Carl von Ossietzky Universität Oldenburg
Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Monitoring of Traffic Manoeuvres with Imprecise Information

Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften

vorgelegt von

Heinrich Ody

Gutachter: Prof. Dr. Ernst-Rüdiger Olderog
Prof. Dr. Michael Reichhardt Hansen

Datum der Einreichung: 4. November 2019
Datum der Verteidigung: 27. März 2020

Abstract

There is a trend in developing more and more autonomous driver assistance systems. For these systems it is difficult to be certain that they behave as desired because of their wide range of responsibilities. Here, automation of formal spatio-temporal reasoning about traffic manoeuvres can help to analyse highly autonomous driver assistance systems.

Multi-Lane Spatial Logic is a logic tailored towards formal reasoning about spatial aspects of motorways. For this logic we investigate means of automation, while paying special attention to the nature of imprecise information when working with sensor readings. We show the infeasibility of using the full power of this logic in automated reasoning, even when we restrict the power of the logic by assuming imprecise information. To achieve automation, we define an extension of this logic for which automation of a relevant fragment is feasible. We connect our extension to timed words and show how we can use our extension to reason automatically about single traffic manoeuvres under consideration of imprecise information. That is, we show how we can use Multi-Lane Spatial Logic to perform *monitoring of traffic manoeuvres with imprecise information.*

However, often a Boolean yes/no analysis of systems is not sufficient. To tackle this, we define an extension of the temporal logic Duration Calculus that facilitates a quantification of how well a temporal property holds, i.e. it allows for "beyond yes/no" analysis of real-time systems. We connect this to our work on traffic manoeuvres by using the temporal logic to analyse a hazard warning protocol for motorways.

Zusammenfassung

Fahrerassistenzsysteme werden immer autonomer und damit auch immer komplexer. Dementsprechend wird es schwieriger sicher zu sein, dass ein autonomes Fahrerassistenzsystem wie erwartet funktioniert. Hier kann automatisiertes logisches Schließen für Verkehrsmanöver helfen, um hochautomatisierte Fahrerassistenzsysteme zu analysieren.

Multi-Lane Spatial Logic ist eine räumliche Logik, welche auf die Analyse von Verkehrsmanövern spezialisiert ist. Für diese Logik untersuchen wir Möglichkeiten der Automatisierung unter Berücksichtigung von ungenauen Sensordaten. Wir zeigen, dass im Allgemeinen automatisiertes logisches Schließen mit dieser Logik nicht möglich ist. Dies gilt, selbst wenn wir die Ausdrucksfähigkeit der Logik einschränken, indem wir annehmen, dass Sensordaten ungenau sind. Um Automatisierung zu ermöglichen, definieren wir eine Erweiterung der Logik und zeigen, dass für ein relevantes Fragment unserer Erweiterung Automatisierung möglich ist. Wir verbinden unsere Erweiterung mit Realzeitwörtern und verwenden die entstehende Logik, um einzelne Verkehrsmanöver, unter Berücksichtigung ungenauer Sensordaten, automatisiert zu analysieren.

Jedoch reichen boolesche ja/nein-Aussagen zur Systemanalyse häufig nicht aus. Deshalb definieren wir eine Erweiterung der temporalen Logik Duration Calculus, mit der man ausdrücken kann, zu welchem Grad eine temporale Eigenschaft erfüllt ist. Wir verbinden dies mit unserer Arbeit an Verkehrsmanövern, indem wir unsere Erweiterung verwenden, um ein Gefahrenwarnprotokoll für Autobahnen zu analysieren.

Acknowledgements

First of all, I thank my advisor Ernst-Rüdiger Olderog for trusting me to let me work on my thesis independently, for giving me guidance when I needed it and for taking the time for in-depth discussions. Your careful and deliberate approach at work has been an example to me.

Further, I thank Martin Fränzle, Michael Reichhardt Hansen and Astrid Rakow for being members of my board of examiners and for seeing my disputation through despite the corona chaos in Spring 2020. I am especially grateful to you Martin: Without your efforts on clarifying the legal aspects of a virtual disputation my defence could not have taken place. I also thank you for inviting me to work with you and Michael on the paper for Ernst-Rüdiger Olderogs Festschrift. I fondly remember our presentation, where Michael hid behind the speaker's desk. Furthermore, thanks to this collaboration I could visit Michael in Denmark for an extended research visit, for which I am very grateful to you Michael.

I thank you Annegret Habel for having me as your teaching assistant for Theoretische Informatik II. I learned a lot from being your teaching assistant.

I thank Manuel Gieseking, Martin Hilscher, Sven Linker and Maike Schwammberger for the many discussions and the proof reading. For the discussions I especially want to thank you Manuel. I always had the feeling that I could discuss every topic with you and that I learned a lot from our discussions.

For the time spend together and the interesting discussions I thank Christopher Bischopink, Evgeny Erofeev, Nils Erik Flick, Thomas Hujsa, Christoph Peuser, Mani Swaminathan and Nick Würdemann. I like to remember Columbia, Dagstuhl, the poker rounds and the shared evenings.

I am grateful to Marion Bramkamp, Nicolai Degen, Andrea Göken, Mark Kettner, Patrick Uven and Ira Wempe for their help with administrative and technical issues. I am especially grateful to you Ira, for the support with business trips. For example, I remember when I made a mistake with the bills for the reimbursement and you smoothly resolved the problem.

In general, I thank all associates of the theoretical computer science department for the nice and open lunch breaks.

Furthermore, I thank Bayram Kiran and Krishna Narasimhan for our many trips together. I love that I can discuss everything with you.

I thank Sören Dierkes, Renke Grunwald and Vincent Hamann for our many board game evenings.

Lastly, and most importantly I thank my long time girlfriend Ludmila Tsesarska. You have always believed in me, even when I had doubts about the feasibility of my work. Without you I could not have completed this dissertation.

Contents

1 Introduction

Recently many companies started to test their autonomously driving cars on the road [GAA+12]. So far, a human always has to be ready to intercept, should problems arise [WHL+15]. One stimulus that caused this surge of interest in autonomously driving cars was the DARPA Urban Challenge in 2007. One goal of developing autonomously driving cars is to make the roads safer and reduce the number of traffic deaths, which was 1.35 million people in 2018 [WHO18]. Other goals include increasing road efficiency, increasing passenger comfort and cost reduction [MGL+15].

However, to be usable the safety of the driving system, consisting of hardware and software, needs to be certified. An autonomously driving car needs to perceive its environment, identify the relevant traffic participants and their intentions, decide a course of action that leads to the achievement of its mission while obeying local traffic laws and ensuring its own safety and the safety of other traffic participants [WHL+15]. Adding to that, autonomously driving cars combine digital computation and control with physical movement, which makes them cyber-physical systems [Pla18]. Furthermore, the environment of the car consists of other traffic participants, which are constantly leaving and entering the vicinity of our car. Thus, we have a dynamically changing system of systems. As controllers of autonomously driving cars evidently are highly complex and need to satisfy a wide range of constraints, some of which are safety critical, formal methods should be used to analyse autonomously driving cars.

Multi-Lane Spatial Logic (MLSL) is a logic tailored towards spatio-temporal reasoning of autonomously driving cars in an abstract manner [HLO+11; LH15; Lin15]. At first, the logic was restricted to reason about cars on a motorway. However, this has been extended to reasoning about cars on streets with bidirectional traffic [HLO13] and to urban traffic [Sch18a]. A typical spatio-temporal property easily expressible with MLSL is

$$\text{“there are never two cars occupying overlapping space”}\ . \tag{1.1}$$

MLSL has been used to manually prove the safety of controllers given as automata [HLO+11; Sch18a], or as MLSL formulas [Lin15].

However, often it is not sufficient to simply check if a system, such as an autonomously driving car, satisfies a property. Instead, we want to quantitatively compare systems. For temporal properties there has been a lot of work on quantifying how *robustly* a system satisfies a property [DM10; FH05; SFK08; FP09]. A system satisfies a property robustly if small perturbations do not affect the satisfaction. This robust satisfaction of properties allows us to transfer properties shown in the formalism to properties of the physical world. But this does not give the means to quantitatively analyse how early a desired event occurs. This has been termed as *temporal quality* [ABK14; dHM03; dFH+05]. For an example consider the property

$$\text{“soon all cars are informed of the hazard on the motorway”}\ . \tag{1.2}$$

In economics, *discounting* represents that money earned earlier is worth more than money earned later. To formalise reasoning about the temporal quality of systems, discounting has been introduced into discrete time temporal logics [ABK14; dHM03; dFH+05]. The idea is to let formulas evaluate to a real-valued truth value from the interval $[0, 1]$. Then, the longer we have to wait for a desired event, the closer the resulting truth value will be to 0.

Contributions

In this thesis we are mainly interested in whether automation of specific problems is possible for our formalism of interest and how we can tweak the problem to make it possible. We outline the contributions of this thesis. We

1. show that a variation of the classical satisfiability problem for MLSL properties is undecidable. For this we reduce the language emptiness problem of the intersection of two context-free languages to the adapted satisfiability problem of MLSL.

2. then extend the previous result to show that the undecidability of MLSL is not an artefact of infinite precision. That is, we show that even with *imprecise information* the variation of the satisfiability problem remains undecidable.

3. define an extension of MLSL and find a fragment where the original satisfiability problem and our modified satisfiability problem are decidable. We define an algorithm to check if a given model satisfies a given formula

of our extension of MLSL. To prove this algorithm correct we define operations on MLSL models to combine and restrict models.

4. extend MLSL to be able to express spatio-temporal properties about single *traffic manoeuvres*, rather than all behaviours of a controller. For temporal logics this is called *monitoring* [MN04]. Considering single behaviours instead of the complete behaviour of a controller is useful, e.g. to reduce the complexity of the problem to facilitate automation. We develop an algorithm to automate monitoring for MLSL. To prove correctness of this algorithm we extend our aforementioned operations on MLSL models to this case.

5. extend the algorithm of Item 4 to consider imprecise spatio-temporal information in traffic manoeuvres. That is, we formalise what it means for the Equation (1.1) to hold robustly.

6. extend the kind of temporal properties expressible for autonomously driving cars. We introduce discounting into a variation of the dense time temporal logic Duration Calculus [ZHR91]. In our logic we can formalise properties such as the one in Equation (1.2) for dense time systems. While model checking for Duration Calculus usually is undecidable [ZHS93], it becomes decidable for a relevant fragment of our logic, where we use timed automata [AD94] as models.

Structure of this Thesis

In Table 1.1 we show the structure of this thesis. The table shows the general setting of each chapter and on which of our peer-reviewed publications the chapter is based on. Note that we present the preliminaries in Chapter 2 and the conclusion in Chapter 8. Further, we provide related work at the end of each chapter.

Table 1.1: Overview over the assumptions on data, the properties considered in each chapter, and where the basis of the chapters is published. With a Boolean semantics properties are either satisfied, or not. A multi-valued semantics quantifies how strongly a property is satisfied.

chapter	data	property	semantics	based on
3	precise	spatial	Boolean	[Ody15b; FHO15]
4	imprecise	spatial	Boolean	[Ody15b]
5	precise	spatio-temporal	Boolean	[Ody17]
6	imprecise	spatio-temporal	Boolean	[Ody17]
7	precise	temporal	multi-valued	[OFH16]

2 Preliminaries

In this chapter we introduce the concepts this thesis is based on. In Section 2.1 we introduce some mathematical notation and structures, in Section 2.2 we define a variation of timed automata, in Section 2.3 we introduce Multi-Lane Spatial Logic and in Section 2.4 we present some arithmetic first-order theories.

2.1 Basic Concepts

We use $\equiv$ to denote *syntactic equality*. Commonly we use $\equiv$ to define syntactic abbreviations of formulas, i.e. in propositional logic for atomic propositions Q, P we write $\psi \equiv P \land Q$ to define an abbreviation for $P \land Q$. In contrast to this we use $=$ to express semantic equality, i.e. in arithmetic $3 = 2 + 1$ holds as $2 + 1$ evaluates to 3, while $3 \equiv 2 + 1$ does not hold as the term 3 and the term $2 + 1$ are syntactically different.

Often we use multi-letter names to abbreviate formulas, e.g. the formula step. We write these abbreviations in sans serif font, to make clear that they are mathematical symbols. Variables are written in slanted italics, e.g. x and i. Also we always write numbers in roman upright font and annotations that are themselves not variables or numbers in sans serif, for example D.

For a set S we denote the *powerset* of S with $\mathcal{P}(S)$. For a finite set S we denote its *size*, or the number of elements S contains, with $|S|$. Further, for two sets S, S' we use common operations on sets such as the union $(S \cup S')$, intersection $(S \cap S')$, difference $(S \setminus S')$ and complementation $(\overline{S})$ with their usual meaning.

With $\mathbb{N}$ we denote the set of natural numbers including 0. With $\mathbb{R}$ we denote the real numbers and with $\mathbb{Q}$ the rational numbers. With $\mathbb{N}_{\geq 1}$ we denote the natural numbers greater or equal than 1, and similarly for other notations like $\mathbb{R}_{>0}$.

For a tuple $(x_1, \dots, x_n) \in X_1 \times \dots \times X_n$ we use $\mathsf{set}(x_1, \dots, x_n)$ to refer to the set $\{x_1, \dots, x_n\}$.

2.1.1 Z Notation

We use several operators as defined in Z Notation [Spi92]. For a tuple (x, y) with arbitrary x, y let $\mathsf{first}\ (x, y) = x$ and $\mathsf{second}\ (x, y) = y$. For the following definitions let X, Y, S be arbitrary sets and $R \subseteq X \times Y$. Then the domain of R (denoted $\mathsf{dom}\, R$) is the subset of X that is related through R to Y, i.e. $\mathsf{dom}\, R = \{x \mid x \in X \wedge \exists y \in Y.\, (x, y) \in R\}$ or equivalently $\mathsf{dom}\, R = \{\mathsf{first}\ (x, y) \mid (x, y) \in R\}$. Similarly, the range of R (denoted $\mathsf{ran}\, R$) is the set of all elements from Y that are related to X through R, i.e. $\mathsf{ran}\, R = \{y \mid y \in Y \wedge \exists x \in X.\, (x, y) \in R\}$ or equivalently $\mathsf{ran}\, R = \{\mathsf{second}\ (x, y) \mid (x, y) \in R\}$.

We define *domain restriction* and *range restriction* of a relation, together with the corresponding *anti-restriction*, as

$$\begin{aligned}
S \lhd R &= \{(x, y) \mid (x, y) \in R \wedge x \in S\} \ , \\
R \rhd S &= \{(x, y) \mid (x, y) \in R \wedge y \in S\} \ , \\
S \mathbin{⩤} R &= (X \setminus S) \lhd R \ , \\
R \mathbin{⩥} S &= R \rhd (Y \setminus S) \ .
\end{aligned}$$

The *inverse* of a relation R (denoted with $R^{\sim}$) is defined as $R^{\sim} = \{(y, x) \mid (x, y) \in R\}$. We use the relational image $R(\!| S |\!) = \mathsf{ran}(S \lhd R)$. For relations Q and R we use the override operation $Q \oplus R = ((\mathsf{dom}\, R) \mathbin{⩤} Q) \cup R$. That means, $Q \oplus R$ relates like R and additionally, relates everything not in the domain of R like Q. For two relations $R \subseteq X \times Y, Q \subseteq Y \times Z$ we define the *composition* of R and Q (denoted $Q \circ R$) as $Q \circ R = \{(x, z) \mid x \in X \wedge z \in Z \wedge \exists y \in Y.\, ((x, y) \in R \wedge (y, z) \in Q)\}$.

For two sets S_1, S_2 we write $S_1 \uplus S_2$ to consider the union of the two sets while stating at the same time that $S_1 \cap S_2 = \emptyset$ holds. For tuples of relations $t = (R_1, \ldots, R_n)$ and $t' = R'_1, \ldots, R'_n$ we define $t \uplus t' = (R_1 \uplus R'_1, \ldots, R_n \uplus R'_n)$. Further, we define $t \lhd S = (R_1 \lhd S, \ldots, R_n \lhd S)$. Similarly, we use $⩤, \cap$ on tuples of relations.

We shall use operations on relations, e.g. domain restriction and disjoint union, also on functions.

A *sequence* is a partial function $s : \mathbb{N}_{\geq 1} \to S$ for some set S. The domain of a sequence is a list of consecutive natural numbers starting with 1, i.e. $\mathsf{dom}\, s = \{1, \ldots, k\}$ for some $k \in \mathbb{N}$. We denote for example the sequence of numbers from 1 to 10 with $\langle 1, 2, 3, \ldots, 10\rangle$ and we denote the empty sequence with $\langle\rangle$. Let s be a sequence. With $\#s$ we denote the length of the sequence, i.e. $\#s = |\,\mathsf{dom}\, s|$. Furthermore, for sequences s_1, s_2 and an element x we define prepending and appending of elements, and *concatenation* of sequences as

$$\begin{aligned} x :: s_1 &= \{1 \mapsto x\} \uplus \{i+1 \mapsto s_1(i) \mid i \in \mathsf{dom}\, s_1\} \ , \\ s_1 :: x &= s_1 \uplus \{1 + \#s_1 \mapsto x\} \ , \\ s_1 \cdot s_2 &= s_1 \uplus \{i + \#s_1 \mapsto s_2(i) \mid i \in \mathsf{dom}\, s_2\} \ . \end{aligned}$$

Let s be a nonempty sequence. We define the first and last element of s with $\mathsf{head}\, s = s(1)$ and $\mathsf{last}\, s = s(\#s)$. The sequences without its first (resp. last) element are defined as $\mathsf{front}\, s = \{\#s\} \lhd s$ and $\mathsf{tail}\, s = \{i \mapsto s(i+1) \mid i \in \{1, \ldots, \#s - 1\}\}$. Additionally, for $i \in \mathbb{N}$ and a sequence s let $s[..i]$ be the prefix of s of length $\mathsf{min}(i, \#s)$ and for $j \in \mathbb{N} \setminus \{0\}$ let $s[j..]$ be the suffix of s starting at j, i.e. the suffix of length $\mathsf{max}(0, \#s - i + 1)$. We point out that for all sequences s we have $s[1..] = s$, $s[\#s + 1..] = \langle\rangle$, $s[..0] = \langle\rangle$ and $s[..\#s + 1] = s$. For two sequences s, s' we say that s is a *subsequence* of s' iff we can generate s from s' by removing elements from s'. Formally, s is a subsequence of s' iff there is a set $N \subseteq \mathbb{N}$ such that $N \lhd s' = s$. For some set S we denote with $\mathsf{Seq}\, S$ the set of all sequences over S and with $\mathsf{Seq}_{\geq 1}\, S$ the set of all nonempty sequences over S. Finally, we lift the operations on sequences to operations on tuples of sequences, where we assume that the sequences in such a tuple all have equal length.

2.1.2 Words and Languages

For an alphabet Σ we call a set $L \subseteq \Sigma^*$ a *language*, where $*$ is the *Kleene star*. Let L_1, L_2 be languages. We use operations common in formal languages like concatenation (denoted $L_1 \cdot L_2$), union (denoted $L_1 \cup L_2$), intersection (denoted $L_1 \cap L_2$) and complementation (denoted $\overline{L_1}$). We call an element of a language a *word*. Words and sequences are very similar in that we join objects from a set in a specific order. Hence, we sometimes use notations defined for sequences also with words. As an example, as in Z notation we denote the length of a word w with $\#w$. For $u, w \in \Sigma^*$ we define the *prefix* relation as $u \sqsubseteq v = \exists w \in \Sigma^*.\, uw = v$. Then $u \sqsubset v$ iff $u \sqsubseteq v$ and $u \neq v$. For an alphabet Σ equipped with a total order $\prec$ the *lexicographic order* $\prec_{\mathsf{l}}$ is defined as

$$\begin{aligned} u \prec_{\mathsf{l}} v \ \text{ iff } (u \sqsubset v \ \text{ or } \ (u[..i-1] = v[..i-1] \ \text{ and } \ u(i) \prec v(i)) \ , \\ \text{for some } i \in \{1, \ldots, \mathsf{min}(\#u, \#v)\} \ . \end{aligned}$$

This means that u is lexicographically smaller than v iff u is a proper prefix of v, or at the first position where u and v differ u is smaller than v.

In this thesis we consider timed words, where multiple events can occur simultaneously. Hence, we consider a timed word to be a time-stamped sequence of sets of events. Throughout this work we shall use dense time. We introduce $\mathbb{T}$ as the temporal domain, which is equal to the nonnegative real numbers.

