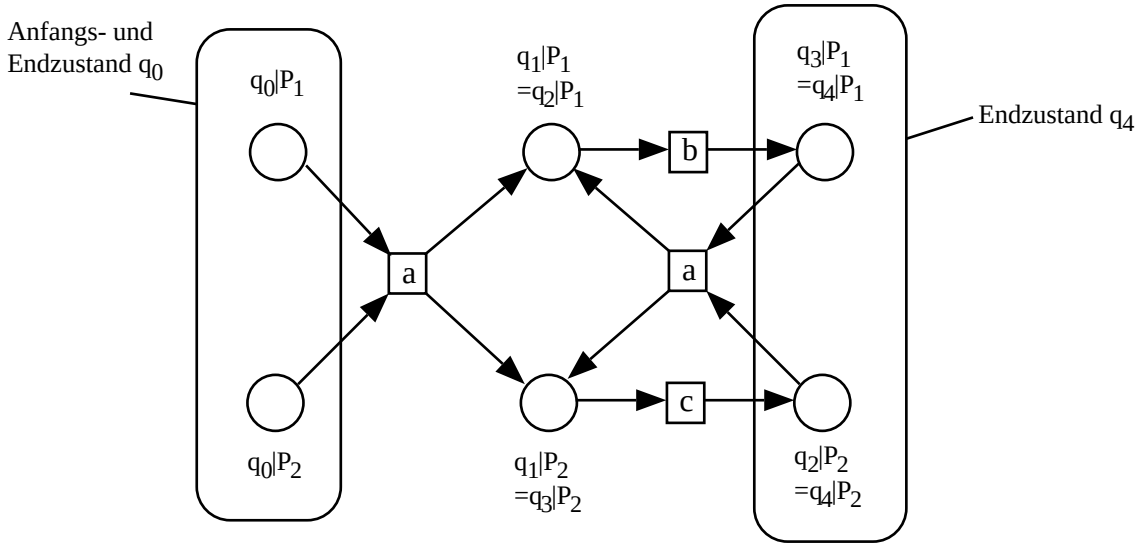

Abbildung 4.5: EAA zu $T = [(abc)^*]$

d-Übergänge liegt nur bei den Teilautomaten $\mathcal{P}_3$ und $\mathcal{P}_4$. $\mathcal{P}_3$ ist im Zustand $\langle P_3([(abc)^3a])\rangle = \langle[(abc)^3]\rangle$ und $\mathcal{P}_4$ im Zustand $\langle P_4([(abc)^3a])\rangle = \langle[(abc)^2aba]\rangle$. Von diesem Zustandspaar muß ein anderer d-Übergang ausgehen, folglich ist $\langle[abcaba]\rangle \neq \langle[(abc)^2aba]\rangle$. Die Unterscheidung dieser beiden Zustände wird durch die Relation $\approx_E$ geleistet. Es gilt

$$[abcaba] \approx_T [(abc)^2aba]$$

aber nicht

$$[abcaba] \approx_E [(abc)^2aba],$$

weil sich die beiden Traces in der Beschriftung (Def. 4.1.5) ihrer letzten Zeichenvorkommen unterscheiden.

4.2 EAA-Konstruktion für Trace-Sprachen über trianguierten Alphabeten

Die Beispiele haben gezeigt, daß im allgemeinen Fall eine korrekte Konstruktion nach Zielonkas Ansatz weder ohne die Äquivalenzrelation $\approx_T$ noch ohne die von der gegeben Trace-Sprache T unabhängige Äquivalenzrelation $\approx_E$ auskommt. In diesem Abschnitt soll die Frage untersucht werden, in welchem Fall allein die Relation $\approx_T$ als Grundlage der Konstruktion ausreicht.

Zwei Traces t und r stehen in der Relation $\approx_T$, wenn ihre Schlußstücke $S_\alpha(t)$ und $S_\alpha(r)$ für jedes $\alpha \subseteq \{1, \ldots, n\}$ syntaktisch kongruent sind. Intuitiv läßt sich jeder Trace als Verkettung eines Anfangsstücks und mehrerer Schlußstücke von Anfangsstücken schreiben:

$$t = P_1(t)S_{\{2,\ldots,n\}}(P_2(t))S_{\{3,\ldots,n\}}(P_3(t)) \ldots S_{\{n\}}(P_n(t))$$

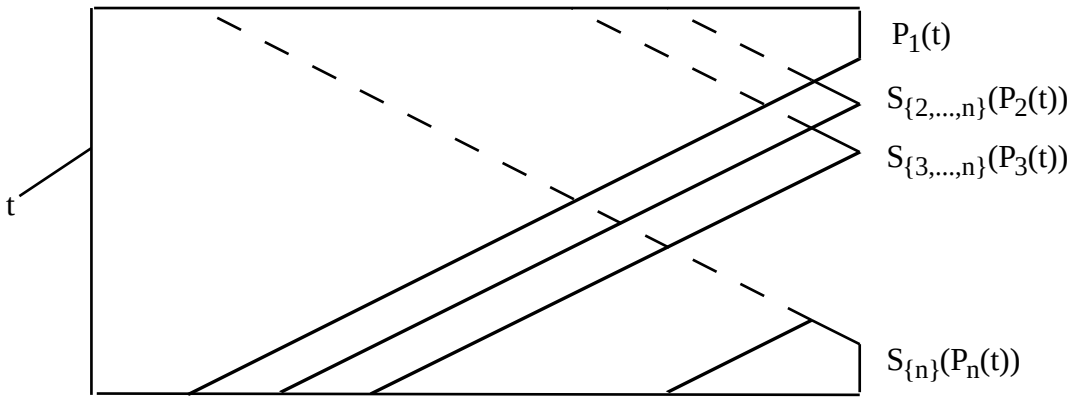

Abbildung 4.6: Zerlegung eines Traces

Die Idee wird durch Abbildung 4.6 veranschaulicht.

Sind alle i-Anfangsstücke zweier Traces t und r $\approx_T$-äquivalent, so sind die einander entsprechenden Schlußstücke der i-Anfangsstücke und somit t und r selber syntaktisch kongruent. Das folgende Beispiel gibt einen Hinweis darauf, unter welchen Bedingungen sich aus diesen Überlegungen ein korrektes Konstruktionsverfahren für EAAs auf der Basis der Relation $\approx_T$ ableiten läßt.

Beispiel: Sei $\Sigma = \{a, b, c, d\}$ mit $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{c, d\}$ und $\Sigma_3 = \{b, c\}$.
Für $t = [(abcd)^2aba]$ gilt

$$t \neq [abcdabcaba][dcd][\varepsilon] = P_1(t)S_{\{2,3\}}(P_2(t))S_{\{3\}}(P_3(t))$$

Für die Ungleichheit ist die Reihenfolge der drei maximalen D-Cliquen entscheidend. Die beiden abhängigen Zeichen b und c sind bereits in $\Sigma_1 \cup \Sigma_2$ enthalten, treten jedoch erst in Σ_3 gemeinsam auf. Diese Konstellation wird vermieden, wenn $\Sigma_1 = \{a, b\}$, $\Sigma_2 = \{b, c\}$ und $\Sigma_3 = \{c, d\}$ gesetzt wird. In diesem Fall gilt stets die Zerlegung $t = P_1(t)S_{\{2,3\}}(P_2(t))S_{\{3\}}(P_3(t))$ (was zu zeigen sein wird).

Die maximalen D-Cliquen können dann und nur dann in eine die Zerlegung garantierende Reihenfolge gebracht werden, wenn jeder minimale Kreis in (Σ, D) ganz in einer maximalen D-Clique liegt, also nicht mehr als drei Elemente enthält. Genau dann ist der Graph von (Σ, D) trianguliert. Für EAAs zu erkennbaren Trace-Sprachen über Alphabeten mit Unabhängigkeitsrelation, deren Graph von (Σ, D) trianguliert ist, kann also ein einfacheres Konstruktionsverfahren angegeben werden, das zudem einen kleineren EAA konstruiert.

Das hier dargestellte Verfahren findet sich im wesentlichen ebenfalls in [DM95]. Dort wird die Automatenkonstruktion für erkennbare Trace-Sprachen über zyklenfreien Alphabeten (vgl. [Mét87]) auf triangulierte Alphabete verallgemeinert. Es werden zelluläre EAAs konstruiert.

Von Diekert und Muscholl stammt der Hinweis auf die Charakterisierung der triangulierten Graphen in [Gol80], die hier sonst explizit bewiesen werden müßte.

Definition 4.2.1 (Triangulierter Graph, [Gol80])

Sei $G = (V, E)$ ein Graph.
G heißt trianguliert $\Longleftrightarrow_{Df}$
Jeder Kreis in G bestehend aus mindestens vier Knoten enthält eine Sehne
(d. h. wenn $a_1, \ldots, a_n \in V, n > 3$, und $\{a_i, a_{i+1}\} \in E$ für $i = 1, \ldots, n-1$ sowie $\{a_1, a_n\} \in E$, dann gibt es $\{a_i, a_j\} \in E$ mit $1 < |i - j| < n - 1$).

Definition 4.2.2 (Perfekte Ordnung, [Gol80])

Sei $G = (V, E)$ ein Graph mit $V = \{v_1, \ldots, v_n\}$. Für einen Knoten $v \in V$ ist $N(v) = \{v' \in V | \{v, v'\} \in E\}$ die Menge seiner Nachbarknoten.
Die Folge $\langle v_1, \ldots, v_n \rangle$ ist eine perfekte Ordnung der Knoten, wenn für alle $i \in \{1, \ldots, n\}$ der Knoten v_i mit der Menge $N(v_i) \cap \{v_i, \ldots, v_n\}$ eine Clique bildet, d. h. $E|_{(N(v_i) \cap \{v_i, \ldots, v_n\}) \cup \{v_i\}}$ vollständig ist.

Die perfekte Ordnung ergibt ein Knoteneliminierungsschema, bei dem der aus dem Restgraphen herauszunehmende Knoten mit seinen Nachbarknoten eine Clique bildet.

Satz 4.2.3 (Charakterisierung von triangulierten Graphen, [FG65])

i) Ein Graph G ist trianguliert $\Longleftrightarrow$ Es gibt eine perfekte Ordnung seiner Knoten.

ii) Die perfekte Ordnung kann mit jedem Knoten von G starten, der mit seinen Nachbarknoten zusammen eine Clique bildet.

Der folgende Satz stellt einen Zusammenhang her zwischen der in der Literatur behandelten Charakterisierung der triangulierten Graphen und der erwähnten Reihenfolge der maximalen D-Cliquen in (Σ, I). Es muß zuvor geklärt werden, wann ein (Σ, I) trianguliert zu nennen ist.

Definition 4.2.4 (Trianguliertes Alphabet (Σ, I))

Ein Alphabet mit Unabhängigkeitsrelation (Σ, I) heißt trianguliert $\Longleftrightarrow_{Df}$ Der Graph von $(\Sigma, \Sigma^2 \setminus (I \cup id))$ ist trianguliert.

Satz 4.2.5 (Ordnung der max. Cliquen in triangulierten Alphabeten)

Sei (Σ, I) ein trianguliertes Alphabet mit Unabhängigkeitsrelation.
Die maximalen D-Cliquen von (Σ, I) können geordnet werden als $\Sigma_1, \ldots, \Sigma_n$, so daß gilt:

$$\bigwedge_{k \in \{1,\ldots,n\}} \bigwedge_{a,b \in \Sigma_k} \{a, b\} \cap (\Sigma_k \setminus \bigcup_{l<k} \Sigma_l) \neq \emptyset \vee \bigvee_{l<k} \{a, b\} \subseteq \Sigma_l$$

Beweis:
Wenn (Σ, I) trianguliert ist, gibt es nach Satz 4.2.3 eine perfekte Ordnung $\langle a_1, \ldots, a_m \rangle$ seiner Zeichen. Wenn das Zeichen a_1 zu zwei maximalen D-Cliquen von (Σ, I) gehören würde, müßte es zwei Zeichen $b, c \in \Sigma$ mit $(a, b) \in D, (a, c) \in D$ und $(b, c) \notin D$ geben. Dies widerspricht jedoch der Eigenschaft von a_1, nach der der Knoten a_1 zusammen mit seinen Nachbarknoten im Graphen $(\Sigma, D \setminus id)$ eine Clique bildet. Somit wird durch a_1 eine maximale Clique Σ_n eindeutig bestimmt.

Nach der Eliminierung von a_1 ist der Restgraph trianguliert, weil es für ihn eine perfekte Ordnung der Knoten gibt. Die Knoten in $\Sigma_n \setminus \bigcup_{k<n} \Sigma_k$ bilden zusammen mit ihren Nachbarknoten Cliquen, so daß nach Satz 4.2.3 eine perfekte Ordnung des Restgraphen mit jedem dieser Knoten starten kann. Folglich kann für (Σ, I) eine perfekte Ordnung angegeben werden, deren Eliminationsschema zunächst die Knoten aus $\Sigma_n \setminus \bigcup_{k<n} \Sigma_k$ löscht. Es kann dann auf die gleiche Weise eine maximale Clique Σ_{n-1} bestimmt werden, um die Knoten aus $\Sigma_{n-1} \setminus \bigcup_{k<n-1} \Sigma_k$ zu eliminieren. Durch fortgesetzte Anwendung dieses Verfahrens bekommen wir so eine Folge $\Sigma_n, \ldots, \Sigma_1$ der maximalen Cliquen.

Sei nun $a, b \in \Sigma_k$. Wenn es einen Index $l < k$ mit $\{a, b\} \subseteq \Sigma_l$ gibt, ist die im Satz behauptete Eigenschaft bereits gezeigt. Sei o. B. d. A. also k der minimale Index mit $a, b \in \Sigma_k$. a und b sind in dem Restgraphen enthalten, der die Knotenmenge $\Sigma_1 \cup \ldots \cup \Sigma_k$ umfaßt. Nach dem Verfahren werden nur solche Knoten eliminiert, die bezogen auf den Restgraphen nur in einer maximalen Clique liegen. Wenn o. B. d. A. a vor b eliminiert wird und Σ_l mit $l \leq k$ die in $\Sigma_n, \ldots, \Sigma_1$ letzte maximale Clique mit $a \in \Sigma_l$ ist, so ist daher wegen $(a, b) \in D$ auch $b \in \Sigma_l$. Dann haben wir aber $l = k$ und $a \in \Sigma_k \setminus \bigcup_{l<k} \Sigma_l$. □

Für den Beweis des Satzes über die Zerlegbarkeit von Traces über triangulierten Alphabeten mit Unabhängigkeitsrelation wird folgendes Lemma benötigt:

Lemma 4.2.6 ($S(P(t))$-Lemma)

Sei (Σ, I) ein trianguliertes Alphabet mit Unabhängigkeitsrelation und seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen in der durch die perfekte Ordnung nach Satz 4.2.5 gegebenen Reihenfolge.
Für jeden Trace $t \in E(\Sigma, I)$ gilt:

$$S_{\{k,\ldots,n\}}(P_{\{1,\ldots,k\}}(t)) = S_{\{k,\ldots,n\}}(P_k(t))$$

Beweis:
Die Halbordnung der Zeichenvorkommen wird in den Traces auf beiden Seiten der Gleichung durch $\leq_t$ bestimmt. Es muß daher nur bewiesen werden, daß beide Traces dieselben Zeichenvorkommen enthalten. Zu zeigen ist demnach

$$Vor(S_{\{k,\ldots,n\}}(P_{\{1,\ldots,k\}}(t))) = Vor(S_{\{k,\ldots,n\}}(P_k(t))),$$

wobei

$Vor(S_{\{k,\ldots,n\}}(P_{\{1,\ldots,k\}}(t))) =$
$\{x \in Vor(t) | \bigvee_{j\in\{1,\ldots,k\}} x \leq_t last_j^j(t) \wedge \bigwedge_{j\in\{1,\ldots,k\}} \bigwedge_{y\in Vor(t)} x \leq_t y \wedge y \leq_t last_j^j(t) \Rightarrow Dom(y) \subseteq \{k,\ldots,n\}\}$

und

$Vor(S_{\{k,\ldots,n\}}(P_k(t))) =$
$\{x \in Vor(t) | x \leq_t last_k^k(t) \wedge \bigwedge_{y\in Vor(t)} x \leq_t y \wedge y \leq_t last_k^k(t) \Rightarrow Dom(y) \subseteq \{k,\ldots,n\}\}.$

$\subseteq$: Zu einem $x \in Vor(S_{\{k,\ldots,n\}}(P_{\{1,\ldots,k\}}(t)))$ gibt es ein $j \in \{1,\ldots,k\}$ mit $x \leq_t last_j^j(t)$. Für alle $y \in Vor(t)$ mit $x \leq_t y$ und $y \leq_t last_j^j(t)$ gilt $Dom(y) \subseteq \{k,\ldots,n\}$. Insbesondere ist also $last_j^j(t) \in \Sigma_k$, folglich ist $x \leq_t last_j^j(t) \leq_t last_k^k(t)$, und für alle $y \in Vor(t)$ mit $x \leq_t y$ und $y \leq_t last_k^k(t)$ gilt $Dom(y) \subseteq \{k,\ldots,n\}$. Somit $x \in Vor(S_{\{k,\ldots,n\}}(P_k(t)))$.

$\supseteq$: Sei $x \in Vor(S_{\{k,\ldots,n\}}(P_k(t)))$. Es wird $x \notin Vor(S_{\{k,\ldots,n\}}(P_{\{1,\ldots,k\}}(t)))$ zum Widerspruch geführt.
Wegen $x \leq_t last_k^k(t)$ und $x \notin Vor(S_{\{k,\ldots,n\}}(P_{\{1,\ldots,k\}}(t)))$ muß es ein Zeichenvorkommen $last_j^j(t)$ mit $j < k$ und $x \leq_t last_j^j(t)$ geben. Sei $(a_1,n_1),\ldots,(a_m,n_m)$ die kürzeste Folge von Zeichenvorkommen mit $x = (a_1,n_1) <_t \ldots <_t (a_m,n_m) = last_j^j(t)$ und $(a_j,a_{j+1}) \in D$. Sei (a_g,n_g) das letzte Vorkommen in dieser Folge mit $Dom(a_g) \subseteq \{k,\ldots,n\}$ und $(a_g,n_g) \leq_t last_k^k(t)$. Sei (a_i,n_i) das letzte Vorkommen in dieser Folge mit $Dom(a_i) \subseteq \{k,\ldots,n\}$. Es gilt $x \leq_t (a_g,n_g) \leq_t (a_i,n_i) <_t last_j^j(t)$ und $(a_g,n_g) \leq_t last_k^k(t)$ (siehe Abbildung 4.7).

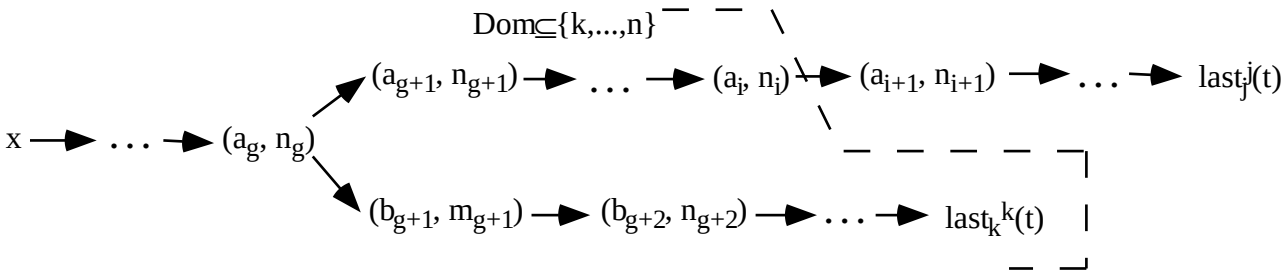

Abbildung 4.7: Abfolge der Zeichenvorkommen in t

(a_{i+1},n_{i+1}) ist das erste Vorkommen in $\langle x,(a_1,n_1),\ldots,(a_m,n_m),last_j^j(t)\rangle$ mit $Dom(a_{i+1}) \not\subseteq \{k,\ldots,n\}$. Also tritt a_{i+1} im Eliminationsschema der maximalen Cliquen erst nach den Zeichen aus $\Sigma_k \setminus \bigcup_{l<k} \Sigma_l$ auf. Weil a_i spätestens mit diesen Zeichen gelöscht wird, wird a_{i+1} echt nach a_i eliminiert. a_{i-1} und a_{i+1} sind unabhängig, weil die kürzeste Folge von Vorkommen gewählt wurde. Würde a_{i-1} nicht vor a_i eliminiert, müßte es bei Elimination von a_i mit a_i und a_{i+1} in derselben maximalen Clique und somit mit a_{i+1} abhängig sein. Daher wird a_{i-1} vor a_i eliminiert. Die fortgesetzte Anwendung dieser Argumentation ergibt: a_g wird spätestens mit a_i, also vor a_{i+1} eliminiert.

$(a_g,n_g) = last_k^k(t)$ ergibt einen Widerspruch, denn $a_g \in \Sigma_k$ würde erst mit den Zeichen aus $\Sigma_k \setminus \bigcup_{l<k} \Sigma_l$ eliminiert werden, a_g müßte also mit a_i oder später eliminiert werden und alle drei Zeichen a_g, a_i sowie a_{i+1} wären in der maximalen Clique, mit der auch a_i gelöscht wird. Weil

a_g und a_{i+1} so abhängig wären, a_{i+1} aber nach a_g eliminiert wird, müßte a_{i+1} in Σ_k liegen, wenn a_g mit Zeichen aus $\Sigma_k \setminus \bigcup_{l<k} \Sigma_l$ gelöscht wird. Dann aber wäre $(a_g, n_g) <_t (a_{i+1}, n_{i+1}) \leq_t last_k^k(t)$. Folglich ist $(a_g, n_g) <_t last_k^k(t)$. Sei $(a_g, n_g) <_t (b_{g+1}, m_{g+1}) <_t (b_{g+2}, m_{g+2}) <_t \ldots <_t last_k^k(t)$ die kürzeste Folge von aufeinanderfolgenden abhängigen Zeichenvorkommen. b_{g+1} wird vor a_g eliminiert, denn wenn a_g spätestens mit a_{g+1} und b_{g+1} gelöscht wird, müssen alle drei in einer maximalen D-Clique und somit abhängig sein. Weil wieder die kürzeste Folge von Zeichenvorkommen gewählt wurde, sind b_{g+2} und a_g unabhängig. Mit der bekannten Argumentation ergibt sich abermals, daß b_{g+2} vor b_{g+1} eliminiert wird und daß das Zeichen zum Vorkommen $last_k^k(t)$ spätestens mit (b_{g+1}, m_{g+1}), also vor (a_g, n_g) eliminiert wird.

Wegen $x \in Vor(S_{\{k,\ldots,n\}}(P_k(t)))$ und $x \leq_t last_k^k(t)$ tritt das Zeichen von $last_k^k(t)$ nur in maximalen Cliquen Σ_l mit $l \geq k$ auf und wird daher erst mit den Zeichen aus $\Sigma_k \setminus \bigcup_{l<k} \Sigma_l$ aus dem trianguliertem Graphen gelöscht. Weil es wie gerade gesehen vor (a_g, n_g) eliminiert wird, (a_g, n_g) aber spätestens mit (a_i, n_i) und dieses (a_i, n_i) spätestens mit den Zeichen aus $\Sigma_k \setminus \bigcup_{l<k} \Sigma_l$, ergibt sich hier ein Widerspruch, der nicht weiter aufgelöst werden kann. □

Mit diesem technischen Ergebnis läßt sich leicht der Satz über die Zerlegung der Traces beweisen:

Satz 4.2.7 (Zerlegung eines Traces über trianguliertem (Σ, I))

Sei (Σ, I) ein trianguliertes Alphabet mit Unabhängigkeitsrelation und $t \in E(\Sigma, I)$ ein Trace. Seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen von (Σ, I) in der durch Satz 4.2.5 gegebenen Reihenfolge. Es gilt:

$$t = P_1(t)S_{\{2,\ldots,n\}}(P_2(t))S_{\{3,\ldots,n\}}(P_3(t))\ldots S_{\{n\}}(P_n(t))$$

Beweis:
Wegen $t' = S_{\{1,\ldots,n\}}(t')$ und $t' = P_{\{1,\ldots,n\}}(t')$ für jedes $t' \in E(\Sigma, I)$ kann die Behauptung als

$$P_{\{1,\ldots,n\}}(t) = S_{\{1,\ldots,n\}}(P_1(t))S_{\{2,\ldots,n\}}(P_2(t))S_{\{3,\ldots,n\}}(P_3(t))\ldots S_{\{n\}}(P_n(t))$$

formuliert werden. Es wird die stärkere Aussage

$$\bigwedge_{k\in\{1,\ldots,n\}} P_{\{1,\ldots,k\}}(t) = S_{\{1,\ldots,n\}}(P_1(t))S_{\{2,\ldots,n\}}(P_2(t))S_{\{3,\ldots,n\}}(P_3(t))\ldots S_{\{k,\ldots,n\}}(P_k(t))$$

durch Induktion bewiesen.

Für k=1 ergibt sich die Behauptung aus $t' = S_{\{1,\ldots,n\}}(t')$. Im Induktionsschritt ist

$$P_{\{1,\ldots,k+1\}}(t) = P_{\{1,\ldots,k\}}(t)S_{\{k+1,\ldots,n\}}(P_{k+1}(t))$$

zu zeigen.

Nach der Definition der Schlußstücke ist für alle $t' \in E(\Sigma, I)$ $t' = P_{\{1,\ldots,k\}}(t')S_{\{k+1,\ldots,n\}}(t')$. Also gilt insbesondere

$$P_{\{1,\ldots,k+1\}}(t) = P_{\{1,\ldots,k\}}(P_{\{1,\ldots,k+1\}}(t))S_{\{k+1,\ldots,n\}}(P_{\{1,\ldots,k+1\}}(t)).$$

Für $P_{\{1,\ldots,k\}}(P_{\{1,\ldots,k+1\}}(t)) = P_{\{1,\ldots,k\}}(t)$ muß die Triangularität von (Σ, I) nicht ausgenutzt werden:

Es gilt $Vor(P_{\{1,\ldots,k+1\}}(t)) = \{x \in Vor(t) | \bigvee_{i\in\{1,\ldots,k+1\}} x \leq_t last_i^i(t)\}$. Folglich ist $last_i^i(t) \in Vor(P_{\{1,\ldots,k+1\}}(t))$ für $i \in \{1,\ldots,k\}$. Demnach gilt:

$$\begin{aligned} Vor(P_i(t)) &= \{x \in Vor(t) | x \leq_t last_i^i(t)\} \\ &= \{x \in Vor(P_{\{1,\ldots,k+1\}}(t)) | x \leq_t last_i^i(t)\} \\ &= Vor(P_i(P_{\{1,\ldots,k+1\}}(t))) \end{aligned}$$

Also haben wir

$$\begin{aligned} Vor(P_{\{1,\ldots,k\}}(P_{\{1,\ldots,k+1\}}(t))) &= \bigcup_{i\in\{1,\ldots,k\}} Vor(P_i(P_{\{1,\ldots,k+1\}}(t))) \\ &= \bigcup_{i\in\{1,\ldots,k\}} Vor(P_i(t)) \\ &= Vor(P_{\{1,\ldots,k\}}(t)) \end{aligned}$$

Die Gleichheit von $S_{\{k+1,\ldots,n\}}(P_{k+1}(t))$ und $S_{\{k+1,\ldots,n\}}(P_{\{1,\ldots,k+1\}}(t))$ wurde bereits in Lemma 4.2.6 bewiesen. □

Lemma 4.2.8 (Korrektheit eines a-Übergangs im zu konstruierenden EAA)

Sei (Σ, I) ein trianguliertes Alphabet mit Unabhängigkeitsrelation und seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen von (Σ, I) in der durch Satz 4.2.5 gegebenen Reihenfolge. Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I). Für zwei Traces $t, r \in E(\Sigma, I)$ und jedes Zeichen $a \in \Sigma$ gilt:

$$\bigwedge_{i\in Dom(a)} S_{\{i,\ldots,n\}}(P_i(t)) \sim_T S_{\{i,\ldots,n\}}(P_i(r)) \Rightarrow \bigwedge_{i\in Dom(a)} S_{\{i,\ldots,n\}}(P_i(t[a])) \sim_T S_{\{i,\ldots,n\}}(P_i(r[a]))$$

Beweis:
Sei $Dom(a) = \alpha = \{i_1, \ldots, i_k\}$ mit $i_1 < \ldots < i_k$. Für $i \in \{i_2, \ldots, i_k\}$ gehört das letzte Vorkommen von a in $t[a]$ auch zu $P_i(t[a])$, und alle Zeichenvorkommen in $P_i(t[a])$ liegen bezüglich $<_{P_i(t[a])}$ vor diesem a-Vorkommen. Weil $Dom(a) \not\subseteq \{i, \ldots, n\}$ ist, gilt $S_{\{i,\ldots,n\}}(P_i(t[a])) = S_{\{i,\ldots,n\}}(P_i(r[a])) = [\varepsilon]$ für $i \in \{i_2, \ldots, i_k\}$. Es muß nun noch der Fall $i = i_1$ betrachtet werden.

Für $i \in \alpha$ ist $P_i(t[a]) = P_\alpha(t)[a]$. Wegen $\alpha = Dom(a) \subseteq \{i_1, \ldots, n\}$ ist

$$S_{\{i_1,\ldots,n\}}(P_i(t[a])) = S_{\{i_1,\ldots,n\}}(P_\alpha(t)[a]) = S_{\{i_1,\ldots,n\}}(P_\alpha(t))[a].$$

Es wird nun gezeigt, daß der Trace $S_{\{i_1,\ldots,n\}}(P_\alpha(t))$ eine Verkettung der $S_{\{i_j,\ldots,n\}}(P_{i_j}(t)), j = 1, \ldots, k$, ist, so daß die Prämisse der behaupteten Implikation ausgenutzt werden kann.

Durch Einschränkung des Graphen zu (Σ, D) auf die maximalen Cliquen Σ_j mit $j \in Dom(a)$ bekommen wir wieder einen triangulierten Graphen, weil in diesem Teilgraphen jeder Knoten

mit a benachbart ist. Die maximalen Cliquen Σ_j lassen sich daher nach Satz 4.2.5 ordnen. Die Folge $\langle \Sigma_{i_1}, \ldots, \Sigma_{i_k} \rangle$ als Teilfolge von $\langle \Sigma_1, \ldots, \Sigma_n \rangle$ ist eine solche Ordnung, die sich durch Einschränkung des Eliminationsschemas für das volle Alphabet (vgl. Beweis zu Satz 4.2.5) auf die Zeichen in den Σ_j mit $j \in Dom(a)$ ergibt. Wegen $P_\alpha(t) = P_{\{i_1,\ldots,i_k\}}(t)$ und $P_{i_j}(t) = P_{i_j}(P_\alpha(t))$ für $j = 1, \ldots, k$ folgt mit Satz 4.2.7:

$$P_\alpha(t) = P_{i_1}(t) S_{\overline{\{i1\}}}(P_{i_2}(t)) \ldots S_{\overline{\{i_1,\ldots,i_{k-1}\}}}(P_{i_k}(t))$$

(Komplement $\overline{\{x\}}$ hier bezüglich $\{1, \ldots, n\}$.) Durch einen Widerspruchsbeweis wird

$$S_{\overline{\{i_1,\ldots,i_j\}}}(P_{i_{j+1}}(t)) = S_{\{i_{j+1},\ldots,n\}}(P_{i_{j+1}}(t))$$

für $j = 1, \ldots, k-1$ gezeigt.

Es müssen nur die Mengen der Zeichenvorkommen auf Gleichheit geprüft werden.
$Vor(S_{\overline{\{i_1,\ldots,i_j\}}}(P_{i_{j+1}}(t))) \supseteq Vor(S_{\{i_{j+1},\ldots,n\}}(P_{i_{j+1}}(t)))$ ergibt sich aus $\overline{\{i_1, \ldots, i_j\}} \supseteq \{i_{j+1}, \ldots, n\}$.
Sei $x \in Vor(S_{\overline{\{i_1,\ldots,i_j\}}}(P_{i_{j+1}}(t)))$. Es gilt $x \leq_t last^{i_{j+1}}_{i_{j+1}}(t)$, und für jedes Vorkommen (b, m) mit $x \leq_t (b, m) \leq_t last^{i_{j+1}}_{i_{j+1}}(t)$ haben wir $Dom(b) \subseteq \overline{\{i_1, \ldots, i_j\}}$.
Annahme: $x \notin Vor(S_{\{i_{j+1},\ldots,n\}}(P_{i_{j+1}}(t)))$. Folglich gibt es ein Vorkommen (b, m) mit $x \leq_t (b, m) \leq_t last^{i_{j+1}}_{i_{j+1}}(t)$ und $Dom(b) \not\subseteq \{i_{j+1}, \ldots, n\}$. Es gibt also eine maximale D-Clique Σ_l mit $b \in \Sigma_l$ und $l < i_{j+1}$ und $l \notin \{i_1, \ldots, i_j\}$. Sei l o. B. d. A. der kleinste Index mit dieser Eigenschaft. Im Eliminationsschema wird a mit den Zeichen aus $\Sigma_{i_1} \setminus \bigcup_{p<i_1} \Sigma_p$ gelöscht. Wenn $(a, b) \in D$ ist, müßte nach den Regeln des Eliminationsschemas $b \in \Sigma_{i_1}$ oder $a \in \Sigma_l$ sein, da b mit $\Sigma_l \setminus \bigcup_{p<l} \Sigma_p$ eliminiert wird. Beides ist ausgeschlossen, weil i_{j+1} der kleinste Index p mit $a, b \in \Sigma_p$ ist. Folglich gilt $(a, b) \in I$.
Sei $(b, m) = (b_0, m_0) <_t (b_1, m_1) <_t \ldots <_t last^{i_{j+1}}_{i_{j+1}}(t)$ die kürzeste Folge von aufeinanderfolgenden abhängigen Zeichenvorkommen. Zeichen b_q in dieser Folge, die nach den Zeichen aus $\Sigma_{i_{j+1}} \setminus \bigcup_{p<i_{j+1}} \Sigma_p$ aus dem Graphen eliminiert werden, sind mit der gleichen Begründung wie für b mit a unabhängig. Sei (b_r, m_r) das erste Vorkommen in der Folge, dessen Zeichen b_r spätestens mit den $\Sigma_{i_{j+1}} \setminus \bigcup_{p<i_{j+1}} \Sigma_p$ gelöscht wird. b_{r-1} wird nach b_r eliminiert und es ist $(a, b_{r-1}) \in I$. b_r wird vor a aus dem Graphen zu $(\Sigma, D \setminus id)$ gelöscht, weil $a \in \Sigma_{i_1}$. Wenn $(a, b_r) \in D$ wäre, müßten b_{r-1} und a mit b_r in einer D-Clique sein, wenn b_r gelöscht wird. Also gilt $(a, b_r) \in I$ und daher wegen $a \in \Sigma_{i_{j+1}}$: $b_r \notin \Sigma_{i_{j+1}}$. b_{r+1} und alle in der Folge nachkommenden Zeichen müssen vor $\Sigma_{i_{j+1}} \setminus \bigcup_{p<i_{j+1}} \Sigma_p$ eliminiert werden, weil $(b_{r+1}, b_r) \in D$ und $(b_{r+1}, b_{r-1}) \in I$. Das ergibt den Widerspruch, weil das Zeichen zu $last^{i_{j+1}}_{i_{j+1}}(t)$ mit $\Sigma_{i_{j+1}} \setminus \bigcup_{p<i_{j+1}} \Sigma_p$ eliminiert wird.