Definition 2.1.1 (Timed Words)**.** Let Σ be an alphabet. Then a *timed word* is a structure $\varrho \in \mathsf{Seq}_{\geq 1}(\mathcal{P}(\Sigma) \times \mathbb{T})$.[1] Often we write $\varrho = (w, \tau)$ where $w \in \mathcal{P}(\Sigma)^+$ and τ is a strictly monotonically increasing sequence of time stamps over $\mathbb{T}$. Also we write $\varrho = \langle (A_1, t_1), \ldots, (A_n, t_n) \rangle$ with $A_1, \ldots, A_n \in \mathcal{P}(\Sigma)$ and $t_1, \ldots, t_n \in \mathbb{T}$. The *span* of a timed word is defined as $\mathsf{span} = [0, \mathsf{last}\,\tau]$. We use the abbreviation $(\mathcal{P}(\Sigma) \times \mathbb{T})^+$ to denote $\mathsf{Seq}_{\geq 1}(\mathcal{P}(\Sigma) \times \mathbb{T})$. △

Note that we sometimes use $\mathsf{Seq}_{\geq 1}(\mathcal{P}(\Sigma) \times \mathbb{T})$ and the usual $(\mathcal{P}(\Sigma) \times \mathbb{T})^+$ interchangeably. It is clear that these two definitions are equivalent. Further, note that we disallow the empty timed word because it complicates some definitions.

We define concatenation of timed words. Note that concatenation of timed words is only defined if all time stamps of the trailing timed word are greater than all time stamps of the leading timed word. In effect this is ensured if the first time stamp of the trailing timed word is greater than the last time stamp of the leading timed word. In general we do not consider the empty timed word. However, sometimes it is beneficial to allow the concatenation of a timed word with the empty timed word, which we define yields the timed word.

Definition 2.1.2 (Concatenation for Timed Words)**.** Let ϱ, ϱ' be two timed words with $\mathsf{last}\,\tau < \tau'(1)$. Then we define the *concatenation* $\varrho \cdot \varrho'$ as the concatenation of their sequences. We extend this to allow the empty sequence by defining $\varrho \cdot \langle\rangle = \langle\rangle \cdot \varrho = \varrho$. △

2.1.3 Trees

Here we like to consider trees as sets of words. A binary *tree* is a nonempty prefix closed set $\Upsilon \subseteq \{0,1\}^*$. A node is defined as the path to the node, which is a word over $\{0,1\}$. A *labelled tree* is a tuple (Υ, f), where $f : \{0,1\}^* \to \Sigma$ is the labelling function and Σ is some set of labels. We lift the labelling function f to sequences of nodes.

[1]The symbol ϱ is a variant of ρ (rho). We use the variant to set it apart from the Latin letter p.

For $x \in \{0,1\}^*$ let $x.\mathsf{children}$ be the sequence of children of x such that $x.\mathsf{children}(v) = xv$ with $v \in \{0,1\}$. For a tree Υ the *depth* of a node $x \in \Upsilon$ is its distance from the root, i.e. $\text{depth}(\Upsilon) = \#x$. Usually the height of a node is defined as the length of the longest downward path from that node and the height of the tree is the height of its root. As we do not need the height of a node we simplify this to the following: the *height* of a tree is the maximal depth of all its nodes, i.e. $\mathsf{height}(\Upsilon) = \mathsf{max}_{x\in\Upsilon}(\text{depth}(x))$. Additionally, let xy, xz be nodes, then $\mathsf{dis}(xy, xz) = \#y + \#z$, i.e. the *distance* of two nodes is the length of the shortest path from one node to the other. For $x \in \Upsilon$ we call x an *internal node* if x has a child. Furthermore, for two nodes x, y with $x \sqsubseteq y$ we say that x is an *ancestor* of y and that y is a *descendant* of x. If two different nodes have the same parent we call them *siblings*. We use $\mathsf{T}(\Sigma)$ for the set of all Σ-labelled trees. See Figure 2.1 for an example of (labelled) binary trees.

2.1.4 Context-Free Languages

A context-free grammar consists of nonterminals, terminals and rewrite rules. The rewrite rules define how we can replace single nonterminals by words containing nonterminals and terminals.

Definition 2.1.3 (Context-Free Grammar)**.** A *context-free grammar* (CFG) is a tuple $G = (\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$, where $\mathcal{N}$ is the set of *nonterminals*, $\mathcal{T}$ with $\mathcal{T} \cap \mathcal{N} = \emptyset$ is the set of *terminals*, $S \in \mathcal{N}$ is the *starting* nonterminal and $\mathcal{R} \subseteq \mathcal{N} \times (\mathcal{T} \cup \mathcal{N})^*$ is the set of *rewrite rules*. A rewrite rule $(N, w) \in \mathcal{R}$ is usually written as $N \to w$. We extend $\to$ as follows. Let $\mathcal{R}^N = \{w \mid (N, w) \in \mathcal{R}\}$, i.e. the set of words that may replace N. Let $u, v, w \in (\mathcal{T} \cup \mathcal{N})^*, N \in \mathcal{N}$ then $uNv \to uwv$ with $(N, w) \in \mathcal{R}$ is a *rewrite step*. A sequence of rewrite steps is a *derivation*. The *language* $\mathcal{L}(G)$ of a CFG G is the set of terminal words reachable with a derivation that starts with S. We refer to $\sigma \in \mathcal{N} \cup \mathcal{T}$ as *letter*. △

All grammars we consider are in Chomsky normal form. This means that a nonterminal always is replaced by either two nonterminals or a terminal or the empty word.

Definition 2.1.4 (Chomsky Normal Form)**.** A CFG $G = (\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$ is in *Chomsky normal form* (CNF) iff all rewrite rules have the form $N_0 \to \tau$, $N_0 \to \epsilon$ or $N_0 \to N_1 N_2$, where $N_0, N_1, N_2 \in \mathcal{N}, \tau \in \mathcal{T}$. △

We call a rewrite rule an *ϵ-rule* if it replaces the nonterminal by the empty word. We make the additional restriction that our grammars are in Chomsky normal form without ϵ-rules (abbreviated CNF$^{-\epsilon}$). We point out that the emptiness problem of the intersection of such grammars in CNF$^{-\epsilon}$ is undecidable. This follows from the undecidability of the emptiness of the intersection of context-free grammars and the decidability of the word problem of context-free grammars.

Lemma 2.1.5. *The emptiness problem of the intersection of context-free grammars in* CNF$^{-\epsilon}$ *is undecidable.*

Example 2.1.6. Let $G_\mathsf{D} = (\mathcal{N}_\mathsf{D}, \mathcal{T}, \mathcal{R}_\mathsf{D}, S_\mathsf{D})$ and $G_\mathsf{U} = (\mathcal{N}_\mathsf{U}, \mathcal{T}, \mathcal{R}_\mathsf{U}, S_\mathsf{U})$ be two grammars in CNF$^{-\epsilon}$, where $\mathcal{N}_\mathsf{D} = \{A_\mathsf{D}, B_\mathsf{D}, S_\mathsf{D}\}$, $\mathcal{T} = \{a, b\}$, $\mathcal{N}_\mathsf{U} = \{C_\mathsf{U}, S_\mathsf{U}\}$ and $\mathcal{R}_\mathsf{D}, \mathcal{R}_\mathsf{U}$ are given in BNF-like notation:

$$\begin{aligned} S_\mathsf{D} &\to A_\mathsf{D} B_\mathsf{D} & \qquad S_\mathsf{U} &\to C_\mathsf{U} C_\mathsf{U} \\ A_\mathsf{D} &\to A_\mathsf{D} A_\mathsf{D} \mid a & C_\mathsf{U} &\to C_\mathsf{U} C_\mathsf{U} \mid a \mid b \\ B_\mathsf{D} &\to B_\mathsf{D} B_\mathsf{D} \mid b & & \end{aligned}$$

Two derivations of G_D and G_U are $S_\mathsf{D} \to A_\mathsf{D} B_\mathsf{D} \to A_\mathsf{D} A_\mathsf{D} B_\mathsf{D} \to A_\mathsf{D} A_\mathsf{D} b \to a A_\mathsf{D} b \to aab$ and $S_\mathsf{U} \to C_\mathsf{U} C_\mathsf{U} \to a C_\mathsf{U} \to a C_\mathsf{U} C_\mathsf{U} \to aa C_\mathsf{U} \to aab$. △

A derivation tree represents a derivation starting from a single letter and ending in a word consisting only of terminals, i.e. $\sigma \to^* w$ with $\sigma \in \mathcal{T} \cup \mathcal{N}, w \in \mathcal{T}$. Note that if $\sigma \in \mathcal{T}$, then the derivation performs zero rewrite steps.

Definition 2.1.7 (Derivation Tree)**.** For a grammar $G = (\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$ in CNF$^{-\epsilon}$ a *derivation tree* is a binary $\mathcal{N} \cup \mathcal{T}$-labelled tree (Υ, f), where the following conditions hold.

- All internal nodes are labelled by nonterminals,
- all leafs are labelled by terminals and
- for any internal node labelled by a nonterminal N let $w = \sigma_0 \sigma_1 \dots \sigma_{k-1}$ be the word formed by its k children. Then $(N, w) \in \mathcal{R}$.

If the root of a derivation tree is labelled by the starting nonterminal, then the word formed by concatenating the labels of the leafs in lexicographic order is in the language of G. △

Note that any subtree of a derivation tree again is a derivation tree. The derivations from Example 2.1.6 are shown in Figure 2.1 as derivation trees.

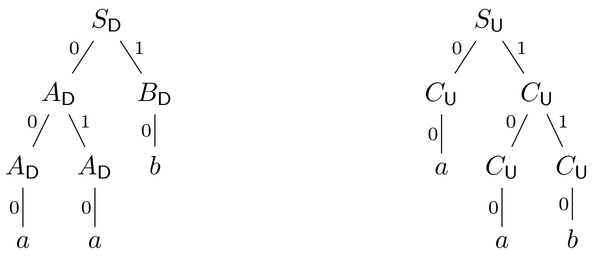


Figure 2.1: Derivation trees for the derivations from Example 2.1.6. The set of nodes for the left tree is $\{\epsilon, 0, 1, 00, 01, 10, 000, 010\}$ and for the right tree it is $\{\epsilon, 0, 1, 00, 10, 11, 100, 110\}$. Let the labelling function (f for the left tree) be given by the labels of the nodes. For $0 \prec 1$ we have $000 \prec_\mathsf{l} 010 \prec_\mathsf{l} 10$. Hence the word $\langle f(000), f(010), f(10)\rangle = \langle a, a, b\rangle$ is in $\mathcal{L}(G_\mathsf{D})$

2.1.5 Decision Problems

In our definition of decision problems and reduction of problems we follow [HU79]. A *decision problem* is a question to which the answer is either yes or no. An example is:

Is the language of a given context-free grammar empty?

Let A be the set of all context-free grammars with a nonempty language. Then the previous decision problem can be restated as whether for a given grammar G we have $G \in A$. Note that we abstract from how a grammar is represented.

To show that one decision problem A is at least as difficult as another problem B ($B \leq A$) we *reduce* B to A. Formally, we define a function $f : B \to A$ such that

$$b \in B \text{ iff } f(b) \in A \enspace ,$$

where the requirements on f depend on the particular reduction we use. In this work we only use reduction to show undecidability. Hence, we require f to be computable.

2.1.6 Other

For two values $x, x' \in \mathbb{R}$ let $x \pm x'$ denote the set $\{x + x'' \mid x'' \in [-x', x']\}$. Further, for $x \in \mathbb{R}$ we denote the *absolute value* of x as $|x|$, which is equal to x if x is nonnegative and $-x$ otherwise.

For coordinates $(x_1, y_1), (x_2, y_2) \in \mathbb{R} \times \mathbb{R}$ with $x_1 \leq x_2$ we can create a function $f : \mathbb{R} \to \mathbb{R}$ that for $x \in [x_1, x_2]$ *linearly interpolates* y as

$$f(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1} * (x - x_1) \ .$$

Intuitively, we set a ruler going through (x_1, y_1) at a gradient of $\frac{y_2 - y_1}{x_2 - x_1}$. Then we go $(x - x_1)$ units along the ruler to find the value of $f(x)$.

Metric Spaces

A set X together with a function $d : X \times X \to \mathbb{R}$ forms a *metric space* iff the following conditions hold:

$$\begin{aligned}
&\mathrm{M}_1 : d(x, x) = 0 \ , \\
&\mathrm{M}_2 : d(x, z) \leq d(x, y) + d(y, z) \ , \\
&\mathrm{M}_3 : d(x, y) = d(y, x) \ , \\
&\mathrm{M}_4 : \text{if } x \neq y \text{ then } d(x, y) > 0 \ .
\end{aligned}$$

Intuitively, d assigns to an two elements in X a distance. If the domain X is known we simply call d a metric. If we have a function $d : X \times X \to \mathbb{R}$ that only satisfies $\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3$ then we call (X, d) a *pseudo metric space* [SS78].

Upper and Lower Bounds

For a set $X \subseteq \mathbb{R}$ we call x an upper (resp. lower bound) of X iff x is greater (resp. less) or equal than all elements in X. We call x the *least upper bound*, or in other words the *supremum* of X, if x is less or equal than all other upper bounds of X. Similarly, if x is greater or equal all lower bounds of X, then x is the *greatest lower bound*, or the *infimum* of X. We denote this with $\mathsf{inf}(X)$ and $\mathsf{sup}(X)$. If the set X is given by the range of a function f we also write $\mathsf{sup}\, f$ for $\mathsf{sup}_{y \in \mathsf{dom}\, f}(f(y))$ and similarly for the infimum.

Let s be an infinite sequence with elements from X. Then s converges to $x \in \mathbb{R}$ (denoted $\mathsf{lim}_{n \to \infty} s(n) = x$) iff the elements in the sequence get arbitrary close to the limit y. Formally,

$$\lim_{n\to\infty} s(n) = x \text{ iff } \forall \epsilon \in \mathbb{R}_{>0}.\, \exists n_\epsilon \in \mathbb{N}.\, \forall n \in \mathbb{N}.\, n \geq n_\epsilon \implies |s(n) - x| < \epsilon \ .$$

We can relate limits of sequences with suprema and infima of sets. If a set X has a supremum, then there is a sequence s of elements from X such that $\lim_{n\to\infty} s(n) = \sup X$. This works analogously for the infimum.

Intervals

We call an interval $[r,t]$ an *proper interval* if $r < t$, and a *point-interval* if $r = t$. Let $\mathbb{IR}$ be the set of all closed, proper intervals over the nonnegative real numbers, i.e. $\mathbb{IR} = \{[r,t] \mid r,t \in \mathbb{R} \wedge r < t\}$. For intervals $[r,t],[r',t'] \in \mathbb{IR}$ we define

$$\begin{aligned} [r',t'] \subset [r,t] &\text{ iff } r < r' \wedge t > t' \ , \\ \underline{[r,t]} &= r \ , \\ \overline{[r,t]} &= t \ . \end{aligned}$$

For any natural numbers $l, l' \in \mathbb{N}$ we define that $[l,l']$ is a discrete interval. Note that we allow $l > l'$ and treat the resulting interval as being equal to $\emptyset$, i.e. we treat an interval as the set of numbers lying within the intervals borders. For a real-valued interval $[r,t] \in \mathbb{IR}$ we define the *length* of $[r,t]$ as

$$\|[r,t]\| = \mathsf{max}(0, t - r) \ .$$

Note that we use the maximum to avoid negative sizes, if $r > t$. Similarly, for a discrete interval $[l,l']$ with $l,l' \in \mathbb{N}$ we define the length of $[l,l']$ as

$$\|[l,l']\| = |\{l, \ldots, l'\}| \ .$$

Properties of Relations

We define some properties of relations following [Sch11]. For two sets X, Y let $R \subseteq X \times Y$ be a relation. Then, R is univalent (or deterministic) if every $x \in X$ is related to at most one $y \in Y$. Further, R is total if every $x \in X$ is related to some $y \in Y$. Formally,

$$\begin{aligned} R \text{ is univalent if } &\forall x \in X.\, \forall y_1, y_2 \in Y.\, ((x,y_1) \in R \wedge (x,y_2) \in R) \implies y_1 = y_2 \ , \\ R \text{ is total if } &\forall x \in X.\, \exists y \in Y.\, (x,y) \in R \ . \end{aligned}$$

For the following definitions we remind that $R^{\smile}$ is the inverse relation of R (cf. Page 6). We define that

- R is *injective* if $R^{\smile}$ is univalent,
- R is *surjective* if $R^{\smile}$ is total and
- R is *bijective* if it is injective and surjective.

2.2 Timed Automata

We introduce a version of timed automata [AD94] that we use to accept trajectories, rather than timed words. These timed automata are similar to Kripke structures, in that in each location a number of state variables take a value. Different from Kripke structures the allowed values in a state of our timed automata are defined by invariants over state variables, similar to invariants over clocks in classical timed automata. Our definition of timed automata mostly is taken from [OD08] with some inspiration from [Hoe06] on how to include state variables.

Before we introduce timed automata we define trajectories. We shall use V as our set of *state variables*. A *trajectory* is a function that assigns to a variable and every point in time a value. While a trajectory usually is defined over unbounded time, here we consider finite and infinite time. Thus, for some interval $T \subseteq \mathbb{T}$, where we allow T to be infinite, and a set V a trajectory τ is a function

$$\tau : V \to T \to \{0,1\} \ .$$

Note that we represent the truth value $\bot$ as 0 and $\top$ as 1 because later we take the integral of a trajectory. Further, we point out that the symbol τ is a variation of the Greek letter τ (tau).

We lift trajectories to Boolean formulas over V by a point-wise extension in a straightforward manner, for example, $\tau(S_0 \vee S_1)(t) = \mathsf{max}(\tau(S_0)(t), \tau(S_1)(t))$. We use the abbreviation S_τ for $\tau(S)$. We require that for every state variable $P \in V$, every finite part of P_τ has a finite number of discontinuity points, i.e. P_τ is of *finite variability.* Similar to timed words we use $\mathsf{span}(\tau)$ to denote the interval on which τ is defined.

First we define the constraints we may use in locations and edges of timed automata. For a set of Boolean variables V let $\mathbb{B}(V)$ be the set of Boolean formulas over the variables in V using the classical Boolean operators and equality. Further, for a set of variables X ranging over the real numbers let $\mathbb{B}(X)$ be constraints of the form $x - y \bowtie c$ or $x \bowtie c$ with $x, y \in X, c \in \mathbb{Q}, \bowtie \in \{<, >, \geq, \leq\}$.

For a set of variables S with data domain $\mathcal{D}$ let $\mathsf{Val}(S)$ be the set of all valuations $f : S \to \mathcal{D}$ that assign each variable a value.

Timed automata may synchronise their transitions via so called *channels*. For a set of channels Chan let

$$\mathsf{Lab} = \{a! \mid a \in \mathsf{Chan}\} \cup \{a? \mid a \in \mathsf{Chan}\} \cup \{\tau\}$$

be the corresponding set of actions. We refer to this set as Lab as these actions are used as labels on edges and to avoid confusion with an unrelated set of actions that we define later. Here, $a!$ and $a?$ form complementary actions that may synchronise with each other and τ is an internal action. Note that we use τ for internal actions, while we use $\mathtt{\tau}$ for trajectories.

Definition 2.2.1 (Timed Automata)**.** Let $\mathcal{X}$ be a finite set of variables ranging over the nonnegative real numbers. Further, let V be a finite set of state variables. A *timed automaton* is a tuple $A = (L, \to, \mathsf{Init}, I, \mathsf{Lab}, V, \Lambda, \mathcal{X})$, where L is the set of *locations*, $\to \subseteq L \times \mathsf{Lab} \times \mathbb{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times L$ is the set of edges, $\mathsf{Init} \subseteq (L \times \mathsf{Val}(V) \times \mathsf{Val}(\mathcal{X}))$ is the set of *initial configurations*, $I : L \to \mathbb{B}(\mathcal{X})$ and $\Lambda : L \to \mathbb{B}(V)$ are the clock- and state variable invariants per location. A *configuration* of a timed automaton is a tuple $(l, \beta, \nu) \in (L \times \mathsf{Val}(V) \times \mathsf{Val}(\mathcal{X}))$. △

Note that commonly Init simply is a set of locations L' and that initially all clocks have the value 0. In our setting this would be

$$\mathsf{Init} = \{(l, \beta, \mathbf{0}) \mid l \in L', \beta \in \mathsf{Val}(V), \beta \models \Lambda(l)\} \ ,$$

where $\mathbf{0}$ is the clock valuation where all clocks have value 0. Depending on the set Init, analysing timed automata can be arbitrary complex. An example would be an automaton with the set of initial configurations containing arbitrary locations and arbitrary state variable valuations and those clock valuations where all clocks evaluate to irrational numbers. However, we restrict ourselves to sets of configurations representable with linear real arithmetic (cf. Page 31). In Figure 2.2 we show a timed automaton modelling a torch. The example is originally due to K. G. Larsen.

We define the parallel composition of timed automata, where different automata can synchronise via channels. Such a parallel composition results in a so

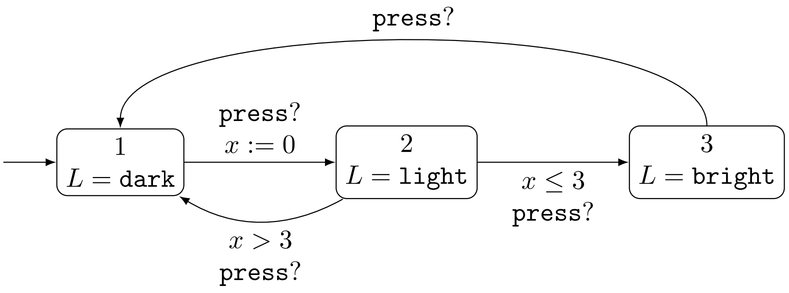


Figure 2.2: A timed automaton modelling a torch due to K. G. Larsen. The automaton contains the three locations 1, 2 and 3, a channel `press` to synchronise with another automaton, a clock x to measure how quickly press-signals arrive and the state variable L (for luminosity) with the domain $\{\texttt{dark}, \texttt{light}, \texttt{bright}\}$. When the torch is off, one press-signal turns the torch on, another press-signal quickly afterwards turns the torch bright and a third signal turns it off again.

called network of timed automata. We define the operational semantics of these networks as a labelled transition system. The only change in the semantics as we define it here to the classical semantics as it is defined for example in [OD08], is that here we also consider state variables. The addition of state variables is taken from [Hoe06].

Definition 2.2.2 (Operational Semantics of Networks)**.** Consider timed automata $A_i = (L_i, \rightarrow_i, \mathsf{Init}_i, I_i, \mathsf{Lab}_i, V_i, \Lambda_i, \mathcal{X}_i)$ with $i \in \{1, \dots, n\}$ with mutually disjoint sets L_i, V_i and $\mathcal{X}_i$. Then a network of timed automata is denoted by $A_1 \| \cdots \| A_n$. Its semantics is given by a transition system with the set of configurations

$$(L_1 \times \cdots \times L_n) \times (\mathsf{Val}(\mathcal{X}_1 \cup \cdots \cup \mathcal{X}_n)) \times (\mathsf{Val}(V_1 \cup \cdots \cup V_n)) \enspace .$$

The transition system has a

- *delay transition* $(\vec{l}, \nu, \beta) \xrightarrow{t} (\vec{l}, \nu + t, \beta)$ if $\nu + t' \models \bigwedge_{i=1}^{n} I_i(l_i)$ for all $t' \in [0, t]$,

- *local transition* $(\vec{l}, \nu, \beta) \xrightarrow{\tau} (\vec{l'}, \nu', \beta')$ if for some $i \in \{1, \dots, n\}$ there is an edge $(l_i, \tau, g, R, l_i') \in \rightarrow_i$ such that $\nu \models g$, $\vec{l'} = \vec{l}[l_i := l_i']$, $\nu' = \nu[R := 0]$, and $\nu' \models I_i(l_i')$, $\beta' \models \Lambda_i(l_i')$ and $V_i \lhd \beta = V_i \lhd \beta'$ and
- a *synchronising transition* $(\vec{l}, \nu, \beta) \xrightarrow{\tau} (\vec{l'}, \nu, \beta)$ if for two different natural numbers $i, j \in \{1, \dots, n\}$ and some channel b there are two transitions $(l_i, b!, g_i, R_i, l_i') \in \rightarrow_i$ and $(l_j, b?, g_j, R_j, l_j') \in \rightarrow_j$ such that $\nu \models g_i \wedge g_j$, $\vec{l'} = \vec{l}[l_i := l_i'][l_j := l_j']$, $\nu' = \nu[R_i := 0][R_j := 0]$, $\nu' \models I_i(l_i') \wedge I_j(l_j')$, $\beta \models \Lambda_i(l_i') \wedge \Lambda_j(l_j')$ and $(V_i \cup V_j) \lhd \beta = (V_i \cup V_j) \lhd \beta'$. $\triangle$

We point out that with $V_i \lhd \beta = V_i \lhd \beta'$ and $(V_i \cup V_j) \lhd \beta = (V_i \cup V_j) \lhd \beta'$ in the discrete transitions we ensure that only automata participating in a transition may change the values of their state variables.

If a network of timed automata does not allow multiple value changes of the same state variable without delay in between we call such a network *delaying.* A sufficient syntactic constraint for a delaying network is that for every automaton in the network, either the set of state variables of that automaton is empty or for every location of that automaton there is a clock x that is reset along all incoming transitions and the guards of all outgoing transitions require $x > 0$. Note that incoming and outgoing transitions here includes loops. We consider a single timed automaton as a special case of a network of timed automata. Thus, henceforth we shall always consider networks of timed automata and abbreviate them simply as timed automata. Note that in the definition above only delay transitions and τ-transitions are defined.

Consider the automaton from Figure 2.2. If we put it in parallel with an automaton that allows zero time to pass between two press-signals, then the resulting network is not delaying. However, if we build a network with an automaton that has no state variables and requires nonzero time to pass between two press-signals, then the resulting network is delaying.

A *run* of a timed automaton is a possibly infinite time-stamped sequence of configurations

$$\pi = \langle ((l_0, \beta_0, \nu_0), t_0) \dots ((l_i, \beta_i, \nu_i), t_i) \dots \rangle$$

such that there are transitions

$$(l_0, \beta_0, \nu_0) \xrightarrow{\lambda_1} \cdots \xrightarrow{\lambda_i - 1} (l_i, \beta_i, \nu_i) \cdots$$

with $\lambda_i \in \mathbb{R} \cup \{\tau\}$ and $i \in N$, where $N = \mathbb{N}$ if π is infinite and otherwise $N = \{0, \dots, \#\pi\}$.

We define how to get trajectories, which are signals over time, from runs of a timed automaton.

Definition 2.2.3. For a (possibly infinite) run

$$\langle((l_0, \beta_0, \nu_0), t_0) \ldots ((l_i, \beta_i, \nu_i), t_i) \ldots\rangle$$

with $\beta_i \in \mathsf{Val}(V)$ for some V such that for all $i, j \in N$ and all $P \in V$ if $\beta_i(P) \neq \beta_j(P)$ then $t_i \neq t_j$. Then our run *matches* a trajectory τ iff for all $i, i+1 \in N$, all variables $P \in V$ and almost all $t \in [t_i, t_i + 1)$ we have $\tau(P)(t) = \beta_i(P)$. $\triangle$

With $\mathcal{T}(A)$ we denote the set of all trajectories for which there exists a run on A.

In the definition above the condition "if $\beta_i(P) \neq \beta_j(P)$ then $t_i \neq t_j$" ensures that at every point in time we assign exactly one value to $\tau(P)(t)$.

To connect the slightly adapted timed automata to trajectories we use the following lemma.

Lemma 2.2.4. *Let $A_1 \| \cdots \| A_n$ be a delaying network of timed automata. Then for any run of the network there is a matching a trajectory.*

Proof. Proof by contradiction. Assume the lemma does not hold and let $A_k = (L_k, \rightarrow_k, \mathsf{Init}_k, I_k, \mathsf{Lab}_k, V_k, \Lambda_k, \mathcal{X}_k)$ with $k \in \{1, \ldots, n\}$. Further, let $\langle((l_0, \beta_0, \nu_0), t_0) \ldots\rangle$ be our violating run for which there is no matching trajectory. Then there are $i, j \in N$, $k \in \{1, \ldots, n\}$ and $P \in V_k$ with $\beta_i(P) \neq \beta_j(P)$ and $t_i = t_j$. However, then A_k is not a delaying timed automaton and also $V_k \neq \emptyset$, which violates our assumptions. We have a contradiction, which means that the lemma holds. $\square$

In this thesis we always connect timed automata to trajectories. Hence, we only consider delaying timed automata, and usually do not explicitly mention that they are delaying.

Discussion of our Definition of Timed Automata In [Hoe06], the semantics of the real-time automata (there called Phase Event Automata) is defined by giving properties of a run. Here, we define the semantics of timed automata via a transition system. Then a run is a sequence of configurations in the transition system. In our definition via a transition system it is very difficulty to define that the transition system ensures that nonzero time passes between value changes of state variables. Hence, we added this requirement as a semantic constraint and termed automata satisfying it as delaying. In [Hoe06] this constraint can easily be ensured in the definition of a legal run.

Timed Automata in Uppaal

Uppaal is a popular tool to analyse timed automata. As we use Uppaal in one of our longer examples we briefly explain it here. For a more detailed exposition we refer to [BDL]. The timed automata Uppaal uses have some differences to the timed automata we introduced.

Data variables One difference is that Uppaal has *data variables.* These data variables are similar to the state variables of our timed automata. One difference is that between two value changes of a data variable zero time may pass, while we require nonzero time to pass between value changes of state variables.

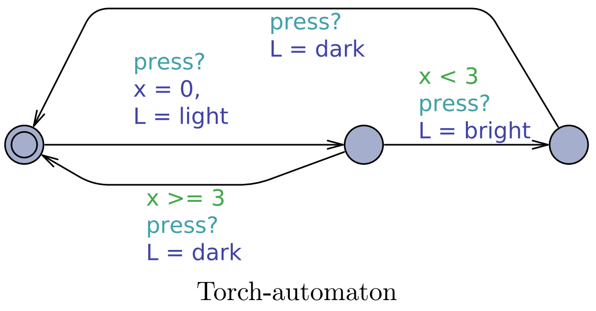
Priority of locations In Uppaal we can designate locations as *committed.* If one or more automata in a network is in a committed location, then the next transition must involve one of the currently committed locations. If this is not possible a deadlock occurs.

Synchronisation Uppaal supports broadcast synchronisation. A broadcast channel is a channel a where the transition labelled with the sending action $a!$ is not blocked if no complementing receiving action $a?$ is enabled. Furthermore, multiple actions $a?$ may synchronise with a single sending $a!$ action. However, if possible synchronisation must occur. That is, all $a?$-labelled enabled transitions must synchronise with an $!a$ action.

Note that all data variables, clocks and channels in a network of timed automata may be local to a single a automaton, shared between multiple, or even all automata of a network. Finally, in Figure 2.3 we provide an example of a Uppaal timed automaton. We point out that in Uppaal variable assignments and checking equality are done with = and ==, while in our graphical representation we use := for variable assignments and = for checking equality.

2.3 Multi-Lane Spatial Logic

In this section we describe Multi-Lane Spatial Logic (MLSL), as it has been defined in [HLO+11; LH15]. The logic uses an abstract formal model for



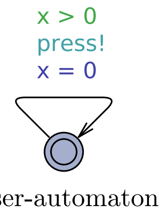


Figure 2.3: On the left we show a Uppaal representation of the automaton from Figure 2.2 that models a torch. On the right we show a user of a torch. Both automata have a local clock x. With the guard $x > 0$ in the User automaton we ensure that the parallel composition of the two automata is delaying. We modelled the state variable as a data variable that is updated whenever a transition is taken.

motorway traffic [HLO+11], where the traffic configuration at a specific point in time is given by a traffic snapshot. In a traffic snapshot the motorway is represented by two dimensions; the vertical discrete dimension represents the different lanes and the horizontal dense dimension represents the extension of the lanes. Then a *reservation* of a car represents space the car physically occupies plus some safety margin, which we assume to be the braking distance. When a car changes lanes it may have multiple adjacent reservations. A *claim* of a car represents that the car would like to reserve the claimed space. With claims we model the turn-signal of a real car. Additionally, a traffic snapshot has information about the speed and acceleration of each car. The evolution of traffic over time is modelled as a labelled transition system, where each state is a traffic snapshot. We give an example traffic snapshot and some MLSL formulas to develop some intuition for the formalism.

Example 2.3.1. MLSL Formulas are evaluated on a restricted area of a traffic snapshot called view. We show an example traffic snapshot and view in Figure 2.4. In the traffic snapshot, with the given view, the formula

$$\langle \mathsf{free} \frown \mathsf{re}(c_2) \frown \mathsf{free} \rangle$$

holds. Here, $\langle \cdot \rangle$ is an abbreviation and means that the subformula holds somewhere in the view, $\frown$ is used to separate adjacent segments within the lane,

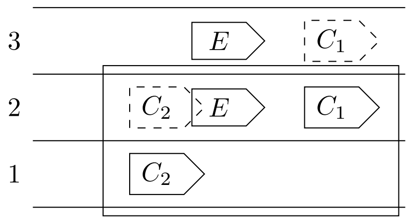


Figure 2.4: Visualisation of a traffic snapshot, where car C_2 has a reservation (solid line) and a claim (dashed line), car E has two reservations and car C_1 also has a reservation and a claim. The claim of car C_2 and a reservation of car E overlap. Additionally, we show a view (big rectangle). Note that one reservation of car E and the claim of car C_1 are outside of the view.

free indicates that the lane segment is free of claims and reservations and $\mathsf{re}(c_2)$ means that the segment has a reservation from car c_2. Note that in formulas we use lower case letters to refer to cars. The formula

$$\langle \mathsf{free} \frown \mathsf{cl}(c_2) \frown \mathsf{re}(\mathsf{ego}) \frown \mathsf{free} \frown \mathsf{re}(c_1) \frown \mathsf{free} \rangle$$

is also satisfied by the traffic snapshot and the view in Figure 2.4. With $\mathsf{cl}(c_2)$ we indicate that the lane segment has a claim of car c_2. Note that $\mathsf{cl}(c_2)$ and $\mathsf{re}(\mathsf{ego})$ are not exclusive, i.e. in the lane segment where the claim of C_2 and the reservation of E overlap, both, $\mathsf{cl}(c_2)$ and $\mathsf{re}(\mathsf{ego})$ are satisfied. We can stack formulas to express that on the lower lane the lower formula holds, and that on the upper lane the upper formula holds. That is, the formula

$$\begin{array}{c} \mathsf{free} \frown \mathsf{cl}(c_2) \frown \mathsf{re}(\mathsf{ego}) \frown \mathsf{free} \frown \mathsf{re}(c_1) \frown \mathsf{free} \\ \mathsf{free} \frown \mathsf{re}(c_2) \frown \mathsf{free} \end{array}$$

is satisfied with the complete view, not just somewhere within the view.

When describing locations within a traffic snapshot relative to each other we often take the bird's eye view. That is, in Figure 2.4 car E is *left* of car C_1 and the reservation of C_1 is *below* the claim of C_1. $\triangle$

2.3.1 The Model

In this thesis we consider only motorway traffic. We assume a countably infinite set of *car identifiers* $\mathbb{I}$ and an arbitrary but fixed set of *lanes* $\mathbb{L} = \{0, \dots, k\}$, for some $k \in \mathbb{N}_{\geq 1}$ to be given. Let $\mathcal{P}(\mathbb{L})$ denote the powerset of $\mathbb{L}$. Often we denote elements from $\mathbb{I}$ with C.

Definition 2.3.2 (Traffic Snapshot [HLO+11; LH15])**.** A *traffic snapshot* TS is a structure $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$, where

- $\mathsf{res} : \mathbb{I} \to \mathcal{P}(\mathbb{L})$ maps cars to their reserved lanes,
- $\mathsf{clm} : \mathbb{I} \to \mathcal{P}(\mathbb{L})$ maps cars to their claimed lanes,
- $\mathsf{pos} : \mathbb{I} \to \mathbb{R}$ maps cars to the position of their rear along the lanes,
- $\mathsf{spd} : \mathbb{I} \to \mathbb{R}$ maps cars to their speed and
- $\mathsf{acc} : \mathbb{I} \to \mathbb{R}$ maps cars to their acceleration.

We denote the set of all traffic snapshots by $\mathbb{TS}$. △

Furthermore, we require the following *sanity conditions* to hold for all cars.

Definition 2.3.3 (Sanity Conditions on Traffic Snapshots)**.** We call a traffic snapshot $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$ *sane* if for all $C \in \mathbb{I}$ the following holds:

1. Car C cannot both reserve and claim the same lane: $\mathsf{res}(C) \cap \mathsf{clm}(C) = \emptyset$.
2. Car C can reserve at most two lanes: $1 \leq |\mathsf{res}(C)| \leq 2$.
3. Reserved lanes must be next to each other:
$$|\mathsf{res}(c)| = 2 \text{ implies } \exists n \in \mathbb{L}.\, \mathsf{res}(C) = \{n, n+1\} \enspace .$$
4. Car C can claim at most one lane: $0 \leq |\mathsf{clm}(C)| \leq 1$.
5. Car C can reserve or claim at most two lanes: $1 \leq |\mathsf{res}(C)| + |\mathsf{clm}(C)| \leq 2$.
6. A claimed lane must be next to a reserved lane for car C:
$$\mathsf{clm}(C) \neq \emptyset \text{ implies } \exists n \in \mathbb{L}.\, \mathsf{res}(C) \cup \mathsf{clm}(C) = \{n, n+1\} \enspace .$$

7. Only finitely many cars participate or initiate in lane changing manoeuvres:

$$|\mathsf{res}(C)| = 2 \text{ or } |\mathsf{clm}(C)| = 1 \text{ holds only for finitely many } C \in \mathbb{I} \ .$$

△

We model the evolution of traffic snapshots as labelled transitions, where we use discrete and continuous transitions. The discrete transitions for a car C are to change the acceleration ($\mathrm{a}(C, a)$ with $a \in \mathbb{R}$), set a claim for a lane ($\mathrm{c}(C, n)$ with $n \in \mathbb{L}$), change an existing claim into a reservation $\mathrm{r}(C)$), withdraw an existing claim ($\mathrm{wd\ c}(C)$) and withdraw a reservation from a lane ($\mathrm{wd\ r}(C, n)$ with $n \in \mathbb{L}$). The continuous transitions are similar to delay transitions in timed automata, i.e. we update the data affected by time (here position, speed and the derived braking distance). To define the transitions we use substitution and function overriding, i.e. let $TS[f/f \oplus \{C \mapsto x\}]$ be TS, except that the function f is replaced by $f \oplus \{C \mapsto x\}$, which maps C to the value x and agrees on everything else with f.

Definition 2.3.4 (Transitions between Traffic Snapshots). Given a traffic snapshot $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$, a car $C \in \mathbb{I}$ and values $n \in \mathbb{L}, a \in \mathbb{R}, z \in \mathbb{T}$ we define

$$TS \xrightarrow{\mathrm{c}(C,n)} TS' \Leftrightarrow \quad \begin{aligned} & TS' = (\mathsf{res}, \mathsf{clm}', \mathsf{pos}, \mathsf{spd}, \mathsf{acc}) \\ \wedge \quad & |\mathsf{clm}(C)| = 0 \wedge |\mathsf{res}(C)| = 1 \\ \wedge \quad & \mathsf{res}(C) \cap \{n+1, n-1\} \neq \emptyset \\ \wedge \quad & \mathsf{clm}' = \mathsf{clm} \oplus \{C \mapsto \{n\}\} \end{aligned}$$

$$TS \xrightarrow{\mathrm{wd\ c}(C)} TS' \Leftrightarrow \quad \begin{aligned} & TS' = (\mathsf{res}, \mathsf{clm}', \mathsf{pos}, \mathsf{spd}, \mathsf{acc}) \\ \wedge \quad & \mathsf{clm}' = \mathsf{clm} \oplus \{C \mapsto \emptyset\} \end{aligned}$$

$$TS \xrightarrow{\mathrm{r}(C)} TS' \Leftrightarrow \quad \begin{aligned} & TS' = (\mathsf{res}', \mathsf{clm}', \mathsf{pos}, \mathsf{spd}, \mathsf{acc}) \\ \wedge \quad & \mathsf{clm}' = \mathsf{clm} \oplus \{C \mapsto \emptyset\} \\ \wedge \quad & \mathsf{res}' = \mathsf{res} \oplus \{C \mapsto \mathsf{res}(C) \cup \mathsf{clm}(C)\} \end{aligned}$$

$$TS \xrightarrow{\mathrm{wd\ r}(C,n)} TS' \Leftrightarrow \quad \begin{aligned} & TS' = (\mathsf{res}', \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc}) \\ \wedge \quad & \mathsf{res}' = \mathsf{res} \oplus \{C \mapsto \{n\}\} \\ \wedge \quad & n \in \mathsf{res}(C) \wedge |\mathsf{res}(C)| = 2 \end{aligned}$$

$$TS \xrightarrow{z} TS' \Leftrightarrow \quad \begin{aligned} & TS' = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}', \mathsf{spd}', \mathsf{acc}) \\ \wedge \quad & \forall C \in \mathbb{I}.\, \mathsf{pos}'(C) = \\ & \qquad \mathsf{pos}(C) + \mathsf{spd}(C) * z + \frac{1}{2}\mathsf{acc}(C) * z^2 \end{aligned}$$

$$TS \xrightarrow{\mathrm{a}(C,a)} TS' \Leftrightarrow \begin{array}{rl} & TS' = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc}') \\ \wedge & \mathsf{acc}' = \mathsf{acc} \oplus \{C \mapsto a\} \end{array}$$

We denote the set of all *discrete actions* with Act. △

We wish to restrict our attention within a traffic snapshot to an area localised around some car. To this end we introduce a view.