Somit gilt $S_{\overline{\{i_1,\ldots,i_j\}}}(P_{i_{j+1}}(t)) = S_{\{i_{j+1},\ldots,n\}}(P_{i_{j+1}}(t))$ für $j = 1, \ldots, k-1$, und es kann geschrieben werden

$$P_\alpha(t) = P_{i_1}(t) S_{\{i_2,\ldots,n\}}(P_{i_2}(t)) \ldots S_{\{i_k,\ldots,n\}}(P_{i_k}(t)).$$

In den Traces $S_{\{i_2,\ldots,n\}}(P_{i_2}(t)), \ldots, S_{\{i_k,\ldots,n\}}(P_{i_k}(t))$ treten nur Vorkommen (b, m) mit $Dom(b) \subseteq \{i_1, \ldots, n\}$ auf. Daher gilt:

$$S_{\{i_1,\ldots,n\}}(P_\alpha(t)) = S_{\{i_1,\ldots,n\}}(P_{i_1}(t)) S_{\{i_2,\ldots,n\}}(P_{i_2}(t)) \ldots S_{\{i_k,\ldots,n\}}(P_{i_k}(t))$$

Derselbe Zusammenhang gilt für den Trace r.
Mit $\bigwedge_{i \in Dom(a)} S_{\{i,\ldots,n\}}(P_i(t)) \sim_T S_{\{i,\ldots,n\}}(P_i(r))$ bekommen wir also

$$S_{\{i_1,\ldots,n\}}(P_\alpha(t)) \sim_T S_{\{i_1,\ldots,n\}}(P_\alpha(r))$$

und $S_{\{i_1,\ldots,n\}}(P_i(t[a])) = S_{\{i_1,\ldots,n\}}(P_\alpha(t))[a] \sim_T S_{\{i_1,\ldots,n\}}(P_\alpha(r))[a] = S_{\{i_1,\ldots,n\}}(P_i(r[a]))$. □

Satz 4.2.9 (EAA-Konstruktion zu $T \in Erk(\Sigma, I)$ über trianguliertem (Σ, I))

Sei (Σ, I) ein trianguliertes Alphabet mit Unabhängigkeitsrelation und seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen von (Σ, I) in der Reihenfolge gemäß Satz 4.2.5. Sei $T \in Erk(\Sigma, I)$ eine erkennbare Trace-Sprache über (Σ, I).

Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ ein EAA mit

i) $\mathcal{P}_i =_{Df} (\Sigma_i, S_i)$ mit $S_i =_{Df} \{\langle S_{\{i,\ldots,n\}}(P_i(t))\rangle_T | t \in E(\Sigma, I)\}$ für $i \in \{1, \ldots, n\}$. Dabei bezeichnet $\langle t\rangle_T$ die Klasse der syntaktischen Kongruenz von $t \in E(\Sigma, I)$ bezüglich $T \subseteq E(\Sigma, I)$.

ii) $\mathcal{I} =_{Df} \{(s_1^0, \ldots, s_n^0)\}$ mit $s_i^0 =_{Df} \langle[\varepsilon]\rangle_T$ für $i = 1, \ldots, n$ ist der Anfangszustand.

iii) Sei $a \in \Sigma$ und $Dom(a) = \{i_1, \ldots, i_k\}$. $\mathcal{D} =_{Df} \{\delta_a | a \in \Sigma\}$ ist die Familie der Übergangsfunktionen mit
$\delta_a\Big(\langle S_{\{i_1,\ldots,n\}}(P_{i_1}(t))\rangle_T, \ldots, \langle S_{\{i_k,\ldots,n\}}(P_{i_k}(t))\rangle_T\Big) =_{Df}$

$$\Big\{\Big(\langle S_{\{i_1,\ldots,n\}}(P_{i_1}(t[a]))\rangle_T, \ldots, \langle S_{\{i_k,\ldots,n\}}(P_{i_k}(t[a]))\rangle_T\Big)\Big\}$$

iv) $\mathcal{F} =_{Df} \{(\langle S_{\{1,\ldots,n\}}(P_1(t))\rangle_T, \ldots, \langle S_{\{n\}}(P_n(t))\rangle_T) | t \in T\}$ ist die Menge der Endzustände.

$\mathcal{A}$ ist korrekt definiert und es gilt $T(\mathcal{A}) = T$.

Beweis:
Die Korrektheit der Definition der Übergangsfunktion in iii) wurde in Lemma 4.2.8 bewiesen.

Sei $t = [a_1 \ldots a_m] \in E(\Sigma, I)$. Durch Induktion wird gezeigt:
T überführt $\mathcal{A}$ in den Zustand $(\langle S_{\{1,\ldots,n\}}(P_1(t))\rangle_T, \ldots, \langle S_{\{n\}}(P_n(t))\rangle_T)$.
Für $m = 0$ ist $t = [\varepsilon]$ und $(\langle S_{\{1,\ldots,n\}}(P_1(t))\rangle_T, \ldots, \langle S_{\{n\}}(P_n(t))\rangle_T) = (\langle[\varepsilon]\rangle_T, \ldots, \langle[\varepsilon]\rangle_T)$. Gelte die Behauptung für $t = [a_1 \ldots a_m], m \geq 0$.
Ein Übergang mit a_{m+1} führt von $(\langle S_{\{1,\ldots,n\}}(P_1(t))\rangle_T, \ldots, \langle S_{\{n\}}(P_n(t))\rangle_T)$ aus gemäß $\delta_{a_{m+1}}$ in den Folgezustand $(s_1, \ldots, s_n)$ mit $s_i = \langle S_{\{i,\ldots,n\}}(P_i(t[a_{m+1}]))\rangle_T$ für $i \in Dom(a_{m+1})$. Für $i \notin Dom(a_{m+1})$ bleibt $\mathcal{P}_i$ beim a_{m+1}-Übergang im selben Zustand, also $s_i = \langle S_{\{i,\ldots,n\}}(P_i(t))\rangle_T$. Wegen $P_i(t[a_{m+1}]) = P_i(t)$ für $i \notin Dom(a_{m+1})$ überführt also $t[a_{m+1}]$ den Anfangszustand in den Zustand $(\langle S_{\{1,\ldots,n\}}(P_1(t[a_{m+1}]))\rangle_T, \ldots, \langle S_{\{n\}}(P_n(t[a_{m+1}]))\rangle_T)$.

Sei $t \in T$. Mit t erreicht $\mathcal{A}$ folglich einen Endzustand, so daß $t \in T(\mathcal{A})$.
Sei umgekehrt $t \in T(\mathcal{A})$. Für t gilt $s_0 \xrightarrow{t} s_f$ mit einem Endzustand $s_f \in \mathcal{F}$. Nach Definition

von $\mathcal{F}$ gibt es einen Trace $t' \in T$ mit $s_0 \stackrel{t'}{\Longrightarrow} s_f$. Nach Satz 4.2.7 können t und t' zerlegt werden:

$t = S_{\{1,\ldots,n\}}(P_1(t))S_{\{2,\ldots,n\}}(P_2(t)) \ldots S_{\{n\}}(P_n(t))$

$t' = S_{\{1,\ldots,n\}}(P_1(t'))S_{\{2,\ldots,n\}}(P_2(t')) \ldots S_{\{n\}}(P_n(t'))$

Weil t und t' den konstruierten EAA in denselben globalen Endzustand s_f überführen, stimmen die lokalen Zustände $\langle S_{\{i,\ldots,n\}}(P_i(t))\rangle_T$ und $\langle S_{\{i,\ldots,n\}}(P_i(t'))\rangle_T$ für alle $i \in \{1,\ldots,n\}$ überein. $S_{\{i,\ldots,n\}}(P_i(t))$ und $S_{\{i,\ldots,n\}}(P_i(t'))$ sind syntaktisch kongruent und somit auch t und t' selber. Mit $t' \in T$ ist schließlich ebenso $t \in T$. □

Beispiel: Gegeben ist $\Sigma = \{a, b, c, d\}$ mit der Abhängigkeitsrelation $D \setminus id = a - b - c - d$. Die maximalen D-Cliquen in der Reihenfolge nach Satz 4.2.5 sind $\Sigma_1 = \{a, b\},\ \Sigma_2 = \{b, c\}$ und $\Sigma_3 = \{c, d\}$.

Konstruiert werden soll ein EAA für die erkennbare Trace-Sprache $T = [(abcd)^*]$.

Zustände des Teilautomaten $\mathcal{P}_1$:

$s_1^0 = \langle[\varepsilon]\rangle_T,\ \ s_1^1 = \langle[a]\rangle_T,\ \ s_1^2 = \langle[ab]\rangle_T,\ \ s_1^3 = \langle[aba]\rangle_T,\ \ s_1^4 = \langle[abcab]\rangle_T,\ \ s_1^5 = \langle[abcaba]\rangle_T$

Zustände des Teilautomaten $\mathcal{P}_2$:

$s_2^0 = \langle[\varepsilon]\rangle_T,\ \ s_2^1 = \langle[c]\rangle_T,\ \ s_2^2 = \langle[dc]\rangle_T$

Zustände des Teilautomaten $\mathcal{P}_3$:

$s_3^0 = \langle[\varepsilon]\rangle_T,\ \ s_3^1 = \langle[d]\rangle_T$

(Es wurden nur die erreichbaren lokalen Zustände angegeben, die zu einem globalen Zustand gehören, von dem aus ein Endzustand erreichbar ist. Diese Zustände ergeben sich aus der Betrachtung der Traces in $Pref(T)$.)

Der Anfangszustand ist $\mathcal{I} = \{(s_1^0, s_2^0, s_3^0)\}$.
Die Menge der Endzustände ist $\mathcal{F} = \{(s_1^0, s_2^0, s_3^0), (s_1^2, s_2^1, s_3^1), (s_1^4, s_2^2, s_3^1)\}$.

Die Übergangsrelation geht aus dem Petrinetz in Abb. 4.8 hervor. $\mathcal{F}$ ist in Abb. 4.8 aus Gründen der Übersichtlichkeit nicht eingezeichnet.

Der konstruierte EAA ist nicht sicher, da er nach $[ac]$ verklemmt. Weil T synchronisiert ist, läßt sich durch Projektion von T auf die Teilalphabete und anschließende freie Synchronisation der zu den Projektionssprachen konstruierten endlichen Automaten ein einfacherer EAA konstruieren. Dieser kleinere EAA ergibt sich aus dem hier konstruierten EAA durch Streichen der lokalen Zustände $s_1^0, s_1^1, s_1^2, s_1^3$ und s_2^1 und mit $\mathcal{I}' = \mathcal{F}' = \{(s_1^4, s_2^2, s_3^1)\}$.

Zielonkas Konstruktion angesetzt auf diese Trace-Sprache ergibt einen größeren EAA, weil die Relation $\approx_E$ durch Numerierung der Zeichenvorkommen (vgl. Abb. 4.2) die Zahl der Äquivalenzklassen und somit die Zahl der Zustände erhöht.

Aus dem Beweis zu Lemma 4.2.8 wird deutlich, wie der konstruierte EAA arbeitet. Wenn ein Zeichen a akzeptiert wird, und es ist $Dom(a) = \{i_1, \ldots, i_k\}$ mit $i_1 < \ldots < i_k$, so werden die Teilautomaten $\mathcal{P}_{i_2}, \ldots, \mathcal{P}_{i_k}$ auf ihren Anfangszustand $\langle[\varepsilon]\rangle_T$ zurückgesetzt. Der Teilautomat $\mathcal{P}_{i_1}$ kann die Zustände der Teilautomaten $\mathcal{P}_{i_2}, \ldots, \mathcal{P}_{i_k}$ lesen und führt in Abhängigkeit

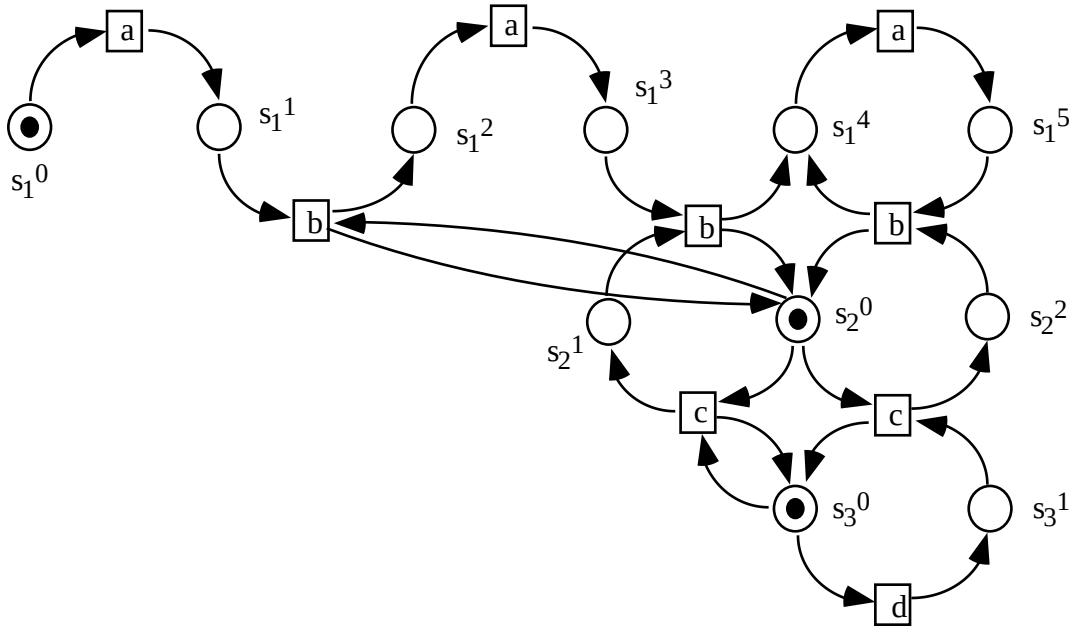

Abbildung 4.8: EAA für $T = [(abcd)^*]$

von diesen Zuständen einen Zustandsübergang aus. Es ergibt sich eine Hierarchie der Teilautomaten. Ein in der Hierarchie tiefer stehender Teilautomat $\mathcal{P}_j$ muß arbeiten, bis sein Zustand von einem höher stehenden Teilautomaten $\mathcal{P}_i, i < j$ abgelesen und er im Zustand zurückgesetzt wird.

4.3 Die modulare Konstruktion von nichtdeterministischen EAAs

Pighizzini präsentiert in [Pig93] zwei wichtige Ergebnisse: Er gibt eine algebraische Charakterisierung der erkennbaren Trace-Sprachen an, die ähnlich zu der von Ochmański ist. Darauf aufbauend formuliert er ein neues Konstruktionsverfahren für endliche asynchrone Automaten.

Der Unterschied zu Ochmańskis Charakterisierung der Sprachklasse liegt in der verwendeten Iterationsoperation. Pighizzini benutzt die gewöhnliche Iteration, wendet sie jedoch nur auf bestimmte Trace-Sprachen an.

Definition 4.3.1 (Eingeschränkte Iteration, [Pig93])

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation und sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache.

Die eingeschränkte Iteration von T ist

$$T^{\oplus} =_{Df} \begin{cases} T^{+} & \text{wenn } \bigvee_{\alpha\subseteq\Sigma} \bigwedge_{t\in T} Alph(t) = \alpha \wedge D|_{\alpha} \text{ ist zusammenhängend} \\ \emptyset & \text{sonst} \end{cases}$$

Die in der Definition für T verlangte Eigenschaft ist für erkennbare Trace-Sprachen entscheidbar: Alle in Frage kommenden Teilmengen α können in einem endlichen Verfahren erzeugt werden. Der nach Zielonka konstruierte EAA zu T kann für ein zu untersuchendes α so erweitert werden, daß er die in einem Trace vorkommenden Zeichen registriert und nur noch Traces $t \in T$ mit $Alph(t) = \alpha$ akzeptiert. Die Äquivalenz dieses modifizierten Automaten mit dem EAA zu T ist entscheidbar.

Satz 4.3.2 (Charakterisierung von $Erk(\Sigma, I)$, [Pig93])

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation. Die Klasse $Erk(\Sigma, I)$ der erkennbaren Trace-Sprachen über (Σ, I) ist die kleinste Klasse von Trace-Sprachen, die alle endlichen Trace-Sprachen über (Σ, I) enthält und bezüglich Vereinigung, Verkettung und eingeschränkter Iteration abgeschlossen ist.

Im Beweis dieses Satzes nutzt Pighizzini das eng verwandte Ergebnis von Ochmański aus. Er muß zusätzlich zeigen, daß die von Ochmański verwendete Iteration von zusammenhängenden erkennbaren Trace-Sprachen durch Vereinigung, Verkettung und eingeschränkte Iteration ausgedrückt werden kann.

Für jede der drei Operationen gibt Pighizzini ein Konstruktionsschema auf EAAs an. Wie schon bei den Konstruktionsschemata zu den rationalen Operationen für gewöhnliche endliche Automaten ist auch hier wieder die Verwendung von ε-Übergängen in den Automaten hilfreich.

Definition 4.3.3 (EAA mit ε-Übergängen, [Pig93])

Ein EAA mit ε-Übergängen bestehend aus n Teilautomaten ist ein $n+3$-Tupel $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit

i) $\mathcal{P}_i = (\Sigma_i \cup \{e_i\}, S_i)$, wobei S_i die Zustandsmenge und $\Sigma_i \cup \{e_i\}$ das Alphabet des Teilautomaten ist. e_i ist ein neues Zeichen, also $e_i \notin \Sigma_1 \cup \ldots \cup \Sigma_n$ und $e_i \neq e_j$ für $i \neq j$.

ii) Das Alphabet von $\mathcal{A}$ ist $\Sigma = \Sigma_1 \cup \ldots \cup \Sigma_n \cup \{e_1, \ldots, e_n\}$.
$\mathcal{I}, \mathcal{D}, \mathcal{F}, S$ und $I_{\mathcal{A}} \subseteq \Sigma \times \Sigma$ sind wie bei EAAs ohne ε-Übergängen (Def. 2.6.1) definiert.

iii) Die von $\mathcal{A}$ erkannte Trace-Sprache mit ε-Übergängen ist

$$T_{\varepsilon}(\mathcal{A}) =_{Df} \{t \in E(\Sigma, I_{\mathcal{A}}) | \bigvee_{s_0\in\mathcal{I}} \bigvee_{s\in\mathcal{F}} s_0 \stackrel{t}{\Longrightarrow} s\}$$

iv) Die von $\mathcal{A}$ erkannte Trace-Sprache ist $T(\mathcal{A}) =_{Df} T_\varepsilon(\mathcal{A})|_{\Sigma_1 \cup \ldots \cup \Sigma_n}$

Gewöhnliche EAAs können als spezielle „EAAs mit ε-Übergängen“ gesehen werden, in denen keine ε-Übergänge vorkommen. Trivialerweise kann demnach jede erkennbare Trace-Sprache auch von einem EAA mit ε-Übergängen erkannt werden. Die Umkehrung:

Satz 4.3.4 (Eliminierung von ε-Übergängen im EAA, [Pig93])

Sei $\mathcal{A}_\varepsilon$ ein EAA mit ε-Übergängen. Zu $\mathcal{A}_\varepsilon$ kann konstruktiv ein gewöhnlicher EAA $\mathcal{A}$ mit $T(\mathcal{A}) = T(\mathcal{A}_\varepsilon)$ angegeben werden.

Die Eliminierung der ε-Übergänge erfolgt nach dem gleichen Muster wie bei gewöhnlichen endlichen Automaten.

Mit diesen Vorbemerkungen kann nun eine modulare Konstruktion von EAAs entsprechend Satz 4.3.2 angegeben werden. Die endlichen Trace-Sprachen ergeben sich durch endlich häufige Anwendung der Vereinigung und Verkettung aus den „elementaren“ Trace-Sprachen $\emptyset, [\varepsilon]$ und $[a], a \in \Sigma$. Für diese Sprachen ist die Konstruktion eines EAAs trivial. Es müssen also nun die rationalen Operationen auf Trace-Sprachen betrachtet werden.

Satz 4.3.5 („Vereinigung“ von EAAs)

Seien $\mathcal{A}_1$ und $\mathcal{A}_2$ zwei EAAs, die über demselben Alphabet mit Unabhängigkeitsrelation (Σ, I) mit den Teilalphabeten $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ definiert sind. Zu $\mathcal{A}_1$ und $\mathcal{A}_2$ kann durch einfache Vereinigung ein EAA $\mathcal{A}_1 \cup \mathcal{A}_2$ konstruiert werden mit $T(\mathcal{A}_1 \cup \mathcal{A}_2) = T(\mathcal{A}_1) \cup T(\mathcal{A}_2)$.

Im Beweis dieses Satzes wird ausgenutzt, daß EAAs mehrere Anfangszustände haben können. Wenn $\mathcal{A}_1$ und $\mathcal{A}_2$ jeweils mindestens einen Anfangszustand haben, so ist $\mathcal{A}_1 \cup \mathcal{A}_2$ nichtdeterministisch, weil die Zustandsmengen von $\mathcal{A}_1$ und $\mathcal{A}_2$ sowie die Mengen der Anfangszustände disjunkt vereinigt werden.

Definition 4.3.6 („Verkettung“ von EAAs, [Pig93])

Seien $\mathcal{A}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, \mathcal{I}', \mathcal{D}', \mathcal{F}')$ und $\mathcal{A}'' = (\mathcal{P}''_1, \ldots, \mathcal{P}''_n, \mathcal{I}'', \mathcal{D}'', \mathcal{F}'')$ zwei EAAs nach Definition 2.6.1, wobei $\mathcal{P}'_i = (\Sigma_i, S'_i)$ und $\mathcal{P}''_i = (\Sigma_i, S''_i)$ für $i = 1, \ldots, n$.
Sei $\mathcal{F}' = \{(s_f^{1\prime}, \ldots, s_f^{n\prime})\}$ und sei $\mathcal{I}'' = \{(s_0^{1\prime\prime}, \ldots, s_0^{n\prime\prime})\}$.
$\mathcal{A} = \mathcal{A}' \circ \mathcal{A}'' = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ ist der durch Verkettung von $\mathcal{A}'$ und $\mathcal{A}''$ entstehende EAA mit ε-Übergängen, mit

i) $\mathcal{P}_i = (\Sigma_i \cup \{e_i\}, S_i)$, wobei $S_i = S'_i \cup S''_i$ für $i = 1, \ldots, n$ und e_i ein neues Zeichen ist.

ii) $\mathcal{I} =_{Df} \mathcal{I}', \quad \mathcal{F} =_{Df} \mathcal{F}''$

iii) Für $a \in \Sigma_1 \cup \ldots \cup \Sigma_n$ sei $Dom(a) = \{i_1, \ldots, i_k\}$. Es wird δ_a definiert:

$$\delta_a(s_{i_1}, \ldots, s_{i_k}) =_{Df} \begin{cases} \delta'_a(s_{i_1}, \ldots, s_{i_k}) & \text{für } (s_{i_1}, \ldots, s_{i_k}) \in S'_{i_1} \times \ldots \times S'_{i_k} \\ \delta''_a(s_{i_1}, \ldots, s_{i_k}) & \text{für } (s_{i_1}, \ldots, s_{i_k}) \in S''_{i_1} \times \ldots \times S''_{i_k} \\ \emptyset & \text{sonst} \end{cases}$$

iv) Für $e_i, i = 1, \ldots, n$, wird definiert: $\delta_{e_i}(s_f^{i\prime}) = \{s_0^{i\prime\prime}\}$ und $\delta_{e_i}(x) = \emptyset$ sonst.

Man beachte, daß der erste EAA in der Verkettung nur einen Endzustand und der zweite EAA nur einen Anfangszustand hat. Das reicht zusammen mit Satz 4.3.5 für unsere Zwecke aus, weil die Sprache eines EAAs mit mehreren Anfangs- und Endzuständen die Vereinigung der Sprachen von strukturgleichen EAAs mit allen einelementigen Kombinationen von Anfangs- und Endzuständen ist.

Satz 4.3.7 (Verkettung von EAAs, [Pig93])

Seien $\mathcal{A}', \mathcal{A}''$ und $\mathcal{A}' \circ \mathcal{A}''$ wie in Definition 4.3.6 definiert. Es gilt:

$$T_\varepsilon(\mathcal{A}' \circ \mathcal{A}'') = T(\mathcal{A}')[e_1 \ldots e_n]T(\mathcal{A}''),$$

also insbesondere

$$T(\mathcal{A}' \circ \mathcal{A}'') = T(\mathcal{A}')T(\mathcal{A}'').$$

Die ε-Übergänge zwischen dem ersten und dem zweiten EAA sind nötig, um zu verhindern, daß eine akzeptierende Zustandsfolge von $\mathcal{A}''$ auf $\mathcal{A}'$ zurückwechselt, wenn der Endzustand von $\mathcal{A}'$ und der Anfangszustand von $\mathcal{A}''$ von sich selber aus erreichbar sind. Im Beweis des Satzes ist hauptsächlich zu zeigen, daß in einer akzeptierten Zeichenfolge von $\mathcal{A}' \circ \mathcal{A}''$ die Vorkommen entsprechend $T(\mathcal{A}')[e_1 \ldots e_n]T(\mathcal{A}'')$ geordnet werden können.

Der Konstruktionsschritt für die eingeschränkte Iteration einer erkennbaren Trace-Sprache ist wesentlich komplizierter. Weil ein EAA zur leeren Trace-Sprache sofort angegeben werden kann, muß die Konstruktion nur für Trace-Sprachen T mit $\bigwedge_{t\in T} \bigvee_{\alpha\subseteq\Sigma} Alph(t) = \alpha \wedge D|_\alpha$ *zusammenhängend* durchgeführt werden. O. B. d. A. reicht es aus, Sprachen mit $\bigwedge_{t\in T} Alph(t) = \Sigma$ über einem zusammenhängenden Alphabet zu betrachten.

Pighizzini erklärt die Konstruktion in zwei Schritten: Nachdem er den Spezialfall $|\mathcal{I}| = |\mathcal{F}| = 1$ erläutert hat, geht er zu EAAs mit beliebigen Anfangs- und Endzustandsmengen über. In einem ersten Ansatz für den Fall $|\mathcal{I}| = |\mathcal{F}| = 1$ würde die Iteration eines EAAs realisiert werden, indem jeder Teilautomat $\mathcal{P}_i$ mit einem ε-Übergang vom Endzustand in den Anfangszustand zurückspringt. Diese Strategie schlägt fehl, wie Pighizzini an einem Beispiel zeigt. Das Problem wird durch eine Verdoppelung des gegebenen EAAs gelöst:

Definition 4.3.8 (Iteration eines EAAs mit $|\mathcal{I}| = |\mathcal{F}| = 1$)

Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ ein EAA mit $\mathcal{P}_i = (\Sigma_i, S_i)$ für $i = 1, \ldots, n$, $\mathcal{I} = \{(s_0^1, \ldots, s_0^n)\}$ und $\mathcal{F} = \{(s_f^1, \ldots, s_f^n)\}$.
Zu $\mathcal{A}$ wird der Iterationsautomat $\mathcal{A}^+ = (\mathcal{P}_1', \ldots, \mathcal{P}_n', \mathcal{I}', \mathcal{D}', \mathcal{F}')$ mit ε-Übergängen definiert:

i) $\mathcal{P}_i' =_{Df} (\Sigma_i \cup \{e_i\}, S_i')$ mit $S_i' =_{Df} S_i \times \{0, 1\}$ für $i = 1, \ldots, n$

ii) $\mathcal{I}' =_{Df} \{((s_0^1, 0), \ldots, (s_0^n, 0))\}$

iii) *Sei* $a \in \Sigma_1 \cup \ldots \cup \Sigma_n$, $Dom(a) = \{i_1, \ldots, i_k\}$, $(s_1, \ldots, s_k) \in S_{i_1} \times \ldots \times S_{i_k}$ *und* $(b_1, \ldots, b_k) \in \{0,1\}^k$. δ'_a *wird definiert als*
$\delta'_a((s_1, b_1), \ldots, (s_k, b_k)) =_{Df}$

$$=_{Df} \begin{cases} \{((u_1, b_1) \ldots, (u_k, b_k))|(u_1, \ldots, u_k) \in \delta_a(s_1, \ldots, s_k)\} & \text{wenn } b_1 = \ldots = b_k \\ \emptyset & \text{sonst} \end{cases}$$

iv) *Für* $i = 1, \ldots, n$ *und* $b \in \{0,1\}$ *ist*

$$\delta'_{e_i}(s^i_f, b) =_{Df} \{s^i_0, 1-b\}$$

und $\delta'_{e_i}(x) = \emptyset$ *sonst.*

v) $\mathcal{F}' =_{Df} \{((s^1_f, b), \ldots, (s^n_f, b))|b \in \{0,1\}\}$

Satz 4.3.9 (Iteration eines EAAs mit $|\mathcal{I}| = |\mathcal{F}| = 1$, [Pig93])

Sei $\mathcal{A}$ *ein EAA mit* $|\mathcal{I}| = |\mathcal{F}| = 1$ *und gelte* $\bigwedge_{t \in T(\mathcal{A})} Alph(t) = \Sigma$. *Sei* $(\Sigma, D_{\mathcal{A}})$ *zusammenhängend.*
Für den in Definition 4.3.8 konstruierten Iterationsautomaten $\mathcal{A}^+$ *mit* ε*-Übergängen gilt:*

$$T_\varepsilon(\mathcal{A}^+) = \{t_1[e_1 \ldots e_n]t_2 \ldots [e_1 \ldots e_n]t_m | m \geq 1 \wedge t_1, \ldots, t_m \in T(\mathcal{A})\}$$

Die Eigenschaft $\bigwedge_{t \in T(\mathcal{A})} Alph(t) = \Sigma$ und der Zusammenhang von $(\Sigma, D_{\mathcal{A}})$ werden im Beweis ausgenutzt, um zu zeigen, daß zwischen zwei Iterationen $t_j, t_{j+1} \in T(\mathcal{A})$ in allen Teilautomaten ein ε-Übergang stattfindet.

Bei mehreren Anfangs- und Endzuständen werden für jedes Paar aus $\mathcal{I} \times \mathcal{F}$ zwei zum gegebenen EAA strukturgleiche Automaten definiert. Die Übergänge zwischen diesen EAAs sind ε-Übergänge.

Definition 4.3.10 (Iteration eines EAAs, [Pig93])

Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ *ein EAA mit* $\mathcal{P}_i = (\Sigma_i, S_i)$ *für* $i = 1, \ldots, n$.
Die Iteration von $\mathcal{A}$ *ist* $\mathcal{A}^+ = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, \mathcal{I}', \mathcal{D}', \mathcal{F}')$ *mit*

i) $\mathcal{P}'_i =_{Df} (\Sigma_i \cup \{e_i\}, S'_i)$, *wobei* $S'_i =_{Df} S_i \times \mathcal{I} \times \mathcal{F} \times \{0,1\}$ *für* $i = 1, \ldots, n$

ii) $\mathcal{I}' =_{Df} \{((s^1_0, s_0, s_f, 0), \ldots, (s^n_0, s_0, s_f, 0))|s_0 = (s^1_0, \ldots, s^n_0) \in \mathcal{I}, s_f \in \mathcal{F}\}$

iii) *Für* $a \in \Sigma_1 \cup \ldots \cup \Sigma_n$ *wird* δ'_a *definiert:*
Sei $Dom(a) = \{i_1, \ldots, i_k\}$, $(s_1, \ldots, s_k) \in S_{i_1} \times \ldots \times S_{i_k}$, $s_0 \in \mathcal{I}$, $s_f \in \mathcal{F}$ *und* $b \in \{0,1\}$.
Dann ist

$\delta'_a((s_1, s_0, s_f, b), \ldots, (s_k, s_0, s_f, b))$

$$= \{((u_1, s_0, s_f, b), \ldots, (u_k, s_0, s_f, b))|(u_1, \ldots, u_k) \in \delta_a(s_1, \ldots, s_k)\}$$

und $\delta'_a(s'_1, \ldots, s'_k) = \emptyset$ *sonst.*

iv) *Für* $i = 1, \ldots, n$, $s_i \in S_i$, $s_0 \in \mathcal{I}$, $s_f = (s_f^1, \ldots, s_f^n) \in \mathcal{F}$ *und* $b \in \{0,1\}$ *ist* $\delta'_{e_i}(s_i, s_0, s_f, b) =_{Df}$

$$=_{Df} \begin{cases} \{(s_0^{i\prime}, s'_0, s'_f, 1-b) | (s_0^{1\prime}, \ldots, s_0^{n\prime}) = s'_0 \in \mathcal{I} \wedge s'_f \in \mathcal{F}\} & \textit{wenn } s_i = s_f^i \\ \emptyset & \textit{sonst} \end{cases}$$

v) $\mathcal{F}' =_{Df} \{((s_f^1, s_0, s_f, b), \ldots, (s_f^n, s_0, s_f, b)) | s_0 \in \mathcal{I}, s_f = (s_f^1, \ldots, s_f^n) \in \mathcal{F}, b \in \{0,1\}\}$

Satz 4.3.9 kann mit dieser Definition auf EAAs mit mehreren Anfangs- und Endzuständen verallgemeinert werden. Der konstruierte Automat erkennt $T(\mathcal{A})^+$, wenn $T(\mathcal{A})$ auf einem zusammenhängenden Alphabet definiert ist und $\bigwedge_{t \in T(\mathcal{A})} Alph(t) = \Sigma$ gilt. Somit läßt sich zu jeder erkennbaren Trace-Sprache, die durch einen regulären Ausdruck mit Operatoren für Vereinigung, Verkettung und eingeschränkte Iteration gegeben ist, ein nichtdeterministischer EAA konstruieren. Mit jedem Konstruktionsschritt wächst die Anzahl der Zustände des EAAs dabei höchstens polynomiell.

Wie schon Zielonka geht Pighizzini auf die Repräsentation der erkennbaren Trace-Sprachen nicht ein. Es ist nicht zu erkennen, wie das Verfahren auf eine durch einen Automaten oder durch einen gewöhnlichen regulären Ausdruck gegebene erkennbare Trace-Sprache angewandt werden kann. Demgegenüber steht die Einfachheit von Pighizzinis Konstruktion: Ihr Prinzip ist leichter zu durchschauen.

Kapitel 5

Teilklassen der erkennbaren Trace-Sprachen

In Abschnitt 3.4 wurden bereits Teilklassen der erkennbaren Trace-Sprachen definiert. In diesem Kapitel werden weitere Eigenschaften dieser Sprachklassen herausgearbeitet und Beziehungen zu noch zu definierenden Klassen hergestellt.