Definition 2.3.5 (View)**.** A *view* V is defined as a structure $V = (L, X, E)$, where

- $L = [l, l'] \subseteq \mathbb{L}$ is an interval of lanes that are visible in V,
- $X = [r, r'] \subseteq \mathbb{R}$ is an interval representing the *extension* of the lanes visible in V and
- $E \in \mathbb{I}$ is the *owner* of V (or sometimes *ego car*).

We denote the set of all views with $\mathbb{V}$. A *subview* of V is obtained by restricting the lanes and extension we observe. Let L', X' be subintervals of L and X, then we define

$$V^{L'} = (L', X, E) \qquad \text{and} \qquad V_{X'} = (L, X', E) \ .$$ △

We define chopping operations on discrete and one dense intervals.

Definition 2.3.6 (Chopping of Intervals)**.** Let $V_i = (L_i, X_i)$ be views with $i \in \{0, 1, 2\}$ and $X_i = [r_i, t_i]$. Then we define *vertical chopping* (denoted by $\ominus$) and *horizontal chopping* (denoted by $\oplus$) of V_0 into V_1 and V_2 as

$$\begin{aligned} V_0 = V_1 \ominus V_2 \ \text{ iff } & L_0 = L_1 \cup L_2 \ \text{ and } \ L_1 \cap L_2 = \emptyset \ \text{ and } \ X_0 = X_1 = X_2 \ \text{ and} \\ & (L_1 = \emptyset \ \text{ or } \ L_2 = \emptyset \ \text{ or } \ \mathsf{max}(L_1) + 1 = \mathsf{min}(L_2)) \ , \\ V_0 = V_1 \oplus V_2 \ \text{ iff } & t_1 = r_2 \ \text{ and } \ r_0 = r_1 \ \text{ and } \ t_0 = t_2 \ \text{ and } \ L_0 = L_1 = L_2 \ . \end{aligned}$$

△

We use the chopping operations to define relations on views.

Definition 2.3.7 (Relation of Views)**.** Let V, V_1, V_2 be three views with $V = (L, X, E)$. Then we define $V = V_1 \ominus V_2$ iff $L = L_1 \ominus L_2$, $V_1 = V^{L_1}$ and $V_2 = V^{L_2}$. Similarly, we define $V = V_1 \oplus V_2$ iff $X = X_1 \oplus X_2$, $V_1 = V_{X_1}$ and $V_2 = V_{X_2}$. △

For views V, V_1, V_2 with $V = V_1 \ominus V_2$ we say that V_1 is *below* V_2. Similarly, if $V = V_1 \oplus V_2$, then V_1 is *left* of V_2.

The empty set of lanes takes a special role for the vertical composition of views.

Remark 2.3.8. For all views V the following holds:

$$V = V^{\emptyset} \ominus V \ ,$$
$$V = V \ominus V^{\emptyset} \ ,$$
$$V^{\emptyset} = V^{\emptyset} \ominus V^{\emptyset} \ . \qquad \triangle$$

Our definition of horizontal chopping only works for nonempty intervals, i.e. intervals that contain at least a single point. We define the following sanity condition on views to make this assumption explicit.

Definition 2.3.9 (Sanity Condition for Views)**.** For all views $V = (L, X, E)$ with $X = [r, r']$ we have $r \leq r'$. $\triangle$

In [Lin15] an abstract *sensor function* $\Omega : \mathbb{I} \times \mathbb{TS} \to \mathbb{R}_{>0}$ is used with the intuition that it returns the physical size of a car plus its braking distance. In this work we deviate from the classical papers where MLSL was defined and take the sensor function into the model. This seemed to give more clarity in some proofs and constructions.

Let CVar be a set of variables ranging over car identifiers. In the logic we use a special constant ego to refer to the owner of the current view. A valuation maps variables and the special symbol ego to car identifiers, i.e., a *valuation* is a function $\nu : \mathsf{CVar} \cup \{\mathsf{ego}\} \to \mathbb{I}$.

Now we have all the ingredients to define what an MLSL model is. Note that different from the original definition we include the set of car identifiers and the sensor function in the model. We do this to make some proofs later on clearer.

Definition 2.3.10 (MLSL Model)**.** Given a set of car identifiers $\mathbb{I}$, a traffic snapshot TS, a sensor function Ω, a view V and a valuation ν a *model* is a structure $M = (TS, \Omega, V, \nu)$. We call a model *sane* if the view and the traffic snapshot satisfy their sanity conditions. We denote the set of all MLSL models with $\mathbb{M}$. $\triangle$

In this work we only consider sane MLSL models.

To define the semantics of the logic we introduce the following abbreviating functions. Let $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$ be a traffic snapshot, $V = (L, X, E)$ a view and Ω a sensor function. Then

$$\begin{aligned}
\mathsf{res}_V &: \mathbb{I} \to \mathcal{P}(L) \text{ with } \mathsf{res}(C) \cap L,\\
\mathsf{clm}_V &: \mathbb{I} \to \mathcal{P}(L) \text{ with } \mathsf{clm}(C) \cap L,\\
\mathrm{len}_V &: \mathbb{I} \to \mathcal{P}(X) \text{ with } [\mathsf{pos}(C), \mathsf{pos}(C) + \Omega(C, TS)] \cap X \ .
\end{aligned}$$

The function res_V and clm_V restrict the reservations and claims to the current view and len_V restricts the space occupied by a car to the view. Additionally, we use the following abbreviation to denote the set of cars in the current view V:

$$I_V = \{C \in \mathbb{I} : \|\mathrm{len}_V(C)\| > 0 \text{ and } \mathsf{res}_V(C) \cup \mathsf{clm}_V(C) \neq \emptyset\} \ .$$

If the model is not clear from the context we use I_V^M.

Given a sensor function Ω and a traffic snapshot TS the *safety envelope* of the car C for the traffic snapshot TS is given as

$$\mathsf{se}(C, TS, \Omega) = [\mathsf{pos}(C), \mathsf{pos}(C) + \Omega(C, TS)] \ .$$

The safety envelope represents the horizontal space used by a car as perceived by another.

We define transitions between MLSL models. For this we first define a function mv that moves the extension of a view according to the movement of the owner of the view.

Definition 2.3.11 (Transitions between MLSL Models)**.** For $i \in \{1, 2\}$ let $M_i = (TS_i, \Omega, V_i, \nu)$ with $TS_i = (\mathsf{res}_i, \mathsf{clm}_i, \mathsf{pos}_i, \mathsf{spd}_i, \mathsf{acc}_i)$ and $V_i = (L, [r_i, r_i'], E)$ be two MLSL models. Then the result of moving V_1 from TS_1 to TS_2 gives a new view

$$\mathsf{mv}_{TS_2}^{TS_1}(V_1) = (L, [r_1 + x, r_1' + x], E) \ ,$$

where $x = \mathsf{pos}_2(E) - \mathsf{pos}_1(E)$. For $\lambda \in \mathsf{Act} \cup \mathbb{R}_{\geq 0}$ we define

$$M_1 \xrightarrow{\lambda} M_2 \text{ iff } TS_1 \xrightarrow{\lambda} TS_2 \text{ and } V_2 = \mathsf{mv}_{TS_2}^{TS_1}(V_1) \ .$$

$\triangle$

2.3.2 The Logic

The atoms of MLSL are used to express that some part of a lane is filled by a reservation or completely free of reservations. The chop operators in the logic are defined using the chop operators on views. The horizontal chop formula $\phi_0 \frown \phi_1$ expresses that on the left subview ϕ_0 and on the right subview ϕ_1 holds. With the vertical chop formula $\binom{\phi_0}{\phi_1}$ we require that the lower subview satisfies ϕ_0 and that the upper subview satisfies ϕ_1. Note that for both chop formulas the satisfying subviews might be empty. Additionally, the logic is closed under first order operators. Note that we introduce is a blend of MLSL as it is defined in [HLO+11] and [LH15; Lin15]. That means, we take the semantics from [LH15; Lin15] as it is the most mature semantics. However, of the operators we only take the length measurement and we leave the branching temporal operators quantification over distances out. From [HLO+11] we take the atom free.

Let RVar and LVar sets of variables ranging over the reals, and over the set of lanes. Then, let Var be the union of RVar, LVar and CVar.

Definition 2.3.12 (Syntax)**.** Given $\gamma, \gamma' \in \mathsf{CVar} \cup \{\mathsf{ego}\}, c \in \mathsf{CVar}, k \in \mathbb{R}_{\geq 0}$ the syntax of MLSL is given by

$$\phi ::= \gamma = \gamma \mid \ell = k \mid \mathsf{free} \mid \mathsf{re}(\gamma) \mid \mathsf{cl}(\gamma) \mid \phi \wedge \phi \mid \neg\phi \mid \exists c.\, \phi \mid \phi \frown \phi \mid \begin{matrix}\phi\\ \phi\end{matrix} \,. \quad \triangle$$

Now we can define the semantics of MLSL. The formula free is true for one-lane views containing no cars, $\mathsf{re}(c)$ and $\mathsf{cl}(c)$ are true for one-lane views that are fully covered by the safety envelope of a reservation or claim, respectively, by c. Furthermore, $\phi_1 \frown \phi_2$ denotes horizontal and $\binom{\phi_2}{\phi_1}$ vertical partitioning of a view.

Definition 2.3.13 (Semantics)**.** Let $\gamma, \gamma' \in \mathsf{CVar} \cup \{\mathsf{ego}\}, c \in \mathsf{CVar}, k \in \mathbb{R}$. Then, given a traffic snapshot TS, a view $V = (L, X, E)$, with $X = [r, r']$, a sensor function Ω and a valuation ν we define the *satisfaction* of a formula by a model $M = (TS, \Omega, V, \nu)$ as

$$\begin{aligned}
&M \models \gamma = \gamma' && \text{iff } \nu(\gamma) = \nu(\gamma'),\\
&M \models \ell = k && \text{iff } \|X\| = k,\\
&M \models \mathsf{free} && \text{iff } |L| = 1 \text{ and } \|X\| > 0 \text{ and}
\end{aligned}$$

$$
\begin{array}{lll}
 & & (\forall C \in \mathbb{I}.\, L \neq \mathsf{res}_V(C) \cup \mathsf{clm}_V(C) \ \text{ or } \ \mathrm{len}_V(C) \cap (r, r') = \emptyset), \\
M \models \mathsf{re}(\gamma) & \text{iff} & |L| = 1 \ \text{ and } \ \|X\| > 0 \ \text{ and} \\
 & & \mathsf{res}_V(\nu(\gamma)) = L \ \text{ and } \ X = \mathrm{len}_V(\nu(\gamma)), \\
M \models \mathsf{cl}(\gamma) & \text{iff} & |L| = 1 \ \text{ and } \ \|X\| > 0 \ \text{ and} \\
 & & \mathsf{clm}_V(\nu(\gamma)) = L \ \text{ and } \ X = \mathrm{len}_V(\nu(\gamma)), \\
M \models \phi_1 \wedge \phi_2 & \text{iff} & M \models \phi_1 \ \text{ and } \ M \models \phi_2, \\
M \models \neg \phi_1 & \text{iff} & M \not\models \phi_1, \\
M \models \exists c.\, \phi & \text{iff} & \exists C \in \mathbb{I}.\ TS, \Omega, V, \nu \oplus \{c \mapsto C\} \models \phi, \\
M \models \phi_1 \frown \phi_2 & \text{iff} & \exists V_1, V_1.\, V = V_1 \varobar V_2 \ \text{ and} \\
 & & TS, \Omega, V_1, \nu \models \phi_1 \ \text{ and } \ TS, \Omega, V_2, \nu \models \phi_1, \\
M \models \begin{matrix} \phi_2 \\ \phi_1 \end{matrix} & \text{iff} & \exists V_1, V_2.\, V = V_1 \ominus V_2 \ \text{ and} \\
 & & TS, \Omega, V_1, \nu \models \phi_1 \ \text{ and } \ TS, \Omega, V_2, \nu \models \phi_2 \ .
\end{array}
$$

△

In addition we make use of the standard abbreviations such as $\mathsf{true}, \vee, \forall$. Also, we use a derived modality to express that *somewhere* on the motorway ϕ holds. It is defined by using both chop operators as

$$\langle \phi \rangle = \mathsf{true} \frown \begin{pmatrix} \mathsf{true} \\ \phi \\ \mathsf{true} \end{pmatrix} \frown \mathsf{true} \ .$$

2.4 First-Order Theories

The general idea of first-order theories is taken from [BM07]. A first-order theory consists of a *signature* that defines the syntactically allowed formulas in this theory and a set of *axioms* that define the meaning of the constants, functions and predicates in the signature. In this thesis, when we introduce a first-order theory, we provide the signature and additionally a grammar describing the syntax. For the axioms we refer to [BM07]. A formula of a first-order theory is *satisfiable* if there is an *interpretation* that satisfies the formula and the axioms. Otherwise the formula is unsatisfiable. A formula is *valid* if every interpretation that satisfies the axioms also satisfies the formula. If a formula is valid, then its negation is *unsatisfiable.* To determine whether an interpretation satisfies a formula is the *model problem.* Note that we denote the satisfaction relation with $\models$. Furthermore, we often use *variable assignment* or simply *assignment* instead of interpretation.

We consider formulas from the *first-order theory of real-closed fields* (*FORCF*), also known as *elementary algebra*, with the signature $\{0, 1, +, -, *, \leq\}$ and standard axioms, where $*$ denotes multiplication.[2] For a formal account of the axioms we refer to [Tar51] or [BM07] for a more modern exposition. We denote the set of real-valued variables as RVar. This logic shares symbols with MLSL, such as $=$, $\neg$ and $\wedge$. However, from the context it will be clear to which logic symbols belong.

Definition 2.4.1 (FORCF Syntax)**.** The set of *FORCF formulas* has the following syntax:

$$\begin{aligned}\psi &::= \exists x.\, \psi \mid \theta \leq \theta \mid \neg\psi \mid \psi \wedge \psi \ ,\\ \theta &::= 0 \mid 1 \mid x \mid \theta * \theta \mid \theta + \theta \mid -\theta \ ,\end{aligned}$$

where $x \in \mathsf{RVar}$. △

A variable assignment for a FORCF formula is given by h. The satisfiability problem of this logic is decidable [Tar51] with a complexity that is double-exponential in the length of the formula [BM07].

Lemma 2.4.2 (Complexity of FORCF ([BM07]))**.** *The time complexity of deciding the satisfiability of a FORCF formula is double-exponential in the length of the formula.*

In the *first-order theory of mixed linear arithmetic* (*FOMLA*), with the signature $\{0, 1, +, -, \leq, \lfloor\rfloor\}$ we use variables ranging over the real numbers, as well as the operations of linear arithmetic and rounding to the next smaller integer. For the axioms of this theory we refer the reader to [BM07].

Definition 2.4.3 (FOMLA Syntax)**.** The set of FOMLA formulas is generated by the following grammar:

$$\begin{aligned}\psi &::= \exists x.\, \psi \mid \theta \leq \theta \mid \neg\psi \mid \psi \wedge \psi \ ,\\ \theta &::= 0 \mid 1 \mid x \mid \lfloor\theta\rfloor \mid \theta + \theta \mid -\theta \ ,\end{aligned}$$

where $x \in \mathsf{RVar}$. We denote the set of FOMLA formulas with Ψ. △

[2]We use $*$ (asterisk) for multiplication instead of $\cdot$ (centered dot). The reason for this is that we use multiplication together with . (dot) to multiply a value a associated with an object O with another value b, i.e $O.a * b$.

The interpretation of the symbols is the standard one. For $\lfloor\theta\rfloor$ we point out that this term rounds the value of θ down to the next integer. We use the remaining propositional connectives and $=, <, \geq$ and $>$ as abbreviations. Furthermore, $\exists i \in \mathbb{N}.\, \psi$ is an abbreviation for $\exists i \in \mathbb{R}.\, i = \lfloor i\rfloor \wedge 0 \leq i \wedge \psi$. When the constraint $i = \lfloor i\rfloor \wedge 0 \leq i$ is associated with a variable i, then i ranges over natural numbers and we say that $i \in \mathsf{NVar}$, that is, NVar is a set of variables ranging over natural numbers. A variable assignment for a FOMLA formula is given by κ.

In [Wei99] Weisspfennig proves that the satisfiability problem of FOMLA is decidable. For this, he first brings the formula into a form that separates the reasoning about integers and the reals. Then he applies quantifier elimination for linear integer arithmetic formulas (Pressburger arithmetic) and for linear real arithmetic to the separate formulas. For the following lemma the lower bound is shown in [FR74], and the upper bound is shown in [Wei99]. Note that [FR74] shows that Pressburger arithmetic, which is a fragment of FOMLA, is at least double-exponential.

Lemma 2.4.4 (Complexity of FOMLA ([FR74] and [Wei99]))**.** *The time complexity of deciding the satisfiability of a FOMLA formula is at best double-exponential and at worst triple-exponential in the length of the formula.*

We introduce the *first-order theory of linear real arithmetic* (*FOLRA*), for which the signature is $\{0, 1, +, -, \leq\}$. The axioms of this theory are presented in [BM07].

Definition 2.4.5 (FOLRA Syntax)**.** The set of FOLRA formulas is generated by the following grammar:

$$\begin{aligned} \psi &::= \exists x.\, \psi \mid \theta \leq \theta \mid \neg\psi \mid \psi \wedge \psi \ , \\ \theta &::= 0 \mid 1 \mid x \mid \theta + \theta \mid -\theta \ , \end{aligned}$$

where $x \in \mathsf{RVar}$. △

The interpretation is standard and thus equal to the interpretation of FORCF or FOMLA. In FOLRA we can only use operators that appear in both FOMLA and FORCF. Thus, any FOLRA formula also is a formula of the other two arithmetic logics. For the lemma below the lower bound is due to [FR74] and the upper bound is due to [FR75].

Lemma 2.4.6 (Complexity of FOLRA ([FR74] and [FR75])). *The time complexity of deciding the satisfiability of a FOLRA formula is at best single-exponential and at worst double-exponential in the length of the formula.*

We call the set of quantifier free FOLRA formulas *linear real arithmetic*. For linear real arithmetic checking satisfiability takes nondeterministic polynomial time. This complexity is due to the Boolean structure. For the conjunctive fragment of linear real arithmetic (without disjunctions) checking satisfiability is possible in deterministic polynomial time [Kha79].

For all of the arithmetic logics we defined, we introduce some abbreviations. For terms $\theta_0, \theta_1, \theta_2, \theta_3$ and $\theta'_0, \ldots, \theta'_k$ with $k \in \mathbb{N}_{\geq 1}$ we define

$$\begin{aligned}
[\theta_0, \theta_1] \subseteq [\theta_2, \theta_3] &\equiv \theta_0 > \theta_1 \vee (\theta_2 \leq \theta_0 \wedge \theta_1 \leq \theta_3) \ , \\
\theta_0 \in \{\theta'_1, \ldots, \theta'_k\} &\equiv \bigvee_{\theta' \in \{\theta'_1, \ldots, \theta'_k\}} \theta_0 = \theta' \ , \\
[\theta_0, \theta_1] \cap (\theta_2, \theta_3) = \emptyset &\equiv \theta_1 < \theta_0 \vee \theta_3 \leq \theta_2 \vee \theta_1 \leq \theta_2 \vee \theta_3 \leq \theta_0 \ .
\end{aligned}$$

With $\theta_0 > \theta_1$ in the first line we check if the interval is empty. If it is, then it has to be a subset of any other interval. Note that both intervals in the subset definition are closed, but for the intersection definition the interval (θ_2, θ_3) is open. Thus, their intersection is empty if either interval is empty, or if the end of one interval is less or equal than the start of the other interval.

Further, we use an abbreviation to ensure the equality of finite sets. For finite sets Z, Z' we can ensure the equality of the sets with the formula

$$Z = Z' \equiv (\bigwedge_{z \in Z} \bigvee_{z' \in Z'} z = z') \wedge (\bigwedge_{z' \in Z'} \bigvee_{z \in Z} z' = z) \ .$$

We denote the set of all real-valued terms as RTerm and the set of terms ranging over $\mathbb{N}$ as NTerm. Note that these sets of terms overlap and that we do not distinguish if they originate from FOMLA, FORCF or FOLRA, as this will be clear from the context. Also, whether the real terms may contain multiplication or not will be clear from the context. Furthermore, in the next chapters we use θ specifically for terms from RTerm.

3 Satisfiability of Spatial Properties with Precise Information

We are interested in automating spatial reasoning with MLSL. To this end, we investigate whether the satisfiability problem of MLSL is decidable under two restrictions. First, we trade length measurement ($\ell = k$) for an unbounded number of lanes. This is interesting because the known undecidability result for MLSL heavily relies on length measurement [LH15]. We give a novel construction showing that the resulting satisfiability problem is undecidable. Second, we extend MLSL with a *scope operator*. With the scope operator we can restrict our reasoning to specific cars. For the resulting logic we study the satisfiability problem with a bounded number of cars and we show that this problem is decidable.

In this chapter we assume that information, such as the position of cars, is exact. By this we mean that we do not perturb the model or the formula before evaluating the satisfaction relation.

The idea of the undecidability of MLSL without length measurement is introduced in [Ody15b; Ody15a]. The definition of MLSL with scopes together with the decision procedure for a fragment of this extension of MLSL is taken from [FHO15]. However, in [FHO15] we do not provide a proof of correctness. To be able to prove the correctness with the desired rigour, in this thesis we introduce new concepts such as operations and a weak form of equality for MLSL models with scope and its representation in arithmetic logic.

In Section 3.1 we show that MLSL remains undecidable if we do not use length measurement but allow for an unbounded number of lanes. In Section 3.2 we extend MLSL with the scope operator. Then, in Section 3.3 we use the scope operator to define a fragment for which the satisfiability problem is decidable. Afterwards, in Section 3.4 we use the construction of the previous section to algorithmically solve the model problem. At the end of the chapter we consider related work and discuss our findings.

3.1 Undecidability of Satisfiability of MLSL

Undecidability of satisfiability of Duration Calculus was proven by a reduction of the halting problem of two counter machines to the satisfiability of Duration Calculus (2CM-Halting $\leq$ DC-SAT) [ZHS93]. This construction has been adapted to prove undecidability of MLSL with length measurement, where length measurement allows to compare the size of the current view to a constant [LH15]. The authors define that the representation of a configuration of a two counter machine is of length $5k$, where k is a natural number. The value m of a counter is represented in an interval of length k and consists of m reservations and the remaining space of a configuration is used for markers to separate the counters. To increase the value of a counter they require that for all reservations which are part of the representation of the counter, there is a reservation $5k$ space units later (in the next representation of the counter's value) and additionally, there is exactly one reservation in the later representation for which there is no reservation $5k$ space units earlier. For this construction it is important to be able to specify the distance between reservations.

Ultimately, we want to understand the border of undecidability for MLSL. We believe that even without length measurement the satisfiability problem of MLSL is undecidable. As a step to better understand the decidability problem of MLSL, in this chapter we trade length measurement for an unbounded number of lanes. We prove that the resulting variation of the satisfiability problem remains undecidable. To show this, we reduce the emptiness problem of the intersection of two context-free languages, which is undecidable [HU79], to the satisfiability problem of MLSL with unbounded lanes.

First of all, we formalise our two notions of satisfiability. In [LH15] the authors used the following notion of satisfiability.

Definition 3.1.1. Given a *finite* set of lanes $\mathbb{L}$ and an MLSL formula ϕ we say that ϕ is *lane-boundedly satisfiable* iff there exists a model M such that $M \models \phi$. △

Here we investigate a variation of this form of satisfiability in that we consider an infinite set of lanes. The purpose of this is to see how decidability is affected if we increase expressiveness in one way (infinite lanes) and decrease the expressiveness in another way (removing length measurement).

Definition 3.1.2. Given an *infinite* set of lanes $\mathbb{L}$ and an MLSL formula ϕ we say that ϕ is *lane-unboundedly satisfiable* iff there exists a model M such that $M \models \phi$. △

Note that with the definition of lane-unbounded satisfiability any model has an *finite* but *unbounded* amount of lanes. That is, only the set from which a model chooses its lanes is infinite. In this work, when we speak of satisfiability of MLSL formulas we always mean lane-unbounded satisfiability.

3.1.1 Construction

We encode two context-free grammars (CFG) in Chomsky normal form without ϵ-rules (CNF$^{-\epsilon}$) G_D and G_U in one formula $F(G_\mathsf{D}, G_\mathsf{U})$ such that a satisfying model represents two derivation trees, one from each grammar. The resulting formula is satisfiable iff both CFGs can produce the same terminal word.

We give an intuitive explanation of how a model satisfying the formula we construct in this section looks like. The representation of a derivation tree for G_D has its root on the top lane and grows *downward*, whereas the tree for G_U has its root on the bottom lane and grows *upward*. In a satisfying model every letter is represented as a number of successive reservations without space in between. Representations of adjacent letters are separated by free space and different letters are represented by a different number of reservations. If a nonterminal from the downwards growing grammar is replaced by a word, the word is represented on the lane below the nonterminal such that the horizontal space used to represent the word is strictly contained in the space used for the nonterminal. Equality of the derived words is represented as horizontal alignment of terminals that differ only in their subscripts (e.g. a_U and a_D in Figure 3.1). In Figure 3.1 we depict an MLSL model representing the derivation trees of the derivations from Example 2.1.6 (see Page 10).

Given two CFGs $G_\mathsf{D} = (\mathcal{N}_\mathsf{D}, \mathcal{T}, \mathcal{R}_\mathsf{D}, S_\mathsf{D})$ and $G_\mathsf{U} = (\mathcal{N}_\mathsf{U}, \mathcal{T}, \mathcal{R}_\mathsf{U}, S_\mathsf{U})$, we assume two sets $\mathcal{T}_\mathsf{D}$, $\mathcal{T}_\mathsf{U}$ and bijective functions $\pi_\mathsf{D} : \mathcal{T} \to \mathcal{T}_\mathsf{D}$ and $\pi_\mathsf{U} : \mathcal{T} \to \mathcal{T}_\mathsf{U}$ such that $\mathcal{T}_\mathsf{D}$, $\mathcal{N}_\mathsf{D}$, $\mathcal{T}_\mathsf{U}$ and $\mathcal{N}_\mathsf{U}$ are all disjoint. The idea behind the two functions is that we want to differentiate between the MLSL encoding of terminals from G_U and from G_D.

Letter

Let $\lambda : \mathcal{T}_\mathsf{D} \cup \mathcal{N}_\mathsf{D} \cup \mathcal{T}_\mathsf{U} \cup \mathcal{N}_\mathsf{U} \to \mathbb{N}_{>0}$ be an injective function. In MLSL we represent every letter $\sigma \in \mathcal{T}_\mathsf{D} \cup \mathcal{N}_\mathsf{D} \cup \mathcal{T}_\mathsf{U} \cup \mathcal{N}_\mathsf{U}$ as $\lambda(\sigma)$ successive reservations from different

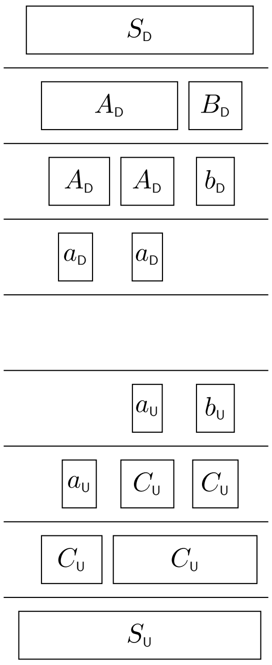


Figure 3.1: Visualisation of an MLSL model representing two derivation trees of derivations taken from Example 2.1.6. The boxes correspond to letters. Reservations inside the letters and the view are not shown.

cars, without free space in between. We formalise this as

$$\mathsf{letter}(\sigma, c_1) \equiv \mathsf{re}(c_1) \frown \exists c_2, \ldots, c_{\lambda(\sigma)}.\,(\mathsf{re}(c_2) \frown \ldots \frown \mathsf{re}(c_{\lambda(\sigma)}) \land \bigwedge_{i,j \in \{1,\ldots,\lambda(\sigma)\}}^{i \neq j} c_i \neq c_j) \ ,$$

where $c_1 \in \mathsf{CVar}$ is a car variable. We use c_1 as an identifier to uniquely differentiate letters within a formula.

Assume that the letter a_{D} is represented by one reservation, and that the letter b_{D} is represented by two reservations. We have to distinguish between two occurrences of a_{D} and one occurrence of b_{D}. To be able to recognise letters we demand that before and after every letter there is some free space. For this we define

$$\mathsf{letter}_{\mathsf{free}}(\sigma, c) \equiv \mathsf{free} \frown \mathsf{letter}(\sigma, c) \frown \mathsf{free} \ .$$

The formulas letter and $\mathsf{letter}_{\mathsf{free}}$ are used in other formulas, which will always bind the car variable, here c, with quantifiers.

Start

To ensure that there is a starting letter we express that the topmost lane contains the starting nonterminal S_{D} as

$$\mathsf{start}_{\mathsf{D}} \equiv \exists c.\, \begin{pmatrix} \mathsf{letter}_{\mathsf{free}}(S_{\mathsf{D}}, c) \\ \mathsf{true} \end{pmatrix} \ .$$

Step

Now we encode the rewrite relation as a formula. Recall that we consider grammars in Chomsky normal form without ϵ-rules, so any right-hand side of a rewrite rule has one or two letters. We represent concatenation of letters and words in grammars with the chop operator of MLSL. For a word w we define

$$\mathsf{word}(w) \equiv \begin{cases} \exists c_0.\, \mathsf{letter}_{\mathsf{free}}(\sigma_0, c_0) & \text{if } \#w = 1\,, \\ \exists c_0, c_1.\,(c_0 \neq c_1 \land \mathsf{letter}_{\mathsf{free}}(\sigma_0, c_0) \frown \mathsf{letter}_{\mathsf{free}}(\sigma_1, c_1)) & \text{if } \#w = 2\,, \end{cases}$$

where σ_j is the j-th letter of w.

To define that a nonterminal N is replaced by a word w, according to the rewrite rule $\mathcal{R}_{\mathsf{D}}$, we define

$$\mathsf{step}_{\mathsf{D}}(N, c) \equiv \begin{pmatrix} \mathsf{free} \\ \mathsf{free} \end{pmatrix} \frown \begin{pmatrix} \mathsf{letter}(N, c) \\ \bigvee_{w \in \mathcal{R}_{\mathsf{D}}^{N}} \mathsf{word}(w) \end{pmatrix} \frown \begin{pmatrix} \mathsf{free} \\ \mathsf{free} \end{pmatrix} .$$

This means that we replace a nonterminal on the lane below it with any of the words from the rewrite relation. As we use $\mathsf{letter_{free}}$ in the definition of word, we ensure that the replaced letter is horizontally larger than its replacement. Note that we subscripted the formula with D, because we only use it to encode derivation trees growing downward.

As *all* nonterminals should be replaced on the next lane we define

$$\mathsf{stepAll_D} \equiv \forall c. \bigwedge_{N \in \mathcal{N}_\mathsf{D}} \left(\langle \mathsf{letter_{free}}(N, c) \rangle \implies \langle \mathsf{step_D}(N, c) \rangle \right).$$

In the premise we test whether somewhere the car variable c is used as identifier for an occurrence of the nonterminal N. Intuitively, we bind the variable c to the occurrence matched in the premise. For this to work as intended we have to assume that c is used as identifier only for this one occurrence. We formalise this later. In the conclusion we use this c, bound to one specific occurrence of a nonterminal, to state that below this occurrence there should be a word as defined by the rewrite relation.

Side conditions

We do forbid overlapping reservations, cars with claims or two reservations. To exclude these we define

$$\begin{aligned}
\mathsf{mutex} &\equiv \neg \exists c, c'. \left(c \neq c' \land \langle \mathsf{re}(c) \land \mathsf{re}(c') \rangle \right) \ , \\
\mathsf{noTwoRes} &\equiv \neg \exists c. \begin{pmatrix} \langle \mathsf{re}(c) \rangle \\ \langle \mathsf{re}(c) \rangle \end{pmatrix} \ , \\
\mathsf{noClaims} &\equiv \neg \exists c. \langle \mathsf{cl}(c) \rangle \ , \\
\mathsf{asm} &\equiv \mathsf{mutex} \land \mathsf{noTwoRes} \land \mathsf{noClaims} \ .
\end{aligned}$$

As we want to encode derivations, all reservations should be part of the representation of the derivation. Thus, all reservations should belong to the representation of some letter, as ensured by

$$\begin{aligned}
\mathsf{allResInLetter} \equiv \forall c. (\langle \mathsf{re}(c) \rangle &\implies \\
& \exists c'. \bigvee_{\substack{\sigma \in \mathcal{N}_\mathsf{D} \cup \mathcal{N}_\mathsf{U} \cup \\ \mathcal{T}_\mathsf{D} \cup \mathcal{T}_\mathsf{U}}} \langle \mathsf{letter_{free}}(\sigma, c') \land (\mathsf{true} \frown \mathsf{re}(c) \frown \mathsf{true}) \rangle) \ ,
\end{aligned}$$

under the assumption that the formula $\mathsf{noTwoRes}$ holds (see also the complete formula $F(G_\mathsf{D}, G_\mathsf{U})$ on Page 41). We point out that $\mathsf{re}(c)$ in the premise and $\mathsf{re}(c)$ in the conclusion refer to the same reservation, as every car has only one reservation. Further, $\mathsf{letter}_\mathsf{free}(\sigma, c')$ is evaluated on the same view as $(\mathsf{true} \frown \mathsf{re}(c) \frown \mathsf{true})$. Thus, this formula ensures that every reservation is inside the representation of a letter.

We want to ensure that all representations of letters are part of a derivation, i.e. we want to forbid orphaned letters. For this we demand that for all letters not on the topmost lane, there is a nonterminal above them. The formula

$$\begin{aligned}\mathsf{letterNextToLetter}_\mathsf{D} \equiv \quad & \forall c. \bigwedge_{\sigma \in \mathcal{N}_\mathsf{D} \cup \mathcal{T}_\mathsf{D}} \left(\left\langle \begin{matrix} \langle \exists c'.\, \mathsf{free} \vee \mathsf{re}(c') \rangle \\ \mathsf{letter}_\mathsf{free}(\sigma, c) \end{matrix} \right\rangle \implies \right. \\ & \left. \exists c''. \bigvee_{N \in \mathcal{N}_\mathsf{D}} \left(\langle \mathsf{letter}_\mathsf{free}(N, c'') \rangle \wedge \left\langle \begin{matrix} \mathsf{letter}(N, c'') \\ \mathsf{true} \frown \mathsf{letter}_\mathsf{free}(\sigma, c) \frown \mathsf{true} \end{matrix} \right\rangle \right) \right)\end{aligned}$$

ensures this, where we use $\langle \exists c'.\, \mathsf{free} \vee \mathsf{re}(c') \rangle$ to match at least one lane, without regard for what is on the lane. Similar as in $\mathsf{stepAll}_\mathsf{D}$ we bind a car variable c to the occurrence of a letter σ in the premise of the implication. However, in the premise we additionally require that the letter is not located on the topmost lane. This is necessary because we do not want to demand that there is another nonterminal above the starting nonterminal. In the conclusion we bind a new variable c'' to the occurrence of a nonterminal N and require that the representation of N is above and strictly larger than the representation of σ. Note that we surround $\mathsf{letter}_\mathsf{free}(\sigma, c)$ with true to allow for other letters next to this one. Intuitively, $\mathsf{letterNextToLetter}_\mathsf{D}$ ensures that from any representation of a letter there is a sequence of vertically adjacent representations leading to S_D on the top lane. As $\mathsf{stepAll}_\mathsf{D}$ requires that all of these sequences obey the rewrite rules, a satisfying model represents a derivation.

Second grammar

The formulas so far can be used to ensure that the MLSL representation of a derivation from the grammar G_D starts at the top lane and grows downwards. Now we add formulas that demand that a derivation from G_U starts at the bottom and grows upwards.

For all formulas ϕ_D defined so far we create a formula ϕ_U by replacing indices D with U in ϕ_D and swapping the order of formulas in vertical chop operators.

For example we define

$$\mathsf{start}_{\mathsf{U}} \equiv \exists c.\begin{pmatrix}\mathsf{true}\\ \mathsf{letter}_{\mathsf{free}}(S_{\mathsf{U}}, c)\end{pmatrix},$$

$$\mathsf{step}_{\mathsf{U}}(N,c) \equiv \begin{pmatrix}\mathsf{free}\\ \mathsf{free}\end{pmatrix} \frown \begin{pmatrix}\bigvee_{w\in\mathcal{R}_{\mathsf{U}}^{N}} \mathsf{word}(w)\\ \mathsf{letter}(N,c)\end{pmatrix} \frown \begin{pmatrix}\mathsf{free}\\ \mathsf{free}\end{pmatrix},$$

and the other formulas are defined similarly.

The formula

$$\begin{array}{rl} & \mathsf{true}\\ \mathsf{freeLane} \equiv & \mathsf{free}\\ & \mathsf{true}\end{array}$$

requires that there is at least one lane without any reservations. This, together with $\mathsf{letterNextToLetter}_{\mathsf{D}}$, ensures that below this free lane there are no representations of letters from G_{D}. If there was such a letter, then from that letter the sequence of vertically adjacent representations would be interrupted by a free lane. Symmetrically, there is no letter from G_{U} above this free lane.

We say for two words w, w' that w is a *subsequence* of w' iff we can create w from w' by only removing letters from w'. We now define that the derived word of one grammar is a subsequence of the derived word of the other grammar. For $i \in \{\mathsf{D}, \mathsf{U}\}$ and $\tau \in \mathcal{T}$ we remind that we use π_i to map terminals to disjoint sets (cf. Page 35). Let c, c' be car variables, then we define

$$\phi(\tau, c, c') \equiv \left\langle \begin{pmatrix}\mathsf{free}\\ \mathsf{true}\\ \mathsf{free}\end{pmatrix} \frown \begin{pmatrix}\mathsf{letter}(\pi_{\mathsf{D}}(\tau), c)\\ \mathsf{true}\\ \mathsf{letter}(\pi_{\mathsf{U}}(\tau), c')\end{pmatrix} \frown \begin{pmatrix}\mathsf{free}\\ \mathsf{true}\\ \mathsf{free}\end{pmatrix} \right\rangle,$$

which requires that the representations of the terminal τ using the variables c, c' are horizontally aligned. The horizontal alignment is enforced by ensuring that the formulas $\mathsf{letter}(\pi_{\mathsf{D}}(\tau), c)$ and $\mathsf{letter}(\pi_{\mathsf{U}}(\tau), c')$ are evaluated on the same extension. This is done by evaluating the horizontal chops before the vertical chops. We define the subsequence relation in MLSL as

$$\begin{aligned}\mathsf{subseq}_{\mathsf{D}} \equiv \bigwedge_{\tau\in\mathcal{T}} \forall c.\, (\langle \mathsf{letter}_{\mathsf{free}}(\pi_{\mathsf{D}}(\tau), c)\rangle \implies \\ \exists c'.\, (\langle \mathsf{letter}_{\mathsf{free}}(\pi_{\mathsf{U}}(\tau), c')\rangle \wedge \phi(\tau, c, c'))) \enspace ,\\ \mathsf{subseq}_{\mathsf{U}} \equiv \bigwedge_{\tau\in\mathcal{T}} \forall c'.\, (\langle \mathsf{letter}_{\mathsf{free}}(\pi_{\mathsf{U}}(\tau), c')\rangle \implies \\ \exists c.\, (\langle \mathsf{letter}_{\mathsf{free}}(\pi_{\mathsf{D}}(\tau), c)\rangle \wedge \phi(\tau, c, c'))) \enspace .\end{aligned}$$

Note that in $\mathsf{subseq}_{\mathsf{D}}$ and $\mathsf{subseq}_{\mathsf{U}}$ the subformula $\phi(\tau, c, c')$ is the same. However, the car variable names and the subscripts D and U outside $\phi(\tau, c, c')$ are swapped. The formula $\mathsf{subseq}_{\mathsf{D}}$ ensures that for every terminal τ, when the downward derivation contains the downward encoding $\pi_{\mathsf{D}}(\tau)$ of τ, then the upwards derivation contains the upward encoding $\pi_{\mathsf{U}}(\tau)$ of τ, horizontally aligned. In other words, $\mathsf{subseq}_{\mathsf{D}}$ requires that each terminal from the downwards derivation has a horizontally aligned corresponding terminal from the upwards derivation. The horizontal alignment prevents that two downwards terminals share the same corresponding upwards terminal. The explanation for $\mathsf{subseq}_{\mathsf{U}}$ is symmetric. Because one word is a subsequence of the other word and vice versa, the two derived words are equal.