5.1 Abschlußeigenschaften von $SYN(\Sigma, I)$

Eine synchronisierte Trace-Sprache ist als Synchronisation von regulären Wortsprachen darstellbar (Def. 3.4.4). Es gilt das folgende Lemma:

Lemma 5.1.1 (Abschluß von SYN gegen Synchronisation)

Seien $T' \in SYN(\Sigma', I')$ und $T'' \in SYN(\Sigma'', I'')$ zwei synchronisierte Trace-Sprachen. Sei (Σ, I) das Alphabet mit Unabhängigkeitsrelation, das bei der Synchronisation von Trace-Systemen über (Σ', I') und (Σ'', I'') entsteht.
Es gilt:

$$T'||T'' \in SYN(\Sigma, I)$$

Beweis:
$T'||T''$ ist nach Satz 3.1.3 eine erkennbare Trace-Sprache.
Seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen Cliquen der Abhängigkeitsrelation $D = \Sigma^2 \setminus I$. Für $T =_{Df} T'\|T''$ ist zu zeigen: $T = T|_{(\Sigma_1,\emptyset)}\| \ldots \|T|_{(\Sigma_n,\emptyset)}$.
$\subseteq$: Sei $t \in T$. Für $i = 1, \ldots, n$ gilt $t|_{(\Sigma_i,\emptyset)} \in T|_{(\Sigma_i,\emptyset)}$ und somit $t \in T|_{(\Sigma_1,\emptyset)}\| \ldots \|T|_{(\Sigma_n,\emptyset)}$.
$\supseteq$: Sei $t \in T|_{(\Sigma_1,\emptyset)}\| \ldots \|T|_{(\Sigma_n,\emptyset)}$, also $t|_{(\Sigma_i,\emptyset)} \in T|_{(\Sigma_i,\emptyset)}$ für alle $i \in \{1, \ldots, n\}$.
Seien $\Sigma'_1, \ldots, \Sigma'_m \subseteq \Sigma'$ die maximalen Cliquen der Abhängigkeitsrelation $D' = \Sigma'^2 \setminus I'$. Laut Definition der Synchronisation ist $D' \subseteq D|_{\Sigma'}$. Folglich sind zwei in (Σ', I') abhängige Zeichen in (Σ, I) ebenfalls abhängig. Somit ist jede maximale Clique $\Sigma'_j, j \in \{1, \ldots, m\}$,

abhängiger Zeichen in (Σ', I') Teilmenge einer maximalen Clique $\Sigma_i \subseteq \Sigma$, und demnach ist mit $t|_{(\Sigma_i,\emptyset)} \in T|_{(\Sigma_i,\emptyset)}$

$$t|_{(\Sigma'_j,\emptyset)} = t|_{(\Sigma_i,\emptyset)}|_{(\Sigma'_j,\emptyset)} \in T|_{(\Sigma_i,\emptyset)}|_{(\Sigma'_j,\emptyset)} = T|_{(\Sigma'_j,\emptyset)}.$$

Andererseits gilt wegen $\Sigma'_j \subseteq \Sigma'$

$$t|_{(\Sigma',I')}|_{(\Sigma'_j,\emptyset)} = t|_{(\Sigma'_j,\emptyset)} \text{ und } T|_{(\Sigma',I')}|_{(\Sigma'_j,\emptyset)} = T|_{(\Sigma'_j,\emptyset)}.$$

Folglich gilt $t|_{(\Sigma',I')}|_{(\Sigma'_j,\emptyset)} \in T|_{(\Sigma',I')}|_{(\Sigma'_j,\emptyset)}$.
Wegen $T = T'\|T''$ ist $T|_{(\Sigma',I')} \subseteq T'$ und somit $t|_{(\Sigma',I')}|_{(\Sigma'_j,\emptyset)} \in T'|_{(\Sigma'_j,\emptyset)}$ für jede maximale D-Clique Σ'_j in (Σ', I'). Weil T' synchronisiert ist, bekommen wir also $t|_{(\Sigma',I')} \in T'$. Die gleiche Argumentation für T'' führt zu $t|_{(\Sigma'',I'')} \in T''$. Beides zusammen ergibt schließlich $t \in T$, weil die Synchronisation als $T = T'\|T'' = \{t \in E(\Sigma, I) | t|_{(\Sigma',I')} \in T' \wedge t|_{(\Sigma'',I'')} \in T''\}$ definiert ist. □

Lemma 5.1.2 (Abschluß von $SYN(\Sigma, I)$ gegen Schnittbildung)

Seien $T', T'' \in SYN(\Sigma, I)$ zwei synchronisierte Trace-Sprachen.
Es gilt:

$$T' \cap T'' \in SYN(\Sigma, I)$$

Beweis:
$T' \cap T''$ ist nach Satz 3.1.3 eine erkennbare Trace-Sprache.
Seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen von (Σ, I).
Für $T =_{Df} T' \cap T''$ ist zu zeigen: $T = T|_{(\Sigma_1,\emptyset)}\| \ldots \|T|_{(\Sigma_n,\emptyset)}$.
$\subseteq$: Diese Teilmengenbeziehung ist stets trivial (vgl. Lemma 5.1.1).
$\supseteq$: Sei $t \in T|_{(\Sigma_1,\emptyset)}\| \ldots \|T|_{(\Sigma_n,\emptyset)}$, also $t|_{(\Sigma_i,\emptyset)} \in T|_{(\Sigma_i,\emptyset)}$ für alle $i \in \{1, \ldots, n\}$. Mithin gibt es einen Trace $r_i \in T$ mit $t|_{(\Sigma_i,\emptyset)} = r_i|_{(\Sigma_i,\emptyset)}$. Wegen $T = T' \cap T''$ gehört r_i auch zu T' und T'', folglich ist $t|_{(\Sigma_i,\emptyset)} \in T'|_{(\Sigma_i,\emptyset)}$ und $t|_{(\Sigma_i,\emptyset)} \in T''|_{(\Sigma_i,\emptyset)}$. Demnach liegt t sowohl in $T'|_{(\Sigma_1,\emptyset)}\| \ldots \|T'|_{(\Sigma_n,\emptyset)}$ als auch in $T''|_{(\Sigma_1,\emptyset)}\| \ldots \|T''|_{(\Sigma_n,\emptyset)}$. Weil T' und T'' beide synchronisiert sind, gilt $t \in T'$ und $t \in T''$, schließlich $t \in T$. □

Lemma 5.1.3 (Abschluß von $SYN(\Sigma, I)$ gegen Spiegelung)

Sei $T \in SYN(\Sigma, I)$ eine synchronisierte Trace-Sprache.
Dann ist die Trace-Spiegelsprache T^R ebenfalls synchronisiert.

Beweis:
Es ist $Lin(T^R) = Lin(T)^R$. Mit $Lin(T)$ ist auch $Lin(T)^R$ regulär und folglich T^R eine erkennbare Trace-Sprache.
Seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen von (Σ, I). Sei $t \in T^R|_{(\Sigma_1,\emptyset)}\| \ldots \|T^R|_{(\Sigma_n,\emptyset)}$. Die Definition der Synchronisation ergibt somit $t|_{(\Sigma_i,\emptyset)} \in T^R|_{(\Sigma_i,\emptyset)}$ und $t|_{(\Sigma_i,\emptyset)} = r_i|_{(\Sigma_i,\emptyset)}$ mit $r_i \in T^R$ für alle $i \in \{1, \ldots, n\}$. Es gilt $t^R|_{(\Sigma_i,\emptyset)} = (t|_{(\Sigma_i,\emptyset)})^R = (r_i|_{(\Sigma_i,\emptyset)})^R = r_i^R|_{(\Sigma_i,\emptyset)}$. Wegen $r_i^R \in (T^R)^R = T$ ist $t^R|_{(\Sigma_i,\emptyset)} \in T|_{(\Sigma_i,\emptyset)}$ für alle $i \in \{1, \ldots, n\}$, und aufgrund der Synchronisiertheit von T somit $t^R \in T$. Daraus ergibt sich schließlich $t = (t^R)^R \in T^R$. □

Das Beispiel $T = \{[a], [b]\} \subseteq E(\{a, b\}, \{(a, b), (b, a)\})$ hat bereits gezeigt, daß $SYN(\Sigma, I)$ mit $I \neq \emptyset$ gegen Vereinigung nicht abgeschlossen ist. Mit Lemma 5.1.2 gilt

Lemma 5.1.4 (Vereinigung und Komplement auf $SYN(\Sigma, I)$)

Die Klasse $SYN(\Sigma, I)$ ist gegen Vereinigung und Komplement abgeschlossen $\iff$ $I = \emptyset$

Beweis:
Nachdem der Fall $I \neq \emptyset$ klar ist, muß nur noch $I = \emptyset$ betrachtet werden. Wenn es nur abhängige Zeichen gibt, gilt $SYN(\Sigma, I) = Erk(\Sigma, I)$. Die erkennbaren Trace-Sprachen sind nach Lemma 3.1.1 gegen Vereinigung und Komplement abgeschlossen. □

Im folgenden Lemma werden Abschlußeigenschaften für Operationen auf Trace-Sprachen über Alphabeten mit nichttrivialen Unabhängigkeitsrelationen betrachtet.

Lemma 5.1.5 (Präfixabschluß und Verkettung auf $SYN(\Sigma, I)$)

Die Klasse $SYN(\Sigma, I)$ ist gegen Präfixabschluß bzw. Verkettung abgeschlossen
$\iff$ *$D = \Sigma^2 \setminus I$ ist transitiv*

Beweis:
$\Rightarrow$: Wenn die Abhängigkeitsrelation D nicht transitiv ist, gibt es drei Zeichen $a, b, c \in \Sigma$ mit $(a, c), (b, c) \in D$ und $(a, b) \in I$. Seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen der Unabhängigkeitsrelation und sei o. B. d. A. $a, c \in \Sigma_1$ und $b, c \in \Sigma_2$. Die Zeichen a, b und c können in weiteren maximalen D-Cliquen vorkommen, jedoch wird dadurch die folgende Argumentation nicht berührt.

i) Betrachtet wird die Trace-Sprache $T = \{[ac], [b]\} \in Erk(\Sigma, I)$. Die Projektionen auf Σ_1 und Σ_2 ergeben:

$$\begin{aligned} T_1 &=_{Df} T|_{\Sigma_1} = \{[ac], [\varepsilon]\} \\ T_2 &=_{Df} T|_{\Sigma_2} = \{[c], [b]\} \end{aligned}$$

Von den vier zu betrachtenden Trace-Kombinationen sind nur zwei synchronisierbar:

$$\begin{aligned} [ac] \| [c] &= [ac] \\ [\varepsilon] \| [b] &= [b] \end{aligned}$$

Durch Projektion auf die maximalen D-Cliquen und anschließende Synchronisation ergeben sich, selbst wenn alle D-Cliquen betrachtet werden, höchstens wieder die Traces $[ac]$ und $[b]$. Somit ist T synchronisiert.
Sei $T' = Pref(T) = \{[\varepsilon], [a], [b], [ac]\}$. Die Projektionen:

$$\begin{aligned} T'_1 &=_{Df} T'|_{\Sigma_1} = \{[\varepsilon], [a], [ac]\} \\ T'_2 &=_{Df} T'|_{\Sigma_2} = \{[\varepsilon], [b], [c]\} \end{aligned}$$

Für $a \in \Sigma_i$ enthält $T'|_{\Sigma_i}$ den Trace $[a]$.
Für $b \in \Sigma_i$ enthält $T'|_{\Sigma_i}$ den Trace $[b]$.
Ferner gilt $[\varepsilon] \in T'|_{\Sigma_i}$ für alle $i = 1, \ldots, n$. Also kann sich $[ab]$ durch Synchronisation aller $T'|_{\Sigma_i}$, $i = 1, \ldots, n$, ergeben. Weil aber $[ab] \notin T'$ ist, ist $T' = Pref(T)$ nicht synchronisiert.

ii) Für die Verkettung werden die Sprachen $T_1 = [a^*bc]$ und $T_2 = [a^*]$ betrachtet. T_1 und T_2 sind synchronisiert. Die Trace-Sprache $T_1 \circ T_2 = [a^*bca^*]$ ist nicht synchronisiert, denn $[bac]|_{\Sigma_1} \in (T_1 \circ T_2)|_{\Sigma_1}$ und $[bac]|_{\Sigma_2} \in (T_1 \circ T_2)|_{\Sigma_2}$, aber $[bac] \notin T_1 \circ T_2$.

$\Leftarrow$: Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation mit transitivem $D = \Sigma^2 \setminus I$. Seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die (paarweise disjunkten) maximalen D-Cliquen in (Σ, I).

i) Sei T eine synchronisierte Trace-Sprache. Zu zeigen ist

$$Pref(T) = Pref(T)|_{\Sigma_1} \| \ldots \| Pref(T)|_{\Sigma_n}$$

Die Inklusion $\subseteq$ gilt stets für die Synchronisation.
Sei also $t \in Pref(T)|_{\Sigma_1} \| \ldots \| Pref(T)|_{\Sigma_n}$. Dann gilt $t|_{\Sigma_i} \in Pref(T)|_{\Sigma_i}$ für alle $i = 1, \ldots, n$. Weil Projektion und Verkettung harmonieren, haben wir ebenfalls $t|_{\Sigma_i} \in Pref(T|_{\Sigma_i})$. Es gibt also einen Trace $t_i \in E(\Sigma_i, \emptyset)$ mit $t|_{\Sigma_i} t_i \in T|_{\Sigma_i}$. Weil die Teilalphabete Σ_i paarweise disjunkt sind, ergibt die Synchronisation der $t|_{\Sigma_i} t_i$ den Trace $t|_{\Sigma_1} \ldots t|_{\Sigma_n} t_1 \ldots t_n$. Da T synchronisiert ist, gilt insbesondere $t|_{\Sigma_1} \ldots t|_{\Sigma_n} t_1 \ldots t_n \in T$. Wegen $\Sigma_i \times \Sigma_j \subseteq I$ für alle $i, j \in \{1, \ldots, n\}$ mit $i \neq j$ ist $t = t|_{\Sigma_1} \ldots t|_{\Sigma_n}$, also $t \in Pref(T)$.

ii) Seien $T_1, T_2 \in SYN(\Sigma, I)$ synchronisierte Trace-Sprachen. Zu zeigen ist

$$T_1 \circ T_2 = (T_1 \circ T_2)|_{\Sigma_1} \| \ldots \| (T_1 \circ T_2)|_{\Sigma_n}.$$

Sei $t \in (T_1 \circ T_2)|_{\Sigma_1} \| \ldots \| (T_1 \circ T_2)|_{\Sigma_n}$. Dann gibt es Traces $t_1, \ldots, t_n \in T_1$ und $t'_1, \ldots, t'_n \in T_2$ mit $t|_{\Sigma_i} = (t_i t'_i)|_{\Sigma_i}$ für $i = 1, \ldots, n$. Da die Σ_i disjunkt sind, ist

$$t = t|_{\Sigma_1} \ldots t|_{\Sigma_n} = (t_1 t'_1)|_{\Sigma_1} \ldots (t_n t'_n)|_{\Sigma_n} = t_1|_{\Sigma_1} \ldots t_n|_{\Sigma_n} t'_1|_{\Sigma_1} \ldots t'_n|_{\Sigma_n}.$$

Wiederum weil die Σ_i disjunkt und T_1 und T_2 synchronisiert sind, ist $t_1|_{\Sigma_1} \ldots t_n|_{\Sigma_n} \in T_1$ und $t'_1|_{\Sigma_1} \ldots t'_n|_{\Sigma_n} \in T_2$, demnach $t \in T_1 \circ T_2$. □

5.2 Abschlußeigenschaften von $SKA(\Sigma, I)$

In [Ziel87] wurde bereits die Abgeschlossenheit der Sprachklasse bezüglich Vereinigung, Schnitt und Synchronisation gezeigt (hier Satz 3.4.9). Zu untersuchen sind noch das Komplement, die Verkettung und der Präfixabschluß.

Lemma 5.2.1 (Abschluß von $SKA(\Sigma, I)$ gegen Komplement)

Sei $T \in SKA(\Sigma, I)$ eien Trace-Sprache, die von einem schwach kooperierenden EAA erkannt wird.
Dann gibt es zu $\overline{T} =_{Df} E(\Sigma, I) \setminus T$ einen schwach kooperierenden EAA $\mathcal{A}$ mit $I_{\mathcal{A}} = I$ und $T(\mathcal{A}) = \overline{T}$.

Beweis:
Zu T gibt es nach Satz 3.4.7 einen deterministischen schwach kooperierenden EAA in Normalform. Sei $\mathcal{A}_T = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\mathcal{P}_i = (\Sigma_i, S_i)$ dieser Automat. Nach Definition 3.4.6 können die Übergangsrelationen $\delta_a, a \in \Sigma$, durch auf den Teilautomaten definierte lokale Übergangsrelationen $\delta_j, j \in \{1, \ldots, n\}$, dargestellt werden. O. B. d. A. kann von $|\delta_j(s,a)| \leq 1$ für alle $a \in \Sigma, j \in Dom(a)$ und $s \in S_j$ ausgegangen werden, denn $|\delta_j(s,a)| > 1$ führt zum Widerspruch mit der Determiniertheit von $\mathcal{A}_T$, wenn es in jedem Teilautomaten aus $Dom(a)$ mindestens einen a-Übergang gibt, und $\delta_j(s,a)$ kann $\emptyset$ gesetzt werden andernfalls. Auf der Grundlage der Relationen δ_j wird der EAA zu einem vollständigen Automaten $\mathcal{A}'_T = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, \mathcal{I}, \mathcal{D}', \mathcal{F})$ ergänzt:

i) Die Teilautomaten werden um einen neuen Zustand s_i^q erweitert:

$$\mathcal{P}'_i = (\Sigma_i, S'_i) \text{ mit } S'_i = S_i \cup \{s_i^q\}$$

ii) Die Schar von Übergangsrelationen $\delta'_j, j \in \{1, \ldots, n\}$, des neuen Automaten wird definiert als

$$\delta'_j(s,a) = \begin{cases} \delta_j(s,a) & falls \quad s \in S_j \wedge |\delta_j(s,a)| = 1 \\ \{s_j^q\} & falls \quad (s \in S_j \wedge \delta_j(s,a) = \emptyset) \vee s = s_j^q \end{cases}$$

für alle $a \in \Sigma, j \in Dom(a)$ und $s \in S'_j$.

Die neuen Übergangsrelationen $\delta'_a, a \in \Sigma$, lassen sich durch Produktbildung aus den $\delta'_j, j \in \{1, \ldots, n\}$, berechnen ($Dom(a) = \{i_1, \ldots, i_k\}$):

$$\delta'_a(s_{i_1}, \ldots, s_{i_k}) = \prod_{j \in \{i_1, \ldots, i_k\}} \delta'_j(s_j, a).$$

Der Automat $\mathcal{A}'_T$ ist deterministisch, weil alle $\delta'_j(s,a)$ und somit alle $\delta'_a(s_{i_1}, \ldots, s_{i_k})$ einelementig sind. Aus der Definition der δ'_j ist ebenfalls abzulesen, daß ein Teilautomat $\mathcal{P}'_i$, der während der Rechnung in den Zustand s_i^q gerät, diesen Zustand nicht mehr verlassen kann. Wenn $\mathcal{A}'_T$ ein Eingabewort akzeptiert, wird also keiner der globalen Zustände in der akzeptierenden Zustandsfolge einen der neuen Zustände enthalten, so daß das Eingabewort ebenso von $\mathcal{A}_T$ akzeptiert wird. Umgekehrt werden alle von $\mathcal{A}_T$ akzeptierbaren Wörter auch $\mathcal{A}'_T$ in einen Endzustand überführen, weil jeder Zustandsübergang in $\mathcal{A}_T$ auch in $\mathcal{A}'_T$ ausgeführt werden kann. Folglich gilt $T = T(\mathcal{A}'_T) = T(\mathcal{A}_T)$.
Weil $\mathcal{A}'_T$ vollständig und deterministisch ist, überführt jedes Eingabewort $\mathcal{A}'_T$ in einen eindeutig definierten globalen Zustand. Das Eingabewort gehört genau dann zur Wortsprache des Automaten, wenn der erreichte Zustand ein Endzustand ist. Ein schwach kooperierender EAA für die Komplementsprache $\overline{T}$ ist somit $\overline{\mathcal{A}'_T} = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, \mathcal{I}, \mathcal{D}', S' \setminus \mathcal{F})$. □

Lemma 5.2.2 (Abschluß von $SKA(\Sigma, I)$ gegen $Pref$)

Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache.
Es gilt:

$$T \in SKA(\Sigma, I) \Rightarrow Pref(T) \in SKA(\Sigma, I)$$

Beweis:
Nach Definition 3.4.8 gibt es einen schwach kooperierenden EAA, der T erkennt. Sei $\mathcal{A}_T = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\mathcal{P}_i = (\Sigma_i, S_i)$ dieser Automat. Der Automat für $Pref(T)$ soll sein $\mathcal{A}'_T = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F}')$ mit

$$\mathcal{F}' =_{Df} \{s \in S | \bigvee_{w \in \Sigma^*} \bigvee_{s_f \in \mathcal{F}} s \stackrel{w}{\Longrightarrow} s_f\}.$$

$\mathcal{A}'_T$ ist wie $\mathcal{A}_T$ schwach kooperierend.
Zu zeigen: $Pref(T) = T(\mathcal{A}'_T)$
$\subseteq$: Sei $t \in Pref(T)$. Zu t gibt es ein Wort $w \in \Sigma^*$ mit $t[w] \in T$. Der Trace $t[w]$ wird von $\mathcal{A}_T$ akzeptiert. Sei $s \in S$ der globale Zustand, den $\mathcal{A}_T$ in einer akzeptierenden Zustandsfolge für $t[w]$ mit t erreicht. Der Zustand s wird durch w in einen Endzustand überführt, ist also Endzustand von $\mathcal{A}'_T$. Da $\mathcal{A}'_T$ bezüglich Zustandsmenge und Zustandsübergängen mit $\mathcal{A}_T$ übereinstimmt, erreicht $\mathcal{A}'_T$ mit t ebenfalls den Zustand $s \in \mathcal{F}'$.
$\supseteq$: Sei $t \in T(\mathcal{A}'_T)$. t wird im Zustand $s \in \mathcal{F}'$ von $\mathcal{A}'_T$ akzeptiert. Zu s gibt es ein Wort $w \in \Sigma^*$ und einen Endzustand $s_f \in \mathcal{F}$ mit $s \stackrel{w}{\Longrightarrow} s_f$ in $\mathcal{A}_T$. Demnach akzeptiert $\mathcal{A}_T$ den Trace $t[w]$. Folglich gilt $t[w] \in T$ und schließlich $t \in Pref(T)$. □

Lemma 5.2.3 (Abschluß von $SKA(\Sigma, I)$ gegen Spiegelung)

Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache.
Es gilt:

$$T \in SKA(\Sigma, I) \Rightarrow T^R \in SKA(\Sigma, I)$$

Beweis:
Sei $T \in SKA(\Sigma, I)$. Zu T gibt es einen schwach kooperierenden endlichen asynchronen Automaten $\mathcal{A}_T = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$. Die Schar von Übergangsrelationen kann durch lokal definierte Übergangsrelationen $\delta_j, j \in \{1, \ldots, n\}$, dargestellt werden. Zu $\mathcal{A}_T$ wird ein neuer schwach kooperierender EAA definiert, indem Anfangs- und Endzustandsmenge vertauscht und die Übergänge δ_j umgedreht werden.
Es ist $\mathcal{A}'_T = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{F}, \mathcal{D}', \mathcal{I})$, wobei $\mathcal{D}'$ sich aus den $\delta'_j, j \in \{1, \ldots, n\}$, ergibt mit

$$\delta'_j(s, a) =_{Df} \{s' \in S_j | s \in \delta_j(s', a)\}$$

für $a \in \Sigma, j \in Dom(a)$ und $s \in S_j$. Seien δ_j^* und δ'^*_j die transitiven und reflexiven Fortsetzungen von δ_j bzw. δ'_j auf Wörter. Der Induktionsschritt für den Beweis von $s' \in \delta'^*_j(s, w) \iff s \in \delta_j^*(s', w^R)$ ist

$$\begin{aligned} s' \in \delta'^*_j(s, va) &\iff \bigvee_{s'' \in S_j} s'' \in \delta'^*_j(s, v) \wedge s' \in \delta'_j(s'', a) \\ &\stackrel{I.V.}{\iff} \bigvee_{s'' \in S_j} s \in \delta_j^*(s'', v^R) \wedge s'' \in \delta_j(s', a) \\ &\iff s \in \delta_j^*(s', av^R) \end{aligned}$$

Somit läßt sich zeigen:

$$\begin{aligned} w \in L(\mathcal{A}_T) &\iff \bigvee_{(s_1^0,\ldots,s_n^0)\in\mathcal{I}} \bigvee_{(s_1^f,\ldots,s_n^f)\in\mathcal{F}} \bigwedge_{j\in\{1,\ldots,n\}} s_j^f \in \delta_j^*(s_j^0, w|_{\Sigma_j}) \\ &\iff \bigvee_{(s_1^0,\ldots,s_n^0)\in\mathcal{I}} \bigvee_{(s_1^f,\ldots,s_n^f)\in\mathcal{F}} \bigwedge_{j\in\{1,\ldots,n\}} s_j^0 \in \delta'^*_j(s_j^f, w^R|_{\Sigma_j}) \\ &\iff w^R \in L(\mathcal{A}'_T) \end{aligned}$$

Folglich ist $L(\mathcal{A}_T) = L(\mathcal{A}'_T)^R$. Demnach ist $\mathcal{A}'_T$ ein schwach kooperierender EAA für die Trace-Spiegelsprache von T. □

Die einzige der hier behandelten Operationen, bezüglich der $SKA(\Sigma, I)$ nicht abgeschlossen ist, ist die Verkettung.

Lemma 5.2.4 (Verkettung auf $SKA(\Sigma, I)$)

$SKA(\Sigma, I)$ ist gegen Verkettung abgeschlossen $\iff$ $D = \Sigma^2 \setminus I$ ist transitiv

Beweis:
$\Rightarrow$: Wenn D nicht transitiv ist, gibt es drei Zeichen $a, b, c \in \Sigma$ mit $(a, b), (a, c) \in D$ und $(b, c) \in I$. Die Trace-Sprachen $T = [a^*bc]$ und $T' = [a^*]$ sind synchronisiert und gehören daher zu $SKA(\Sigma, I)$.
$T \circ T' = [a^*bca^*] \in SKA(\Sigma, I)$ wird zum Widerspruch geführt: $T \circ T' \in SKA(\Sigma, I)$ läßt sich nach Satz 3.4.12 zerlegen in k synchronisierte Trace-Sprachen über (Σ, I).

$$T \circ T' = T_1 \cup \ldots \cup T_k \quad \text{mit} \quad T_1, \ldots, T_k \in SYN(\Sigma, I)$$

Betrachtet wird die Trace-Sprache $\tilde{T} = [\{a^m bca^n \in \Sigma^* | m + n = k\}]$. $\tilde{T}$ ist eine Teilmenge von $T \circ T'$, und es ist $|\tilde{T}| = k + 1$. Die Traces aus $\tilde{T}$ müssen in den $T_1, \ldots, T_k$ vorkommen. Mindestens eine der k Trace-Sprachen muß mindestens zwei Traces aus $\tilde{T}$ enthalten. Sei

$T_j,\ j \in \{1, \ldots, k\}$, diese Trace-Sprache und seien

$[a^{m_1}bca^{n_1}], [a^{m_2}bca^{n_2}] \in T_j \cap \tilde{T}$ mit $m_1 + n_1 = m_2 + n_2 = k$ und $m_1 \neq m_2$ die fraglichen Traces.

Sei o. B. d. A. $m_1 < m_2$. Weil T_j synchronisiert ist, gehört auch

$$[a^{m_1}bca^{n_1}]|_{\{a,b\}} \| [a^{m_2}bca^{n_2}]|_{\{a,c\}} = [a^{m_1}ba^l ca^{n_2}] \text{ mit } l = k - m_1 - n_2 > 0$$

zu T_j. T_j ist Teilmenge von $T \circ T'$, und somit gehört der durch die Synchronisation entstandene Trace zu $T \circ T'$, im Widerspruch zu $T \circ T' = [a^*bca^*]$.

$\Leftarrow$: Die maximalen D-Cliquen eines (Σ, I) sind bei transitivem D seine D-Äquivalenzklassen. Die Teilautomaten eines EAAs in Normalform mit einem solchen Alphabet arbeiten auf diesen D-Äquivalenzklassen, demnach auf disjunkten Teilalphabeten. Somit ist bei transitiver Abhängigkeitsrelation jeder EAA in Normalform schwach kooperierend, und es gilt $SKA(\Sigma, I) = Erk(\Sigma, I)$. Die erkennbaren Trace-Sprachen sind gegen Verkettung abgeschlossen (Lemma 3.1.1), also ist es auch $SKA(\Sigma, I)$ bei transitivem D. □

5.3 EAAs mit lokal definierbaren Endzuständen

In Kapitel 3 wurden mit $SYN(\Sigma, I)$ und $SKA(\Sigma, I)$ zwei Sprachklassen definiert, die sich durch Einschränkung des EAA-Konzepts ergeben. Trace-Sprachen aus $SYN(\Sigma, I)$ werden von sog. modularisierten EAAs erkannt, deren Teilautomaten vollkommen autonom arbeitende endliche Automaten sind. In den schwach kooperierenden EAAs sind gegenüber den modularisierten EAAs beliebige Mengen von globalen Zuständen als Anfangs- und Endzustandsmengen zugelassen. Der schwach kooperierende EAA ist also auf dem Weg vom modularisierten zum uneingeschränkten EAA nur die halbe Strecke: Die Zustandsübergänge können im schwach kooperierenden EAA lokal definiert werden.

In diesem Abschnitt soll eine Teilklasse der EAAs betrachtet werden, die auf einem anderen Weg vom modularisierten zum uneingeschränkten EAA liegt. In den EAAs dieser Teilklasse sind die Anfangs- und Endzustandsmengen lokal definierbar (Kreuzprodukte von Anfangs- bzw. Endzustandsmengen der Teilautomaten), während die Übergangsrelationen so allgemein wie bei den uneingeschränkten EAAs definiert sind.

Definition 5.3.1 (EAA mit lokal definierbaren Mengen $\mathcal{I}$ und $\mathcal{F}$)

Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\mathcal{P}_i = (\Sigma_i, S_i), i \in \{1, \ldots, n\}$, ein EAA.

i) Die Anfangs- und Endzustandsmengen sind lokal definierbar
$\iff$ Es gibt in jedem Teilautomaten $\mathcal{P}_i$ Mengen von Anfangs- bzw. Endzuständen $\mathcal{I}_i, \mathcal{F}_i \subseteq S_i$ mit

$$\mathcal{I} = \mathcal{I}_1 \times \ldots \times \mathcal{I}_n \quad \text{und } \mathcal{F} = \mathcal{F}_1 \times \ldots \times \mathcal{F}_n.$$

ii) Die Klasse der lokal definierbaren Trace-Sprachen über (Σ, I) ist $LAE(\Sigma, I)$:

$$T \in LAE(\Sigma, I) \iff_{Df} \text{Es gibt einen EAA } \mathcal{A} \text{ mit lokal definierbaren Anfangs- und Endzustandsmengen mit } I_{\mathcal{A}} = I \text{ und } T(\mathcal{A}) = T.$$

Zunächst wird die Mächtigkeit der neuen Sprachklasse betrachtet.

Lemma 5.3.2 (Mächtigkeit von $LAE(\Sigma, I)$)

$$LAE(\Sigma, I) = Erk(\Sigma, I) \iff I = \emptyset$$

Beweis:
$\Rightarrow$: Es reicht aus, ein aus zwei unabhängigen Zeichen bestehendes Alphabet zu betrachten. Sei $T = \{[a], [b]\} \subseteq E(\Sigma, I)$ mit $\Sigma = \{a, b\}$ und $I = \{(a, b), (b, a)\}$. T ist endlich, also auch erkennbar. Ein EAA für T mit lokal definierbaren Anfangs- und Endzustandsmengen wäre modularisiert, weil der eine Teilautomat nur auf $\Sigma_1 = \{a\}$ und der andere nur auf $\Sigma_2 = \{b\}$ arbeiten könnte und $\Sigma_1 \cap \Sigma_2$ leer ist. T ist aber nicht synchronisiert, weil

$$T \neq T|_{\Sigma_1} \| T|_{\Sigma_2} = \{[\varepsilon], [a], [b], [ab]\},$$

und deswegen gibt es zu T keinen modularisierten EAA. Folglich ist $T \notin LAE(\Sigma, I)$.
$\Leftarrow$: Zu $T \in Erk(\Sigma, \emptyset)$ gibt es einen EAA $\mathcal{A}$ in Normalform mit $T(\mathcal{A}) = T$ und $I_{\mathcal{A}} = \emptyset$. Ein EAA in Normalform mit $I_{\mathcal{A}} = \emptyset$ besteht aus nur einem Teilautomaten, so daß die Anfangs- und Endzustandsmengen trivialerweise lokal definierbar sind. □

Es wurde in Abschnitt 3.4 bereits das Ergebnis von Duboc dargestellt, daß nämlich die endlichen Vereinigungen von synchronisierten Trace-Sprachen genau die von schwach kooperierenden EAAs erkennbaren Trace-Sprachen sind. Die Anwendung von I-erhaltenden elementaren Homomorphismen auf die Trace-Sprachen dieser Sprachklasse ergeben wiederum genau die Klasse der erkennbaren Trace-Sprachen. Beide Zusammenhänge lassen sich dual zu $SKA(\Sigma, I)$ für die Klasse $LAE(\Sigma, I)$ formulieren (siehe Abb. 5.1).

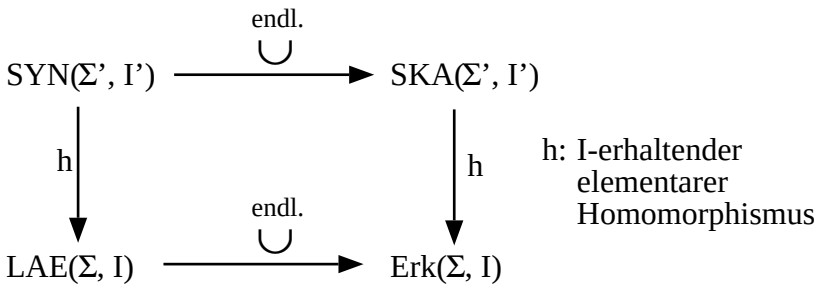

Abbildung 5.1: Homomorphismen und endl. Vereinigung auf Sprachklassen

Lemma 5.3.3 (Beziehung zwischen $SYN(\Sigma, I)$ und $LAE(\Sigma, I)$)

Sei $T \in LAE(\Sigma, I)$ eine lokal definierbare Trace-Sprache über dem Alphabet mit Unabhängigkeitsrelation (Σ, I). Seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen von (Σ, I).
Zu T gibt es

i) ein Alphabet mit Unabhängigkeitsrelation (Σ', I') mit den maximalen D-Cliquen $\Sigma'_1, \ldots, \Sigma'_n$,

ii) einen I-erhaltenden elementaren Homomorphismus $h : E(\Sigma', I') \longrightarrow E(\Sigma, I)$ und

iii) eine synchronisierten Trace-Sprache $T' \in SYN(\Sigma', I')$,

so daß $T = h(T')$.

Beweis:
Der Beweis folgt der Idee Dubocs für den gleichen Zusammenhang zwischen $SKA(\Sigma', I')$ und $Erk(\Sigma, I)$.
Zu $T \in LAE(\Sigma, I)$ gibt es einen EAA $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$, $\mathcal{P}_i = (\Sigma_i, S_i)$, mit lokal definierten Anfangs- und Endzustandsmengen und $I_{\mathcal{A}} = I$, $T(\mathcal{A}) = T$. Das neue Alphabet ist die Menge aller Übergänge in $\mathcal{A}$:

$$\Sigma' =_{Df} \{(s, a, \tilde{s}) \in (S_{i_1} \times \ldots \times S_{i_k}) \times \Sigma \times (S_{i_1} \times \ldots \times S_{i_k}) | \{i_1, \ldots, i_k\} = Dom(a) \wedge \tilde{s} \in \delta_a(s)\}$$

Es wird definiert:

$h(s, a, \tilde{s}) =_{Df} a$ für alle $a \in \Sigma$ und

$(\overline{a}, \overline{b}) \in I' \Longleftrightarrow_{Df} (h(\overline{a}), h(\overline{b})) \in I$ für alle $\overline{a}, \overline{b} \in \Sigma'$.

Der Homomorphismus $h : E(\Sigma', I') \to E(\Sigma, I)$ ist I-erhaltend und elementar. h bildet die D-Cliquen von (Σ', I') auf die D-Cliquen von (Σ, I) ab.
Der neue EAA $\mathcal{A}'$ unterscheidet sich von $\mathcal{A}$ nur im Alphabet und der Schar von Übergangsrelationen. Die Zustände werden übernommen.