For the final formula we conjoin the downward and the upward formulas. The following formula has the property that it is lane-unboundedly satisfiable iff the intersection of the languages of G_{D} and G_{U} is empty. We prove this claim in the next section. We define

$$\begin{aligned}F(G_{\mathsf{D}}, G_{\mathsf{U}}) \equiv \bigwedge_{i\in\{\mathsf{D},\mathsf{U}\}} \mathsf{stepAll}_i \wedge \mathsf{start}_i \wedge \mathsf{letterNextToLetter}_i \wedge \mathsf{subseq}_i \\ \wedge\, \mathsf{freeLane} \wedge \mathsf{allResInLetter} \wedge \mathsf{asm} \enspace .\end{aligned}$$

3.1.2 Proving Undecidability

In this section we proof that for two context-free grammars G_{D} and G_{U} the MLSL formula $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ is lane-unboundedly satisfiable iff the intersection of the languages of G_{D} and G_{U} is empty. For this proof we use interval-labelled trees. For these trees we have the condition that the interval labelling should represent (or respect) the structure of the tree. That means that all interval labels are proper (Equation (3.1)), that the interval of a child should be strictly contained in the interval of its parent (Equation (3.2)) and that the interval of a right child is greater than the interval of a left child (Equation (3.3)). For this section we remind that $\mathbb{IR}$ is the set of closed real-valued intervals, $\underline{I}$ and $\overline{I}$ with $I \in \mathbb{IR}$ define the left and right border of the interval I and for two intervals $I, I' \in \mathbb{IR}$ we define with $I \subset I'$ that I is strictly in I' on both ends of the interval (cf. Page 13).

Definition 3.1.3. Let (Υ, g) with $g : \{0,1\}^* \to \mathbb{IR}$ be an interval-labelled binary tree. Then we say that the interval-labelling function respects the structure of the tree Υ iff the following conditions are satisfied:

$$\underline{g(x)} < \overline{g(x)} \ , \tag{3.1}$$

$$g(y) \subset g(\mathsf{front}\, y) \ , \tag{3.2}$$

$$\text{if } x0, x1 \in \Upsilon \text{ then } \underline{g(x0)} < \overline{g(x1)} \ , \tag{3.3}$$

where $x \in \Upsilon$ and $y \in \Upsilon \setminus \{\epsilon\}$. △

We get the following lemma for nodes that are not in a descendant/ascendant relationship: the first property states that the order induced by interval labelling is equal to the lexicographic order on nodes. The second property states that the interval labels of such nodes do not overlap.

Lemma 3.1.4. *Let (Υ, g) with $g : \Upsilon \to \mathbb{IR}$ be an interval-labelled binary tree such that g respects the tree structure. Then for all $x, x' \in \Upsilon$ we have*

$$x \prec_{\mathsf{l}} x' \text{ and } x \not\sqsubseteq x' \text{ implies } \overline{g(x)} < \underline{g(x')} \ ,$$
$$(x \not\sqsubseteq x' \text{ and } x' \not\sqsubseteq x) \text{ implies } g(x) \cap g(x') = \emptyset \ ,$$

where $\prec_{\mathsf{l}}$ is the lexicographic order and $\sqsubseteq$ is the prefix relation (cf. Page 7).

Proof. We start with the first property. Assume $x \prec_{\mathsf{l}} x'$ and $x \not\sqsubseteq x'$. Then there is a unique node $x'' \in \Upsilon$ that is the closest common ancestor of x and x', i.e. the node where the paths to x and x' diverge. For this node there are words $y, z \in \{0,1\}^*$ such that $x = x''0y$ and $x' = x''1z$. From Equation (3.3) we know that the interval label of $x''0$ is smaller than the interval label of $x''1$, i.e. $\underline{g(x''0)} < \overline{g(x''1)}$. From Equation (3.2) we know that the interval labels of x and x' are strictly contained in the intervals of $x''0$ and $x''1$, which implies $\overline{g(x)} < \underline{g(x')}$.

The second property follows from the first. □

For our proof we first need a lemma for binary trees. The lemma states that for two binary trees with the same number of leafs, the parent/child and sibling relationship can be encoded in an interval labelling function such that the j-th leaf in one tree has the same interval assigned as the j-th leaf in the other tree.

Note that the following lemma also holds when we only consider intervals formed by rational or natural numbers. However, we do not make use of this.

Lemma 3.1.5. *Let $(\Upsilon_\mathsf{D}, f_\mathsf{D})$ and $(\Upsilon_\mathsf{U}, f_\mathsf{U})$ be two binary trees with both trees having the same number $k \in \mathbb{N}_{\geq 1}$ of leafs. We can create two labelling functions $g_\mathsf{D} : \Upsilon_\mathsf{D} \to \mathbb{IR}$ and $g_\mathsf{U} : \Upsilon_\mathsf{U} \to \mathbb{IR}$ such that*

$$g_i \textit{ respects the structure of the tree (cf. Def. 3.1.3)} \qquad \text{and} \tag{3.4}$$

$$g_\mathsf{D}(y_j) = g_\mathsf{U}(z_j)\ , \tag{3.5}$$

where $i \in \{D, U\}$ and $y_j \in \Upsilon_\mathsf{D}$ (resp. $z_j \in \Upsilon_\mathsf{U}$) is the j-th leaf of its tree according to the lexicographic ordering.

Proof. To construct the functions $g_\mathsf{D}, g_\mathsf{U}$, we start to define them for the leafs. For this, let $r_0, \dots, r_{k-1}, t_0, \dots, t_{k-1} \in \mathbb{R}$ be values such that

$$r_0 < t_0 < r_1 < t_1 < \cdots < r_{k-1} < t_{k-1} \qquad \text{and} \tag{3.6}$$

$$r_{j+1} - t_j = 2 * \mathsf{max}(|\Upsilon_\mathsf{D}|, |\Upsilon_\mathsf{U}|) + 1 \qquad \text{and} \tag{3.7}$$

$$g_\mathsf{D}(y_{j'}) = g_\mathsf{U}(z_{j'}) = [r_{j'}, t_{j'}]\ , \tag{3.8}$$

with $j \in [0, k-2]$ and $j' \in [0, k-1]$. This means that the distance of the rightmost point of $g_i(x_j)$ to the leftmost point of $g_i(x_{j+1})$ is defined by the number of nodes in the trees.

Let $x \in \Upsilon_i$ be a node with children. Then we define

$$g_i(x) = \begin{cases} [r-1, t+1] & \text{if } \#(x.\mathsf{children}) = 1 \text{ with } g_i(x0) = [r, t] \\ [r-1, t'+1] & \text{else with } g_i(x0) = [r, t] \text{ and } g_i(x1) = [r', t']\ . \end{cases} \tag{3.9}$$

Essentially, the interval of a parent is the interval of the children increased by one in each direction. This ensures that the parent of a node x is a strict superset of x and a possible sibling of x. In Figure 3.2 a visualisation of g_D and g_U for the derivations from Example 2.1.6 on Page 10 is depicted.

Thus, so far it is clear that our construction satisfies Equations (3.1), (3.2) and (3.5). To prove that g_i respects the structure for the tree it remains to show that g_i satisfies Equation (3.3). We say that $x0$ is the left child of x and

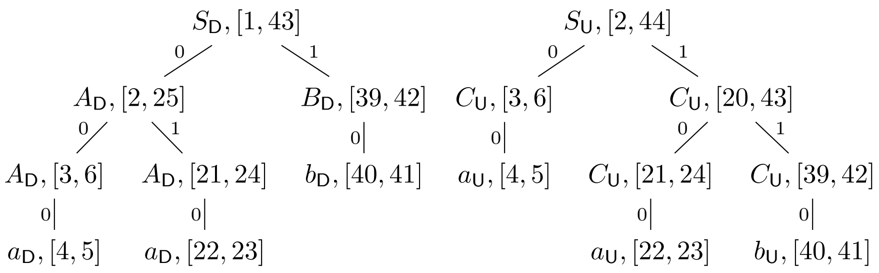

$S_{\mathsf{D}}, [1, 43]$
$A_{\mathsf{D}}, [2, 25]$
$B_{\mathsf{D}}, [39, 42]$
$A_{\mathsf{D}}, [3, 6]$
$A_{\mathsf{D}}, [21, 24]$
$b_{\mathsf{D}}, [40, 41]$
$a_{\mathsf{D}}, [4, 5]$
$a_{\mathsf{D}}, [22, 23]$
$S_{\mathsf{U}}, [2, 44]$
$C_{\mathsf{U}}, [3, 6]$
$C_{\mathsf{U}}, [20, 43]$
$a_{\mathsf{U}}, [4, 5]$
$C_{\mathsf{U}}, [21, 24]$
$C_{\mathsf{U}}, [39, 42]$
$a_{\mathsf{U}}, [22, 23]$
$b_{\mathsf{U}}, [40, 41]$
0
1

Figure 3.2: Two binary (in fact derivation) trees from Figure 2.1 on Page 11 extended with interval labelling as defined in Lemma 3.1.5.

that $x1$ is the right child of x. We define the leftmost (resp. rightmost) leaf of a node $x \in \Upsilon_i$ as

$$\begin{aligned}\mathsf{lf_L}(x) &= x_j \in \Upsilon_i \text{ such that } x \sqsubseteq x_j \text{ and } \forall j' < j.\, x_{j'} \not\sqsubseteq x \ , \\ \mathsf{lf_R}(x) &= x_j \in \Upsilon_i \text{ such that } x \sqsubseteq x_j \text{ and } \forall j' > j.\, x_{j'} \not\sqsubseteq x\end{aligned}$$

where $\sqsubseteq$ is the prefix relation for words (cf. Page 7) and x_j (resp. $x_{j'}$) is the j-th (resp. j'-th) leaf by the lexicographic order. We reformulate Equation (3.9) to show that Equation (3.3) holds. For any node $x \in \Upsilon_i$ the interval labelling is given by

$$g_i(x) = [\underline{g_i(\mathsf{lf_L}(x))} - \mathsf{dis}(x, \mathsf{lf_L}(x)), \overline{g_i(\mathsf{lf_R}(x))} + \mathsf{dis}(x, \mathsf{lf_R}(x))] \, . \tag{3.10}$$

This formalises that for a node x the left border of its interval label is given by the left border of the leftmost leaf that is a descendant of x and the distance from that leftmost leaf to x, and similarly for the right border.

We make a small detour and show that Equation (3.10) holds. For this we argue inductively, starting with the leafs. For a leaf x we have $\mathsf{lf_L}(x) = \mathsf{lf_R}(x) = x$, which means

$$\begin{aligned}&[\underline{g_i(\mathsf{lf_L}(x))} - \mathsf{dis}(x, \mathsf{lf_L}(x)), \overline{g_i(\mathsf{lf_R}(x))} + \mathsf{dis}(x, \mathsf{lf_R}(x))] \\ &= [\underline{g_i(x)} - 0, \overline{g_i(x)} + 0] = g_i(x) \ .\end{aligned}$$

The equality holds for the base case. Now, let x be a node with children, and let us first consider the case that x has two children. We use the fact that

$\mathsf{lf}_{\mathsf{L}}(x0) = \mathsf{lf}_{\mathsf{L}}(x)$ and $\mathsf{lf}_{\mathsf{R}}(x1) = \mathsf{lf}_{\mathsf{R}}(x)$. Then

$$\begin{aligned} g_i(x) &\overset{\text{def}}{=} [\underline{g_i(x0)} - 1, \overline{g_i(x1)} + 1] \\ &\overset{\text{IH}}{=} [\underline{g_i(\mathsf{lf}_{\mathsf{L}}(x0))} - \mathsf{dis}(x0, \mathsf{lf}_{\mathsf{L}}(x0)) - 1, \overline{g_i(\mathsf{lf}_{\mathsf{R}}(x1))} + \mathsf{dis}(x1, \mathsf{lf}_{\mathsf{R}}(x1)) + 1] \\ &= [\underline{g_i(\mathsf{lf}_{\mathsf{L}}(x))} - \mathsf{dis}(x0, \mathsf{lf}_{\mathsf{L}}(x)) - 1, \overline{g_i(\mathsf{lf}_{\mathsf{R}}(x))} + \mathsf{dis}(x1, \mathsf{lf}_{\mathsf{R}}(x)) + 1] \\ &= [\underline{g_i(\mathsf{lf}_{\mathsf{L}}(x))} - \mathsf{dis}(x, \mathsf{lf}_{\mathsf{L}}(x)), \overline{g_i(\mathsf{lf}_{\mathsf{R}}(x))} + \mathsf{dis}(x, \mathsf{lf}_{\mathsf{R}}(x))] \ . \end{aligned}$$

Thus, the equality holds for nodes with two children. For nodes with a single child similar reasoning applies.

We return from our detour. For the following equations we point out that the distance of a child from its parent is less or equal the number of nodes in the tree, i.e.

$$\forall x, x' \in \Upsilon.\, x \sqsubseteq x' \implies \mathsf{dis}(x, x') \leq |\Upsilon| \,. \tag{3.11}$$

Further, note that if x has two children and $\mathsf{lf}_{\mathsf{R}}(x0) = x_j$ then $\mathsf{lf}_{\mathsf{L}}(x1) = x_{j+1}$, i.e. for a node x the rightmost leaf of the left child of x is adjacent to the leftmost leaf of the right child of x. Given

$$\underline{g_i(x0)} < \overline{g_i(x1)} \iff \underline{g_i(x1)} - \overline{g_i(x0)} > 0$$

we show that our construction satisfies Equation (3.3):

$$\begin{array}{lll} & \underline{g_i(x1)} - \overline{g_i(x0)} & \text{with (3.10)} \\ = & \underline{g_i(\mathsf{lf}_{\mathsf{L}}(x1))} - \mathsf{dis}(x1, \mathsf{lf}_{\mathsf{L}}(x1)) - (\overline{g_i(\mathsf{lf}_{\mathsf{R}}(x0))} + \mathsf{dis}(x0, \mathsf{lf}_{\mathsf{R}}(x0))) & \\ = & \underline{g_i(\mathsf{lf}_{\mathsf{L}}(x1))} - \overline{g_i(\mathsf{lf}_{\mathsf{R}}(x0))} - \mathsf{dis}(x1, \mathsf{lf}_{\mathsf{L}}(x1)) - \mathsf{dis}(x0, \mathsf{lf}_{\mathsf{R}}(x0)) & \text{for some } j \\ = & \underline{g_i(x_{j+1})} - \overline{g_i(x_j)} - \mathsf{dis}(x1, \mathsf{lf}_{\mathsf{L}}(x1)) - \mathsf{dis}(x0, \mathsf{lf}_{\mathsf{R}}(x0)) & \text{with (3.7)} \\ = & 2 * \mathsf{max}(|\Upsilon_{\mathsf{D}}|, |\Upsilon_{\mathsf{U}}|) + 1 - \mathsf{dis}(x1, \mathsf{lf}_{\mathsf{L}}(x1)) - \mathsf{dis}(x0, \mathsf{lf}_{\mathsf{R}}(x0)) & \text{with (3.11)} \\ \geq & 2 * \mathsf{max}(|\Upsilon_{\mathsf{D}}|, |\Upsilon_{\mathsf{U}}|) + 1 - 2 * \mathsf{max}(|\Upsilon_{\mathsf{D}}|, |\Upsilon_{\mathsf{U}}|) & \\ > & 0 & \square \end{array}$$

Now we can proceed to the actual reduction of the lane-unbounded satisfiability problem of MLSL to the emptiness problem of the intersection of two context-free languages. First we give a definition used in both directions of the proof. We remind that we used $\pi_i : \mathcal{T} \to \mathcal{T}_i$ with $i \in \{\mathsf{D}, \mathsf{U}\}$ to distinguish terminals from the downward and the upward derivation (see Page 35). Let $\pi'_i : \mathcal{N}_i \cup \mathcal{T} \to \mathcal{T}_i \cup \mathcal{N}_i$ be an extension of π_i that maps nonterminals to their identity. We define π'_i as

$$\pi'_i(\sigma) = \begin{cases} \pi_i(\sigma) & \text{if } \sigma \in \mathcal{T} \ , \\ \sigma & \text{otherwise} \ . \end{cases}$$

Lemma 3.1.6. *Let $G_\mathsf{D}, G_\mathsf{U}$ be two context-free grammars in Chomsky normal form without ϵ-rules such that $\mathcal{L}(G_\mathsf{D}) \cap \mathcal{L}(G_\mathsf{U}) \neq \emptyset$. Then $F(G_\mathsf{D}, G_\mathsf{U})$ from Page 41 is lane-unboundedly satisfiable.*

Proof. Note that whenever we talk of derivation trees in this proof, we assume that the root is labelled by the starting nonterminal of the grammar.

Let $G_i = (\mathcal{N}_i, \mathcal{T}, \mathcal{R}_i, S_i)$ with $i \in \{\mathsf{D}, \mathsf{U}\}$ and let us assume w.l.o.g. that $\mathcal{N}_\mathsf{D}$ and $\mathcal{N}_\mathsf{U}$ are disjoint. As $\mathcal{L}(G_\mathsf{D}) \cap \mathcal{L}(G_\mathsf{U}) \neq \emptyset$ there is a word $w \in \mathcal{L}(G_\mathsf{D}) \cap \mathcal{L}(G_\mathsf{U})$ and let (Υ_i, f_i) be a derivation tree of w from G_i. Let g_i with $i \in \{\mathsf{D}, \mathsf{U}\}$ be the two interval labelling functions we can create for Υ_i according to Lemma 3.1.5. We show that $F(G_\mathsf{D}, G_\mathsf{U})$ is satisfiable by constructing an MLSL model M_all such that $M_\text{all} \models F(G_\mathsf{D}, G_\mathsf{U})$ (for $F(G_\mathsf{D}, G_\mathsf{U})$ see Page 41).

We can define the interval of lanes as $L_\text{all} = [0, l_\text{all}]$ with $l_\text{all} = \mathsf{height}(\Upsilon_\mathsf{D}) + \mathsf{height}(\Upsilon_\mathsf{U}) + 2$. The $+2$ is necessary because the root of a tree does not count towards its height but still is represented on its own lane. We define the extension of the view as $X_\text{all} = [r - \delta, r' + \delta']$, where $r = \mathsf{min}(g_\mathsf{D}(\epsilon) \cup g_\mathsf{U}(\epsilon))$, $\delta, \delta' \in \mathbb{R}_{>0}$ and $r' = \mathsf{max}(g_\mathsf{D}(\epsilon) \cup g_\mathsf{U}(\epsilon))$. Now we assign to all nodes $x \in \Upsilon_i$ an MLSL representation using reservations. Let $k = \lambda(\pi'_i(f_i(x)))$ be the number of cars needed to represent the label of x in MLSL, where λ assigns the number of cars used to represent a letter (cf. Page 35). Further, let $C_0^{i,x}, \ldots, C_{k-1}^{i,x}$ be different cars for which we did not assign res_all, pos_all and Ω_all yet and let $s_0, \ldots, s_k \in \mathbb{R}$ be real values such that $\underline{g_i(x)} = s_0^{i,x} < s_1^{i,x} < \cdots < s_k^{i,x} = \overline{g_i(x)}$. Then we define

$$\mathsf{res}_\text{all}(C_j^{i,x}) = \begin{cases} \{l_\text{all} - \#x\} & \text{if } x \in \Upsilon_\mathsf{D} \\ \{\#x\} & \text{if } x \in \Upsilon_\mathsf{U} \ , \end{cases}$$
$$\mathsf{pos}_\text{all}(C_j^{i,x}) = s_j^{i,x} \ ,$$
$$\Omega_\text{all}(C_j^{i,x}, TS_\text{all}) = s_{j+1}^{i,x} - s_j^{i,x} \ ,$$

for $j \in [0, k-1]$. This fills the interval $g_i(x)$ with k nonoverlapping reservations without free space in between. As an example, $C_1^{\mathsf{D},\epsilon}$ is the second car in the representation of the letter $\pi_\mathsf{D}(f_\mathsf{D}(\epsilon))$ of the node ϵ in the tree Υ_D. All cars which we do not use to represent nodes in the trees have reservations outside the view. The idea behind the definition of res is that Υ_{D} is represented in the upper part of the view growing downward, and Υ_{U} is represented in the lower part of the view growing upward. The number of lanes is chosen large enough so the representations do not interfere with each other. If we create M_{all} for the derivations from Figure 3.2 the resulting model looks like the model shown in Figure 3.1.