$\mathcal{A}' =_{Df} (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, \mathcal{I}, \mathcal{D}', \mathcal{F})$ mit

$\mathcal{P}_i =_{Df} (\Sigma'_i, S_i)$ und $\Sigma'_i =_{Df} h^{-1}(\Sigma_i)$ für $i = 1, \ldots, n$,

$\mathcal{D}' =_{Df} \{\delta'_{\overline{a}} | \overline{a} \in \Sigma'\}$, wobei

$$\delta'_{(s,a,\tilde{s})}(s) = \{\tilde{s}\} \text{ für } a \in \Sigma,\ \{i_1, \ldots, i_k\} = Dom(a) \text{ und } s, \tilde{s} \in S_{i_1} \times \ldots \times S_{i_k}.$$

Für den neuen Automaten gilt $I'_{\mathcal{A}} = I'$:

$$\begin{aligned}
(\overline{a}, \overline{b}) \in I'_{\mathcal{A}} &\iff \neg \bigvee_{i \in \{1,\ldots,n\}} \overline{a}, \overline{b} \in \Sigma'_i = h^{-1}(\Sigma_i) \\
&\iff \neg \bigvee_{i \in \{1,\ldots,n\}} h(\overline{a}), h(\overline{b}) \in \Sigma_i \\
&\iff (h(\overline{a}), h(\overline{b})) \in I_{\mathcal{A}} = I \\
&\iff (\overline{a}, \overline{b}) \in I'
\end{aligned}$$

$h(T(\mathcal{A}')) \subseteq T(\mathcal{A})$ ergibt sich aus

$$s \overset{\overline{a}}{\Longrightarrow} s' \text{ in } \mathcal{A}' \Rightarrow s \overset{h(\overline{a})}{\Longrightarrow} s' \text{ in } \mathcal{A} \text{ für alle } \overline{a} \in \Sigma',\quad s, s' \in S_1 \times \ldots \times S_n$$

und der Tatsache, daß bei der Konstruktion von $\mathcal{A}'$ die Zustandsmengen $\mathcal{I}$ und $\mathcal{F}$ von $\mathcal{A}$ übernommen wurden.
Sei $[w] \in T(\mathcal{A})$ mit $w = a_1 \ldots a_m$. Für w gibt es in $\mathcal{A}$ eine akzeptierende Folge von Zuständen $s_0, s_1, \ldots, s_m \in S$ mit $s_0 \in \mathcal{I}$, $s_{i-1} \overset{a_i}{\Longrightarrow} s_i$ für $i = 1, \ldots, m$ und $s_m \in \mathcal{F}$. Dann gibt es eine Folge $b_1, \ldots, b_m \in \Sigma'$ von Zeichen mit $b_i = (s_{i-1}, a_i, s_i)$, $i = 1, \ldots, m$. Es gilt offensichtlich $s_0 \overset{[b_1 \ldots b_m]}{\Longrightarrow} s_m$, $[b_1 \ldots b_m] \in T(\mathcal{A}')$ und $h([b_1 \ldots b_m]) = [w]$, also $T(\mathcal{A}) \subseteq h(T(\mathcal{A}'))$.
Nun ist zu zeigen, daß $T(\mathcal{A}')$ synchronisiert ist. Zu diesem Zweck werden die Teilautomaten von $\mathcal{A}'$ als gewöhnliche endliche Automaten mit eigener Übergangsrelation definiert, und es wird nachgewiesen, daß $T(\mathcal{A}')$ die Synchronisation der von diesen Automaten erkannten Wortsprachen ist.
Sei also für $i = 1, \ldots, n$ $A_i = (\Sigma'_i, S_i, \mathcal{I}_i, \delta_i, \mathcal{F}_i)$ ein endlicher Automat, wobei $\mathcal{I}_i$ und $\mathcal{F}_i$ die lokalen Anfangs- bzw. Endzustandsmengen des Teilautomaten $\mathcal{P}_i$ sind, und die Übergangsrelation $\delta_i : S_i \times \Sigma'_i \to \wp(S_i)$ definiert wird als
$\delta_i(s_i, \overline{a}) =_{Df} \Big\{ s'_i \in S_i \Big| Dom(\overline{a}) = \{i_1, \ldots, i_k\}$

$$\wedge \bigvee_{s, \tilde{s} \in S_{i_1} \times \ldots \times S_{i_k}} (s = (s_{i_1}, \ldots, s_i, \ldots, s_{i_k}) \wedge \tilde{s} = (s'_{i_1}, \ldots, s'_i, \ldots, s'_{i_k}) \wedge \delta'_{\overline{a}}(s) = \{\tilde{s}\}) \Big\}.$$

Sei $L(A_i)$ die vom Automaten A_i erkannte Sprache und sei $T' = L(A_1)\| \ldots \|L(A_n)$.
Beh.: $T' = T(\mathcal{A}')$
$\subseteq$: Sei $t \in T'$, also $t|_{(\Sigma'_i,\emptyset)}$ ein Trace, dessen einziger Repräsentant von A_i erkannt wird (für alle $i = 1, \ldots, n$). Sei $\overline{a} \in Alph(t)$ ein Zeichen aus t, und sei $t = [\overline{wav}]$ mit $\overline{w}, \overline{v} \in \Sigma'^*$. Nach Abarbeitung von $\overline{w}|_{(\Sigma'_i,\emptyset)}$ ist der Automat $A_i, i \in Dom(\overline{a})$, in einem Zustand, in dem er einen $\overline{a}$-Übergang ausführen kann. Wegen $\overline{a} = (s, a, \tilde{s})$ ist der Zustand als Komponente des Tupels $s \in S_{i_1} \times \ldots \times S_{i_k}$ eindeutig bestimmt. s gibt die Zustände der Automaten $A_i, i \in Dom(\overline{a})$, vor Abarbeitung von $\overline{a}$ an, und alle A_i können den $\overline{a}$-Übergang simultan ausführen, was gleichbedeutend mit einem $\overline{a}$-Übergang im EAA $\mathcal{A}'$ ist. Weil die lokalen Anfangs- und Endzustandsmengen von $\mathcal{A}'$ in die Automaten A_i übernommen wurden, kann jedes von den A_i simultan akzeptierte Wort auch von $\mathcal{A}'$ akzeptiert werden.
$\supseteq$: Sei $t \in T(\mathcal{A}')$. Zu t gibt es eine akzeptierende Zustandsfolge in $\mathcal{A}'$. Die Projektion auf die Zustandsmenge S_i ergibt eine Folge von Zuständen im Teilautomaten $\mathcal{P}'_i$ von $\mathcal{A}'$. Nach Definition der Übergangsrelation δ_i im Automaten A_i ist diese Folge eine akzeptierende Zustandsfolge des Repräsentanten von $t|_{(\Sigma'_i,\emptyset)}$ in A_i. Somit ist $t|_{(\Sigma'_i,\emptyset)} \in L(A_i)$ für alle $i = 1, \ldots, n$. Schließlich ist $t \in T'$, weil $T' = L(A_1)\| \ldots \|L(A_n)$.
$T(\mathcal{A}')$ ist als Synchronisation von regulären Wortsprachen, also Trace-Sprachen über einem Alphabet mit leerer Unabhängigkeitsrelation, darstellbar. Die Wortsprachen sind synchronisiert, weil ihr Alphabet mit seiner maximalen D-Clique übereinstimmt. Die synchronisierten Trace-Sprachen sind nach Lemma 5.1.1 gegen Synchronisation abgeschlossen, so daß $T(\mathcal{A}')$ ebenfalls synchronisiert ist. □

Lemma 5.3.4 (Beziehung zwischen $LAE(\Sigma, I)$ und $Erk(\Sigma, I)$)

Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache.
Es gibt lokal definierbare Trace-Sprachen $T_1, \ldots, T_k \in LAE(\Sigma, I)$ mit

$$T = T_1 \cup \ldots \cup T_k.$$

Beweis:
Zu T gibt es einen EAA $\mathcal{A}$ mit Anfangszustandsmenge $\mathcal{I}$ und Endzustandsmenge $\mathcal{F}$. Nach Definition ist die von $\mathcal{A}$ erkannte Sprache $T = T(\mathcal{A}) = \{t \in E(\Sigma, I) | \bigvee_{s_0 \in \mathcal{I}} \bigvee_{s_f \in \mathcal{F}} s_0 \stackrel{t}{\Longrightarrow} s_f\}$. Für jedes Paar $(s_0, s_f) \in \mathcal{I} \times \mathcal{F}$ wird nun ein EAA $\mathcal{A}_{(s_0,s_f)}$ definiert, der mit $\mathcal{A}$ übereinstimmt, aber s_0 als einzigen Anfangs- und s_f als einzigen Endzustand hat. Die neu definierten EAAs haben somit einelementige und daher lokal definierbare Anfangs- und Endzustandsmengen. Offensichtlich gilt:

$$\begin{aligned}
t \in T \iff t \in T(\mathcal{A}) &\iff \bigvee_{s_0 \in \mathcal{I}} \bigvee_{s_f \in \mathcal{F}} s_0 \stackrel{t}{\Longrightarrow} s_f \text{ in } \mathcal{A} \\
&\iff \bigvee_{s_0 \in \mathcal{I}} \bigvee_{s_f \in \mathcal{F}} s_0 \stackrel{t}{\Longrightarrow} s_f \text{ in } \mathcal{A}_{(s_0,s_f)} \\
&\iff \bigvee_{s_0 \in \mathcal{I}} \bigvee_{s_f \in \mathcal{F}} t \in T(\mathcal{A}_{(s_0,s_f)}) \\
&\iff t \in \bigcup_{s_0 \in \mathcal{I}, s_f \in \mathcal{F}} T(\mathcal{A}_{(s_0,s_f)})
\end{aligned}$$

Die $T(\mathcal{A}_{(s_0,s_f)})$ sind lokal definierbare Trace-Sprachen. □

Im Falle der modularisierten sowie der schwach kooperierenden EAAs bedeutet der Übergang zu deterministischen EAAs keine Einschränkung der Sprachklasse. Bei den lokal definierbaren Trace-Sprachen verhält es sich anders.

Definition 5.3.5 (Sprachklasse $DLE(\Sigma, I) = LAE_{det}(\Sigma, I)$)

Die Sprachklasse $DLE(\Sigma, I)$ der deterministisch lokal definierbaren Trace-Sprachen wird definiert als

$$T \in DLE(\Sigma, I) \iff_{Df} \textit{Es gibt einen deterministischen EAA } \mathcal{A} \textit{ mit lokal definierbarer Endzustandsmenge mit } I_{\mathcal{A}} = I \textit{ und } T(\mathcal{A}) = T.$$

Lemma 5.3.6 (Beziehung $LAE(\Sigma, I)$ — $DLE(\Sigma, I)$)

Für die beiden zuletzt definierten Sprachklassen gilt:

$$DLE(\Sigma, I) = LAE(\Sigma, I) \iff \textit{Die Abhängigkeitsrelation } D = \Sigma^2 \setminus I \textit{ ist transitiv}$$

Beweis:
⇒: Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation, dessen Abhängigkeitsrelation D nicht transitiv ist. Dann gibt es drei Zeichen $a, b, c \in \Sigma$ mit $(a, c), (b, c) \in D$ und $(a, b) \in I$. Sei $T = \{[ca], [cb]\} \in Erk(\Sigma, I)$. Zu T gibt es einen nichtdeterministischen EAA mit einem Anfangs- und einem Endzustand (Abb. 5.2), also $T \in LAE(\Sigma, I)$.

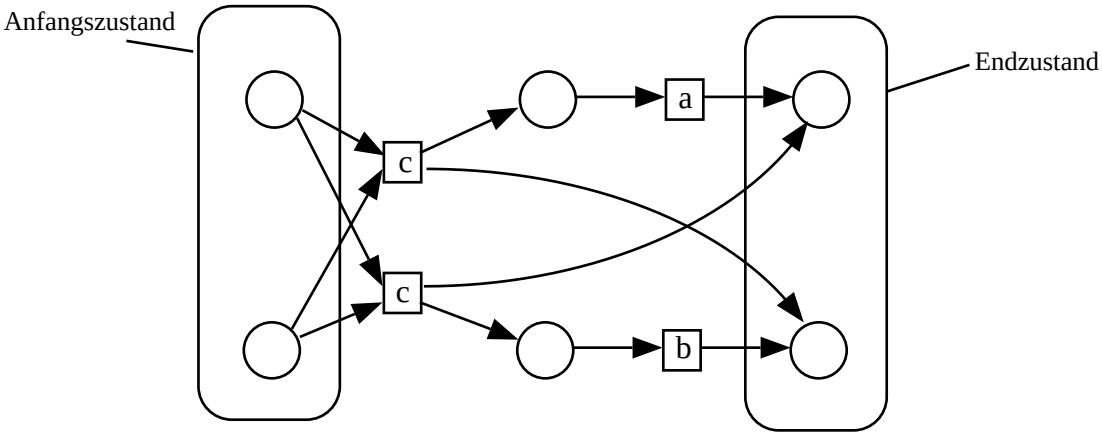

Abbildung 5.2: Nichtdeterministischer EAA für $\{[ca], [cb]\}$

Ein deterministischer EAA zu T würde den einzigen Anfangszustand mit $[c]$ in einen eindeutigen Zustand s überführen. s müßte Anfangszustand für einen EAA $\mathcal{A}'$ mit $T(\mathcal{A}') = \{[a], [b]\}$ sein. Der Beweis von Lemma 5.3.2 hat bereits gezeigt, daß es für diese Sprache keinen EAA mit lokal definierbaren Anfangs- und Endzustandsmengen gibt. Weil die Anfangszustandsmenge von $\mathcal{A}'$ einelementig und daher lokal definierbar ist, kann die Endzustandsmenge von

$\mathcal{A}'$ bzw. $\mathcal{A}$ nicht als Kreuzprodukt von lokalen Endzuständen dargestellt werden. Also ist $T \notin DLE(\Sigma, I)$.

$\Leftarrow$: Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation mit transitiver Abhängigkeitsrelation $D = \Sigma^2 \setminus I$. Die maximalen Mengen abhängiger Zeichen bilden in (Σ, I) die D-Äquivalenzklassen $\mathrm{K}_1, \ldots, \mathrm{K}_m \subseteq \Sigma$.
Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\mathcal{P}_i = (\Sigma_i, S_i)$ ein EAA mit lokal definierbaren Mengen von Anfangs- und Endzuständen mit $I_{\mathcal{A}} = I$ und $T(\mathcal{A}) = T$. Wenn $\mathcal{A}$ in Normalform ist, hat $\mathcal{A}$ m Teilautomaten, die auf den Äquivalenzklassen $\mathrm{K}_1, \ldots, \mathrm{K}_m$ arbeiten. Wegen $|Dom(a)| = 1$ für alle $a \in \Sigma$ ist $\mathcal{A}$ schwach kooperierend. Somit ist $\mathcal{A}$ modularisiert und T synchronisiert (Korollar 3.4.5). Die auf disjunkten Alphabeten autonom arbeitenden Teilautomaten können mit Hilfe der Potenzautomatenkonstruktion in deterministische Teilautomaten überführt werden. Somit gilt $T \in DLE(\Sigma, I)$.
Wenn $\mathcal{A}$ nicht in Normalform ist, kann ein äquivalenter EAA $\mathcal{A}'$ in Normalform mit lokal definierbaren Anfangs- und Endzustandsmengen angegeben werden: $\mathcal{A}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_m, \mathcal{I}', \mathcal{D}', \mathcal{F}')$ mit $\mathcal{P}'_j = (\mathrm{K}_j, S'_j), S'_j = S_{j_1} \times \ldots \times S_{j_k}$, wobei $\mathcal{P}_{j_l}, l = 1, \ldots, k$, die Teilautomaten von $\mathcal{A}$ mit $\Sigma_{j_l} \subseteq \mathrm{K}_j$ sind. Es werden also die Teilautomaten von $\mathcal{A}$, die innerhalb derselben Äquivalenzklasse von Zeichen K_j arbeiten, durch Multiplikation zusammengefaßt. Sie ergeben so den Teilautomaten $\mathcal{P}'_j$ von $\mathcal{A}'$, der auf K_j arbeitet. Wegen der Transitivität von D geht jeder Teilautomat von $\mathcal{A}$ in genau einen Teilautomaten von $\mathcal{A}'$ ein. Die globalen Zustände von $\mathcal{A}$ und $\mathcal{A}'$ entsprechen sich daher bijektiv. Die Anfangs- und Endzustandsmengen $\mathcal{I}'$ und $\mathcal{F}'$ des neuen Automaten sind wieder lokal definierbar, weil die lokalen Anfangs- und Endzustandsmengen der Teilautomaten $\mathcal{P}'_j$ sich durch Multiplikation der lokalen Anfangs- bzw. Endzustandsmengen der entsprechenden Teilautomaten $\mathcal{P}_{j_l}$ ergeben. Die formal aufwendiger zu definierende Menge von Übergangsrelationen $\mathcal{D}'$ zeigt, daß mehrere synchrone Übergänge eines Zeichens in den Teilautomaten $\mathcal{P}_{j_l}$ zu einem Übergang in $\mathcal{P}'_j$ zusammengefaßt werden. $\mathcal{A}'$ ist offensichtlich in Normalform und es gilt $T(\mathcal{A}') = T$. □

Zu den Sprachklassen $SYN(\Sigma, I), SKA(\Sigma, I)$ und $Erk(\Sigma, I)$ sind in den vorangegangenen Kapiteln automatenunabhängige Charakterisierungen angegeben worden. Die Sprachklasse $DLE(\Sigma, I)$ kann durch eine Abschlußeigenschaft charakterisiert werden. Das folgende Lemma wird für den Beweis gebraucht.

Lemma 5.3.7 (EAA in Normalform zu $T \in DLE(\Sigma, I)$)

Sei $T \in DLE(\Sigma, I)$ eine deterministische lokal definierbare Trace-Sprache.
Es gibt einen deterministischen EAA $\mathcal{A}$ in Normalform mit lokal definierbaren Endzuständen und $I_{\mathcal{A}} = I, \quad T(\mathcal{A}) = T$.

Beweis:
Es wird ein deterministischer EAA $\mathcal{A}$ in Normalform mit lokal definierter Endzustandsmenge konstruiert. Dann wird gezeigt, daß $\mathcal{A}$ die Sprache T erkennt.
Sei $\mathcal{A}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_m, \mathcal{I}', \mathcal{D}', \mathcal{F}')$ mit $\mathcal{P}'_j = (\Sigma'_j, S'_j)$ der deterministische EAA mit lokal definierbarer Endzustandsmenge und $T(\mathcal{A}') = T$. Im zu konstruierenden EAA arbeiten die Teilautomaten auf den maximalen D-Cliquen $\Sigma_1, \ldots, \Sigma_n$ von (Σ, I). Ein Teilautomat ist ein Produktautomat von solchen Teilautomaten von $\mathcal{A}'$, die auf Teilmengen derselben maximalen

D-Clique arbeiten. Formal:
$\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\mathcal{P}_i = (\Sigma_i, S_i)$, $S_i = S'_{j_1} \times \ldots \times S'_{j_k}$, wobei $\mathcal{P}'_{j_l}$, $l = 1, \ldots, k$, die Teilautomaten von $\mathcal{A}'$ mit $\Sigma'_{j_l} \subseteq \Sigma_i$ sind.
Um die Beziehung zwischen dem zu konstrierenden EAA $\mathcal{A}$ und dem gegebenen EAA $\mathcal{A}'$ zu klären, wird eine Abbildung g definiert, die die Zustände von $\mathcal{A}'$ auf die von $\mathcal{A}$ abbildet.
$g : S'_1 \times \ldots S'_m \to S_1 \times \ldots \times S_n$ mit

$$g(s'_1, \ldots, s'_m) = (s_1, \ldots, s_n) \Longleftrightarrow_{Df} \bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{\{j_1,\ldots,j_k\} \subseteq \{1,\ldots,m\}} s_i = (s'_{j_1}, \ldots, s'_{j_k}).$$

g stellt die $s'_1, \ldots, s'_m$ zu neuen Tupeln, den Zuständen der Teilautomaten von $\mathcal{A}$, zusammen. Die Abbildung g ist injektiv, weil jedes s'_j als Komponente in mindestens ein s_i eingeht. (Die D-Clique Σ'_j muß in mindestens einer maximalen D-Clique Σ_i liegen.)

Mit g können leicht die Anfangs- und Endzustandsmengen von $\mathcal{A}$ definiert werden.

$\mathcal{I} =_{Df} g(\mathcal{I}')$ ist wie $\mathcal{I}'$ einelementig.

$\mathcal{F}_i =_{Df} \{s_i \in S_i | \bigvee_{(s'_1,\ldots,s'_m) \in \mathcal{F}'} (s_1, \ldots, s_i, \ldots, s_n) = g(s'_1, \ldots, s'_m)\}$ für $i = 1, \ldots, n$, und

$\mathcal{F} =_{Df} \mathcal{F}_1 \times \ldots \times \mathcal{F}_n$.

$\mathcal{F}$ ist offensichtlich lokal definiert und es gilt $g(\mathcal{F}') \subseteq \mathcal{F}$.
Die Schar von Übergangsrelationen $\mathcal{D} = \{\delta_a | a \in \Sigma\}$ ist schwieriger zu formulieren:
Sei $Dom(a) = \{i_1, \ldots, i_k\}$. Es wird definiert:

$$(\tilde{s}_{i_1}, \ldots, \tilde{s}_{i_k}) \in \delta_a(s_{i_1}, \ldots, s_{i_k}) \iff_{Df} \bigvee_{s', \tilde{s}' \in S'} \begin{array}{l} g(s')|_{Dom(a)} = (s_{i_1}, \ldots, s_{i_k}) \wedge \\ g(\tilde{s}')|_{Dom(a)} = (\tilde{s}_{i_1}, \ldots, \tilde{s}_{i_k}) \wedge \\ s' \stackrel{a}{\Longrightarrow} \tilde{s}' \text{ in } \mathcal{A}' \end{array}$$

$\mathcal{A}$ ist deterministisch: Angenommen $|\delta_a(s_{i_1}, \ldots, s_{i_k})| \geq 2$, dann gibt es in $\mathcal{A}'$ vier Zustände $s', \tilde{s}', s'', \tilde{s}'' \in S'$ mit

i) $g(s')|_{Dom(a)} = g(s'')|_{Dom(a)} = (s_{i_1}, \ldots, s_{i_k})$

ii) $g(\tilde{s}')|_{Dom(a)} \neq g(\tilde{s}'')|_{Dom(a)}$

iii) $s' \stackrel{a}{\Longrightarrow} \tilde{s}'$, $s'' \stackrel{a}{\Longrightarrow} \tilde{s}''$ in $\mathcal{A}'$

Sei $Dom'(a)$ die Menge (der Indizes) der Teilautomaten von $\mathcal{A}'$, die mit dem Zeichen a arbeiten:
$Dom'(a) = \{j \in \{1, \ldots, m\} | a \in \Sigma'_j\}$.
Betrachten wir nun i): Jedes s_{i_l}, $l = 1, \ldots, k$, auf der rechten Seite ist ein Tupel von Zuständen $s'_{j_1}, \ldots, s'_{j_r}$ von Teilautomaten von $\mathcal{A}'$. Jeder Teilautomat $\mathcal{P}'_j$ aus $Dom'(a)$ ist

durch einen Zustand vertreten, nämlich als Komponente von s_i, wenn $\Sigma'_j \subseteq \Sigma_i$. Mit i) haben wir daher $s'|_{Dom'(a)} = s''|_{Dom'(a)}$. Der a-Übergang in $\mathcal{A}'$ ist nur von den Teilautomaten in $Dom'(a)$ abhängig und eindeutig. Daher gilt unter Ausnutzung von iii) $\tilde{s}'|_{Dom'(a)} = \tilde{s}''|_{Dom'(a)}$. Die Zustände $\tilde{s}'|_{Dom'(a)}$ und $\tilde{s}''|_{Dom'(a)}$ gehen in $g(\tilde{s}')|_{Dom(a)}$ bzw. $g(\tilde{s}'')|_{Dom(a)}$ ein, so daß sich diese beiden Objekte in den Zuständen der Teilautomaten aus $Dom'(a)$ also nicht unterscheiden können. $g(\tilde{s}')|_{Dom(a)}$ und $g(\tilde{s}'')|_{Dom(a)}$ können außerdem Zustände von Teilautomaten außerhalb von $Dom'(a)$ enthalten. Es stimmten aber schon $g(s')|_{Dom(a)}$ und $g(s'')|_{Dom(a)}$ in diesen Zuständen überein, und sie wurden vom a-Übergang nicht berührt. Folglich gilt $g(\tilde{s}')|_{Dom(a)} = g(\tilde{s}'')|_{Dom(a)}$ im Widerspruch zu ii).
Der EAA $\mathcal{A}$ ist somit deterministisch, ist in Normalform und seine Endzustandsmenge ist lokal definiert. Es bleibt $T(\mathcal{A}') = T(\mathcal{A})$ zu zeigen.
$\subseteq$: Sei $[w] \in T(\mathcal{A}')$. In $\mathcal{A}'$ gibt es eine akzeptierende Zustandsfolge für w. Aus der Definition von δ_a ergibt sich

$$s' \stackrel{a}{\Longrightarrow} \tilde{s}' \text{ in } \mathcal{A}' \quad \Rightarrow \quad g(s') \stackrel{a}{\Longrightarrow} g(\tilde{s}') \text{ in } \mathcal{A},$$

so daß die Anwedung von g auf die Zustandsfolge in $\mathcal{A}'$ eine Zustandsfolge in $\mathcal{A}$ ergibt. Der Endzustand der Folge $s^f \in \mathcal{F}'$ in $\mathcal{A}'$ wird wegen $g(\mathcal{F}') \subseteq \mathcal{F}$ auf einen Endzustand von $\mathcal{A}$ abgebildet, so daß $\mathcal{A}$ das Wort w bzw. den Trace $[w]$ akzeptiert.
$\supseteq$: Ein Zustand $s \in S$ in $\mathcal{A}$ heißt in $\mathcal{A}'$ interpretierbar, wenn es einen Zustand $s' \in S'$ in $\mathcal{A}'$ mit $g(s') = s$ gibt. Da g injektiv ist, ist s', wenn der Zustand existiert, eindeutig bestimmt. Nach Konstruktion von δ_a wird in $\mathcal{A}$ ein interpretierbarer Zustand mit demselben Übergang wie in $\mathcal{A}'$ wieder in einen interpretierbaren Zustand überführt. Der Anfangszustand von $\mathcal{A}$ ist interpretierbar durch den Anfangszustand von $\mathcal{A}'$. Sei $s^f = (s_1^f, \ldots, s_n^f) \in \mathcal{F}$ ein Endzustand von $\mathcal{A}$. Sei $s^{f'} \in S'$ die Interpretation von s^f. Nach Definition von $\mathcal{F}$ sind die Komponenten der s_i^f Komponenten von $s^{f'}$ und lokale Endzustände der Teilautomaten von $\mathcal{A}'$. Weil die Endzustandsmenge von $\mathcal{A}'$ lokal definierbar ist, gilt somit $s^{f'} \in \mathcal{F}'$. Alles in allem läßt sich eine akzeptierende Zustandsfolge in $\mathcal{A}$ durch eine akzeptierende Zustandsfolge für das gleiche Wort in $\mathcal{A}'$ interpretieren: $T(\mathcal{A}) \subseteq T(\mathcal{A}')$. □

Satz 5.3.8 (Charakterisierung von $DLE(\Sigma, I)$)

Sei $T \in Erk(\Sigma, I)$ eine erkennbare Trace-Sprache über dem Alphabet mit Unabhängigkeitsrelation (Σ, I). Seien $\Sigma_1, \ldots, \Sigma_n \subseteq \Sigma$ die maximalen D-Cliquen von (Σ, I).
Es gilt:

$$T \in DLE(\Sigma, I) \iff \bigwedge_{t \in E(\Sigma,I)} \bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in T} P_i(t) = P_i(r) \Rightarrow t \in T$$

Beweis:
$\Rightarrow$: Zu T gibt es nach dem vorangegangenen Lemma einen deterministischen EAA $\mathcal{A}$ in Normalform mit lokal definierbarer Endzustandsmenge.
Sei $t \in E(\Sigma, I)$ ein Trace mit $\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in T} P_i(t) = P_i(r)$. $P_i(r)$ ist das kürzeste Anfangsstück von r, das alle Vorkommen von Zeichen aus der maximalen D-Clique Σ_i enthält. Folglich ist der auf Σ_i arbeitende Teilautomat $\mathcal{P}_i$ nach Abarbeitung von $P_i(r)$ im selben Zustand wie nach Abarbeitung von r. Da dies ein lokaler Endzustand ist, es für jeden Index einen solchen Trace $r \in T$ gibt und die Endzustandsmenge lokal definierbar ist, erreicht $\mathcal{A}$ mit t einen

Endzustand, und es gilt $t \in T$.
$\Leftarrow$: Sei $T \in Erk(\Sigma, I)$ eine erkennbare Trace-Sprache mit

$$\bigwedge_{t \in E(\Sigma,I)} \bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in T} P_i(t) = P_i(r) \Rightarrow t \in T.$$

Zu T wird nach Zielonka der deterministische EAA $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ konstruiert (vgl. Abschnitt 4.1). Es wird weiterhin definiert:

$\mathcal{F}'_i =_{Df} \{\langle P_i(t)\rangle | t \in T\}$ für $i = 1, \ldots, n$

$\mathcal{F}' =_{Df} \mathcal{F}'_1 \times \ldots \times \mathcal{F}'_n$

$\mathcal{A}' =_{Df} (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F}')$

Die Behauptung ist jetzt $T(\mathcal{A}') = T$. Die Teilmengenbeziehung $T(\mathcal{A}) \subseteq T(\mathcal{A}')$ ist klar, weil $\mathcal{F} \subseteq \mathcal{F}'$. Sei $t \in T(\mathcal{A}')$. t überführt $\mathcal{A}'$ in den Zustand $(\langle P_1(t)\rangle, \ldots, \langle P_n(t)\rangle) \in \mathcal{F}'$. Die $\langle P_i(t)\rangle$ sind aufgrund der Definition von $\mathcal{F}'$ lokale Endzustände, und es gibt Traces $r_i \in T$ mit $\langle P_i(r_i)\rangle = \langle P_i(t)\rangle$ für alle $i = 1, \ldots, n$. Wenn $P_i(r_i)$ und $P_i(t)$ also zur selben $\approx$-Klasse gehören, sind sie $\approx$-äquivalent und insbesondere syntaktisch kongruent. Folglich gilt für alle $t_\omega \in E(\Sigma, I)$:

$$P_i(r_i)t_\omega \in T \iff P_i(t)t_\omega \in T.$$

Sei t_{r_i} der Trace mit $P_i(r_i)t_{r_i} = r_i \in T$. t_{r_i} enthält kein Zeichenvorkommen aus Σ_i. Sei r'_i nun $P_i(t)t_{r_i}$. Aus der syntaktischen Kongruenz von $P_i(r_i)$ und $P_i(t)$ ergibt sich $r'_i \in T$. Weil t_{r_i} kein Zeichenvorkommen aus Σ_i enthält, gilt $P_i(r'_i) = P_i(t)$ für alle $i \in \{1, \ldots, n\}$. Mithin bekommen wir aus der Eigenschaft von T: $t \in T = T(\mathcal{A})$. Schließlich gilt also $T = T(\mathcal{A}')$ und $T \in DLE(\Sigma, I)$, weil die Endzustandsmenge von $\mathcal{A}'$ lokal definierbar ist. □

Korollar 5.3.9 (Entscheidbarkeit von $DLE(\Sigma, I)$)

Sei $T \subseteq E(\Sigma, I)$ eine erkennbare Trace-Sprache.
Das Problem „$T \in DLE(\Sigma, I)$?“ ist entscheidbar.

Beweis:
Der Beweis zu Satz 5.3.8 nutzt in der Rückrichtung Zielonkas effektiv ausführbares Konstruktionsverfahren für den deterministischen EAA zu T aus. Der Schritt zum Automaten $\mathcal{A}'$ mit der lokal definierten Endzustandsmenge ist einfach. Beide EAAs sind mit ihren globalen Zuständen gewöhnliche endliche Automaten, deren Äquivalenz entschieden werden kann. □

5.4 Abschlußeigenschaften von $DLE(\Sigma, I)$

Mit der neuen Charakterisierung der DLE-Trace-Sprachen lassen sich die Abschlußeigenschaften der Sprachklasse untersuchen.

Lemma 5.4.1 (Abschluß von $DLE(\Sigma, I)$ gegen Schnitt)

Seien $T_1, T_2 \in DLE(\Sigma, I)$ zwei deterministisch lokal definierbare Trace-Sprachen.
Für den Durchschnitt von T_1 und T_2 gilt:

$$T_1 \cap T_2 \in DLE(\Sigma, I)$$

Beweis:
Sei $t \in E(\Sigma, I)$ ein beliebiger Trace mit

$$\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in T_1 \cap T_2} P_i(t) = P_i(r).$$

Der Trace $r \in T_1 \cap T_2$ (jeweils zu $i \in \{1, \ldots, n\}$) liegt insbesondere in T_1 und in T_2, also

$$\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in T_1} P_i(t) = P_i(r) \quad \text{und} \quad \bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in T_2} P_i(t) = P_i(r).$$

Weil T_1 und T_2 deterministisch lokal definierbar sind, ergibt sich mit der Charakterisierung aus Satz 5.3.8 $t \in T_1$ und $t \in T_2$, somit $t \in T_1 \cap T_2$. Die Trace-Sprache $T_1 \cap T_2$ erfüllt also ebenfalls die in Satz 5.3.8 formulierte Anforderung. Die Ausnutzung der Rückrichtung von Satz 5.3.8 ergibt schließlich $T_1 \cap T_2 \in DLE(\Sigma, I)$. □

Lemma 5.4.2 (Vereinigung und Komplement auf $DLE(\Sigma, I)$)

Die Klasse $DLE(\Sigma, I)$ ist gegen Vereinigung und Komplement bezüglich $E(\Sigma, I)$ abgeschlossen $\iff I = \emptyset$

Beweis:
$\Rightarrow$: Wenn $I \neq \emptyset$ ist, gibt es Zeichen $a, b \in \Sigma$ mit $(a, b) \in I$. Seien o. B. d. A. $\Sigma_1, \Sigma_2, \Sigma_3 \subseteq \Sigma$ maximale D-Cliquen mit $a \in \Sigma_1$, $b \in \Sigma_2$ und $a, b \notin \Sigma_3$. Als Gegenbeispiel für den Abschluß gegen Vereinigung dient die Sprache $T = \{[a], [b]\}$.
Sei $T_1 = \{[a]\}$ und sei $T_2 = \{[b]\}$.
T_1 ist deterministisch lokal definierbar, weil ein deterministischer EAA für T_1 nur einen globalen Endzustand haben muß. Ebenso ist $T_2 \in DLE(\Sigma, I)$. Für $T = T_1 \cup T_2$ wird $t = [ab]$ betrachtet:

$P_1(t) = [a]$ ist $P_1(r_1)$ mit $r_1 = [a] \in T$

$P_2(t) = [b]$ ist $P_2(r_2)$ mit $r_2 = [b] \in T$

$P_3(t) = [\varepsilon]$ ist $P_3(r_3)$ mit $r_3 = [a] \in T$

Wegen $[ab] \notin T$ ist T nach Satz 5.3.8 nicht deterministisch lokal definierbar und daher $DLE(\Sigma, I)$ nicht gegen Vereinigung abgeschlossen.
Zusammen mit Lemma 5.4.1 läßt sich folgern, daß $DLE(\Sigma, I)$ mit $I \neq \emptyset$ auch nicht gegen Komplement abgeschlossen ist.
$\Leftarrow$: Für $I = \emptyset$ gilt $DLE(\Sigma, I) = Erk(\Sigma, I)$. Die erkennbaren Trace-Sprachen sind nach Lemma 3.1.1 gegen Vereinigung und Komplement abgeschlossen. □

Lemma 5.4.3 (Abschluß von DLE gegen Synchronisation)

Seien $T \in DLE(\Sigma, I)$ und $T' \in DLE(\Sigma', I')$ zwei deterministisch lokal definierbare Trace-Srachen. Sei (Σ'', I'') das durch die Synchronisation entstehende Alphabet mit Unabhängigkeitsrelation.
Es gilt:

$$T \| T' \in DLE(\Sigma'', I'')$$

Beweis:
Das Lemma läßt sich einfacher durch Betrachtung des Automatenmodells als mit Hilfe der Charakterisierung beweisen.
Seien $\mathcal{A}$ und $\mathcal{A}'$ deterministische EAAs mit lokal definierbaren Endzustandsmengen und mit $T(\mathcal{A}) = T$ bzw. $T(\mathcal{A}') = T'$. Sei $\mathcal{A}'' = \mathcal{A} \| \mathcal{A}'$ der in Definition 3.4.1 definierte Synchronisationsautomat. Nach Satz 3.4.2 ist $T \| T' = T(\mathcal{A} \| \mathcal{A}')$. Es bleibt also zu zeigen, daß der Synchronisationsautomat $\mathcal{A}''$ deterministisch und seine Endzustandsmenge lokal definierbar ist.

i) Die Menge der Anfangszustände von $\mathcal{A}''$ ergibt sich als Kreuzprodukt der Mengen der Anfangszustände von $\mathcal{A}$ und $\mathcal{A}'$. Beide sind einelementig, weil $\mathcal{A}$ und $\mathcal{A}'$ deterministisch sind. Folglich hat $\mathcal{A}''$ genau einen Anfangszustand.

ii) Die Übergangsrelation δ''_a von $\mathcal{A}''$ wird für $a \in \Sigma \setminus \Sigma'$ und $a \in \Sigma' \setminus \Sigma$ vom Automaten $\mathcal{A}$ bzw. $\mathcal{A}'$ übernommen. Die Übergänge für solche $a \in \Sigma \cup \Sigma'$ sind mithin wie in $\mathcal{A}$ bzw. $\mathcal{A}'$ deterministisch.
Für $a \in \Sigma \cap \Sigma'$ wird die Menge der im a-Übergang erreichbaren Zustände als Kreuzprodukt definiert (vgl. Definition 3.4.1 auf Seite 42):

$$\delta''_a(s_{i_1}, \ldots, s_{i_l}) =_{Df} \delta_a(s_{i_1}, \ldots, s_{i_k}) \times \delta'_a(s_{i_{k+1}}, \ldots, s_{i_l})$$

Die beiden Mengen $\delta_a(s_{i_1}, \ldots, s_{i_k})$ und $\delta'_a(s_{i_{k+1}}, \ldots, s_{i_l})$ sind einelementig oder leer, so daß auch $\delta''_a(s_{i_1}, \ldots, s_{i_l})$ wieder höchstens einelementig ist.
Mit i) und ii) ist nachgewiesen, daß $\mathcal{A}''$ deterministisch ist.

iii) Seien $\mathcal{F}$ und $\mathcal{F}'$ die Endzustandsmengen von $\mathcal{A}$ bzw. $\mathcal{A}'$. Die Menge der Endzustände des Synchronisationsautomaten ist $\mathcal{F}'' = \mathcal{F} \times \mathcal{F}'$, ist also lokal definierbar, weil auch $\mathcal{F}$ und $\mathcal{F}'$ lokal definierbar sind. □

Lemma 5.4.4 (Präfixabschluß, Spiegelung und Verkettung auf $DLE(\Sigma, I)$)

Die Klasse von Trace-Sprachen $DLE(\Sigma, I)$ ist gegen Präfixabschluß, Spiegelung bzw. Verkettung abgeschlossen
$\iff$ *$D = \Sigma^2 \setminus I$ ist transitiv*

Beweis:
$\Rightarrow$: Es reicht aus, $\Sigma = \{a, b, c\}$ mit $I = \{(a, b), (b, a)\}$ zu betrachten. Die maximalen D-Cliquen von (Σ, I) sind $\Sigma_1 = \{a, c\}$ und $\Sigma_2 = \{b, c\}$. Die Trace-Sprache $T = \{[ac], [bc]\}$ ist

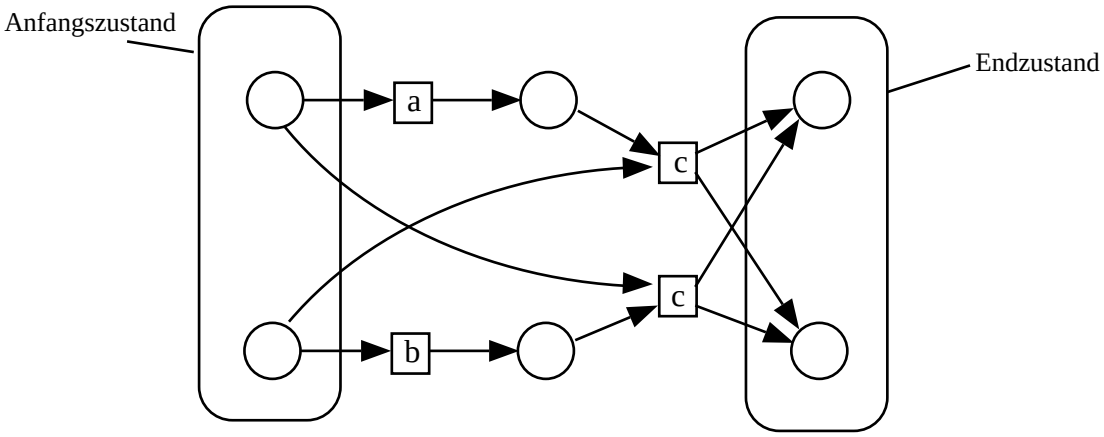

Abbildung 5.3: Deterministischer EAA für $\{[ac], [bc]\}$

deterministisch lokal definierbar. Abbildung 5.3 zeigt einen deterministischen EAA, der T erkennt und nur einen Endzustand hat.

$T \in DLE(\Sigma, I)$ läßt sich auch mit Satz 5.3.8 begründen, wenn man $P_1(T) = \{P_1(t) \in E(\Sigma, I) | t \in T\} = T$ und $P_2(T) = T$ berücksichtigt.

i) $Pref(T) = \{[\varepsilon], [a], [b], [ac], [bc]\}$ ist nicht deterministisch lokal definierbar, weil

$P_1([ab]) = P_1([a])$ und $[a] \in Pref(T)$,
$P_2([ab]) = P_2([b])$ und $[b] \in Pref(T)$,
aber $[ab] \notin Pref(T)$.

ii) $T^R = \{[ca], [cb]\}$ gehört nicht zu $DLE(\Sigma, I)$, weil

$P_1([cab]) = P_1([ca])$ und $[ca] \in T^R$,
$P_2([cab]) = P_2([cb])$ und $[cb] \in T^R$,
aber $[cab] \notin T^R$.

iii) Für die Verkettung werden die Sprachen $T_1 = [(ca)^*]$ und $T_2 = [(cb)^*]$ betrachtet. Beide sind deterministisch lokal definierbar, nicht jedoch $T_1 \circ T_2 = [(ca)^*(cb)^*]$, weil $[ca], [cb] \in T_1 \circ T_2$ und $[cab] \notin T_1 \circ T_2$.

$\Leftarrow$: Wenn D transitiv ist, sind die maximalen D-Cliquen in (Σ, I) die D-Äquivalenzklassen $\mathrm{K}_1, \ldots, \mathrm{K}_n \subseteq \Sigma$ von (Σ, I). Die Teilautomaten eines EAAs in Normalform zu (Σ, I) arbeiten mit den K_i als Teilalphabeten. Jedes Zeichen gehört zu einer Äquivalenzklasse, folglich gilt $|Dom(a)| = 1$ für alle Zeichen $a \in \Sigma$. Somit ist ein solcher EAA schwach kooperierend und wegen der lokal definierbaren Endzustände modularisiert. Also gilt für transitive $D = \Sigma^2 \setminus I$:

$DLE(\Sigma, I) = SYN(\Sigma, I)$. Die synchronisierten Trace-Sprachen sind nach Lemma 5.1.3 gegen Spiegelung und bei transitivem D nach Lemma 5.1.5 gegen Präfixabschluß bzw. Verkettung abgeschlossen. □

An den Lemmas 5.4.2 und 5.4.4 wird deutlich, wie die automatenunabhängige Charakterisierung von $DLE(\Sigma, I)$ die Argumentation erleichtert. Es muß nur nach einem bestimmten Trace gesucht werden, um zu zeigen, daß es zu einer gegebenen erkennbaren Trace-Sprache keinen DLE-Automaten gibt.

5.5 Beziehungen zwischen $DLE(\Sigma, I)$ und bereits definierten Sprachklassen

In diesem Abschnitt werden Teil- und Schnittmengenbeziehungen zwischen den schon definierten Sprachklassen geklärt, indem konkrete typische Trace-Sprachen aus Differenzmengen vorgestellt werden.

Durch Betrachtung des Automatenmodells ergeben sich folgende Inklusionen:

$$SYN(\Sigma, I) \subseteq SKA(\Sigma, I) \subseteq Erk(\Sigma, I)$$

$$SYN(\Sigma, I) \subseteq DLE(\Sigma, I) \subseteq Erk(\Sigma, I)$$

Die folgenden Gegenbeispiele werden zeigen, daß keine weiteren Teilmengenbeziehungen gelten. Insbesondere sind die aufgeführten Inklusionen echt.

Lemma 5.5.1 (Typische Sprache aus $SKA(\Sigma, I)$)

Für die Sprachklassen $SKA(\Sigma, I)$ und $DLE(\Sigma, I)$ gilt:

$$SKA(\Sigma, I) \setminus DLE(\Sigma, I) = \emptyset \iff I = \emptyset$$

Beweis:
$\Rightarrow$: Wenn $I \neq \emptyset$ ist, gibt es Zeichen $a, b \in \Sigma$ mit $(a, b) \in I$. Sei $T^{(1)} = \{[a], [b]\} \in Erk(\Sigma, I)$. Die Trace-Sprache wurde schon mehrfach als Beispiel herangezogen. $T^{(1)}$ ist endlich und daher nach Lemma 3.4.11 in $SKA(\Sigma, I)$. $T^{(1)} \notin DLE(\Sigma, I)$ folgt aus Lemma 5.3.2.
$\Leftarrow$: Für $I = \emptyset$ ist $SKA(\Sigma, I) = DLE(\Sigma, I) = Erk(\Sigma, I)$. □

Lemma 5.5.2 (Typische Sprache aus $DLE(\Sigma, I)$)

Für die Sprachklassen $SKA(\Sigma, I)$ und $DLE(\Sigma, I)$ gilt:

$$DLE(\Sigma, I) \setminus SKA(\Sigma, I) = \emptyset \iff D = \Sigma^2 \setminus I \text{ ist transitiv}$$

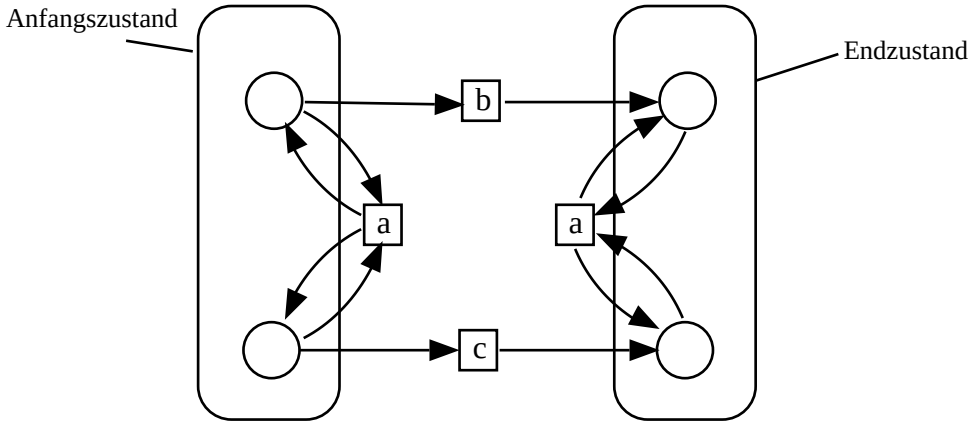

Abbildung 5.4: EAA für $T^{(2)} = [a^*bca^*]$

Beweis:
$\Rightarrow$: Wenn D nicht transitiv ist, gibt es drei Zeichen $a, b, c \in \Sigma$ mit $(a, b), (a, c) \in D$ und $(b, c) \in I$. Es wird die Trace-Sprache $T^{(2)} = [a^*bca^*]$ betrachtet.
$T^{(2)} \in DLE(\Sigma, I)$: Der EAA in Abb. 5.4 ist deterministisch, hat nur einen Anfangs- und einen Endzustand und erkennt $T^{(2)}$.

$T^{(2)} \notin SKA(\Sigma, I)$ wurde bereits im Beweis zu Lemma 5.2.4 gezeigt.

$\Leftarrow$: Wenn D transitiv ist, sind die maximalen D-Cliquen in (Σ, I) seine D-Äquivalenzklassen. Sei $T \in Erk(\Sigma, I)$ eine erkennbare Trace-Sprache über einem solchen Alphabet mit Unabhängigkeitsrelation. Zu T gibt es nach Zielonka einen deterministischen EAA in Normalform, der T erkennt. Die Teilautomaten dieses EAAs haben die D-Äquivalenzklassen als ihre Teilalphabete. Jedes Zeichen gehört zu genau einer Äquivalenzklasse, folglich $|Dom(a)| = 1$ für alle Zeichen $a \in \Sigma$. Somit ist der EAA schwach kooperierend und $T \in SKA(\Sigma, I)$. Bei transitivem D ist also $SKA(\Sigma, I) = Erk(\Sigma, I)$ und $DLE(\Sigma, I) \setminus SKA(\Sigma, I) = DLE(\Sigma, I) \setminus Erk(\Sigma, I) = \emptyset$. □

Mit den beiden Lemmas sind die Teilmengenbeziehungen zwischen den Sprachklassen fast geklärt. $DLE(\Sigma, I) \cup SKA(\Sigma, I)$ füllt für nichttransitive D die Klasse $Erk(\Sigma, I)$ nicht aus, denn aus den Sprachen $T^{(1)}$ und $T^{(2)}$ läßt sich leicht eine erkennbare Sprache konstruieren, die weder zu $DLE(\Sigma, I)$ noch zu $SKA(\Sigma, I)$ gehört (etwa $T^{(2)} \cup \{[d]\}$ mit einem neuen von a, b und c unabhängigen Zeichen d).

Die letzte Frage ist, ob jede Trace-Sprache, die sowohl durch einen schwach kooperierenden EAA als auch durch einen deterministischen EAA mit lokal definierbaren Endzuständen dargestellt werden kann, auch synchronisiert ist.

Lemma 5.5.3 ($SYN(\Sigma, I)$ echte Teilmenge von $DLE(\Sigma, I) \cap SKA(\Sigma, I)$)

$$(DLE(\Sigma, I) \cap SKA(\Sigma, I)) \setminus SYN(\Sigma, I) = \emptyset \iff D = \Sigma^2 \setminus I \text{ ist transitiv}$$

Beweis:
$\Rightarrow$: Es reicht aus, das Alphabet mit Unabhängigkeitsrelation $(\Sigma, I) = (\{a, b, c\}, \{(a, b), (b, a)\})$

zu betrachten. Die Trace-Sprache aus $(DLE(\Sigma, I) \cap SKA(\Sigma, I)) \setminus SYN(\Sigma, I)$ ist nun $T^{(3)} = \{[ac], [bc]\}$. Begründung:
Abb. 5.3 zeigt einen deterministischen EAA für $T^{(3)}$ mit einem Anfangs- und einem Endzustand. $T^{(3)}$ ist daher eine Sprache aus $DLE(\Sigma, I)$. Weil $T^{(3)}$ endlich ist und alle endlichen Trace-Sprachen über (Σ, I) zu $SKA(\Sigma, I)$ gehören (Lemma 3.4.11), gilt $T^{(3)} \in SKA(\Sigma, I)$. $T^{(3)}$ ist nicht synchronisiert, weil

$$T^{(3)}|_{(\{a,c\},\emptyset)} \| T^{(3)}|_{(\{b,c\},\emptyset)} = \{[ac], [c]\} \| \{[c], [bc]\} = \{[c|, [ac], [bc], [abc]\} \neq T^{(3)}.$$

Folglich gilt $T^{(3)} \in (DLE(\Sigma, I) \cap SKA(\Sigma, I)) \setminus SYN(\Sigma, I)$.

$\Leftarrow$: Ein EAA in Normalform zu einer Trace-Sprache über einem Alphabet mit Unabhängigkeitsrelation mit transitiver Abhängigkeitsrelation ist schwach kooperierend, weil jedes Zeichen in nur einem Teilautomaten verarbeitet wird. Wenn der EAA außerdem deterministisch und seine Endzustandsmenge lokal definierbar ist, sind die Teilautomaten autonom arbeitende endliche Automaten. Der EAA ist modularisiert. Bei transitivem D gilt daher $DLE(\Sigma, I) = SYN(\Sigma, I)$ und folglich $(DLE(\Sigma, I) \cap SKA(\Sigma, I)) \setminus SYN(\Sigma, I) = \emptyset$. □

Die Teilmengenbeziehungen bei nichttransitivem D zwischen den behandelten Sprachklassen zeigt Abb. 5.5.

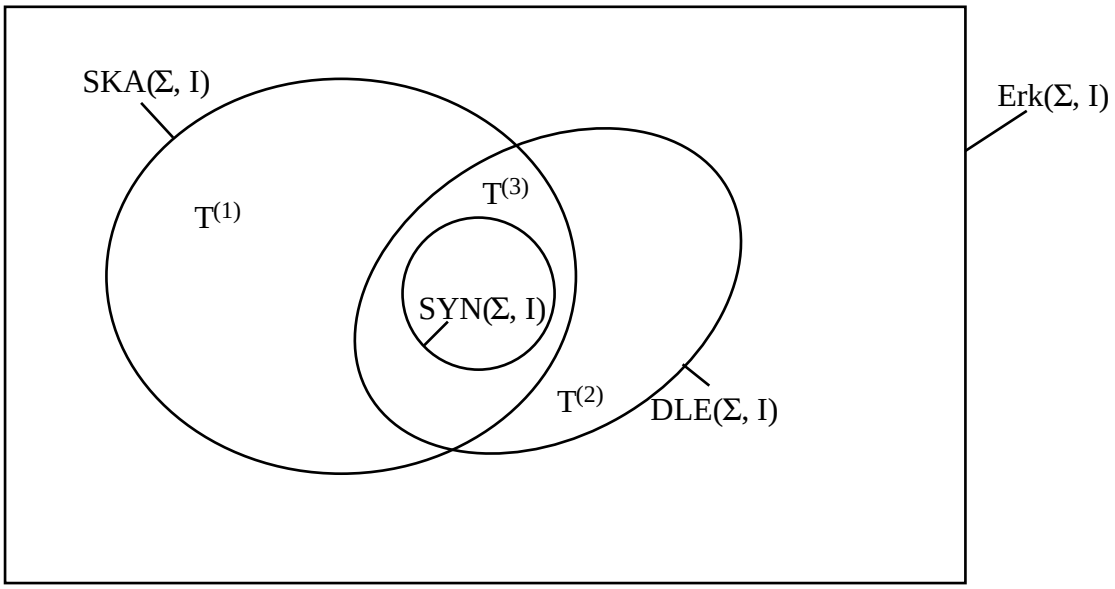

Abbildung 5.5: Teilmengenbeziehungen zwischen den bisher betrachteten Sprachklassen

5.6 Deterministisch-sichere Trace-Sprachen

Wie in Abschnitt 3.6 bereits gezeigt wurde, gibt es erkennbare Trace-Sprachen, zu denen kein deterministischer und sicherer („verklemmungsfreier“) EAA angegeben werden kann. Die Klasse der deterministisch-sicheren Trace-Sprachen ist somit eine echte Teilmenge der erkennbaren Trace-Sprachen.

Definition 5.6.1 (Klasse der deterministisch-sicheren Trace-Sprachen)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über dem Alphabet mit Unabhängigkeitsrelation (Σ, I).

i) T ist determinstisch-sicher
$\Longleftrightarrow_{Df} T = \emptyset$ oder es gibt einen deterministischen und sicheren EAA $\mathcal{A}$ mit $T(\mathcal{A}) = T$ und $I_{\mathcal{A}} = I$.

ii) $DS(\Sigma, I) \subseteq Erk(\Sigma, I)$ ist die Klasse der determinstisch-sicheren Trace-Sprachen über (Σ, I).

Ein EAA verklemmt, wenn zwei Teilautomaten nebenläufig in lokale Zustände laufen, deren Kombination einen globalen Zustand ergibt, von dem aus kein Endzustand erreicht werden kann. Wie bei den deterministischen EAAs mit lokalen Endzuständen wird in den deterministischen sicheren EAAs, diesmal für alle erreichbaren lokalen Zustände, die freie Kombinierbarkeit verlangt. Die deterministisch-sicheren Trace-Sprachen können so automatenunabhängig über $DLE(\Sigma, I)$ charakterisiert werden. Für den Beweis ist ein Lemma notwendig, mit dem von der Erreichbarkeit von lokalen Zuständen auf die Erreichbarkeit von globalen Zuständen geschlossen werden kann.

Lemma 5.6.2 (Levi-Lemma angewendet auf deterministische EAAs)

Sei $\mathcal{A}$ ein deterministischer EAA über (Σ, I) mit der Menge von globalen Zuständen S, und sei $t \in E(\Sigma, I)$ ein Trace, wobei $P_\alpha(t)$ den EAA $\mathcal{A}$ in den globalen Zustand $s_\alpha \in S$ und $P_\beta(t)$ den EAA in den globalen Zustand $s_\beta \in S$ überführt ($\alpha, \beta \subseteq \{1, \ldots, n\}$).
Dann ist mit $P_{\alpha\cup\beta}(t)$ ein Zustand $s_{\alpha\cup\beta} \in S$ erreichbar.

Beweis:
$P_\alpha(t)$ und $P_\beta(t)$ sind Anfangsstücke von t, folglich gibt es Traces $r_\alpha, r_\beta \in E(\Sigma, I)$ mit $P_\alpha(t)r_\alpha = P_\beta(t)r_\beta = t$. Nach dem Levi-Lemma für Traces (Lemma 2.2.11) läßt sich t eindeutig zerlegen als $t = t_0 t_\alpha t_\beta t_r$ mit

i) $P_\alpha(t) = t_0 t_\alpha, \quad r_\alpha = t_\beta t_r$

ii) $P_\beta(t) = t_0 t_\beta, \quad r_\beta = t_\alpha t_r$

iii) $Alph(t_\alpha) \times Alph(t_\beta) \subseteq I$

t_0 ist ein Anfangsstück von $P_\alpha(t)$, so daß $\mathcal{A}$ mit t_0 ebenfalls einen Zustand $s_{t_0} \in S$ erreicht. Da $\mathcal{A}$ deterministisch ist, ist s_{t_0} eindeutig bestimmt. Von den anschließenden Zustandsübergängen bei Abarbeitung von t_α sind nur Teilautomaten $\mathcal{P}_i$ mit $\Sigma_i \cap Alph(t_\alpha) \neq \emptyset$ betroffen. Ebenso läßt sich die Menge der Teilautomaten bestimmen, die an der Abarbeitung von t_β beteiligt sind. Wegen iii) sind diese beiden Teilautomatenmengen disjunkt, so daß beginnend bei s_{t_0} die Zustandsübergänge füt t_α und t_β unabhängig voneinander ausgeführt werden können. Es ergibt sich ein erreichbarer Zustand $s_{\alpha\cup\beta} \in S$. □

Der Zusammenhang zwischen den beiden Sprachklassen $DLE(\Sigma, I)$ und $DS(\Sigma, I)$ kann nun folgendermaßen konkretisiert werden:

Satz 5.6.3 (Beziehung $DLE(\Sigma, I) - DS(\Sigma, I)$)

Sei $T \subseteq Erk(\Sigma, I)$ eine erkennbare Trace-Sprache über (Σ, I). Es gilt:

$$T \in DS(\Sigma, I) \iff Pref(T) \in DLE(\Sigma, I)$$

Beweis:
$\Rightarrow$: Sei $\mathcal{A}$ ein deterministischer und sicherer EAA mit $I_{\mathcal{A}} = I$ und $T(\mathcal{A}) = T$. Sei $t \in E(\Sigma, I)$ ein Trace mit der Eigenschaft

$$\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in Pref(T)} P_i(t) = P_i(r).$$

Offensichtlich gilt für t selber schon $\bigwedge_{i \in \{1,\ldots,n\}} P_i(t) \in Pref(T)$. Die $P_i(t)$ überführen in $\mathcal{A}$ den Anfangszustand in eindeutig bestimmte erreichbare globale Zustände, weil sie Anfangsstücke von Traces aus $T(\mathcal{A})$ sind und $\mathcal{A}$ deterministisch ist. Nach Lemma 5.6.2 wird folglich auch mit $t = P_{\{1,\ldots,n\}}(t)$ ein globaler Zustand erreicht. Von diesem Zustand aus läßt sich die Zustandsfolge in einen Endzustand fortsetzen, denn $\mathcal{A}$ ist sicher. Folglich ist $t \in Pref(T)$ und somit nach Satz 5.3.8 $Pref(T) \in DLE(\Sigma, I)$.

$\Leftarrow$: Sei $T \subseteq E(\Sigma, I)$ eine nichtleere Trace-Sprache mit $Pref(T) \in DLE(\Sigma, I)$. Zu T wird nach Zielonka (vgl. Abschnitt 4.1) der deterministische EAA konstruiert. In diesem EAA werden alle lokalen Zustände gestrichen, die in keiner akzeptierenden Zustandsfolge als Komponenten von globalen Zuständen auftreten. Die erkannte Trace-Sprache bleibt von dieser Operation unberührt. Sei $\mathcal{A}$ der so reduzierte EAA nach Zielonka. Die Zustandsmenge von $\mathcal{A}$ ist nicht leer, weil $T \neq \emptyset$. $\mathcal{A}$ ist deterministisch.
Sei $s = (s_1, \ldots, s_n)$ ein erreichbarer Zustand in $\mathcal{A}$ und $t \in E(\Sigma, I)$ ein Trace, mit dem s vom Anfangszustand aus erreicht wird. Die lokalen Zustände s_i treten in globalen Zuständen $s^i = (s_1^i, \ldots, s_i, \ldots, s_n^i)$ auf, die in akzeptierenden Zustandsfolgen von $\mathcal{A}$ vorkommen.
Sei $r_i \in E(\Sigma, I)$ ein Trace, mit dem s^i vom Anfangszustand aus erreicht wird. Weil von s^i aus ein Endzustand erreichbar ist, gilt $r_i \in Pref(T)$. Die vier Traces $t, r_i, P_i(t)$ und $P_i(r_i)$ überführen den Teilautomaten $\mathcal{P}_i$ in den Zustand s_i. Nach Zielonkas Konstruktion gilt somit $P_i(t) \approx P_i(r_i)$, also insbesondere $P_i(t) \approx_T P_i(r_i)$ und $P_i(t) \sim_T P_i(r_i)$. Weil mit $r_i \in Pref(T)$ auch $P_i(r_i)$ in der Präfixsprache von T liegt und wie gerade gesehen $P_i(t)$ und $P_i(r_i)$ syntaktisch kongruent bezüglich T sind, gilt $P_i(t) \in Pref(T)$. t hat also die Eigenschaft

$$\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in Pref(T)} P_i(t) = P_i(r)$$

(setze $r =_{Df} P_i(t)$), und nach Satz 5.3.8 gilt daher $t \in Pref(T)$, weil $Pref(T) \in DLE(\Sigma, I)$. Der Zustand s war mit t erreichbar, also ist wegen $t \in Pref(T)$ von s aus ein Endzustand erreichbar. Somit ist $\mathcal{A}$ ein deterministischer und sicherer EAA und $T \in DS(\Sigma, I)$. □

Für $DS(\Sigma, I)$ werden wieder die Abschlußeigenschaften bezüglich der mengentheoretischen Operationen, der Spiegelung, der Synchronisation, des Präfixabschlusses und der Verkettung untersucht.

Lemma 5.6.4 ($DS(\Sigma, I)$ abgeschlossen bezüglich $Pref$)

Sei $T \in Erk(\Sigma, I)$ eine erkennbare Trace-Sprache. Es gilt:

$$T \in DS(\Sigma, I) \Rightarrow Pref(T) \in DS(\Sigma, I)$$

Beweis:
Die Behauptung folgt aus Satz 5.6.3. □

Im folgenden Lemma wird die Charakterisierung von $DS(\Sigma, I)$ ausgenutzt, um für konkrete Trace-Sprachen zu zeigen, daß sie nicht deterministisch-sicher sind.

Lemma 5.6.5 (Mengentheoretische Operationen auf $DS(\Sigma, I)$)

Die Klasse $DS(\Sigma, I)$ ist bezüglich der Operationen $\cup$, $\setminus$ und $\cap$ abgeschlossen $\iff I = \emptyset$

Beweis:
$\Rightarrow$: Es reicht aus, das Alphabet mit Unabhängigkeitsrelation $(\Sigma, I) = (\{a, b\}, \{(a, b), (b, a)\})$ heranzuziehen. Für die Vereinigung, das Komplement und den Schnitt können Gegenbeispiele für die Abgeschlossenheit von $DS(\Sigma, I)$ angegeben werden:

i) Vereinigung $\cup$:
Die Trace-Sprachen $T_a = \{[a]\}$ und $T_b = \{[b]\}$ sind deterministisch-sicher. $T = T_a \cup T_b$ ist nicht deterministisch-sicher, wie folgende Überlegung zeigt: Die i-Anfangsstücke von $t = [ab]$ sind $[a]$ und $[b]$ und liegen in $Pref(T)$, es gilt aber $t \notin Pref(T)$. Demnach ist $Pref(T) \notin DLE(\Sigma, I)$ nach Satz 5.3.8 und $T \notin DS(\Sigma, I)$ nach Satz 5.6.3.

ii) Komplement $\setminus$:
Sei $T = \{t \in E(\Sigma, I) || t| \geq 2\}$. T wird vom EAA in Abbildung 5.6 erkannt.

Der Automat ist deterministisch und sicher. Jeder erreichbare Zustand kann in den Endzustand (s_1^2, s_2^2) überführt werden. Das Komplement von T ist $\overline{T} = \{[\varepsilon], [a], [b]\}$. Mit der gleichen Argumentation wie für $\{[a], [b]\}$ in i) kann gezeigt werden: $\overline{T} \notin DS(\Sigma, I)$.

iii) Schnitt $\cap$:
Die Trace-Sprachen $T_1 = \{[a], [b], [aab]\}$ und $T_2 = \{[a], [b], [abb]\}$ sind deterministisch-sicher, wie der EAA in Abbildung 5.7 zeigt.

$T_1 \cap T_2$ ergibt $\{[a], [b]\}$. $\{[a], [b]\} \notin DS(\Sigma, I)$ wurde bereits in i) gezeigt.

$\Leftarrow$: Jede erkennbare Trace-Sprache über $(\Sigma, \emptyset)$ ist determinstsch-sicher. Die erkennbaren Trace-Sprachen sind gegen die mengentheoretischen Operationen abgeschlossen (Lemma 3.1.1). □

Lemma 5.6.6 (Spiegelung und Verkettung auf $DS(\Sigma, I)$)

Die Klasse $DS(\Sigma, I)$ ist gegen Spiegelung bzw. Verkettung abgeschlossen
$\iff I = \emptyset \vee I = \Sigma^2 \setminus id$

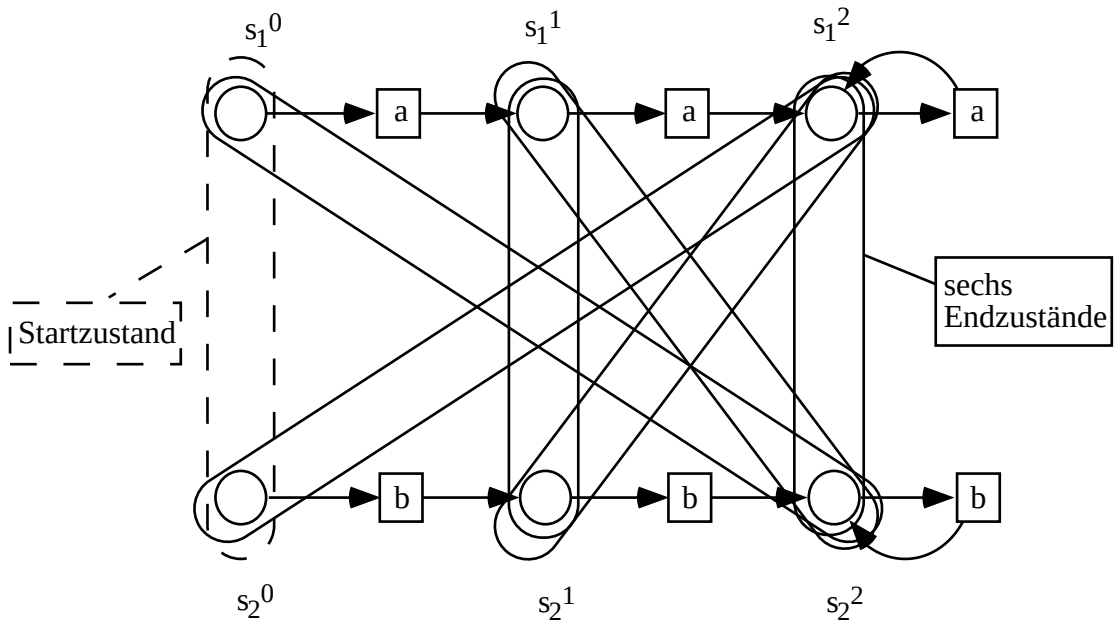

Abbildung 5.6: EAA für die Trace-Sprache $\{t \in E(\Sigma, I) || t| \geq 2\}$

Beweis: $\Rightarrow$: Wenn I nicht trivial ist, gibt es in (Σ, I) zwei abhängige Zeichen und zwei unabhängige Zeichen. Es lassen sich zwei Fälle ableiten, von denen mindestens einer vorliegen muß:

i) Es gibt drei Zeichen $a, b, c \in \Sigma$ mit $(a, b) \in I$ und $(a, c), (b, c) \in D$.

   (a) Für die Spiegeloperation wird die Trace-Sprache $T = \{[ca], [ccb]\}$ untersucht. Einen deterministischen und sicheren EAA für T zeigt Abbildung 5.8, folglich haben wir $T \in DS(\Sigma, I)$.
   Für die Spiegelsprache $T^R = \{[ac], [bcc]\}$ gilt andererseits $[a], [b] \in Pref(T^R)$ und $[ab] \notin Pref(T^R)$. $T^R \notin DS(\Sigma, I)$ ergibt sich wieder aus den Sätzen 5.3.8 und 5.6.3.

   (b) Die Trace-Sprachen $T_1 = [(ca)^*]$ und $T_2 = [(cb)^*]$ sind deterministisch-sicher. $T_1 \circ T_2$ ist nicht deterministisch-sicher, weil $[ca], [cb] \in Pref(T_1 \circ T_2)$ und $[cab] \notin Pref(T_1 \circ T_2)$.

ii) Es gibt drei Zeichen $a, b, d \in \Sigma$ mit $(a, b), (b, d) \in I$ und $(a, d) \in D$.

   (a) Hier spielt die Trace-Sprache $T = [(ad)^*b \cup (ad)^*a]$ die entscheidende Rolle. Ein EAA für T besteht aus einem Teilautomaten für die Zeichen $\{a, d\}$ und einem Teilautomaten für $\{b\}$, die schwach kooperieren. Im deterministischen EAA für T kann jeder erreichbare Zustand, der nicht Endzustand ist, mit b oder d in einen Endzustand überführt werden. Daher $T \in DS(\Sigma, I)$.
   $T^R = [b(da)^* \cup a(da)^*]$ ist mit der gleichen Begründung wie bei i) nicht deterministisch sicher: $[a], [b] \in Pref(T^R), \quad [ab] \notin Pref(T^R)$

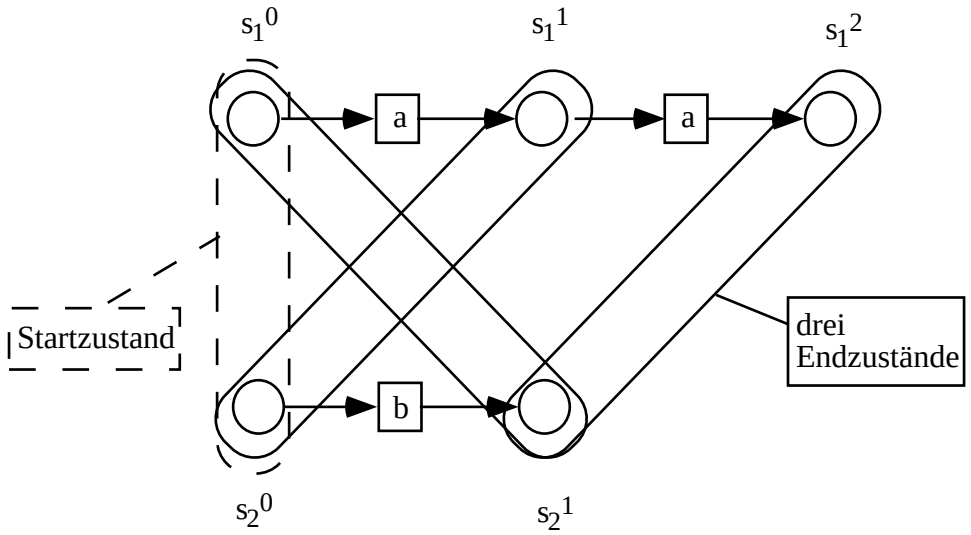

Abbildung 5.7: Deterministischer und sicherer EAA für die Trace-Sprache $\{[a], [b], [aab]\}$

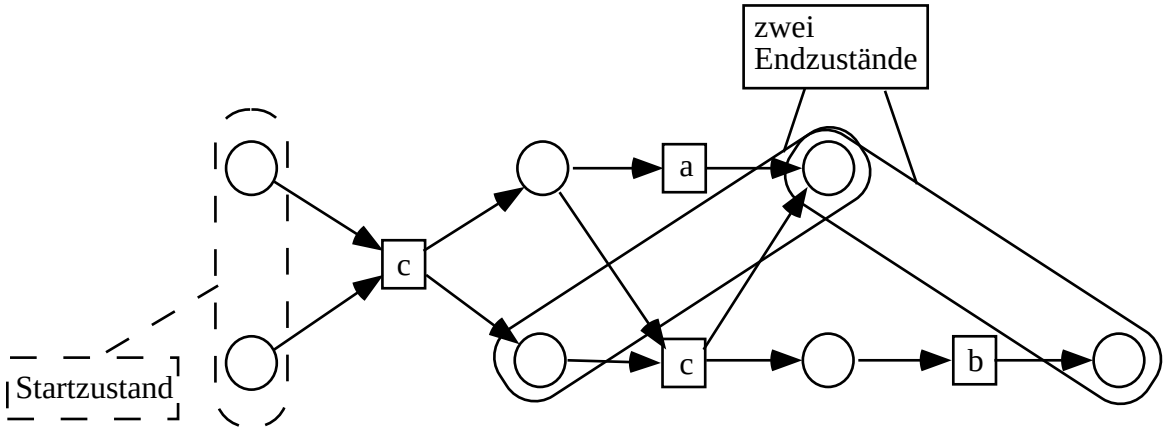

Abbildung 5.8: Deterministischer und sicherer EAA für die Trace-Sprache $\{[ca], [ccb]\}$

(b) $T_1 = \{[\varepsilon], [bd]\}$ und $T_2 = \{[\varepsilon], [a]\}$ sind deterministisch-sicher, nicht jedoch $T_1 \circ T_2 = \{[\varepsilon], [a], [bd], [bda]\}$, weil $[a], [b] \in Pref(T_1 \circ T_2)$ und $[ab] \notin Pref(T_1 \circ T_2)$.