By construction of our interval labels, for any two nodes with equal depth their interval labels are disjoint. Here we assume that in between the interval labels of two nodes of the same level there is at least a distance of one. This can be easily ensured by the interval labelling functions. Formally, for $i \in \{\mathsf{D}, \mathsf{U}\}$ and all nodes $x, x' \in \Upsilon_i$ such that $x \neq x'$ we assume

$$\#x = \#x' \implies \|[\mathsf{min}(g_i(x) \cup g_i(x')), \mathsf{max}(g_i(x) \cup g_i(x'))] \setminus (g_i(x) \cup g_i(x'))\| \geq 1 \ ,$$

where $\|\cdot\|$ is the measure on intervals. The above implies for all $x \in \Upsilon_i$ we have

$$M \models \mathsf{letter}_{\mathsf{free}}(g_i(x), c) \ ,$$

where $M = (TS_{\text{all}}, \Omega_{\text{all}}, V, \nu)$, $V = ([l, l], X, E)$, $l = l_{\text{all}} - \#x$ for $i = \mathsf{D}$ (resp. $l = \#x$ for $i = \mathsf{U}$), $X = [\underline{g_i(x)} - 1, \overline{g_i(x)} + 1]$ and $\nu = \nu_{\text{all}} \oplus \{c \mapsto C_0^{i,x}\}$.

We show that $M_{\text{all}} \models F(G_{\mathsf{D}}, G_{\mathsf{U}})$. The model M_{all} satisfies $\mathsf{start}_{\mathsf{D}}$ (see Page 37) because for $M = (TS_{\text{all}}, \Omega_{\text{all}}, V, \nu)$ with $V = ([l_{\text{all}}, l_{\text{all}}], X_{\text{all}}, E)$ and $\nu = \nu_{\text{all}} \oplus \{c \mapsto C_0^{\mathsf{D},\epsilon}\}$ we have $M \models \mathsf{letter}_{\mathsf{free}}(S_{\mathsf{D}}, C)$.

Furthermore, $\mathsf{freeLane}$ is satisfied because the lane $\mathsf{height}(\Upsilon_{\mathsf{U}}) + 1$ (or equivalently $l_{\text{all}} - \mathsf{height}(\Upsilon_{\mathsf{D}}) - 1$) does not contain any reservations.

First we consider the formula asm and its subformulas mutex, $\mathsf{noTwoRes}$ and $\mathsf{noClaims}$ (see Page 38). The formula mutex holds because the properties of the interval labelling functions g_i ensure that intervals of the same depth do not overlap. As we placed reservations only within these intervals and as we represent labels of nodes of different depth on different lanes we do not have overlapping reservations. The formula $\mathsf{noTwoRes}$ is satisfied because we placed for all cars at most one reservation in the view. Similarly, we did not place any claims, which means that $\mathsf{noClaims}$ is satisfied.

For the formula $\mathsf{allResInLetter}$ (see Page 38) consider a subview V of V_{all} that satisfies the premise of the implication, which means that a view is filled by a reservation of some car C and that the valuation ν maps c to C. That is, for

$V = ([l, l'], X, E)$ and $C = \nu(c)$ we have $X \subseteq \mathsf{se}(C, TS_{\mathrm{all}}, \Omega_{\mathrm{all}})$, $l \in \mathsf{res}_{\mathrm{all}}(C)$ and $l = l'$. As we only placed cars representing nodes we know that for $i \in \{\mathsf{D}, \mathsf{U}\}$ there is $x \in \Upsilon_i$ and $j \in \{0, \dots, \lambda(\pi'_i(f_i(x))) - 1\}$ such that $C = C_j^{i,x}$. First, let us consider the case $i = \mathsf{D}$. Then $l = l_{\mathrm{all}} - \#x$ and $X \subseteq g_{\mathsf{D}}(x)$. Let $M = (TS_{\mathrm{all}}, \Omega_{\mathrm{all}}, V', \nu')$ with $\nu' = \nu \oplus \{c' \mapsto C_0^{\mathsf{D},x}\}$, where $C_0^{\mathsf{D},x}$ and $C_j^{\mathsf{D},x}$ might be the same car, $V' = ([l_{\mathrm{all}} - \#x, l_{\mathrm{all}} - \#x], X, E)$, $X = ([\underline{g_{\mathsf{D}}(x)} - 1, \overline{g_{\mathsf{D}}(x)} + 1]$. Then we have $M' \models \mathsf{letter}_{\mathsf{free}}(\pi'_{\mathsf{D}}(f_{\mathsf{D}}(x)), c')$. As $l = l_{\mathrm{all}} - \#x$ and $X \subseteq g_{\mathsf{D}}(x)$ we know that the same view also satisfies $\mathsf{true} \frown \mathsf{re}(c) \frown \mathsf{true}$. For the case $i = \mathsf{U}$ the argument is symmetric. Thus, in the rest of the proof we shall only talk about representations of letters.

We proceed to $\mathsf{letterNextToLetter}_{\mathsf{D}}$ from Page 39. Let us assume that the premise of $\mathsf{letterNextToLetter}_{\mathsf{D}}$ is satisfied. That means we have a subview $V = ([l, l'], X, E)$, a valuation ν and a letter $\sigma \in \mathcal{T}_{\mathsf{D}} \cup \mathcal{N}_{\mathsf{D}}$ such that for $M = (TS_{\mathrm{all}}, \Omega_{\mathrm{all}}, V, \nu)$ we have $M \models \begin{pmatrix} \langle \exists c'.\, \mathsf{free} \vee \mathsf{re}(c') \rangle \\ \mathsf{letter}_{\mathsf{free}}(\sigma, c) \end{pmatrix}$. This implies that V contains at least two lanes. Further, we point out that $l < l_{\mathrm{all}}$. By construction we know that $\nu(c) = C_0^{\mathsf{D},x}$ for some $x \in \Upsilon_{\mathsf{D}}$. As $l < l_{\mathrm{all}}$ it follows that x is not the root of Υ_{D}, i.e. $y = \mathsf{front}(x)$ is defined and $y \in \Upsilon_{\mathsf{D}}$. As $(\Upsilon_{\mathsf{D}}, f_{\mathsf{D}})$ is a derivation tree all internal nodes are labelled by nonterminals. Thus, $f_{\mathsf{D}}(y) \in \mathcal{N}_{\mathsf{D}}$.

We show that the conclusion of $\mathsf{letterNextToLetter}_{\mathsf{D}}$, which consists of two parts, is satisfied. For the first part of we create $M' = (TS_{\mathrm{all}}, \Omega_{\mathrm{all}}, V', \nu')$ with $\nu' = \nu \oplus \{c'' \mapsto C_0^{\mathsf{D},y}\}$, $V' = ([l'', l''], X', E)$, $X' = [\underline{g_{\mathsf{D}}(y)} - 1, \overline{g_{\mathsf{D}}(y)} + 1]$ and $l'' = l_{\mathrm{all}} - \#y$ and conclude $M' \models \mathsf{letter}_{\mathsf{free}}(\pi'_{\mathsf{D}}(f_{\mathsf{D}}(y)), c'')$. For the second part we create $M'' = (TS_{\mathrm{all}}, \Omega_{\mathrm{all}}, V'', \nu')$ with $V'' = ([l, l''], g_{\mathsf{D}}(y), E)$. Now, from $g_{\mathsf{D}}(x) \subset g_{\mathsf{D}}(y)$ we conclude $M'' \models \begin{pmatrix} \mathsf{letter}(N, c'') \\ \mathsf{true} \frown \mathsf{letter}_{\mathsf{free}}(\sigma, c) \frown \mathsf{true} \end{pmatrix}$. The argumentation for $\mathsf{letterNextToLetter}_{\mathsf{U}}$ is symmetric.

We continue with $\mathsf{stepAll}_{\mathsf{D}}$ from Page 38. Assume that the premise of $\mathsf{stepAll}_{\mathsf{D}}$ is satisfied, i.e. we have a subview $V = (L, X, E)$ of V_{all}, a valuation ν and a nonterminal N s.t. for $M = (TS_{\mathrm{all}}, \Omega_{\mathrm{all}}, V, \nu)$ we have $M \models \mathsf{letter}_{\mathsf{free}}(N, c)$. This means that V contains a single lane; let this lane be l. By construction of M_{all} we know that there is $x \in \Upsilon_{\mathsf{D}}$ such that $\nu(c) = C_0^{\mathsf{D},x}$ and $g_{\mathsf{D}}(x) \subset X$. Further, as $(\Upsilon_{\mathsf{D}}, f_{\mathsf{D}})$ is a derivation tree and as only internal nodes are labelled by nonterminals, we know that x is not a leaf.

We show that the conclusion of $\mathsf{stepAll}_{\mathsf{D}}$ is satisfied. We first consider the case that x has two children $x0, x1$. For $j \in \{0, 1\}$ let $M_j = (TS_{\mathrm{all}}, \Omega_{\mathrm{all}}, V_j, \nu_j)$

with $V_0 = ([l-1, l-1], [\underline{X}, r], E)$, $V_1 = ([l-1, l-1], [r, \overline{X}], E)$, $r = \overline{g_{\mathsf{D}}(x0)} + \frac{1}{2} * (\underline{g_{\mathsf{D}}(x1)} - \overline{g_{\mathsf{D}}(x0)})$ and $\nu_j = \nu \oplus \{c_j \mapsto C_0^{\mathsf{D},xj}\}$. This means that the extensions of V_0 and V_1 meet at their endpoints. Furthermore, the extension of both views is slightly larger than the interval labelling because $g_{\mathsf{D}}(xj) \subset g_{\mathsf{D}}(x)$. By construction we know $M_j \models \mathsf{letter}_{\mathsf{free}}(\pi'_{\mathsf{D}}(f_{\mathsf{D}}(xj)), c_j)$. This implies that the view $V' = V \ominus (V_0 \oplus V_1)$ satisfies the formula $\mathsf{step}_{\mathsf{D}}(N, c)$, i.e. for $M' = (TS_{\text{all}}, \Omega_{\text{all}}, V', \nu)$ we have $M' \models \mathsf{step}_{\mathsf{D}}(N, c)$. The case that x has one child is analogous, albeit easier. We have shown that M_{all} satisfies $\mathsf{stepAll}_{\mathsf{D}}$. For $\mathsf{stepAll}_{\mathsf{U}}$ the reasoning is symmetric.

We remind that we relabelled the terminals $\mathcal{T}$ of both grammars to disjoint sets $\mathcal{T}_{\mathsf{D}}, \mathcal{T}_{\mathsf{U}}$ to treat them independently in the logic. As we now relate representations of terminals from $\mathcal{T}_{\mathsf{D}}$ with representations from $\mathcal{T}_{\mathsf{U}}$ we use the renaming functions π_i and their inverses π_i^{-1}, instead of the primed versions.

Assume that the premise of $\mathsf{subseq}_{\mathsf{D}}$ (see Page 41) is satisfied, i.e. we have a terminal $\tau \in \mathcal{T}$ and a valuation ν such that for $M = (TS_{\text{all}}, \Omega_{\text{all}}, V, \nu)$ with $V = ([l, l], X, E)$ and $\nu = \nu_{\text{all}} \oplus \{c \mapsto C_0^{\mathsf{D},x}\}$ for some $x \in \Upsilon_{\mathsf{D}}$ we have $M \models \mathsf{letter}_{\mathsf{free}}(\pi_{\mathsf{D}}(\tau), c)$. As x is labelled by a terminal we know that x is a leaf. Let x be the j-th leaf of Υ_{D} as defined by the lexicographic order on nodes. As both derivation trees derive the same word we know that for the j-th leaf y of Υ_{U} we have $\pi_{\mathsf{U}}^{-1}(f_{\mathsf{U}}(y)) = \tau$. From Lemma 3.1.5 we know $g_{\mathsf{D}}(x) = g_{\mathsf{U}}(y)$. Thus, by construction, for $M' = (TS_{\text{all}}, \Omega_{\text{all}}, V', \nu')$ with $V' = ([l', l'], X', E)$, $l' = \#y$, $X' = [\underline{g_{\mathsf{D}}(y)} - 1, \overline{g_{\mathsf{D}}(y)} + 1]$ and $\nu' = \nu \oplus \{c' \mapsto C_0^{\mathsf{U},y}\}$ we have $M' \models \mathsf{letter}_{\mathsf{free}}(\pi_{\mathsf{U}}(\tau), c')$, i.e. the first part of the conclusion of $\mathsf{subseq}_{\mathsf{D}}$ is satisfied.

For the second part we know $g_{\mathsf{D}}(x) = g_{\mathsf{U}}(y)$. Thus, for the model $M'' \models (TS_{\text{all}}, \Omega_{\text{all}}, V'', \nu')$ with $V'' = ([l', l], X', E)$ we have

$$M'' \models \begin{pmatrix} \mathsf{free} \\ \mathsf{true} \\ \mathsf{free} \end{pmatrix} \frown \begin{pmatrix} \mathsf{letter}(\tau_{\mathsf{D}}, c) \\ \mathsf{true} \\ \mathsf{letter}(\tau_{\mathsf{U}}, c') \end{pmatrix} \frown \begin{pmatrix} \mathsf{free} \\ \mathsf{true} \\ \mathsf{free} \end{pmatrix}.$$

Hence, the second part of $\mathsf{subseq}_{\mathsf{D}}$ also is satisfied, which means that M_{all} satisfies $\mathsf{subseq}_{\mathsf{D}}$.

The reasoning is symmetric for $\mathsf{subseq}_{\mathsf{U}}$.

We have shown that M_{all} satisfies all subformulas of $F(G_{\mathsf{D}}, G_{\mathsf{U}})$. Thus $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ is lane-unboundedly satisfiable. □

We show the other direction of the reduction. That is, we show that if for two context-free grammars G_{D}, G_{U} the MLSL formula $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ from Page 41 is lane-unboundedly satisfiable, then the intersection of the languages of the grammars is nonempty.

Lemma 3.1.7. *Let $G_{\mathsf{D}}, G_{\mathsf{U}}$ be two context-free grammars in Chomsky normal form without ϵ-rules and let $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ (see Page 41) be lane-unboundedly satisfiable. Then $\mathcal{L}(G_{\mathsf{D}}) \cap \mathcal{L}(G_{\mathsf{U}}) \neq \emptyset$.*

Proof. In this direction we first reason about the structure of a satisfying model. Then we show that we can create a derivation tree of G_{D} from the model and a derivation tree of G_{U}, which works symmetrical to the case for G_{D}. At the end we show that both trees represent the same word.

For $i \in \{\mathsf{D}, \mathsf{U}\}$ let $G_i = (\mathcal{N}_i, \mathcal{T}, \mathcal{R}_i, S_i)$, where we assume w.l.o.g. that $\mathcal{N}_{\mathsf{D}}$ and $\mathcal{N}_{\mathsf{U}}$ are disjoint. We remind that we renamed the terminals $\mathcal{T}$ to disjoint sets $\mathcal{T}_{\mathsf{D}}$ and $\mathcal{T}_{\mathsf{U}}$ using π'_{D} and π'_{U} (see Page 46). As the formula $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ is lane-unboundedly satisfiable, there is an MLSL model $M_{\text{all}} = (TS_{\text{all}}, \Omega_{\text{all}}, V_{\text{all}}, \nu_{\text{all}})$ with $V_{\text{all}} = ([l_{\text{all}}, l'_{\text{all}}], [r_{\text{all}}, r'_{\text{all}}], E)$ and $M_{\text{all}} \models F(G_{\mathsf{D}}, G_{\mathsf{U}})$. Note that as all references to car variables are quantified ν_{all} does not contain any useful information. Further, as we do not use ego, the view owner E also is not helpful.

For the formulas mutex, $\mathsf{noTwoRes}$, $\mathsf{noClaims}$, $\mathsf{allResInLetter}$ we refer to Page 38 The formula mutex ensures that V_{all} contains no overlapping reservations, $\mathsf{noTwoRes}$ ensures that every car has at most one reservation inside the view and $\mathsf{noClaims}$ ensures that there are no claims. The formula $\mathsf{allResInLetter}$ ensures that all reservations are part of the representation of a letter, and thus we reason about representations of letters. For an MLSL model $M = (TS, \Omega, V, \nu)$ with $V = (L, X, E)$, a letter $\sigma \in \mathcal{T}_{\mathsf{D}} \cup \mathcal{N}_{\mathsf{D}} \cup \mathcal{T}_{\mathsf{U}} \cup \mathcal{N}_{\mathsf{U}}$ and a view $V' = (L', X', E)$ with $L' = [l, l], l \in L$, $X' = [r, r']$, $X' \subset X$ and $\|X'\| > 0$ we define

$$\begin{aligned}
\mathsf{repr}(M, \sigma, V') \text{ iff } & (TS, \Omega, ([l, l], [r, r'], E), \nu) \models \exists c.\, \mathsf{letter}(\sigma, c) && \text{and} \\
& \exists \delta \in \mathbb{R}_{>0}.\, (TS, \Omega, ([l, l], [r - \delta, r], E), \nu) \models \mathsf{free} && \text{and} \\
& \exists \delta \in \mathbb{R}_{>0}.\, (TS, \Omega, ([l, l], [r', r' + \delta], E), \nu) \models \mathsf{free},
\end{aligned}$$

which means that on lane l the extension X' contains $\lambda(\sigma)$ (see Page 35 for λ) successive reservations and some free space to the left and to the right.

In the following we argue for formulas subscripted with D, however symmetric arguments apply for formulas subscripted with U. The formula $\mathsf{start}_{\mathsf{D}}$ (see Page 37) ensures that S_{D} is the only letter with a representation on the topmost lane. Let this representation be given by the view V_0.

We show how to construct the derivations, and later we show that they derive the same word. Let $\mathbb{V}_{\mathrm{all}}$ be the set of all subviews V of V_{all}, which satisfy $\mathsf{repr}(M_{\mathrm{all}}, \sigma, V)$ for some σ. We define a function $h_{\mathsf{D}} : \mathbb{V}_{\mathrm{all}} \to \{0,1\}^*$ recursively that assigns every view representing a letter a path over $\{0,1\}^*$. Then Υ_{D} is the image of h_{D}. We start with

$$h_{\mathsf{D}}(V_0) = \epsilon \ .$$

Let $\sigma \in \mathcal{T}_{\mathsf{D}} \cup \mathcal{N}_{\mathsf{D}}$, $l \in [l_{\mathrm{all}}, l'_{\mathrm{all}} - 1]$, $X \in \mathbb{IR}$ such that $\mathsf{repr}(M_{\mathrm{all}}, \sigma, ([l,l], X, E))$ holds. Then from $\mathsf{letterNextToLetter_D}$ (see Page 39) we know that there is a non-terminal $N \in \mathcal{N}_{\mathsf{D}}$ and an interval X_1 with $X \subset X_1$ such that $\mathsf{repr}(M_{\mathrm{all}}, N, ([l+1, l+1], X_1, E))$ holds. For $V = ([l,l], X, E)$ and $V_1 = ([l+1, l+1], X_1, E)$ we define

$$h_{\mathsf{D}}(V) = h_{\mathsf{D}}(V_1) \cdot v \ ,$$

where $\cdot$ is the operator for string concatenation and

$$v = \begin{cases} 1 & \text{if } \exists \sigma' \in \mathcal{T}_{\mathsf{D}} \cup \mathcal{N}_{\mathsf{D}} \text{ and } \exists X' \subset X_1 \text{ such that } \overline{X'} < \underline{X} \\ & \quad \text{and } \mathsf{repr}(M_{\mathrm{all}}, \sigma', ([l,l], X', E)) \text{ holds} \ , \\ 0 & \text{otherwise}. \end{cases}$$

This means that if there is another representation on the same lane left of V and that other representation also is enclosed by the same representation on the lane above, then V becomes child one, otherwise child zero. Note that $\mathsf{stepAll_D}$ from Page 38 ensures that there is at most one such other representation.

Let the image of h_{D} be Υ_{D}, i.e. $\Upsilon_{\mathsf{D}} = \mathsf{ran}\, h_{\mathsf{D}}$. Additionally, we define the labelling functions $f_{\mathsf{D}} \colon \Upsilon_{\mathsf{D}} \to \mathcal{T} \cup \mathcal{N}_{\mathsf{D}}$, which assigns to every node a letter and $g_{\mathsf{D}} \colon \Upsilon_{\mathsf{D}} \to \mathbb{IR}$, which assigns to every node an interval. Note that we defined h_{D} for all views that represent some letter. Let $V = (L, X, E)$ and $\sigma \in \mathcal{T}_{\mathsf{D}} \cup \mathcal{N}_{\mathsf{D}}$ be arbitrary such that $\mathsf{repr}(M_{\mathrm{all}}, \sigma, V)$ holds. Then we define $f_{\mathsf{D}}(h_{\mathsf{D}}(V)) = \tilde{\pi}'_{\mathsf{D}}(\sigma)$ and $g_{\mathsf{D}}(h_{\mathsf{D}}(V)) = X$, where $\tilde{\pi}'_{\mathsf{D}} : \mathcal{T}_{\mathsf{D}} \cup \mathcal{N}_{\mathsf{D}} \to \mathcal{T} \cup \mathcal{N}_{\mathsf{D}}$ is the inverse of π'_{D} from Page 46.