$\Leftarrow$: Für $I = \emptyset$ ist $DS(\Sigma, I) = Erk(\Sigma, I)$. Die erkennbaren Trace-Sprachen sind gegen Spiegelung abgeschlossen. Im Falle $I = \Sigma^2 \setminus id$ können alle Zeichen vertauscht werden. Folglich gilt $([w])^R = [w]$ für alle Wörter $w \in \Sigma^*$. Also ist $T = T^R$.
Mit $I = \Sigma^2 \setminus id$ sind die Traces über (Σ, I) eindeutig durch die Anzahl der Vorkommen der einzelnen Zeichen bestimmt. Sei $\Sigma = \{a_1, \ldots, a_n\}$. Die Klasse $DS(\Sigma, I)$ läßt sich nun einfacher charakterisieren:

$$T \in DS(\Sigma, I) \iff \bigwedge_{m_1,\ldots,m_n \in \mathbb{N}_0} \Big(\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{t_i \in T} m_i \leq |t_i|_{a_i} \Rightarrow \bigvee_{t \in T} \bigwedge_{i \in \{1,\ldots,n\}} m_i \leq |t|_{a_i} \Big)$$

Seien $T_1, T_2 \in DS(\Sigma, I)$ zwei deterministisch-sichere Trace-Sprachen. Seien $m_1, \ldots, m_n \in \mathbb{N}_0$ vorgegebene Häufigkeiten der Zeichen $a_1, \ldots, a_n \in \Sigma$. Zu zeigen ist

$$\bigwedge_{i\in\{1,\ldots,n\}} \bigvee_{t_i\in T_1\circ T_2} m_i \leq |t_i|_{a_i} \Rightarrow \bigvee_{t\in T_1\circ T_2} \bigwedge_{i\in\{1,\ldots,n\}} m_i \leq |t|_{a_i}.$$

Aus $\bigwedge_{i\in\{1,\ldots,n\}} \bigvee_{t_i\in T_1\circ T_2} m_i \leq |t_i|_{a_i}$ folgt $\bigwedge_{i\in\{1,\ldots,n\}} \bigvee_{t'_i\in T_1, t''_i\in T_2} m_i \leq |t'_i|_{a_i} + |t''_i|_{a_i}$. Weil T_1 und T_2 deterministisch-sicher sind, gibt es nach der einfacheren Charakterisierung zwei Traces $t' \in T_1$ und $t'' \in T_2$ mit

$$\bigwedge_{i\in\{1,\ldots,n\}} |t'_i|_{a_i} \leq |t'|_{a_i} \wedge |t''_i|_{a_i} \leq |t''|_{a_i}.$$

Es gilt somit insbesondere für $t = t't'' \in T_1 \circ T_2$:

$$\bigwedge_{i\in\{1,\ldots,n\}} m_i \leq |t'_i|_{a_i} + |t''_i|_{a_i} \leq |t'|_{a_i} + |t''|_{a_i} = |t|_{a_i}$$

Also ist die Implikation nachgewiesen, und nach der Charakterisierung für $DS(\Sigma, I)$ ist $T_1 \circ T_2 \in DS(\Sigma, I)$. □

Lemma 5.6.7 (Synchronisation auf $DS(\Sigma, I)$)

Die Synchronisation von deterministisch-sicheren Trace-Sprachen ergibt nicht in jedem Fall eine determinstisch-sichere Trace-Sprache über dem Synchronisationsalphabet.

Beweis:
Sei $(\Sigma_1, I_1) = (\{a, b\}, \emptyset)$ und $(\Sigma_2, I_2) = (\{b, c\}, \emptyset)$. Wegen $I_1 = I_2 = \emptyset$ sind alle erkennbaren Trace-Sprachen über (Σ_1, I_1) und (Σ_2, I_2) deterministisch-sicher.
Sei $T_1 = \{[\varepsilon], [ab]\} \in Erk(\Sigma_1, I_1)$ und $T_2 = \{[b], [c]\} \in Erk(\Sigma_2, I_2)$. Die Synchronisation von T_1 und T_2 ist über dem Alphabet mit Unabhängigkeitsrelation $(\{a, b, c\}, \{(a, c), (c, a)\})$ definiert:

$$T = T_1 \| T_2 = \{[ab], [c]\}$$

T ist nicht deterministisch-sicher, denn $[a], [c] \in Pref(T)$ und $[ac] \notin Pref(T)$. □

5.7 Erkennbare Quasiverbände

Die letzte der fünf zu betrachtenden Sprachklassen wird unabhängig vom EAA definiert. Sie wird durch eine Abschlußeigenschaft charakterisiert, die auf einer abstrakten Ebene einen Aspekt nebenläufigen Verhaltens beschreibt. Demnach treten unabhängige Ereignisse unabhängig voneinander ein. Für eine Trace-Sprache $T \subseteq E(\Sigma, I)$, die das Verhalten eines nebenläufigen Systems beschreibt, wird mit $[a], [b] \in T$ und $(a, b) \in I$ auch $[\varepsilon], [ab] \in T$ verlangt, weil andernfalls die Eintritte von a und b sich gegenseitig ausschließen.

Für die Übertragung dieser Idee auf längere Traces wird auf die Halbordnung $(E(\Sigma, I), \sqsubseteq)$ (Definition 2.2.10) zurückgegriffen. Mit der Halbordnung können zwei neue Operationen auf Traces definiert werden, die Infimum und Supremum ergeben. Für eine Menge von Traces ist der leere Trace $[\varepsilon]$ stets eine untere Schranke. Eine obere Schranke existiert nicht in jedem Fall.

Definition 5.7.1 (Schnitt und Vereinigung auf Traces)

Seien $t_1, t_2 \in E(\Sigma, I)$ Traces über (Σ, I).

i) Der Schnitt $t = t_1 \sqcap t_2$ von t_1 und t_2 ist der bezüglich $\sqsubseteq$ maximale Trace $t \in E(\Sigma, I)$ mit $t \sqsubseteq t_1$ und $t \sqsubseteq t_2$.

ii) Gibt es einen Trace $s \in E(\Sigma, I)$ mit $t_1 \sqsubseteq s$ und $t_2 \sqsubseteq s$, so ist die Vereinigung $t = t_1 \sqcup t_2$ von t_1 und t_2 der bezüglich $\sqsubseteq$ minimale Trace $t \in E(\Sigma, I)$ mit $t_1 \sqsubseteq t$ und $t_2 \sqsubseteq t$.

Lemma 5.7.2 (Eigenschaften der Operationen)

i) $t_1 \sqcup t_2$ und $t_1 \sqcap t_2$ sind eindeutig bestimmt.

ii) Die Operationen $\sqcup$ und $\sqcap$ sind kommutativ und assoziativ.

iii) Seien $t_1, t_2 \in E(\Sigma, I)$ zwei Traces mit $t_1, t_2 \in Pref(s)$ für einen dritten Trace $s \in E(\Sigma, I)$. Es gilt:

$$Vor(t_1 \sqcup t_2) = Vor(t_1) \cup Vor(t_2)$$

$$Vor(t_1 \sqcap t_2) = Vor(t_1) \cap Vor(t_2)$$

Beweis:

i) Es wird zuerst der Fall $t_1, t_2 \in Pref(s)$ betrachtet. Nach dem Levi-Lemma für Traces gibt es vier Traces $t_0, t', r', r_0 \in E(\Sigma, I)$ mit $t_0 t' r' r_0 = s$, $t_0 t' = t_1$, $t_0 r' = t_2$ und $Alph(t') \times Alph(r') \subseteq I$. Weil t_0 sowohl Anfangsstück von t_1 als auch von t_2 ist, gilt $Vor(t_0) \subseteq Vor(t_1) \cap Vor(t_2)$. Sei (a, n) ein Vorkommen aus $Vor(t_1) \cap Vor(t_2)$. Es ist $a \notin Alph(t') \cap Alph(r')$, da diese Schnittmenge aufgrund $Alph(t') \times Alph(r') \subseteq I$ leer ist. Folglich ist $a \in Alph(t_0)$, somit $(a, n) \in Vor(t_0)$ und $Vor(t_0) = Vor(t_1) \cap Vor(t_2)$. Sei $t_0' \in E(\Sigma, I)$ ein Trace mit $t_0' \sqsubseteq t_1$, $t_0' \sqsubseteq t_2$ und $t_0 \sqsubseteq t_0'$. Aus den Anfangsstückbeziehungen ergibt sich $Vor(t_0) \subseteq Vor(t_0') \subseteq Vor(t_1) \cap Vor(t_2)$. Wegen $Vor(t_0) = Vor(t_1) \cap Vor(t_2)$ bekommen wir $Vor(t_0) = Vor(t_0')$. Mit $t_0 \sqsubseteq t_0'$ gilt demnach $t_0 = t_0'$. Also ist t_0 mit diesen Eigenschaften bezüglich $\sqsubseteq$ maximal und wir haben $t_0 = t_1 \sqcap t_2$. Sei t_0'' ein zweiter maximaler Trace mit $t_0'' \sqsubseteq t_1$, $t_0'' \sqsubseteq t_2$. Für t_0'' gilt ebenfalls $Vor(t_0'') = Vor(t_1) \cap Vor(t_2) = Vor(t_0)$. Eine erneute Anwendung des Levi-Lemmas, diesmal auf $t_0, t_0'' \sqsubseteq t_1$, ergibt $t_0 = t_0''$. Somit ist $t_1 \sqcap t_2$ eindeutig bestimmt.
Die Eindeutigkeit von $t_1 \sqcup t_2$ wird analog unter Ausnutzung des Levi-Lemmas gezeigt. Für beliebige $t_1, t_2 \in E(\Sigma, I)$ ist noch die Eindeutigkeit von $t_1 \sqcap t_2$ zu zeigen. Seien $t, t' \in E(\Sigma, I)$ zwei bezüglich $\sqsubseteq$ maximale Traces mit $t, t' \sqsubseteq t_1$ und $t, t' \sqsubseteq t_2$. Wie gerade gezeigt wurde, ist der Trace $t \sqcup t'$ eindeutig bestimmt und es gilt $t, t' \sqsubseteq t \sqcup t' \sqsubseteq t_1$ und $t, t' \sqsubseteq t \sqcup t' \sqsubseteq t_2$. Aus der Maximalität von t und t' folgt schließlich $t = t \sqcup t' = t'$.

ii) Die Kommutativität von $\sqcup$ und $\sqcap$ ergibt sich aus der Formulierung dieser beiden Operationen. Seien $t_1, t_2, t_3 \in E(\Sigma, I)$ drei Traces. $(t_1 \sqcap t_2) \sqcap t_3$ ist der bezüglich $\sqsubseteq$ maximale Trace t mit $t \sqsubseteq t_1 \sqcap t_2$ und $t \sqsubseteq t_3$. Folglich gilt $t \sqsubseteq t_1$, $t \sqsubseteq t_2$ und $t \sqsubseteq t_3$. Sei $t' \in E(\Sigma, I)$ ein bezüglich $\sqsubseteq$ maximaler Trace mit dieser Eigenschaft. Neben $t \sqsubseteq t_3$ haben wir $t' \sqsubseteq t_1 \sqcap t_2$, weil $t_1 \sqcap t_2$ das maximale eindeutig bestimmte Element unterhalb von t_1 und t_2 ist. Weil t wiederum der maximale und eindeutig bestimmte Trace mit diesen beiden Eigenschaften ist, gilt $t' = t$. Somit ist $(t_1 \sqcap t_2) \sqcap t_3$ unabhängig von der Klammerung.
$(t_1 \sqcup t_2) \sqcup t_3$ ist definiert, wenn es einen Trace $s \in E(\Sigma, I)$ mit $t_1 \sqcup t_2 \sqsubseteq s$ und $t_3 \sqsubseteq s$ gibt. Dann ist aber auch $t_1 \sqsubseteq s$ und $t_2 \sqsubseteq s$, so daß die Definiertheit von der Klammerung des Terms unabhängig ist. $(t_1 \sqcup t_2) \sqcup t_3 = t_1 \sqcup (t_2 \sqcup t_3)$ wird wie für den Schnitt $\sqcap$ bewiesen.

iii) Die Behauptung wurde bereits unter i) bewiesen. □

Definition 5.7.3 (Quasiverbände, Klasse $EQV(\Sigma, I)$)

Sei $T \subseteq E(\Sigma, I)$ eine Trace-Sprache über (Σ, I).

i) T ist ein Quasiverband

$$\Longleftrightarrow_{Df} \bigwedge_{t_1, t_2 \in T} \Big(\bigvee_{t' \in E(\Sigma, I)} t_1 \sqsubseteq t' \wedge t_2 \sqsubseteq t' \Big) \Rightarrow t_1 \sqcup t_2 \in T \wedge t_1 \sqcap t_2 \in T$$

ii) $EQV(\Sigma, I) \subseteq Erk(\Sigma, I)$ ist die Klasse der erkennbaren Quasiverbände über (Σ, I).

Die Eigenschaften eines Quasiverbands werden deutlicher, wenn das Levi-Lemma für Traces (Lemma 2.2.11, Abb. 2.3) herangezogen wird. Für zwei Traces $t_1, t_2 \in T$ mit $t_1 = t_0 t_1'$, $t_2 = t_0 t_2'$ und $Alph(t_1') \times Alph(t_2') \subseteq I$ gilt im Quasiverband T stets $t_0 \in T$ und $t_0 t_1' t_2' = t_0 t_2' t_1' \in T$, d. h. wenn t_1 und t_2 ein gemeinsames Anfangsstück t_0 haben und sich nur in den nebenläufigen Schlußstücken t_1' und t_2' unterscheiden, dann gehören sowohl das Anfangsstück $t_0 = t_1 \sqcap t_2$ als auch die Vereinigung $t_0 t_1' t_2' = t_1 \sqcup t_2$ zum Quasiverband T.

Eine Untersuchung der erkennbaren Quasiverbände auf Abschlußeigenschaften ergibt folgendes Bild:

Lemma 5.7.4 ($EQV(\Sigma, I)$ gegen Schnitt abgeschlossen)

Seien $T_1, T_2 \in EQV(\Sigma, I)$ zwei erkennbare Quasiverbände.
Dann ist der Schnitt $T_1 \cap T_2$ ebenfalls ein Quasiverband: $T_1 \cap T_2 \in EQV(\Sigma, I)$.

Beweis:
Sei $T = T_1 \cap T_2$. Seien $t_1, t_2 \in T$ zwei Traces aus T mit $t_1, t_2 \in Pref(t')$ für einen Trace $t' \in E(\Sigma, I)$.
t_1 und t_2 sind sowohl Traces aus T_1 wie aus T_2. Weil T_1 und T_2 Quasiverbände sind, gilt

$$t_1 \sqcap t_2 \in T_1 \wedge t_1 \sqcap t_2 \in T_2$$

$$t_1 \sqcup t_2 \in T_1 \wedge t_1 \sqcup t_2 \in T_2,$$

folglich

$$t_1 \sqcap t_2 \in T_1 \cap T_2 \wedge t_1 \sqcup t_2 \in T_1 \cap T_2.$$

Somit ist $T_1 \cap T_2$ ein Quasiverband. Weil die erkennbaren Trace-Sprachen gegen Schnitt abgeschlossen (Satz 3.1.1) und T_1 und T_2 erkennbar sind, ergibt sich schließlich $T_1 \cap T_2 \in EQV(\Sigma, I)$. □

Lemma 5.7.5 (Erkennbare Quasiverbände gegen Synchronisation abgeschlossen)

Seien (Σ_1, I_1) und (Σ_2, I_2) zwei Alphabete mit Unabhängigkeitsrelation und sei (Σ, I) das Alphabet mit Unabhängigkeitsrelation, das sich durch Synchronisation von (Σ_1, I_1) und (Σ_2, I_2) ergibt, es gelte also $\Sigma = \Sigma_1 \cup \Sigma_2$ und $\Sigma^2 \setminus I = \Sigma_1^2 \setminus I_1 \cup \Sigma_2^2 \setminus I_2$.
Dann gilt für erkennbare Trace-Sprachen $T_1 \in Erk(\Sigma_1, I_1)$ und $T_2 \in Erk(\Sigma_2, I_2)$:

$$T_1 \in EQV(\Sigma_1, I_1) \wedge T_2 \in EQV(\Sigma_2, I_2) \Rightarrow T_1 \| T_2 \in EQV(\Sigma, I)$$

Beweis:
Seien $t, t' \in T = T_1 \| T_2$ zwei Traces aus der Synchronisationssprache und $r \in E(\Sigma, I)$ ein Trace mit $t, t' \in Pref(r)$. Es gibt folglich Traces $t_f, t'_f \in E(\Sigma, I)$ mit $tt_f = t't'_f = r$. Sei $i \in \{1, 2\}$.
Durch Projektion auf (Σ_i, I_i) erhält man $t|_{\Sigma_i} t_f|_{\Sigma_i} = t'|_{\Sigma_i} t'_f|_{\Sigma_i} = r|_{\Sigma_i}$. Sei nun $t_i = t|_{\Sigma_i}$ und $t'_i = t'|_{\Sigma_i}$ als abkürzende Schreibweise. Weil T_i Quasiverband ist und $t_i, t'_i \in Pref(r|_{\Sigma_i})$, gilt $t_i \sqcup t'_i \in T_i$ und $t_i \sqcap t'_i \in T_i$. Zu zeigen ist nun $t_i \sqcup t'_i = (t \sqcup t')|_{\Sigma_i}$.
Beide Traces enthalten die gleichen Mengen von Zeichenvorkommen, weil

$$\begin{aligned} Vor(t_i \sqcup t'_i) &= Vor(t_i) \cup Vor(t'_i) \\ &= (Vor(t) \cap (\Sigma_i \times I\!N)) \cup (Vor(t') \cap (\Sigma_i \times I\!N)) \\ &= (Vor(t) \cup Vor(t')) \cap (\Sigma_i \times I\!N) \\ &= Vor(t \sqcup t') \cap (\Sigma_i \times I\!N) \\ &= Vor((t \sqcup t')|_{\Sigma_i}) \end{aligned}$$

Sei $\leq$ die Halbordnung der Zeichenvorkommen von $t \sqcup t'$ und sei $\leq_i$ die Halbordnung der Zeichenvorkommen von $t_i \sqcup t'_i$ (Def. 2.2.7). Abhängige Zeichenvorkommen sind in $t_i \sqcup t'_i$ und $(t \sqcup t')|_{\Sigma_i}$ gleich geordnet, wie folgende Überlegung zeigt:
Seien $(a, n), (b, m) \in Vor(t_i \sqcup t'_i)$ Zeichenvorkommen mit $(a, b) \notin I_i$ und $(a, n) \leq_i (b, m)$. Aufgrund der Zerlegbarkeit von $t_i \sqcup t'_i$ in ein gemeinsames Anfangsstück von t_i und t'_i und zwei unabhängige Schlußstücke gehören die abhängigen Vorkommen (a, n) und (b, m) o. B. d. A. beide zu den Vorkommen von t_i. t_i war durch Projektion von t auf (Σ_i, I_i) entstanden, so daß folglich $(a, n), (b, m) \in Vor(t)$. Die Vereinigung $t \sqcup t'$ von t und t' ergibt sich aus t durch Anhängen eines Schlußstücks, so daß $(a, n), (b, m) \in Vor(t \sqcup t')$. Projektion und Vereinigung von Traces haben keinen Einfluß auf die Reihenfolge von (a, n) und (b, m), also $(a, n) \leq (b, m)$ in $t \sqcup t'$. Die Projektion von $t \sqcup t'$ auf (Σ_i, I_i) schließlich respektiert wieder die Reihenfolge der Zeichenvorkommen, mithin sind (a, n) und (b, m) in $(t \sqcup t')|_{\Sigma_i}$ geordnet

wie in $t_i \sqcup t'_i$.
Somit ist $t_i \sqcup t'_i = (t \sqcup t')|_{\Sigma_i}$ nachgewiesen. Der Beweis für $t_i \sqcap t'_i = (t \sqcap t')|_{\Sigma_i}$ kann analog über die Menge der Zeichenvorkommen und ihre Halbordnung geführt werden.
Nachdem $t_i \sqcup t'_i \in T_i$ und $t_i \sqcap t'_i \in T_i$ bereits gezeigt wurde, haben wir nun $(t \sqcup t')|_{\Sigma_i} \in T_i$ und $(t \sqcap t')|_{\Sigma_i} \in T_i$ für $i \in \{1, 2\}$. Nach der Definition der Synchronisation gilt folglich $t \sqcup t' \in T$ und $t \sqcap t' \in T$. Also ist T ein Quasiverband. Weil die erkennbaren Trace-Sprachen gegen Synchronisation abgeschlossen sind, haben wir schließlich $T \in EQV(\Sigma, I)$. □

Lemma 5.7.6 (Vereinigung, Komplement und Verkettung auf $EQV(\Sigma, I)$)

Die Klasse $EQV(\Sigma, I)$ ist gegen Vereinigung, Komplement bzw. Verkettung abgeschlossen $\iff I = \emptyset$

Beweis:
$\Rightarrow$: Seien $a, b \in \Sigma$ zwei unabhängige Zeichen in (Σ, I).

i) Die Trace-Sprachen $\{[a]\}$ und $\{[b]\}$ sind einelementig und daher erkennbare Quasiverbände. $T = \{[a]\} \cup \{[b]\}$ ist kein Quasiverband, weil $[a] \sqcup [b] = [ab] \notin T$. Für $I \neq \emptyset$ ist $EQV(\Sigma, I)$ also nicht gegen Vereinigung abgeschlossen. Zusammen mit Lemma 5.7.4 ergibt sich, daß $EQV(\Sigma, I)$ in diesem Fall auch nicht gegen Komplementbildung abgeschlossen ist.

ii) $T_1 = \{[\varepsilon], [aa]\}$ und $T_2 = \{[\varepsilon], [ab]\}$ sind erkennbare Quasiverbände.
$T_1 \circ T_2 = \{[\varepsilon], [aa], [ab], [aaab]\}$ ist kein Quasiverband, weil $[aa] \sqcap [ab] = [a] \notin T_1 \circ T_2$.

$\Leftarrow$: Wenn alle Zeichen abhängig sind, erfüllt jede erkennbare Trace-Sprache die Anforderungen an einen erkennbaren Quasiverband: $EQV(\Sigma, \emptyset) = Erk(\Sigma, \emptyset)$. Die erkennbaren Trace-Sprachen sind gegen die mengentheoretischen Operationen und gegen Verkettung abgeschlossen. □

Lemma 5.7.7 (Spiegelung und Präfixabschluß auf $EQV(\Sigma, I)$)

Die Klasse $EQV(\Sigma, I)$ ist gegen Spiegelung und Präfixabschluß abgeschlossen
$\iff I = \emptyset \vee I = \Sigma^2 \setminus id$

Beweis:
$\Rightarrow$: Wenn die Unabhängigkeitsrelation nicht trivial ist, liegt einer der beiden folgenden Fälle vor:

i) Es gibt drei Zeichen $a, b, c \in \Sigma$ mit $(a, b) \in I$ und $(a, c), (b, c) \in D$. In diesem Fall wird die Trace-Sprache $T_1 = \{[ac], [bc]\}$ betrachtet. T_1 ist erkennbarer Quasiverband, weil die einzigen beiden Traces in T_1 nicht Anfangsstücke eines dritten Traces sind.
$T_1^R = \{[ca], [cb]\}$ ist kein Quasiverband, weil $[ca], [cb] \in Pref([cab])$ und $[ca] \sqcup [cb] = [cab] \notin T_1^R$.
$Pref(T_1) = \{[\varepsilon], [a], [b], [ac], [bc]\}$ ist ebenfalls kein Quasiverband,
weil $[a], [b] \in Pref([ab])$ und $[a] \sqcup [b] = [ab] \notin Pref(T_1)$.

ii) Es gibt drei Zeichen $a, b, d \in \Sigma$ mit $(a, b), (b, d) \in I$ und $(a, d) \in D$. Hier ist das Gegenbeispiel $T_2 = \{[ab], [da]\}$. Offensichtlich ist $T_2 \in EQV(\Sigma, I)$ mit der gleichen Begründung wie bei i). Es gilt $T_2^R = \{[ab], [ad]\} \notin EQV(\Sigma, I)$, weil $[ab], [ad] \in Pref([abd])$ und $[ab] \sqcup [ad] = [abd] \notin T_2^R$. Ebenso ist $Pref(T_2) = \{[\varepsilon], [a], [b], [d], [ab], [da]\} \notin EQV(\Sigma, I)$, weil $[b], [d] \in Pref([bd])$ und $[b] \sqcup [d] = [bd] \notin Pref(T_2)$.

$\Leftarrow$: Für $I = \emptyset$ ist $Erk(\Sigma, I) = EQV(\Sigma, I)$. Die erkennbaren Trace-Sprachen sind gegen Spiegelung und Präfixabschluß abgeschlossen.
Falls $I = \Sigma^2 \setminus id$ ist, können alle Zeichen in den Traces vertauscht werden, und es gilt $T^R = T$. Zu zeigen ist noch $Pref(T) \in EQV(\Sigma, I)$, wenn $T \in EQV(\Sigma, I)$:
Seien $t_1, t_2 \in Pref(T)$ zwei Traces aus $Pref(T)$. Dann gibt es Traces $t_1', t_2' \in T$ mit $t_1 \in Pref(t_1')$ und $t_2 \in Pref(t_2')$. Weil Traces aus $E(\Sigma, \Sigma^2 \setminus id)$ sich nicht in der Reihenfolge verschiedener Zeichen unterscheiden können, sondern nur in der Menge ihrer Zeichenvorkommen, haben wir $t_1't_2' = t_2't_1'$. Folglich gilt $t_1', t_2' \in Pref(t_1't_2')$, und dann wegen der Quasiverbandseigenschaft von T $t_1' \sqcup t_2' \in T$. Weil $t_1, t_2 \in Pref(t_1't_2')$ ist, existiert $t_1 \sqcup t_2$. Nach Lemma 5.7.2 gilt:

$$Vor(t_1 \sqcup t_2) = Vor(t_1) \cup Vor(t_2) \subseteq Vor(t_1') \cup Vor(t_2') = Vor(t_1' \sqcup t_2')$$

Somit haben wir $t_1 \sqcup t_2 \in Pref(t_1' \sqcup t_2')$, dann $t_1 \sqcup t_2 \in Pref(T)$ und schließlich $t_1 \sqcap t_2 \in Pref(T)$, weil $t_1 \sqcap t_2 \in Pref(t_1 \sqcup t_2)$. Also ist $Pref(T) \in EQV(\Sigma, I)$. □

Die Abschlußeigenschaften der Sprachklassen sind in der folgenden Tabelle zusammengefaßt. Man beachte, daß bei der Synchronisation von Trace-Sprachen das Alphabet (Σ, I) verändert wird. Die Abschlußeigenschaften in der Spalte „$\|$“ beziehen sich daher auf die Sprachklassen über beliebigen (Σ, I).

	$\cup$	$\cap$	*Kompl.*	*Spiegel*	$\|$	*Pref*	$\circ$
$SYN(\Sigma, I)$	$I = \emptyset$	ja	$I = \emptyset$	ja	ja(L)	D transitiv	D transitiv
$SKA(\Sigma, I)$	ja(L)	ja	ja(L)	ja	ja(L)	ja	D transitiv
$DLE(\Sigma, I)$	$I = \emptyset$	ja	$I = \emptyset$	D transitiv	ja	D transitiv	D transitiv
$DS(\Sigma, I)$	$I = \emptyset$	$I = \emptyset$	$I = \emptyset$	I trivial	nein	ja	I trivial
$EQV(\Sigma, I)$	$I = \emptyset$	ja	$I = \emptyset$	I trivial	ja	I trivial	$I = \emptyset$

„ja(L)“ bezeichnet Abschlußeigenschaften, die in der zitierten Literatur behandelt wurden. „I trivial“ meint $I = \emptyset \vee I = \Sigma^2 \setminus id$.

Die Untersuchung der Beziehungen der beiden neudefinierten Sprachklassen zu den Sprachklassen $SYN(\Sigma, I)$, $DLE(\Sigma, I)$ und $SKA(\Sigma, I)$ ergibt, daß jede deterministisch lokal definierbare Trace-Sprache auch Quasiverband ist und daß die Klasse der deterministisch-sicheren Trace-Sprachen quer zu den anderen Sprachklassen liegt.

Satz 5.7.8 ($DLE(\Sigma, I)$ Teilmenge von $EQV(\Sigma, I)$)

Sei $T \in Erk(\Sigma, I)$ eine erkennbare Trace-Sprache. Es gilt:

$$T \in DLE(\Sigma, I) \Rightarrow T \in EQV(\Sigma, I)$$

Beweis:
Sei $T \in DLE(\Sigma, I)$ eine deterministisch lokal definierbare Trace-Sprache und seien $t, t' \in T$ zwei Traces aus T mit $t, t' \in Pref(r)$ für ein $r \in E(\Sigma, I)$. Zu zeigen ist $t \sqcup t', t \sqcap t' \in T$.
Sei $\Sigma_i \subseteq \Sigma$ eine maximale D-Clique in (Σ, I), sei $(a, n) \in Vor(t)$ das letzte Vorkommen eines Zeichens $a \in \Sigma_i$ in t und sei $(b, m) \in Vor(t')$ das letzte Vorkommen eines Zeichens $b \in \Sigma_i$ in t'. Wegen $t, t' \in Pref(r)$ sind (a, n) und (b, m) Vorkommen in r. Weil $a, b \in \Sigma_i$ ist, sind a und b abhängig und es gilt o. B. d. A. $(a, n) \leq_r (b, m)$. Somit ist (a, n) sowohl Zeichenvorkommen in t wie in t', also $(a, n) \in Vor(t) \cap Vor(t')$. Dann aber gilt mit Lemma 5.7.2 auch $(a, n) \in Vor(t \sqcap t')$.
Angenommen es gibt ein Vorkommen $(c, l) \in Vor(t \sqcap t')$ eines Zeichens $c \in \Sigma_i$ mit $(a, n) \leq_{t \sqcap t'} (c, l)$ und $(a, n) \neq (c, l)$. Dann gilt auch $(a, n) \leq_t (c, l)$, weil $t \sqcap t' \in Pref(t)$. Dies widerspricht der Maximalität von (a, n) als Vorkommen eines Zeichens aus Σ_i in t. Also ist (a, n) letztes Vorkommen eines Zeichens aus Σ_i in $t \sqcap t'$ und $P_i(t \sqcap t') = P_i(t)$. Somit ist jedes i-Anfangsstück von $t \sqcap t'$ auch i-Anfangsstück von t oder t' und es gilt

$$\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r_i \in \{t, t'\}} P_i(t \sqcap t') = P_i(r_i).$$

Nach Satz 5.3.8 bekommen wir demnach $t \sqcap t' \in T$, weil $T \in DLE(\Sigma, I)$. $t \sqcup t' \in T$ wird analog bewiesen. Weil T erkennbar ist, gilt schließlich $T \in EQV(\Sigma, I)$. □

Lemma 5.7.9 ($DLE(\Sigma, I) \subseteq DS(\Sigma, I)$ gdw. D transitiv)

Für die Sprachklassen $DLE(\Sigma, I)$ und $DS(\Sigma, I)$ gilt:

$$DLE(\Sigma, I) \subseteq DS(\Sigma, I) \iff D = \Sigma^2 \setminus I \text{ ist transitiv}$$

Beweis:
$\Rightarrow$: Wenn D nicht transitiv ist, gibt es drei Zeichen $a, b, c \in \Sigma$ mit $(a, b) \in I$ und $(a, c), (b, c) \in D$. Die Trace-Sprache $T = \{[ac], [bc]\}$ ist deterministisch lokal definierbar, wie der EAA für T in Abbildung 5.3 zeigt. T ist nicht deterministisch-sicher, weil $[a], [b] \in Pref(T)$ und $[ab] \notin Pref(T)$, d. h. jeder deterministische EAA für T verklemmt nach Eingabe von $[ab]$.

$\Leftarrow$: Die Teilautomaten eines EAAs in Normalform zu einem (Σ, I) mit transitiver Abhängigkeitsrelation arbeiten auf den D-Äquivalenzklassen abhängiger Zeichen. Jedes Zeichen wird also in nur einem Teilautomaten verarbeitet. Wenn der EAA deterministisch und die Endzustandsmenge des EAAs lokal definierbar ist, arbeiten die Teilautomaten autonom auf paarweise disjunkten Teilalphabeten. Jeder der Teilautomaten kann um die lokalen Zustände reduziert werden, von denen aus kein Endzustand erreichbar ist, ohne daß die von ihm akzeptierte Teilsprache davon berührt wird. Ist die vom EAA erkannte Trace-Sprache nicht leer, bleiben nach dieser Reduktion mindestens die Anfangszustände der Teilautomaten zurück. In diesem reduzierten EAA kann aus einem erreichbaren Globalzustand jeder Teilautomat für sich in einen Endzustand überführt werden. Daher ist ein solcher EAA deterministisch und sicher. □

Neben den Teilmengenbeziehungen $SYN(\Sigma, I) \subseteq DLE(\Sigma, I) \cap SKA(\Sigma, I)$ und $DLE(\Sigma, I) \subseteq EQV(\Sigma, I)$ lassen sich bei nichttransitivem D keine weiteren Inklusionen zwischen den fünf

definierten Sprachklassen beweisen. Um dies zu belegen und um die Unterschiede zwischen den Sprachklassen zu verdeutlichen, werden für jede Schnittmenge einfache Beispielsprachen angegeben.

Satz 5.7.10 (Beispielsprachen aus Schnittmengen)

Sei $\Sigma = \{a, b, c, d\}$ *ein Alphabet mit der Unabhängigkeitsrelation*
$I = \{(a, b), (b, a), (b, d), (c, d), (d, b), (d, c)\}$.
$D \setminus id$ *ist* $d \text{ — } a \text{ — } c \text{ — } b$

i) Die folgende Tabelle gibt Trace-Sprachen aus Differenz- und Schnittmengen an:

$\cap$	$DS(\Sigma, I)$	$Erk(\Sigma, I) \setminus DS(\Sigma, I)$
$SYN(\Sigma, I)$	$\{[\varepsilon], [a], [b], [ab]\}$	$\{[ac], [b]\}$
$DLE(\Sigma, I) \setminus SYN(\Sigma, I)$	$\{[c], [abc]\}$	$\{[ac], [bc]\}$
$EQV(\Sigma, I) \setminus DLE(\Sigma, I)$	$\{[\varepsilon], [ab]\}$	$\{[ab], [d]\}$
$Erk(\Sigma, I) \setminus EQV(\Sigma, I)$	$\{[a], [b], [ab]\}$	$\{[a], [b]\}$

ii) Die in i) angeführten Beispielsprachen gehören zu $SKA(\Sigma, I)$. *Durch Substitution des Zeichens a durch die Trace-Sprache* $[a_1^* a_2 a_3 a_1^*]_{I'}$ *(*D' *wie in Abbildung 5.9) in den sechs Trace-Sprachen außerhalb von* $SYN(\Sigma, I)$ *bekommt man Beispielsprachen aus den entsprechenden Schnittmengen mit* $Erk(\Sigma, I) \setminus SKA(\Sigma, I)$.

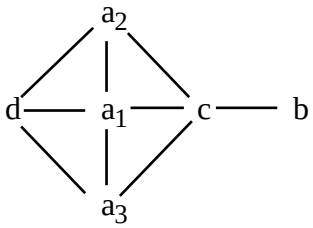

Abbildung 5.9: Abhängigkeitsrelation D' mit a_1, a_2, a_3 für a

Beweis:

i) Die Beweise für die Klassenzuordnungen der einzelnen Sprachen folgen in der Idee den Beweisen der Lemmas 5.6.5, 5.6.6, 5.6.7, 5.7.6 und 5.7.7, in denen Gegenbeispiele für Abschlußeigenschaften angegeben wurden. Insbesondere werden hier erneut die automatenunabhängigen Charakterisierungen der Sprachklassen $SYN(\Sigma, I)$, $DLE(\Sigma, I)$, $EQV(\Sigma, I)$ und $DS(\Sigma, I)$ ausgenutzt.

ii) Die aufgeführten Trace-Sprachen sind endlich, also nach Lemma 3.4.11 in $SKA(\Sigma, I)$. Die Trace-Sprache $[a_1^* a_2 a_3 a_1^*]_{I'}$ mit unabhängigen Zeichen a_2 und a_3 gehört nach Lemma 5.5.2 nicht zu $SKA(\Sigma, I)$. Die Klassenzugehörigkeiten der durch Substitution entstehenden Sprachen werden wieder unter Verwendung der automatenunabhängigen Charakterisierungen der Sprachklassen bewiesen. □

Wenn D transitiv ist, sind einige der Schnittmengen leer. So ist in diesem Fall $SKA(\Sigma, I) = Erk(\Sigma, I)$ und $DLE(\Sigma, I) = SYN(\Sigma, I)$. Nach Lemma 5.7.9 gilt außerdem $DLE(\Sigma, I) \subseteq DS(\Sigma, I)$. In der Tabelle zu Satz 5.7.10 verbleiben mithin nur die fünf Beispielsprachen, die das Zeichen c nicht enthalten.

Abbildung 5.10 zeigt die Teilmengenbeziehungen zwischen den Sprachklassen, wenn D nicht transitiv ist.

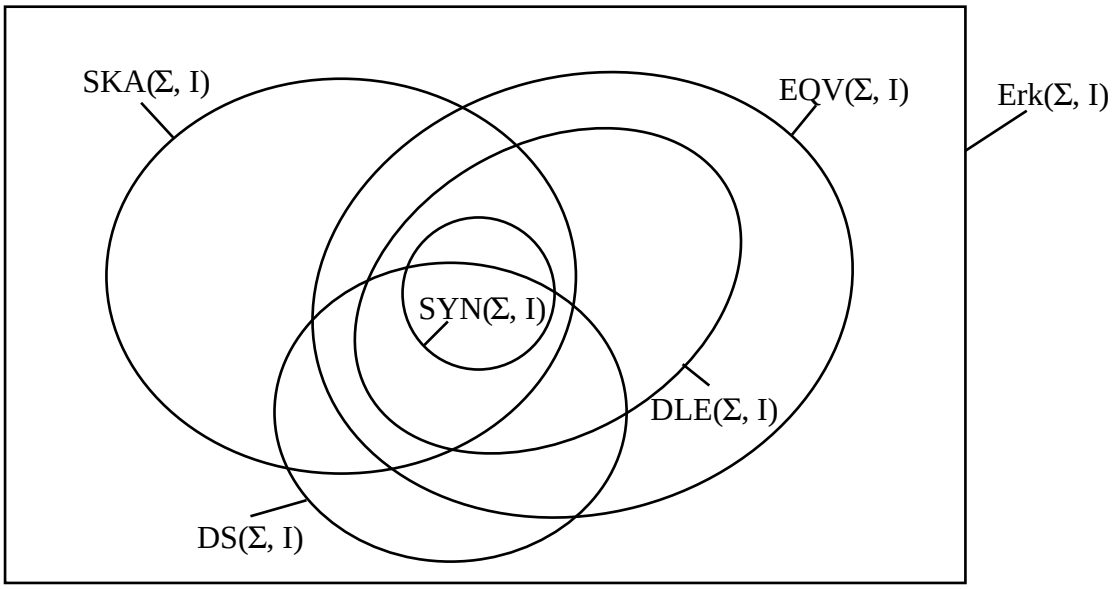

Abbildung 5.10: Teilmengenbeziehungen zwischen den Sprachklassen

5.8 Präfixabgeschlossene Sprachklassen

Zwischen den drei Sprachklassen $DLE(\Sigma, I)$, $DS(\Sigma, I)$ und $EQV(\Sigma, I)$ scheint eine enge Verwandtschaft zu bestehen. Allen dreien ist gemeinsam, daß (bzgl. I) unabhängige Teile eines Traces vom Automaten unabhängig akzeptiert werden. Wie im letzten Abschnitt gezeigt, stimmen die drei Klassen im allgemeinen Fall jedoch nicht überein, weil sie die Unabhängigkeit der Teilautomaten verschieden charakterisieren. Bei Einschränkung auf die präfixabgeschlossenen Trace-Sprachen werden diese Unterschiede aufgehoben.

Satz 5.8.1 (Schnitt mit $Pref(\Sigma, I)$)

Sei $Pref(\Sigma, I)$ die Klasse der präfixabgeschlossenen Trace-Sprachen über dem Alphabet mit Unabhängigkeitsrelation (Σ, I).
Es gilt:

$$EQV(\Sigma, I) \cap Pref(\Sigma, I) = DLE(\Sigma, I) \cap Pref(\Sigma, I) = DS(\Sigma, I) \cap Pref(\Sigma, I)$$

Beweis:
$DLE(\Sigma, I) \cap\ Pref(\Sigma, I) = DS(\Sigma, I) \cap Pref(\Sigma, I)$ folgt aus Satz 5.6.3.
$DLE(\Sigma, I) \cap\ Pref(\Sigma, I) \subseteq EQV(\Sigma, I) \cap\ Pref(\Sigma, I)$ ergibt sich aus Satz 5.7.8.
Sei nun $T \in EQV(\Sigma, I) \cap\ Pref(\Sigma, I)$ und sei $t \in E(\Sigma, I)$ ein Trace mit der Eigenschaft

$$\bigwedge_{i \in \{1,\ldots,n\}} \bigvee_{r \in T} P_i(t) = P_i(r).$$

$T \in Pref(\Sigma, I)$ ergibt dann $P_i(t) \in T$ für alle $i = 1, \ldots, n$. Weil $t = P_1(t) \sqcup P_2(t) \sqcup \ldots \sqcup P_n(t)$ und $T \in EQV(\Sigma, I)$ bekommen wir $t \in T$ und dann mit Satz 5.3.8 $T \in DLE(\Sigma, I)$. □

Der Satz stellt einen Zusammenhang her zwischen der Struktur des deterministischen EAAs ($DLE(\Sigma, I)$), seinem Verhalten ($DS(\Sigma, I)$) und der Abschlußeigenschaft ($EQV(\Sigma, I)$) der von ihm akzeptierten Trace-Sprache. Mazurkiewicz ([Maz87]) bezeichnet die durch den Satz charakterisierten Trace-Sprachen als *trace structures*. Nach Mazurkiewicz werden durch diese Trace-Sprachen das Verhalten von endlichen nebenläufigen Systemen beschrieben. Eine präfixabgeschlossene Trace-Sprache, die anschaulich nicht als Verhalten eines nebenläufigen Systems interpretiert werden kann, ist $T = \{[\varepsilon], [a], [b]\}$ mit $(a, b) \in I$. Auf der anderen Seite umfaßt die Klasse der präfixabgeschlossenen erkennbaren Quasiverbände mehr als die präfixabgeschlossenen synchronisierten Trace-Sprachen.
Sei beispielsweise $(\Sigma, I) = (\{a, b, c\}, \{(a, b), (b, a)\})$ das Alphabet mit Unabhängikeitsrelation und $T = \{[\varepsilon], [a], [b], [ab], [ac], [bc]\}$. T ist ein präfixabgeschlossener erkennbarer Quasiverband. T ist nicht synchronisiert, weil $[abc] \notin T$.

Die im Satz behandelte Sprachklasse soll näher untersucht werden und bekommt deswegen einen Namen:

Definition 5.8.2 (Klasse $PEQ(\Sigma, I)$)

Sei (Σ, I) ein Alphabet mit Unabhängigkeitsrelation. Die Sprachklasse

$$PEQ(\Sigma, I) =_{Df} EQV(\Sigma, I) \cap Pref(\Sigma, I) \subseteq Erk(\Sigma, I)$$

ist die Klasse der präfixabgeschlossenen erkennbaren Quasiverbände über (Σ, I).

Die folgenden Lemmas machen Aussagen zu den Abschlußeigenschaften der präfixabgeschlossenen erkennbaren Quasiverbände.

Lemma 5.8.3 (Abschluß von $PEQ(\Sigma, I)$ gegen $Pref$ und Schnitt)

Seien $T_1, T_2 \in Erk(\Sigma, I)$ erkennbare Trace-Sprachen. Es gilt:

i) $T_1 \in PEQ(\Sigma, I) \Rightarrow Pref(T_1) \in PEQ(\Sigma, I)$

ii) $T_1, T_2 \in PEQ(\Sigma, I) \Rightarrow T_1 \cap T_2 \in PEQ(\Sigma, I)$

Beweis:

i) ist trivial wegen $T_1 \in Pref(\Sigma, I)$.

ii) Die erkennbaren Quasiverbände sind nach Lemma 5.7.4 gegen Schnitt abgeschlossen. Weil der Schnitt von präfixabgeschlossenen Sprachen wieder präfixabgeschlossen ist, ist $PEQ(\Sigma, I)$ somit gegen Mengendurchschnitt abgeschlossen. □

Lemma 5.8.4 (Abschluß von PEQ gegen Synchronisation)

Seien (Σ_1, I_1), (Σ_2, I_2) und (Σ, I) drei Alphabete mit Unabhängigkeitsrelation mit $\Sigma = \Sigma_1 \cup \Sigma_2$ und $\Sigma^2 \setminus I = \Sigma_1^2 \setminus I_1 \cup \Sigma_2^2 \setminus I_2$.
Seien $T_1 \in Erk(\Sigma_1, I_1)$ und $T_2 \in Erk(\Sigma_2, I_2)$ zwei erkennbare Trace-Sprachen. Es gilt:

$$T_1 \in PEQ(\Sigma_1, I_1) \wedge T_2 \in PEQ(\Sigma_2, I_2) \Rightarrow T_1 \| T_2 \in PEQ(\Sigma, I)$$

Beweis:
Mit T_1 und T_2 ist nach Lemma 5.7.5 $T = T_1 \| T_2$ ebenfalls ein erkennbarer Quasiverband. Bleibt zu zeigen, daß $T_1 \| T_2$ präfixabgeschlossen ist.
Sei $t \in T_1 \| T_2$ und $t' \in Pref(t)$. Dann gibt es einen Trace $t_f \in E(\Sigma, I)$ mit $t' t_f = t$. Sei $i \in \{1, 2\}$. Durch Projektion auf (Σ_i, I_i) bekommen wir $t'|_{\Sigma_i} t_f|_{\Sigma_i} = t|_{\Sigma_i}$. Wegen $T = T_1 \| T_2$ gilt $t|_{\Sigma_i} \in T_i$. T_i ist präfixabgeschlossen, so daß folglich $t'|_{\Sigma_i} \in T_i$ für $i \in \{1, 2\}$. Aufgrund von $T = T_1 \| T_2$ ist dann aber wieder $t' \in T$ und also $T \in Pref(\Sigma, I)$. □

Lemma 5.8.5 (Operationen $\cup$, $\setminus$ und ^{R} auf $PEQ(\Sigma, I)$)

i) Die Klasse $PEQ(\Sigma, I)$ ist gegen Vereinigung abgeschlossen $\iff I = \emptyset$

ii) Die Klasse $PEQ(\Sigma, I)$ ist nicht gegen Komplement abgeschlossen.

iii) Die Klasse $PEQ(\Sigma, I)$ ist gegen Spiegelung abgeschlossen $\iff I = \Sigma^2 \setminus id$

Beweis:

i) $\Rightarrow$: Für zwei Zeichen $a, b \in \Sigma$ mit $(a, b) \in I$ läßt sich leicht zeigen, daß die Trace-Sprachen $T_a = \{[\varepsilon], [a]\}$ und $T_b = \{[\varepsilon], [b]\}$ präfixabgeschlossene erkennbare Quasiverbände sind, die Vereinigung von T_a und T_b jedoch aus dieser Sprachklasse herausführt.
$\Leftarrow$: Die erkennbaren Quasiverbände sind für $I = \emptyset$ nach Lemma 5.7.6 gegen Vereinigung abgeschlossen. Weil außerdem die präfixabgeschlossenen Sprachen ebenfalls gegen Vereinigung abgeschlossen sind, gilt die Behauptung.

ii) Die Trace-Sprache $\{[\varepsilon]\}$ ist für jedes Alphabet mit Unabhängigkeitsrelation ein präfixabgeschlossener erkennbarer Quasiverband. Das Komplement von $\{[\varepsilon]\}$ ist nicht leer und enthält den leeren Trace nicht, ist also nicht präfixabgeschlossen.

iii) $\Rightarrow$: Wenn $I \neq \Sigma^2 \backslash id$ ist, gibt es zwei abhängige Zeichen $a, c \in \Sigma$. Die Trace-Sprache $T = \{[\varepsilon], [a], [ac]\}$ ist präfixabgeschlossener erkennbarer Quasiverband. $T^R = \{[\varepsilon], [a], [ca]\}$ ist nicht präfixabgeschlossen.
$\Leftarrow$: Für alle Trace-Sprachen über $(\Sigma, \Sigma^2 \setminus id)$ gilt $T^R = T$. □

Für die Klasse $PEQ(\Sigma, I)$ läßt sich ein einfaches Entscheiungsverfahren angeben, wenn die erkennbare Trace-Sprache T durch einen endlichen Automaten für $Lin(T)$ gegeben ist. Es wird in zwei Schritten vorgegangen: Zuerst wird geprüft, ob T präfixabgeschlossen ist. Wenn ja, kann im zweiten Schritt unter Ausnutzung des folgenden Lemmas entschieden werden, ob T deterministisch lokal definierbar und somit erkennbarer Quasiverband ist.

Lemma 5.8.6 (Kriterium für $T \in DLE(\Sigma, I)$ bei $T \in Pref(\Sigma, I)$)

Sei $T \in Pref(\Sigma, I) \cap Erk(\Sigma, I)$ eine präfixabgeschlossene erkennbare Trace-Sprache. Für T gilt:

$$T \in DLE(\Sigma, I) \quad gdw. \quad \bigwedge_{t \in E(\Sigma, I)} \bigwedge_{(a,b) \in I} t[a] \in T \wedge t[b] \in T \iff t[ab] \in T$$

Beweis:
$\Rightarrow$: Sei $T \in DLE(\Sigma, I) \cap Pref(\Sigma, I)$ eine präfixabgeschlossene deterministisch lokal definierbare Trace-Sprache. Sei $t \in E(\Sigma, I)$ ein Trace und $a, b \in \Sigma$ mit $(a, b) \in I$ zwei unabhängige Zeichen.
$t[ab] \in T \Rightarrow t[a], t[b] \in T$ gilt schon allein wegen $T \in Pref(\Sigma, I)$.
Sei $t[a], t[b] \in T$. Zu T gibt es einen deterministischen EAA $\mathcal{A}$ mit $T(\mathcal{A}) = T$ und lokal definierbarer Endzustandsmenge. t überführt in $\mathcal{A}$ den Anfangszustand in den eindeutig bestimmten globalen Zustand $s = (s_1, \ldots, s_n)$. Mit a wird s in den Endzustand s_f^a überführt, wobei gegenüber s höchstens die lokalen Zustände der Teilautomaten in $Dom(a)$ verändert werden. Ebenso wird s mit b in den Endzustand s_f^b überführt, indem höchstens die Teilautomaten in $Dom(b)$ Zustandsänderungen erfahren. Weil $Dom(a) \cap Dom(b) = \emptyset$ und die Endzustandsmenge lokal definierbar ist, werden die Teilautomaten in $Dom(a)$, $Dom(b)$ bzw. $\{1, \ldots, n\} \setminus (Dom(a) \cup Dom(b))$ durch $t[ab]$ folglich in lokale Endzustände überführt. Demnach gilt $t[ab] \in T$.
$\Leftarrow$: Sei $T \in Pref(\Sigma, I) \cap Erk(\Sigma, I)$ eine präfixabgeschlossene erkennbare Trace-Sprache mit der Eigenschaft

$$\bigwedge_{t \in E(\Sigma, I)} \bigwedge_{(a,b) \in I} t[a] \in T \wedge t[b] \in T \iff t[ab] \in T.$$

Seien $t_1, t_2 \in T$ zwei Traces aus T mit $t_1, t_2 \in Pref(r)$ für einen beliebigen Trace $r \in E(\Sigma, I)$. Nach dem Levi-Lemma für Traces können t_1 und t_2 zerlegt werden als $t_1 = t_0 t_1'$ und $t_2 = t_0 t_2'$ mit $Alph(t_1') \times Alph(t_2') \subseteq I$. Sei $t_1' = [a_1 \ldots a_l]$ und $t_2' = [b_1 \ldots b_m]$. Sei $0 \leq i \leq l$ und $0 \leq j \leq m$. Durch Induktion über die Summe von i und j wird nun bewiesen, daß $t_0[a_1 \ldots a_i b_1 \ldots b_j] \in T$.
Im Falle $i + j = 0$ ist nur t_0 zu betrachten. $t_0 \in T$ gilt wegen $t_0 t_1' \in T$ und $T \in Pref(\Sigma, I)$.
Für $i + j = 1$ ergeben sich $t_0[a_1] \in T$ und $t_0[b_1] \in T$ aus $t_0 t_1', t_0 t_2' \in T$ und $T \in Pref(\Sigma, I)$.
Gelte $t_0[a_1 \ldots a_i b_1 \ldots b_j] \in T$ für alle Indizes i und j mit $0 \leq i \leq l,\ 0 \leq j \leq m$ und $i + j \leq k$.
Es wird $t_0[a_1 \ldots a_i b_1 \ldots b_j]$ mit $0 \leq i \leq l,\ \ 0 \leq j \leq m$ und $2 \leq i + j = k + 1$ betrachtet.

Für $i = 0$ bzw. $j = 0$ folgt $t_0[a_1 \dots a_i b_1 \dots b_j] \in T$ wie im Induktionsanfang aus $t_0 t'_1, t_0 t'_2 \in T$ und $T \in Pref(\Sigma, I)$. Sei also $i \geq 1$ und $j \geq 1$. Sei $t = t_0[a_1 \dots a_{i-1} b_1 \dots b_{j-1}]$. Nach Induktionsvoraussetzung gilt $t[a_i] \in T$ und $t[b_j] \in T$. Mit der Eigenschaft von T ist dann auch $t[a_i b_j] = t_0[a_1 \dots a_i b_1 \dots b_j] \in T$. Also ist $t_0 t'_1 t'_2 = t_1 \sqcup t_2 \in T$ durch Induktion gezeigt. Somit erfüllt T die Eigenschaft eines Quasiverbands, denn $t_1 \sqcap t_2 = t_0 \in T$ gilt aufgrund der Präfixabgeschlossenheit von T. Mit Satz 5.8.1 ist T deterministisch lokal definierbar: $T \in DLE(\Sigma, I)$. □

Das Entscheidungsverfahren für „$T \in PEQ(\Sigma, I)$?“ am deterministischen endlichen Automaten für $Lin(T)$ (mit Endzustandsmenge F) läuft nun so ab: „$T \in Pref(\Sigma, I)$?“ wird überprüft, indem untersucht wird, ob es einen Zustandsübergang von einem erreichbaren Zustand aus $\overline{F}$ in einen Endzustand aus F gibt. Ist das nicht der Fall, so ist $T \in Pref(\Sigma, I)$ und das Verfahren wird fortgesetzt. Jeder erreichbare Zustand $q_f \in F$ wird mit jeder Kombination von zwei seiner direkten Folgezustände $q'_f, q''_f \in F$ betrachtet. Gilt $q_f \xrightarrow{a} q'_f$ und $q_f \xrightarrow{b} q''_f$ mit unabhängigen Zeichen a und b, so muß es nach Lemma 5.8.6 einen vierten Zustand $q''' \in F$ mit $q''_f \xrightarrow{a} q'''_f$ und $q'_f \xrightarrow{b} q'''_f$ geben, wenn $T \in DLE(\Sigma, I)$. Offenbar wächst der Aufwand dieses Verfahrens mit $O(n^3)$, wenn n die Anzahl der Zustände des deterministischen Automaten für $Lin(T)$ ist.

Kapitel 6

Die Konstruktion von EAAs zu Sprachen aus Teilklassen von $Erk(\Sigma, I)$

Die in den Kapiteln 3 und 5 definierten Sprachklassen werden z. T. von EAAs mit besonderen Eigenschaften erkannt. Es ergibt sich die Frage, ob durch Ausnutzung der durch die Sprachklassen charakterisierten Eigenschaften eine einfachere Automatenkonstruktion möglich ist. Im Falle der Klasse $SYN(\Sigma, I)$ kann diese Frage schnell positiv beantwortet werden. Für die Sprachklassen $DLE(\Sigma, I)$, $DS(\Sigma, I)$ und $EQV(\Sigma, I)$ kommen wir zu negativen Ergebnissen.

Für synchronisierte Trace-Sprachen ist die Konstruktion eines modularisierten EAAs einfach. Es reicht aus, zu den durch Projektion auf die maximalen D-Cliquen des Alphabets definierten Wortsprachen die Minimalautomaten zu konstruieren und diese durch Synchronisation (Def. 3.4.1) zu einem modularisierten EAA zusammenzufügen. Unabhängig von der Repräsentation der gegebenen Trace-Sprache reichen die aus der Theorie der gewöhnlichen endlichen Automaten bekannten Methoden zusammen mit der Automatensynchronisation aus.

Für die Klassen $DLE(\Sigma, I)$, $DS(\Sigma, I)$ und $EQV(\Sigma, I)$ zeigen wir durch Reduktion, daß die Automatenkonstruktion für Trace-Sprachen aus diesen Klassen nicht einfacher ist als für erkennbare Trace-Sprachen im allgemeinen. Wenn es für die Teilklassen Konstruktionsverfahren gibt, dann können sie für jede beliebige erkennbare Trace-Sprache genutzt werden.

Satz 6.1 (Konstruktion für $T \in DLE(\Sigma, I)$)

Sei $T \in Erk(\Sigma, I)$ eine erkennbare Trace-Sprache über (Σ, I) und sei $\Sigma' = \Sigma \cup \{z\}$ mit $z \notin \Sigma$ das um das Zeichen z erweiterte Alphabet. Sei z mit allen Zeichen aus Σ abhängig, also $I' =_{Df} I \subseteq \Sigma' \times \Sigma'$. Die Trace-Sprache $T\{[z]\}$ ist deterministisch lokal definierbar:

$$T\{[z]\} \in DLE(\Sigma', I')$$

Beweis:
Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\mathcal{P}_i = (\Sigma_i, S_i)$ ein deterministischer EAA mit $\Sigma_{\mathcal{A}} = \Sigma$, $I_{\mathcal{A}} = I$ und $T(\mathcal{A}) = T$. Es wird auf folgende Weise ein neuer EAA $\mathcal{A}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, \mathcal{I}, \mathcal{D}', \mathcal{F}')$ konstruiert:

i) $\mathcal{P}'_i = (\Sigma'_i, S'_i)$ mit $\Sigma'_i = \Sigma_i \cup \{z\}$ und $S'_i = S_i \cup \{s'_i\}$, wobei s'_i ein neuer Zustand ist.

ii) Sei für $a \in \Sigma$ $Dom(a) = \{i_1, \ldots, i_k\}$. Es wird $\delta'_a : S'_{i_1} \times \ldots \times S'_{i_k} \to \wp(S'_{i_1} \times \ldots \times S'_{i_k})$ definiert:

$$\delta'_a(s_1, \ldots, s_k) =_{Df} \begin{cases} \delta_a(s_1, \ldots, s_k) & \text{für } (s_1, \ldots, s_k) \in S_{i_1} \times \ldots \times S_{i_k} \\ \emptyset & \text{sonst} \end{cases}$$

iii) Für das neue Zeichen z ist $Dom(z) = \{1, \ldots, n\}$. δ'_z wird gesetzt als

$$\delta'_z(s_1, \ldots, s_n) =_{Df} \begin{cases} \{(s'_1, \ldots, s'_n)\} & \text{für } (s_1, \ldots, s_n) \in \mathcal{F} \\ \emptyset & \text{sonst} \end{cases}$$

iv) Die neue Endzustandsmenge ist $\mathcal{F}' =_{Df} \{(s'_1, \ldots, s'_n)\}$.

Offensichtlich ist $\mathcal{A}'$ deterministisch und seine Endzustandsmenge ist lokal definierbar. Es ist nun $T(\mathcal{A}') = T\{[z]\}$ zu zeigen:

$\subseteq$: Sei $t' \in T(\mathcal{A}')$ und sei s_0 der globale Anfangszustand von $\mathcal{A}$ und $\mathcal{A}'$. Dann gilt $s_0 \stackrel{t'}{\Longrightarrow} (s'_1, \ldots, s'_n)$. Nach Definition von $\delta'_a, a \in \Sigma$, und δ'_z führt nur ein z-Übergang in den Zustand $(s'_1, \ldots, s'_n)$, und zwar von einem Endzustand $s_f \in \mathcal{F}$ des Automaten $\mathcal{A}$ aus. Demnach gibt es einen Trace $t \in E(\Sigma, I)$ mit $t' = t[z]$ und $s_0 \stackrel{t}{\Longrightarrow} s_f \stackrel{z}{\Longrightarrow} (s'_1, \ldots, s'_n)$. Also gilt $t' \in T\{[z]\}$.

$\supseteq$: Sei $t' \in T\{[z]\}$, also $t' = t[z]$ mit $t \in T$. t überführt in $\mathcal{A}$ den Anfangs- in einen Endzustand $s_f \in \mathcal{F}$. Nach Definition von δ'_z ist in $\mathcal{A}'$ von s_f ein z-Übergang nach $(s'_1, \ldots, s'_n) \in \mathcal{F}'$ möglich, so daß $t' = t[z] \in T(\mathcal{A}')$. □

Mit einem EAA-Konstruktionsverfahren für deterministisch lokal definierbare Trace-Sprachen bzw. für erkennbare Quasiverbände könnte für jede erkennbare Trace-Sprache T ein EAA in zwei Schritten konstruiert werden. Zunächst wird das Verfahren mit der Trace-Sprache $T\{[z]\}$ zur Konstruktion eines deterministischen EAAs $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ ausgenutzt. Es wird dann eine neue Endzustandsmenge $\mathcal{F}' =_{Df} \{s \in S | \bigvee_{s_f \in \mathcal{F}} s \stackrel{z}{\Longrightarrow} s_f\}$ definiert. Der EAA mit der Endzustandsmenge $\mathcal{F}'$ erkennt die Trace-Sprache T.

Mit der gleichen Argumentation wird gezeigt, daß ein Konstruktionsverfahren für deterministisch-sichere Trace-Sprachen nicht einfacher als ein Konstruktionsverfahren für beliebige erkennbare Trace-Sprachen ist.

Satz 6.2 (Automatenkonstruktion für $T \in DS(\Sigma, I)$)

Sei $T \in Erk(\Sigma, I)$ eine erkennbare Trace-Sprache über (Σ, I). Seien z und $\overline{z}$ zwei neue Zeichen, die mit allen Zeichen aus Σ abhängig sind, also $\Sigma' = \Sigma \cup \{z, \overline{z}\}$ und $I' = I \subseteq \Sigma' \times \Sigma'$. Die Trace-Sprache $T' = T[z] \cup \overline{T}[\overline{z}]$ ist eine deterministisch-sichere Trace-Sprache über (Σ, I).

Beweis:
Sei $\mathcal{A} = (\mathcal{P}_1, \ldots, \mathcal{P}_n, \mathcal{I}, \mathcal{D}, \mathcal{F})$ mit $\mathcal{P}_i = (\Sigma_i, S_i)$ ein deterministischer vollständiger EAA mit $\Sigma_{\mathcal{A}} = \Sigma$, $I_{\mathcal{A}} = I$ und $T(\mathcal{A}) = T$. Es wir ein zweiter EAA $\mathcal{A}' = (\mathcal{P}'_1, \ldots, \mathcal{P}'_n, \mathcal{I}, \mathcal{D}', \mathcal{F}')$ konstruiert:

i) $\mathcal{P}'_i = (\Sigma'_i, S'_i)$ mit $\Sigma'_i = \Sigma_i \cup \{z, \overline{z}\}$ und $S'_i = S_i \cup \{s'_i\}$, wobei s'_i ein neuer Zustand ist.

ii) Sei für $a \in \Sigma$ $Dom(a) = \{i_1, \ldots, i_k\}$. Es wird $\delta'_a : S'_{i_1} \times \ldots \times S'_{i_k} \to \wp(S'_{i_1} \times \ldots \times S'_{i_k})$ definiert:

$$\delta'_a(s_1, \ldots, s_k) =_{Df} \begin{cases} \delta_a(s_1, \ldots, s_k) & \text{für } (s_1, \ldots, s_k) \in S_{i_1} \times \ldots \times S_{i_k} \\ \emptyset & \text{sonst} \end{cases}$$

iii) Für die neuen Zeichen z und $\overline{z}$ ist $Dom(z) = Dom(\overline{z}) = \{1, \ldots, n\}$. Es wird definiert:

$$\delta'_z(s_1, \ldots, s_n) =_{Df} \begin{cases} \{(s'_1, \ldots, s'_n)\} & \text{für } (s_1, \ldots, s_n) \in \mathcal{F} \\ \emptyset & \text{sonst} \end{cases}$$

$$\delta'_{\overline{z}}(s_1, \ldots, s_n) =_{Df} \begin{cases} \{(s'_1, \ldots, s'_n)\} & \text{für } (s_1, \ldots, s_n) \in S \setminus \mathcal{F} \\ \emptyset & \text{sonst} \end{cases}$$

iv) Die neue Endzustandsmenge ist $\mathcal{F}' =_{Df} \{(s'_1, \ldots, s'_n)\}$.

$\mathcal{A}'$ ist deterministisch, und weil jeder erreichbare Globalzustand mit z oder $\overline{z}$ in den Endzustand überführt werden kann, ist $\mathcal{A}'$ sicher. Gezeigt wird $T(\mathcal{A}') = T\{[z]\} \cup \overline{T}\{[\overline{z}]\}$.
$\subseteq$: Sei $t' \in T(\mathcal{A}')$. t' überführt den Anfangszustand in den Endzustand $(s'_1, \ldots, s'_n)$. Nach Definition von $\delta'_a, a \in \Sigma, \delta'_z$ und $\delta'_{\overline{z}}$ gelangt der EAA $\mathcal{A}'$ nur mit einem z- oder $\overline{z}$-Übergang in diesen Endzustand und war vorher in einem Zustand aus $\mathcal{F}$ bzw. $S \setminus \mathcal{F}$ des Automaten $\mathcal{A}$. Folglich gibt es ein $t \in T$ mit $t' = t[z]$ oder ein $t \in E(\Sigma, I) \setminus T$ mit $t' = t[\overline{z}]$, und es ist $t' \in T\{[z]\}$ bzw. $t' \in \overline{T}\{[\overline{z}]\}$.
$\supseteq$: Sei $t' \in \overline{T}\{[\overline{z}]\}$, also $t' = t[\overline{z}]$ mit $t \in E(\Sigma, I) \setminus T$. Weil $\mathcal{A}$ vollständig ist, gelangt $\mathcal{A}$ mit t in einen globalen Zustand $s \in S \setminus \mathcal{F}$. Mit $\delta'_{\overline{z}}$ wird dieser Zustand mittels $\overline{z}$ in den Endzustand von $\mathcal{A}'$ überführt, also $t' = t[\overline{z}] \in T(\mathcal{A}')$. Ebenso wird von $t' \in T\{[z]\}$ auf $t' \in T(\mathcal{A}')$ geschlossen. □

Die Automatenkonstruktion für beliebige erkennbare Trace-Sprachen kann auf der Konstruktion für deterministisch-sichere Trace-Sprachen aufbauen. Statt zu $T \in Erk(\Sigma, I)$ selber wird zu $T\{[z]\} \cup \overline{T}\{[\overline{z}]\}$ der deterministische und sichere EAA $\mathcal{A}$ konstruiert. Mit der neuen Endzustandsmenge $\mathcal{F}' = \{s \in S | \bigvee_{s_f \in \mathcal{F}} s \stackrel{z}{\Longrightarrow} s_f\}$ erkennt der Automat die Trace-Sprache T. Nach Eliminierung der z- und $\overline{z}$-Übergänge im Automaten $\mathcal{A}$ und Streichung der Zeichen z und $\overline{z}$ aus den Alphabeten der Teilautomaten bekommen wir einen EAA über (Σ, I), der T erkennt.

Kapitel 7

Einschätzung und Ausblick

In dieser Arbeit wurden durch Einschränkung des EAA-Konzepts Teilklassen der erkennbaren Trace-Sprachen definiert. Es wurde gezeigt, daß mit den definierten Sprachklassen auch ohne Bezugnahme auf den verteilten Automaten gearbeitet werden kann. Die Sprachklassen sind mit Ausnahme der bereits in der Literatur behandelten Klasse $SKA(\Sigma, I)$ kaum gegen die hier betrachteten Operationen abgeschlossen. Abgesehen von den zu erwartenden Teilmengenbeziehungen konnten alle Inklusionen durch einfache Gegenbeispiele widerlegt werden. Positive Ergebnisse ergaben sich erst mit der Einschränkung des Alphabettyps und bei Beschränkung auf präfixabgeschlossene Sprachen.

Die einfachen Beispiele aus den Differenzmengen der Sprachklassen zeigen, daß die Klassen wesentlich verschiedene Spracheigenschaften bezeichnen. Weil alle Unterschiede bei leerer Unabhängigkeitsrelation des Alphabets aufgehoben sind, ergibt sich die Auffächerung aus der Nebenläufigkeit von Zeichenvorkommen. Somit ist die Betrachtung dieser Sprachklassen eher als die Übertragung von Sprachklassendefinitionen aus der Theorie der Wortsprachen ein Bestandteil der Trace-Theorie. Die in dieser Arbeit definierten Sprachklassen drücken unterschiedliche Aspekte erkennbaren Verhaltens von verteilten Systemen aus: Synchronisiertheit, Sicherheit, verschieden Stufen von Nebenläufigkeit.

Eine wichtige Teilklasse scheinen die präfixabgeschlossenen erkennbaren Quasiverbände zu sein, die anschaulich das Verhalten von verteilten Systemen mit endlicher Zustandsmenge beschreiben. Zielonkas Automatentyp erkennt darüber hinaus alle (präfixabgeschlossenen) erkennbaren Trace-Sprachen, die intuitiv nicht ohne weiteres dieses Kriterium erfüllen, beispielsweise $T = \{[\varepsilon], [a], [b]\}$ mit unabhängigen Zeichen a und b. In dieser Arbeit wurde eine enge Beziehung zwischen den präfixabgeschlossenen erkennbaren Quasiverbänden und dem Verhalten und der Struktur eines EAAs herausgearbeitet.

Die Konstruktion von endlichen asynchronen Automaten ist nur für synchronisierte Trace-Sprachen einfach. Gerade für diese Sprachklasse brauchte des Konzept des EAAs aber nicht eingeführt zu werden, weil hier frei synchronisierte endliche Automaten zur Charakterisierung der Sprachklasse ausreichen. Für größere Sprachklassen wird die Unabhängigkeitsrelation zum bestimmenden Parameter bei der Wahl des Konstruktionsansatzes. In Kapitel 4 wurde anhand einiger einfacher nichtsynchronisierter Beispielsprachen gezeigt, daß schon für

nichttriangulierte vierelementige Alphabete Zielonkas Konstruktion voll ausgenutzt werden muß.

Die Frage nach der Konstruktion von schwach kooperierenden Automaten zu Trace-Sprachen aus $SKA(\Sigma, I)$ ist offen geblieben. Zu diesem Zweck müßte ein Verfahren gefunden werden, das zu einer solchen Sprache eine endliche Menge von synchronisierten Trace-Sprachen berechnet, die zusammen die gegebene Trace-Sprache überdecken.

An die Arbeit können sich weitere Untersuchungen von Trace-Sprachklassen, auch im Zusammenhang mit verteilten Automaten, anschließen. Besonderes Augenmerk ist darauf zu richten, welche Eigenschaften eine Trace-Sprache haben muß, um anschaulich als Verhalten eines verteilten Systems gelten zu können. Im Hinblick auf verteilte endliche Automaten bleibt die Frage, für welche Teilklasse der erkennbaren Trace-Sprachen eine einfache Automatenkonstruktion ausreicht. Verglichen mit den regulären Wortsprachen wissen wir über die erkennbaren Trace-Sprachen wenig, so daß alleine auf diesem Gebiet viele offene Fragen zu beantworten sind.

Anhang A

Modellierung einer Fertigungszelle

Das folgende Beispiel ist [LL93] entnommen. Beschrieben wird dort eine Fertigungszelle, in der Bleche zu Werkstücken gepreßt werden. Die Presse wird über einen Roboter beschickt. Die Bleche werden über ein Förderband dem Roboter zugeführt. Über ein zweites Förderband werden die gepreßten Werkstücke abtransportiert. Die Abbildung A.1 zeigt schematisch ein Modell der Fertigungszelle.

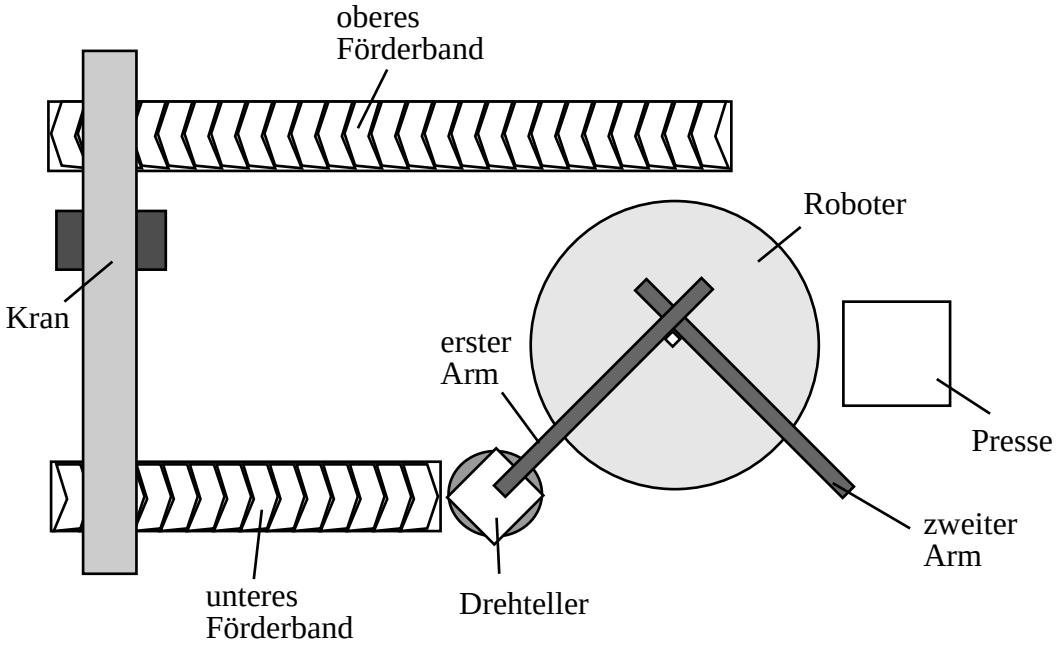

Abbildung A.1: Modell der Fertigungszelle

A.1 Die Komponenten der Fertigungszelle

i) Das untere Förderband übernimmt den Transport der Bleche zum Roboter. Das Förderband wird mit Hilfe einer Lichtschranke gesteuert, die die Ankunft eines Blechs im

hinteren Abschnitt des Bandes registriert. Das Band hält an, wenn das Blech auf den Drehteller geschoben ist, und setzt sich wieder in Bewegung, wenn der (leere) Drehteller seine Ausgangsposition am Ende des Bandes wieder erreicht und ein neues Blech am Kopf des Bandes abgelegt wurde.

ii) Der Drehteller bringt das Blech in eine Position, in der es vom Roboter gegriffen und in die Presse eingelegt werden kann. Das vom Band kommende Blech wird um etwa 45° im Uhrzeigersinn gedreht und auf das Niveau des Roboterarms angehoben. Beide Bewegungen werden unabhängig ausgeführt.

iii) Der Roboter ist zum schnelleren Be- und Entladen der Presse mit zwei Armen ausgestattet. Die waagerecht liegenden Arme sind rechtwinklig angeordnet und können um eine senkrechte Achse geschwenkt werden. Vertikale Bewegungen kann der Roboter nicht ausführen. Die Arme können teleskopartig ein- und ausgefahren werden und mit Elektromagneten die Bleche bzw. Werkstücke greifen. Der Roboter arbeitet in vier Arbeitsschritten:

   (a) Der erste Arm fährt aus, greift ein Blech vom Drehteller und fährt wieder ein. Der Roboter schwenkt links.

   (b) Der zweite (höhergelegene) Arm fährt aus, greift ein gepreßtes Werkstück aus der geöffneten Presse und fährt wieder ein. Der Roboter schwenkt links.

   (c) Der zweite Arm fährt aus, legt das Werkstück auf dem oberen Förderband ab und fährt wieder ein. Der Roboter schwenkt weiter links.

   (d) Der erste Arm fährt aus, legt das Blech in die geöffnete Presse und fährt wieder ein. Der Roboter schwenkt rechts zurück, bis der erste Arm auf den Drehteller zeigt.

iv) Die Presse wird mit einem Blech beschickt, wenn ihr unterer Teil in der tieferen Ladeposition ist. Dann wird das Blech gepreßt. Der bewegliche Teil der Presse fährt in die höhere Entladeposition und nach dem Entladen zurück in die Ladeposition.

v) Das obere Förderband wird wie das untere Förderband über eine Lichtschranke gesteuert, die am Ende des Bandes die Werkstücke registriert. Ein in den letzten Abschnitt des Bandes transportiertes Werkstück wird vom Kran aufgenommen. Das Band wird für diese Aktion angehalten. Der Roboter kann nur dann ein Werkstück am Anfang des Bandes ablegen, wenn durch Anfahren und Anhalten des Bandes dort Platz geschaffen wurde.

vi) Der Kran legt in der Fertigungszelle die Bleche auf das untere Förderband und nimmt die Werkstücke vom oberen Förderband. Im Modell werden beide Aktionen verbunden, um einen zyklischen Ablauf zu bekommen. Wenn ein Werkstück den letzten Abschnitt des oberen Bandes erreicht, fährt der Kran herab und nimmt das Werkstück mit seinem Elektromagneten auf. Das Werkstück wird angehoben, und der Kran fährt zum unteren Förderband. Dort wird das Werkstück abgelegt, wenn das Band nach Abgabe eines Blechs an den Drehteller still steht. Das Band fährt an, wenn im hinteren Teil über die Lichtschranke kontrolliert kein Blech liegt und der Kran am Anfang des Bandes ein Blech abgelegt hat.

A.2 Elementare Aktionen des Trace-Modells

Für die Modellierung muß die Menge der im Modell vorkommenden Aktionen bestimmt und die Abhängigkeiten zwischen diesen Aktionen geklärt werden. Das Ziel der Modellierung ist die Formulierung einer Trace-Sprache, die die Abläufe in der Fertigungszelle beschreibt. Es fließen folgende Überlegungen ein:

i) Auf den Förderbändern liegen die Bleche und Werkstücke wie in einer Warteschlange. Ein neues Objekt kann nur dann am Anfang des Bandes abgelegt werden, wenn am Ende ein Objekt vom Drehteller bzw. Kran aufgenommen wurde und das Band weitertransportiert hat. Es wird angenommen, daß wenn die Lichtschranke ein Blech im hinteren Abschnitt des Bandes registriert, sowohl das Blech an den Drehteller abgegeben als auch ein neues Blech vorne abgelegt wird, bevor das Band weitertransportiert. Meldet die Lichtschranke kein Blech im hinteren Abschnitt, legt der Kran kein neues Blech ab und das Band transportiert unmittelbar weiter. Somit wird die Verminderung der Zahl der Objekte auf dem Band ausgeschlossen. Die elementaren Aktionen sind hier „niederlegen“, „untere_Lichtschranke_abfragen“, „Transport_unteres_Band“ und „Blech_auf_Drehteller_schieben“.

ii) Das Anheben und das Drehen des Drehtellers können als unabhängige Aktionen angesehen werden. Zwischen „Teller_links“ und „Teller_rechts“ sowie „Teller_auf“ und „Teller_ab“ wird unterschieden.

iii) Die Einfahr- und Ausfahrbewegungen der Greifarme des Roboters können mit dem Greifen bzw. Ablegen der Bleche und Werkstücke zusammengefaßt werden. Zusammen mit den Drehbewegungen ergeben sich sechs elementare Aktionen des Roboters, die sich gegenseitig ausschließen.

iv) Für die Presse sind drei Aktionen zu nennen: „pressen“, „Entladeposition_ansteuern“, „Ladeposition_ansteuern“.

v) Das obere Förderband arbeitet im wesentlichen wie das untere Förderband. Die elementaren Aktionen sind „Werkstück_ablegen“, „obere_Lichtschranke_abfragen“, „Transport_oberes_Band“ und „aufnehmen“.

vi) Der Kran fährt vor und zurück, auf und nieder, greift und legt ab. Das Auf- und Niederfahren am oberen Band ist nur von den Transporten dieses Bandes, nicht denen des unteren Bandes abhängig. Für die Formulierung der Abhängigkeitsbeziehungen müßte demnach zwischen den Kranbewegungen am oberen Band und den an sich gleichen Bewegungen am unteren Band unterschieden werden. Diese Differenzierung vergrößert jedoch das Trace-Modell, ohne daß ein wesentlicher Zugewinn an Genauigkeit der Modellierung erzielt wird, denn die Kranbewegungen am oberen Band treten nur zusammen mit dem Greifen und die am unteren Band nur zusammen mit dem Ablegen auf. Von daher bietet es sich an, die Kranbewegungen mit den Aktionen „Greifen“ und „Ablegen“ zu Aktionen „Aufnehmen“ bzw. „Niederlegen“ zusammenzufassen. Es ist klar, daß für eine Verfeinerung des Modells diese beiden Aktionen zerlegt werden können.

A.3 Abhängigkeitsbeziehungen

Für die Modellierung der Abhängigkeitsbeziehungen müssen im Prinzip alle Paare von Aktionen betrachtet werden. Im Hinblick auf den zu modellierenden Prozeß ergeben sich drei Typen von Beziehungen:

- Unabhängigkeit: Aktionen, die in beliebiger Reihenfolge oder gleichzeitig ablaufen können, sind unabhängig. So sind die Drehbewegungen des Roboters unabhängig von den Aktionen der Presse und den Bandtransporten.
- Technische Abhängigkeit: Neben Aktionen, die sich offensichtlich ausschließen (z. B. Kran_vor – Kran_zurück), gibt es andere Aktionen, deren Vertauschung im technischen Ablauf zu Störungen führt. Beispielsweise müssen das Be- und Entladen der Presse abwechselnd erfolgen, weil die Presse nur ein Blech bearbeiten kann.
- Ablaufbedingte Abhängigkeit: Die Aktionen der beiden Roboterarme sind abgesehen vom Be- und Entladen der Presse technisch unabhängig, werden aber aufgrund der Positionierung des Drehtellers, der Presse und des oberen Förderbands durch Drehbewegungen des Roboters getrennt. Im Hinblick auf eine möglichst kleine Anzahl maximaler D-Cliquen werden diese Aktionen als abhängig angesehen.

Der Graph der Abhängigkeitsrelation ist in Abbildung A.2 dargestellt. Die 22 Aktionen gruppieren sich in neun maximalen D-Cliquen.

A.4 Der Ablauf als Trace-Sprache

Für jede der maximalen D-Cliquen kann die Folge von korrekten Abläufen seiner Aktionen angegeben werden.

Ausgangssituation: Wir gehen davon aus, daß

- sich auf dem unteren Band mindestens ein Blech befindet und Bandtransporte notwendig sind, ein Blech in den hinteren Abschnitt zu befördern,
- der Drehteller für die Aufnahme eines Blechs bereit ist,
- der Roboter den ersten Greifarm auf den Drehteller gerichtet und beide Greifarme frei hat,
- die leere Presse sich in Ladeposition befindet,
- im hinteren Abschnitt des oberen Förderbands sich kein Werkstück befindet, und
- der Kran (ohne Werkstück) am oberen Förderband wartet.

Die Aktionenfolgen in den maximalen D-Cliquen:

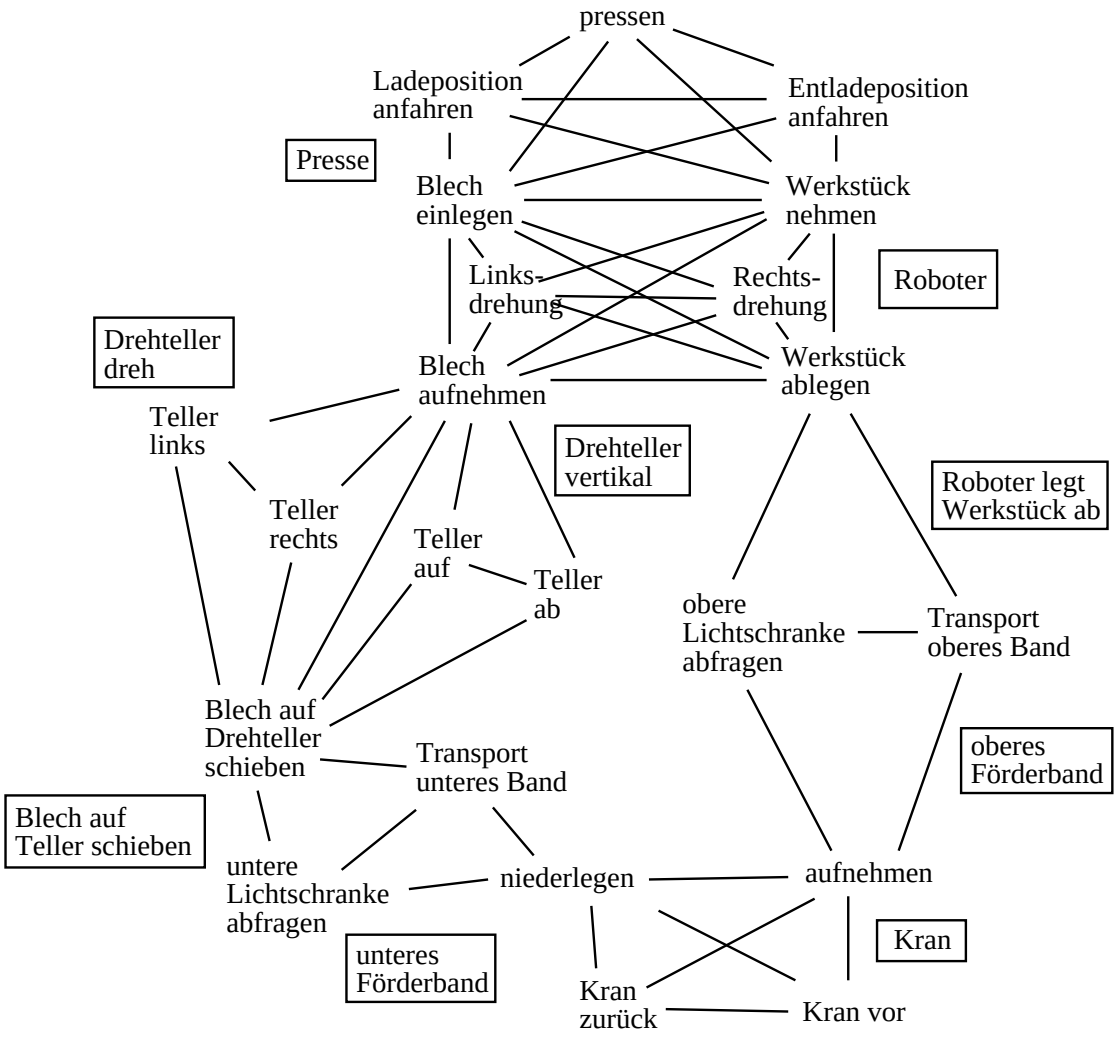

Abbildung A.2: Abhängigkeitsrelation der Aktionen im Modell

i) Drehteller dreh:
(Blech_auf_Drehteller_schieben · Teller_rechts · Blech_aufnehmen · Teller_links)*

ii) Drehteller vertikal:
(Blech_auf_Drehteller_schieben · Teller_auf · Blech_aufnehmen · Teller_ab)*

iii) Roboter:
(Blech_aufnehmen · Linksdrehung · Blech_einlegen · Rechtsdrehung · (Blech_aufnehmen · Linksdrehung · Werkstück_nehmen · Linksdrehung · Werkstück_ablegen · Linksdrehung · Blech_einlegen · Rechtsdrehung)*· Werkstück_nehmen · Linksdrehung · Werkstück_ablegen · Rechtsdrehung)*

iv) Presse:
(Blech_einlegen · pressen · Entladeposition_anfahren · Werkstück_nehmen · Ladeposition_anfahren)*

v) Roboter legt Werkstück ab:
(Werkstück_ablegen · (Transport_oberes_Band · obere_Lichtschranke_abfragen)$^+$)*

vi) oberes Förderband:
((Transport_oberes Band · obere_Lichtschranke abfragen)$^+$ · aufnehmen)*

vii) Kran:
(aufnehmen · Kran_vor · niederlegen · Kran_zurück)*

viii) unteres Förderband:
((Transport_unteres_Band · untere_Lichtschranke_abfragen)$^+$ · niederlegen)*

ix) Blech auf Teller schieben:
((Transport_unteres_Band · untere_Lichtschranke_abfragen)$^+$
· Blech_auf_Drehteller_schieben)*

Das Trace-Verhalten der Fertigungszelle ergibt sich nicht durch freie Synchronisation der Wortsprachen der in den Cliquen i) bis ix) zusammengefaßten Aktionen. Es sind die Sprachen der Cliquen v) und vi) sowie viii) und ix), die nicht frei synchronisiert werden können. Beide Cliquenpaare beschreiben die Steuerung der Aktionen an den Förderbändern durch die Lichtschranken. Bei einer freien Synchronisation der angegebenen Wortsprachen würden sich Aktionenfolgen ergeben, in denen zwischen wiederholten Bandtransporten und Lichtschrankenabfragen nur ein Objekt auf dem Band abgelegt wird (ohne Aufnahme eines Objekts vom Band) oder nur ein Objekt vom Band aufgenommen wird (ohne Ablage eines Objekts auf dem Band). Beide Aktionenfolgen sind nach der Beschreibung der Fertigungszelle nicht zulässig: Zwischen zwei Signalabfragen wird entweder weder ein Objekt vom Band genommen, noch darauf abgelegt, oder es wird sowohl ein Objekt vom Band genommen als auch eins darauf abgelegt. Im zweiten Fall werden die beiden Aktionen unabhängig ausgeführt. Abbildung A.3 zeigt in einem Petrinetz die Abfolge der Aktionen in den Cliquen v) und vi). Die Marken zeigen den Anfangszustand am oberen Förderband. Es wird erst nach Ablage des ersten Werkstücks in Gang gesetzt. Alle weiteren Aktionen „Werkstück_ablegen“ passieren parallel zu Aktionen „aufnehmen“ zwischen Bandtransporten wie oben beschrieben.

Der EAA in Abbildung A.3 ist nicht schwach kooperierend, und es gibt keinen schwach kooperierenden EAA, der die fragliche Trace-Sprache erkennt, wie der Beweis zu Satz 5.5.2 für eine ähnliche Trace-Sprache zeigt. Für die Cliquen viii) und ix) (unteres Förderband) ergibt sich der gleiche EAA, mit dem Unterschied, daß nur die linke Transition (Transport) im Anfangszustand aktiviert ist.

Wir bekommen einen EAA für das Fertigungszellenmodell, indem wir die EAAs für die Cliquen v) und vi) sowie viii) und ix) (Abbildung A.3) mit den fünf endlichen Automaten der Cliquen i) bis iv) und vii) synchronisieren, die jeweils nur eine Aktionenfolge iterieren.

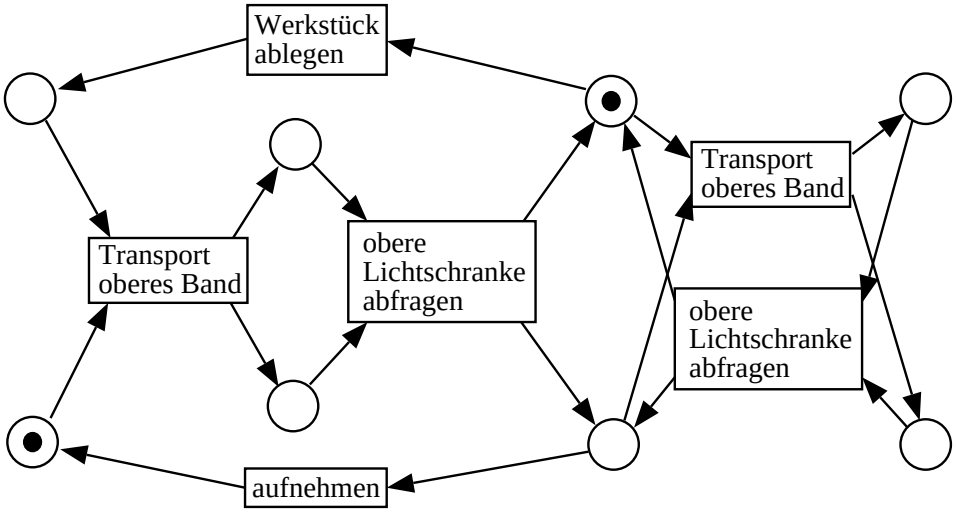

Abbildung A.3: Petrinetz für Aktionen am oberen Förderband

Als Endzustand des EAAs kann zunächst der Anfangszustand genommen werden. Dies entspricht einer vollständigen Bearbeitung aller Werkstücke, die das untere Förderband verlassen. Die einelementige Endzustandsmenge ist lokal definierbar und der beschriebene EAA deterministisch, so daß sein Trace-Verhalten zur Sprachklasse DLE gehört. Werden alle globalen Zustände des EAAs zu Endzuständen erklärt, akzeptiert der EAA alle begonnen korrekten Aktionenfolgen der Fertigungszelle. Das Verhalten ist präfixabgeschlossen und die Endzustandsmenge lokal definierbar, die Trace-Sprache des EAAs also in der Klasse PEQ. Der EAA kann um lokale Fehlerzustände vervollständigt werden, die durch unzulässige Aktionenfolgen erreicht werden und von denen aus kein Endzustand erreichbar ist.

Anhang B

Graphentheoretische Grundbegriffe

Definition B.1 (Gerichteter Graph)

Ein gerichteter Graph *ist ein Paar* (V, E) *bestehend aus einer endlichen nichtleeren Menge* V *von* Knoten *und einer Menge* $E \subseteq V \times V$ *von* Kanten.

Definition B.2 (Weg im gerichteten Graphen)

Sei (V, E) *ein gerichteter Graph. Ein* Weg *von* v_0 *nach* v_n *ist eine Folge* $v_0, \ldots, v_n \in V$ *von Knoten,* $n \geq 1$*, mit* $(v_{i-1}, v_i) \in E$ *für alle* $i \in \{1, \ldots, n\}$.

Definition B.3 (Kreis, Zyklenfreiheit eines gerichteten Graphen)

Sei $G = (V, E)$ *ein gerichteter Graph.*

i) Ein Kreis *in* G *ist ein Weg* $v_0, \ldots, v_n \in V$ *mit* $v_0 = v_n$.

ii) G *ist* zyklenfrei*, wenn es keinen Kreis in* G *gibt.*

Definition B.4 (Zusammenhang eines gerichteten Graphen)

Sei $G = (V, E)$ *ein gerichteter Graph.*

i) Ein ungerichteter Weg *von* v_0 *nach* v_n *ist eine Folge* $v_0, \ldots, v_n \in V$ *von Knoten,* $n \geq 1$*, mit* $(v_{i-1}, v_i) \in E$ *oder* $(v_i, v_{i-1}) \in E$ *für alle* $i \in \{1, \ldots, n\}$.

ii) G *ist* schwach zusammenhängend
$\iff_{Df} \bigwedge_{v_\alpha, v_\omega \in V} v_\alpha \neq v_\omega \Rightarrow$ *Es gibt einen ungerichteten Weg von* v_α *nach* v_ω.

iii) G *ist* stark zusammenhängend
$\iff_{Df} \bigwedge_{v_\alpha, v_\omega \in V} v_\alpha \neq v_\omega \Rightarrow$ *Es gibt einen Weg von* v_α *nach* v_ω.

Definition B.5 (Graph)

Ein (ungerichteter) Graph *ist ein Paar* (V, E) *bestehend aus einer endlichen nichtleeren Menge* V *von* Knoten *und einer Menge von zweielementigen Teilmengen von* V*, der* Kantenmenge E*.*

Anm.: Durch jeden gerichteten Graphen (V, E) mit einer symmetrischen Kantenrelation E wird ein ungerichteter Graph (V, E') definiert mit $E' = \{\{v_1, v_2\} \subseteq V | v_1 \neq v_2 \wedge (v_1, v_2) \in E\}$.

Definition B.6 (Weg im Graphen)

Sei (V, E) *ein Graph.*
Ein Weg *von* v_0 *nach* v_n *ist eine Folge* $v_0, \ldots, v_n \in V$ *von Knoten,* $n \geq 1$*, mit* $\{v_{i-1}, v_i\} \in E$ *für alle* $i \in \{1, \ldots, n\}$*.*

Definition B.7 (Zusammenhang eines Graphen)

Sei Der Graph $G = (V, E)$ *ist* zusammenhängend
$\iff_{Df} \bigwedge_{v_\alpha, v_\omega \in V} v_\alpha \neq v_\omega \Rightarrow$ *Es gibt einen Weg von* v_α *nach* v_ω*.*

Definition B.8 (Clique in einem Graphen)

Sei $G = (V, E)$ *ein Graph.*

i) Die Knotenteilmenge $V' \subseteq V$ *ist eine* Clique *in* G
$\iff_{Df} \bigwedge_{v_1, v_2 \in V'} v_1 \neq v_2 \Rightarrow \{v_1, v_2\} \in E$*.*

ii) Die Knotenteilmenge $V' \subseteq V$ *ist eine* maximale Clique *in* G*, wenn für jede Clique* V'' *mit* $V' \subseteq V''$ *gilt:* $V' = V''$*.*

Literaturverzeichnis

[AH87] Aalbersberg, I. J.; Hoogeboom, H. J.: *Decision Problems for Regular Trace Languages,* LNCS 267(1987), 250–259

[AR88] Aalbersberg, I. J.; Rozenberg, G.: *Theory of Traces,* TCS 60(1988), 1–82

[AW86] Aalbersberg, I. J.; Welzl, E.: *Trace Languages Defined by Regular String Languages,* Theoretical Informatics and Applications 20(1986), 103–119

[BMS82] Bertoni, A.; Mauri, G.; Sabadini, N.: *Equivalence and Membership Problems for Regular Trace Languages,* LNCS 140(1982), 61–71

[BPS88] Bruschi, D.; Pighizzini, D.; Sabadini, N.: *On the Existence of the Minimum Asynchronous Automaton and on Decision Problems for Unambiguous Regular Trace Languages,* LNCS 294(1988), 334–345

[BPS94] Bruschi, D.; Pighizzini, D.; Sabadini, N.: *On the Existence of Minimum Asynchronous Automata and on the Equivalence Problem for Unambiguous Regular Trace Languages,* Information and Computation 108(1994), 262–285

[Brau84] Brauer, W.: *Automatentheorie,* Teubner–Verlag, Stuttgart 1984

[BS86] Berry, G.; Sethi, R.: *From Regular Expressions to Deterministic Automata,* TCS 48(1986), 117–126

[CF69] Cartier, P.; Foata, D.: *Problèmes combinatoires de commutation et réarrangements,* Lecture Notes in Mathematics 85, Springer–Verlag, Berlin Heidelberg 1969

[CLRZ92] Clerbout, M.; Latteux, M.; Roos, Y.; Zielonka, W.: *Semi-Commutations and Rational Expressions,* LNCS 623(1992), 113–125

[CM88] Cori, R.; Métivier, Y.: *Approximation of a Trace, Asynchronous Automata and the Ordering of Events in Distributed Systems,* LNCS 317(1988), 147–161

[CMZ93] Cori, R.; Métivier, Y.; Zielonka, W.: *Asynchronous Mappings and Asynchronous Cellular Automata,* Information and Computation 106(1993), 159–202

[CR89] Czernik, S.; Reineke, H.: *Äquivalenz von Prozeßbegriffen für verteilte Systeme,* Diplomarbeit im Studiengang Informatik der Uni Bremen, 1989

[Diek90] Diekert, V.: *Combinatorics on Traces,* LNCS 454(1990)

[Diek94] Diekert, V.: *A partial trace semantics for Petri nets,* TCS134(1994), 87–105

[DM95] Diekert, V.; Muscholl, A.: *The Construction of Asynchronous Automata for Triangulated Graphs,* erscheint in Fundamenta Informaticae 1995

[DH94] Drusinsky, D.; Harel, D.: *On the Power of Bounded Concurrency I: Finite Automata,* Journal of the Association for Computing Machinery 41(1994), 517–539

[Dub86] Duboc, Chr.: *Mixed Product and Asynchronous Automata,* TCS 48(1986), 183–199

[EHR94] Ehrenfeucht, A.; Hoogeboom, H. J.; Rozenberg, G.: *Combinatorial Properties of Dependence Graphs,* Information and Computation 114(1994), 315–328

[FG65] Fulkerson, D. R.; Gross, O. A.: *Incidence Matrices and Interval Graphs,* Pacific Journal of Mathematics 15(1965), 835–855

[Gast91] Gastin, P.: *Recognizable and Rational Trace Languages of Finite and Infinite Traces,* LNCS 480(1991), 89–104

[GOPR92] Gastin, P.; Ochmański, E.; Petit, A.; Rozoy, B.: *Decidability of the Star Problem in* $A^* \times \{b\}^*$, Information Processing Letters 44(1992), 65–71

[GP92] Gastin, P.; Petit, A.: *Asynchronous Cellular Automata for Infinite Traces,* LNCS 623(1992), 583–593

[Ginz67] Ginzburg, A.: *A Procedure for Checking Equality of Regular Expressions,* Journal of the ACM 14(1967), 355–362

[Gol80] Golumbic, M. Ch.: *Algorithmic Graph Theory and Perfect Graphs,* Academic Press, New York 1980

[Grau88] Graubmann, Peter: *The Construction of EN-Systems from a given Trace Behaviour,* LNCS 340(1988), 133–153

[HKT92] Hoogers, P. W.; Kleijn, H. C. M.; Thiagarajan, P. S.: *A Trace Semantics for Petri Nets,* LNCS 623(1992), 595–604

[Hoa85] Hoare, C. A. R.: *Communicating Sequential Processes,* Prentice Hall International Series in Computer Science, Englewood Cliffs 1985

[Iba78] Ibarra, O. H.: *Reversal-Bounded Multicounter Machines and Their Decision Problems,* Journal of the ACM 25(1978), 116–133

[Jans94] Janssen, W.: *Layered Design of Parallel Systems,* Dissertation Universität Twente 1994

[Jan87] Jantzen, M.: *Language Theory of Petri Nets,* LNCS 254(1987), 397–412

[KMS94] Klarlund, N.; Mukund, M.; Sohoni, M.: *Determinizing Asynchronous Automata,* LNCS 820(1994), 130–141

[LL93] Lewerentz, C.; Lindner, Th.(Hrsg.): *Case Study "Production Cell" – A Comparative Study in Formal Software Development,* FZI-Publication 1/94, Forschungszentrum Informatik, Karlsruhe 1993

[Maz77] Mazurkiewicz, A.: *Concurrent Program Schemes and Their Interpretations,* DAIMI–PB–78, Universität Aarhus 1977

[Maz87] Mazurkiewicz, A.: *Trace Theory,* LNCS 255(1987), 279–324

[Mét87] Métivier, Y.: *An Algorithm for Computing Asynchronous Automata in the Case of Acyclic Non-Commutation Graphs,* LNCS 267(1987), 226–237

[Mil80] Milner, R.: *Calculus of Communicating Processes,* LNCS 92(1980)

[MM65] Moon, J. W.; Moser, L.: *On Cliques in Graphs,* Israel Journal of Mathematics 3(1965), 23–28

[Mus94] Muscholl, A.: *On the complementation of Büchi asynchronous cellular automata,* LNCS 820(1994), 142–153

[Och85] Ochmański, E.: *Regular Behaviour of Concurrent Systems,* Bulletin of the EATCS 27(1985), 56–67

[Och88] Ochmański, E.: *On Morphisms of Trace Monoids,* LNCS 294(1988), 346–355

[Pet93] Petit, A.: *Recognizable Trace Languages, Distributed Automata and The Distribution Problem,* Acta Informatica 30(1993), 89–102

[Petri62] Petri, C. A.: *Kommunikation mit Automaten,* Schriften des Instituts für Instrumentelle Mathematik an der Universität Bonn, Nr. 2, 1962

[Pig93] Pighizzini, G.: *Synthesis of Nondeterministic Asynchronous Automata,* in: Droste, M.; Gurevich, Y. (Eds.): *Semantics of Programming Languages and Model Theory,* Gordon and Breach Science Publ. 1993

[Pig94] Pighizzini, G.: *Asynchronous Automata Versus Asynchronous Cellular Automata,* TCS 132(1994), 179–207

[Rei82] Reisig, W.: *Petrinetze—Eine Einführung,* Springer–Verlag, Berlin Heidelberg 1982

[Sak87] Sakarovitch, J.: *On Regular Trace Languages,* TCS 52(1987), 59–75

[Sak92] Sakarovitch, J.: *The "Last" Decision Problem for Rational Trace Languages,* LNCS 583(1992), 460–473

[Star78] Starke, P. H.: *Free Petri Net Languages,* LNCS 64(1978), 506–515

[Star90] Starke, P. H.: *Analyse von Petri-Netz-Modellen,* Teubner–Verlag, Stuttgart 1990

[Ziel87] Zielonka, W.: *Notes on Finite Asynchronous Automata,* Theoretical Informatics and Applications 21(1987), 99–135

[Ziel89] Zielonka, W.: *Safe Executions of Recognizable Trace Languages by Asynchronous Automata,* LNCS 363(1989), 278–289

Lebenslauf:

Ich, Henning Reineke, wurde am 27. November 1962 als zweites Kind der Eheleute Heinrich und Anni Reineke in Bremen geboren. Mein Vater war Haftrichter von Beruf. Er starb 1989. Meine Mutter ist Hausfrau.

Nach dem Besuch der Grundschule von 1969 bis 1973 wechselte ich auf das Kippenberg-Gymnasium. Dort legte ich 1982 das Abitur ab.

Im Anschluß an den fünfzehnmonatigen Wehrdienst nahm ich im Wintersemester 83/84 an der Universität Bremen das Informatikstudium auf. Das Studium schloß ich im Dezember 1989 mit dem Diplom ab. Die Diplomarbeit „Äquivalenz von Prozeßbegriffen für verteilte Systeme“ (geschrieben zusammen mit S. Czernik) befaßte sich mit einem Thema aus der Petrinetztheorie. Die Arbeit wurde von Prof. K. Döpp betreut.

Seit Juli 1990 bin ich an der Carl-von-Ossietzky-Universität im Fachbereich Informatik beschäftigt.