Let $x \in \Upsilon_{\mathsf{D}}$ and assume that $f_{\mathsf{D}}(x) \in \mathcal{N}_{\mathsf{D}}$. Then $\mathsf{stepAll_D}$, together with the fact that G_{D} and G_{U} are in CNF$^{-\epsilon}$, ensure either

- there is a node $x0 \in \Upsilon_{\mathsf{D}}$ such that $(f_{\mathsf{D}}(x), f_{\mathsf{D}}(x0)) \in \mathcal{R}_{\mathsf{D}}$ and $f_{\mathsf{D}}(x0) \in \mathcal{T}$, or
- there are two nodes $x0, x1 \in \Upsilon_{\mathsf{D}}$ such that $(f_{\mathsf{D}}(x), \langle f_{\mathsf{D}}(x0), f_{\mathsf{D}}(x1)\rangle) \in \mathcal{R}_{\mathsf{D}}$ and $f_{\mathsf{D}}(x0), f_{\mathsf{D}}(x1) \in \mathcal{N}_{\mathsf{D}}$.

Further, if $f_{\mathsf{D}}(x) \in \mathcal{T}$, then the formula $\mathsf{letterNextToLetter}_{\mathsf{D}}$ ensures that x has no children in Υ_{D} because $\mathsf{letterNextToLetter}_{\mathsf{D}}$ requires that above the MLSL representation of a letter there is the representation of a *nonterminal*. Thus, $(\Upsilon_{\mathsf{D}}, f_{\mathsf{D}})$ is a derivation tree of G_{D} with a root labelled by the starting nonterminal. In a symmetric manner we can build functions h_{U}, f_{U}, g_{U} and a tree Υ_{U}.

Let $i \in \{\mathsf{D}, \mathsf{U}\}$ and let $\bar{i} \in \{\mathsf{D}, \mathsf{U}\} \setminus \{i\}$. From $\mathsf{letterNextToLetter}_i$ and $\mathsf{stepAll}_i$ we know that g_i respects the tree structure of Υ_i (cf. Page 42). Thus, g_i defines a total order on the leafs of Υ_i. Let $\tau \in \mathcal{T}$, then from subseq_i we know that for every node $x \in \Upsilon_i$ with $f_i(x) = \tau$ and $g_i(x) = X$ there is a node $x' \in \Upsilon_{\bar{i}}$ such that $f_{\bar{i}}(x') = \tau$ and $g_{\bar{i}}(x') = X$. Hence, subseq_i ensures that the word derived by (Υ_i, f_i) is a subsequence of the word derived by $(\Upsilon_{\bar{i}}, f_{\bar{i}})$, and vice versa for $\bar{i}$. Thus, the two derived words are equal, which implies that $\mathcal{L}(G_{\mathsf{D}}) \cap \mathcal{L}(G_{\mathsf{U}}) \neq \emptyset$. $\square$

As the language emptiness of the intersection two context-free grammars without ϵ-rules is undecidable (see Lemma 2.1.5), we can use the previous two lemmas to obtain our first theorem.

Theorem 3.1.8. *The lane-unbounded satisfiability problem of MLSL is undecidable.*

We point out that if we restrict the horizontal domain to be discrete, then the lane-unbounded satisfiability problem remains undecidable. To see this, we point out that in Lemma 3.1.6 we can restrict the interval labels of the leafs to discrete intervals of the size greater equal $\mathsf{max}_{\sigma \in \mathcal{N}_{\mathsf{D}} \cup \mathcal{N}_{\mathsf{U}} \cup \mathcal{T}_{\mathsf{D}} \cup \mathcal{T}_{\mathsf{U}}}(\lambda(\sigma)) + 1$. Then, to represent the letter σ of a node we can use $\lambda(\sigma)$ reservations, each with a discrete size of at least 1.

3.2 Multi-Lane Spatial Logic with Scope

In the previous sections we have shown that the original MLSL is undecidable even without length measurement, if we allow an arbitrary number of lanes. In this section we define an extension of MLSL called Multi-Lane Spatial Logic with Scope (MLSLS). In this extension it is possible to confine the *scope* for the cars considered in a given traffic situation. We then show that for the fragment where all quantifiers are restricted to a finite and bounded domain, the satisfiability problem is decidable.

3.2.1 The Model

Just like in MLSL in our extension the model has a traffic snapshot, a sensor function, a view and a variable valuation. Additionally, we introduce the notion of a scope to the model to be able to narrow down the cars under consideration in a given situation. This leads to the following definition of a model. Note that as usual, we use $\mathbb{I}$ as the set of car identifiers.

Definition 3.2.1 (Model with Scope)**.** Let $CS \subseteq \mathbb{I}$ be a subset of identifiers, TS a traffic snapshot, V a view and ν a valuation. Then we call $M = (CS, TS, \Omega, V, \nu)$ a *model of MLSLS with scope* CS. We explicitly allow for Ω and the functions in TS to be defined only for a subset of $\mathbb{I}$. $\triangle$

We allow for MLSLS models without a view and for models that only contain information about some cars. This is useful to define composition of models representing disjoint sets of cars and restriction of models to sets of cars. To this end, we define a simplified version of MLSLS models that does not have a view and no ego constant.

Definition 3.2.2 (Simple MLSLS Models)**.** Let $M = (CS, TS, \Omega, \nu)$ where $\nu : \mathsf{CVar} \to \mathbb{I}$. Then we say that M is a *simple MLSLS model.* $\triangle$

Note that in the definition above the variable valuation does not map the ego constant, as $\mathsf{ego} \notin \mathsf{CVar}$. We call a model with view *proper.* If an MLSLS model *might* be simple, and we leave it open on purpose, we say that an MLSLS model is *possibly simple.* Additionally, we use $\mathbb{M}$ to denote the set of all (possibly simple) MLSLS models. Note that we also use $\mathbb{M}$ for the set of all MLSL models. However, it will be clear from the context to which models $\mathbb{M}$ refers to. Further, let $M_\emptyset = (\emptyset, \emptyset, \emptyset, \emptyset)$ be the *empty MLSLS model.*

Definition 3.2.3 (Car Domain)**.** For an MLSLS model M, a traffic snapshot $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$ and a sensor function $\Omega : \mathbb{I} \times \mathbb{TS} \to \mathbb{R}_{>0}$ we define the *car domain* as $\mathsf{cars}\, TS = \mathsf{dom}\, \mathsf{pos}$, $\mathsf{cars}\, \Omega = \mathsf{dom}(\mathsf{dom}\, \Omega)$ and $\mathsf{cars}\, M = \mathsf{cars}\, TS$, where TS is the traffic snapshot in M. $\triangle$

We call a model *finite* if it represents a finite number of cars, i.e. if $\mathsf{cars}\, M$ is finite. For an MLSLS model M we define its size, denoted with $|M|$, as $|M| = |\mathsf{cars}\, M|$.

For our models we have requirements, as to what constitutes a sane model. First, we define a requirement for sensor functions, and then for MLSLS models. We require that the sensor function for each car is independent of other cars.

As an example, we do not want a sensor function with $\Omega(C, TS_1) = 5$ and $\Omega(C, TS_2) = 6$, where TS_1 and TS_2 contain equal values for the car C. We formalise this below.

Definition 3.2.4 (Sanity Condition for Sensor Functions)**.** Let $\Omega : \mathbb{I} \times \mathbb{TS} \to \mathbb{R}_{>0}$ be a sensor function. Then, Ω is *sane* if for all $(C, TS), (C, TS') \in \mathsf{dom}\,\Omega$ we have

$$\{C\} \lhd TS = \{C\} \lhd TS' \quad \text{implies} \quad \Omega(C, TS) = \Omega(C, TS') \ . \qquad \triangle$$

We require that in an MLSLS model the traffic snapshot and the sensor function is defined for all cars in the scope and for all cars in the range of the variable valuation. Furthermore, we require that the traffic snapshot is defined for the same cars as the sensor function and that all functions in the traffic snapshot have the same domain. Finally, we require that the traffic snapshot, the sensor function and the view are sane.

Definition 3.2.5 (Sanity conditions for MLSLS Models)**.** We call an MLSLS model *sane* if it satisfies the following conditions. We require that in a possibly simple model $M = (CS, TS, \Omega, V, \nu)$ (resp. $M = (CS, TS, \Omega, \nu)$)

- $\mathsf{cars}\ TS$ is a superset of the scope and of the range of the variable valuation,
- $\mathsf{cars}\ TS = \mathsf{cars}\,\Omega$,
- all functions in the traffic snapshot have the same domain,
- the sensor function satisfies Definition 3.2.4,
- the traffic snapshot satisfies the classical sanity conditions (cf. Definition 2.3.3) and
- if the model is proper, then the view satisfies Definition 2.3.9. $\triangle$

We assume that all MLSLS models we consider satisfy the sanity conditions above.

Assumption 3.2.6. All possibly simple MLSLS models we consider are sane, i.e. satisfy Definition 3.2.5. $\triangle$

We can transform a proper MLSLS model into a simple model by removing information about the view owner.

Definition 3.2.7. For a proper MLSLS model $M = (CS, TS, \Omega, V, \nu)$ let the simple model $M' = (CS, TS, \Omega, \{\mathsf{ego}\} \mathrel{\unlhd} \nu)$ be the *simple version of* M. $\triangle$

We define that two simple MLSLS models are composable if they do not overlap in their car domains and their car variables. We then extend this definition to define when a proper and a simple model are composable.

Definition 3.2.8 (Composable Models)**.** For $i \in \{1, 2\}$ we say that two simple MLSLS models $M_i = (CS_i, TS_i, \Omega_i, \nu_i)$ are *composable* iff $CS_1 \cap CS_2 = \emptyset$, $\mathsf{cars}\, TS_1 \cap \mathsf{cars}\, TS_2 = \emptyset$, $\mathsf{cars}\, \Omega_1 \cap \mathsf{cars}\, \Omega_2 = \emptyset$ and $\mathsf{dom}\, \nu_1 \cap \mathsf{dom}\, \nu_2 = \emptyset$. We define that a proper MLSLS model M and a simple MLSLS model M' are *composable* if the simple version of M and M' are composable. $\triangle$

Note that $M_\emptyset$ is composable with every MLSLS model, including itself and that two proper models are never composable.

We define three operations on simple MLSLS models. The first is the *disjoint union* of two MLSLS models, denoted by $\boxplus$. The second is the *restriction* of an MLSLS model to a set of cars, denoted by $\boxed{\upharpoonright}$. The last operation is the *anti-restriction* of an MLSLS model to a set of cars, denoted by $\boxminus$.

Definition 3.2.9 (Operations on Simple Models)**.** Let $M_i = (CS_i, TS_i, \Omega_i, \nu_i)$ with $i \in \{1, 2\}$ be two composable MLSLS models and let $CS \subseteq \mathbb{I}$ be a set of car identifiers. Then we define three functions: the *disjoint union* $\boxplus : \mathbb{M} \times \mathbb{M} \to \mathbb{M}$, the *restriction* $\boxed{\upharpoonright} : \mathbb{M} \times \mathcal{P}(\mathbb{I}) \to \mathbb{M}$ and the *anti-restriction* $\boxminus : \mathbb{M} \times \mathcal{P}(\mathbb{I}) \to \mathbb{M}$ as

$$\begin{aligned}
&M_1 \boxplus M_2 = (CS_1 \uplus CS_2, TS_1 \uplus TS_2, \Omega_1 \uplus \Omega_2, \nu_1 \uplus \nu_2) \ , \\
&M_1 \boxed{\upharpoonright} CS = (CS_1 \cap CS, CS \lhd TS_1, (CS \times \mathbb{TS}) \lhd \Omega_1, \nu_1 \rhd CS) \ , \\
&M_1 \boxminus CS = M_1 \boxed{\upharpoonright} (\mathbb{I} \setminus CS) \ .
\end{aligned}$$

$\triangle$

We extend the operations above to proper MLSLS models by allowing for $M_1 \boxplus M_2$ one model to be a proper MLSLS model, while the other remains a simple model. The result also is a proper MLSLS model. For a proper MLSLS model M_1 the operation $M_1 \boxed{\upharpoonright} CS$ results in a simple model if the view owner is not in CS. In this case we remove the ego variable from the valuation. For a proper MLSLS model M_1 the operation $M_1 \boxminus CS$ results in a simple model if the view owner is in CS. We point out that Definition 3.2.4 is helpful to have usable definitions of operations on MLSLS models.

Definition 3.2.10 (Operations on Proper Models)**.** For two composable models $M_1 = (CS_1, TS_1, \Omega_1, \nu_1)$ and $M_2 = (CS_2, TS_2, \Omega_2, V_2, \nu_2)$ with $V_2 = (L, X, E)$

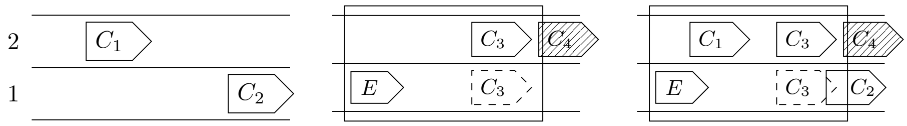


Figure 3.3: We depict a simple MLSLS model (M_1 on the left) and two proper MLSLS models (M_2 in the middle and M_3 on the right). Cars not in the scope are shaded.

and a set of car identifiers $CS \subseteq \mathbb{I}$ we define

$$M_1 \boxplus M_2 = M_2 \boxplus M_1 = (CS_1 \uplus CS_2, TS_1 \uplus TS_2, \Omega_1 \uplus \Omega_2, V_2, \nu_1 \uplus \nu_2) \ ,$$

$$M_2 \boxdot CS = \begin{cases} (CS_2 \cap CS, TS', \Omega', V_2, \nu_2 \rhd CS) & \text{if } E \in CS \ , \\ (CS_2 \cap CS, TS', \Omega', \mathsf{CVar} \lhd (\nu_2 \rhd CS)) & \text{otherwise} \ , \end{cases}$$

$$\text{where } TS' = CS \lhd TS_2 \text{ and } \Omega' = (CS \times \mathbb{TS}) \lhd \Omega_2 \ ,$$

$$M_2 \boxminus CS = M_2 \boxdot (\mathbb{I} \setminus CS) \ .$$

△

Example 3.2.11. In Figure 3.3 we show three MLSLS models: one simple model M_1 and two proper models M_2, M_3. The models M_1 and M_2 are composable because they contain data for different cars, their variable valuations do not overlap and only M_2 is a proper model. The model M_3 is equal to the composition of M_1 and M_2 and contains all information from these models. Further, we have $M_1 = M_3 \boxminus \{C_3, C_4, E, C_5\}$ even though C_5 is not represented in M_3. △

Algebraic Properties

We observe the following helpful algebraic properties. Note that these properties only hold for composable models and not for all possible models as one might hope. This requirement stems from the fact that each value (car) represents a distinct physical entity. The proofs are a little cumbersome with their distinctions on whether the original and the resulting models of the operations are proper or simple.

Lemma 3.2.12 (Model Composition is Associative and Commutative)**.** *Let M_1, M_2 and M_3 be three composable and possibly simple models. Then*

$$\begin{aligned} M_1 \boxplus M_2 &= M_2 \boxplus M_1 \quad \text{and} \\ (M_1 \boxplus M_2) \boxplus M_3 &= M_1 \boxplus (M_2 \boxplus M_3) \ . \end{aligned}$$

Proof. The operator $\boxplus$ is defined via the union of sets and (tuples of) functions. There, union is associative and commutative. Hence, $\boxplus$ is associative and commutative. □

Lemma 3.2.13 ((Anti-)Restriction can be Joined)**.** *Let M_1 be a possibly simple MLSLS model and let $CS, CS' \subseteq \mathbb{I}$ be sets of car identifiers. Then*

$$(M_1 \mathbin{\boxed{\upharpoonright}} CS) \mathbin{\boxed{\upharpoonright}} CS' = M_1 \mathbin{\boxed{\upharpoonright}} (CS \cap CS') \quad \text{and} \tag{3.12}$$

$$(M_1 \boxminus CS) \boxminus CS' = M_1 \boxminus (CS \cup CS') \ . \tag{3.13}$$

Proof. We consider Equation (3.12). We first make a case distinction on whether M_1 is proper.
Case 1 (M_1 is proper). Let $M_1 = (CS_1, TS_1, \Omega_1, V_1, \nu_1)$ with $V_1 = (L, X, E)$. We make a case distinction on whether $E \in CS$ and $E \in CS'$.
Subcase 1.1 ($E \in CS$, $E \in CS'$). Then

$$\begin{aligned} &(M_1 \mathbin{\boxed{\upharpoonright}} CS) \mathbin{\boxed{\upharpoonright}} CS' \\ &= (CS_1 \cap CS, CS \lhd TS_1, (CS \times \mathbb{TS}) \lhd \Omega_1, V_1, \nu_1 \rhd CS) \mathbin{\boxed{\upharpoonright}} CS' \\ &= (CS_1 \cap CS \cap CS', CS' \lhd (CS \lhd TS_1), \\ &\qquad (CS' \times \mathbb{TS}) \lhd ((CS \times \mathbb{TS}) \lhd \Omega_1), V_1, (\nu_1 \rhd CS) \rhd CS') \\ &= (CS_1 \cap CS \cap CS', (CS' \cap CS) \lhd TS_1, \\ &\qquad ((CS' \cap CS) \times \mathbb{TS}) \lhd \Omega_1, V_1, \nu_1 \rhd (CS \cap CS')) \\ &= M_1 \mathbin{\boxed{\upharpoonright}} (CS \cap CS') \ . \end{aligned}$$

Subcase 1.2 ($E \notin CS$). Mostly analogous to the previous case, with the difference that $(M_1 \mathbin{\boxed{\upharpoonright}} CS)$ and thus also $(M_1 \mathbin{\boxed{\upharpoonright}} CS) \mathbin{\boxed{\upharpoonright}} CS'$ does not have a view. Note that it does not matter whether $E \in CS'$. We have

$$\begin{aligned}
&(M_1 \boxbar CS) \boxbar CS' \\
&= (CS_1 \cap CS, CS \lhd TS_1, (CS \times \mathbb{TS}) \lhd \Omega_1, \mathsf{CVar} \lhd (\nu_1 \rhd CS)) \boxbar CS' \\
&= (CS_1 \cap CS \cap CS', CS' \lhd (CS \lhd TS_1), \\
&\qquad (CS' \times \mathbb{TS}) \lhd ((CS \times \mathbb{TS}) \lhd \Omega_1), (\mathsf{CVar} \lhd (\nu_1 \rhd CS)) \rhd CS') \\
&= (CS_1 \cap CS \cap CS', (CS' \cap CS) \lhd TS_1, \\
&\qquad ((CS' \cap CS) \times \mathbb{TS}) \lhd \Omega_1, \mathsf{CVar} \lhd (\nu_1 \rhd (CS \cap CS')) \\
&= M_1 \boxbar (CS \cap CS') \ .
\end{aligned}$$

For the third step we point out that $(\mathsf{CVar} \lhd (\nu_1 \rhd CS)) \rhd CS'$ by associativity of the relation restriction operators is equal to $\mathsf{CVar} \lhd ((\nu_1 \rhd CS) \rhd CS')$.

Subcase 1.3 ($E \in CS$, $E \notin CS'$). Mostly analogous to the previous case.

Case 2 (M_1 is simple). Analogous to the previous case with the difference that there is no view and that the case distinction on whether $E \in CS$ or $E \in CS'$ is not necessary.

Now we consider Equation (3.13). We have

$$\begin{aligned}
(M_1 \boxminus CS) \boxminus CS' &= (M_1 \boxbar (\mathbb{I} \setminus CS)) \boxbar (\mathbb{I} \setminus CS') \qquad \text{with } \textit{Equation } (3.12) \\
&= (M_1 \boxbar ((\mathbb{I} \setminus CS) \cap (\mathbb{I} \setminus CS')) \\
&= M_1 \boxbar (\mathbb{I} \setminus (CS \cup CS')) \\
&= M_1 \boxminus (CS \cup CS') \ .
\end{aligned}$$

□

Lemma 3.2.14 (Neutral Elements). *Let M_1 be a possibly simple MLSLS model and $CS \subseteq \mathbb{I}$. Then*

$$M_1 \boxplus M_\emptyset = M_1 \ ,$$

$$\begin{aligned}
M_\emptyset \boxminus CS &= M_\emptyset \ , & M_\emptyset \boxbar CS &= M_\emptyset \ , \\
M_1 \boxminus \emptyset &= M_1 \ , & M_1 \boxbar \emptyset &= M_\emptyset \ , \\
M_1 \boxminus \mathbb{I} &= M_\emptyset \ , & M_1 \boxbar \mathbb{I} &= M_1 \ .
\end{aligned}$$

Proof. Follows immediately from the definitions. □

Next, we show that restriction and anti-restriction distribute over disjoint union.

Lemma 3.2.15 (Distributivity)**.** *Let M_1 and M_2 be two composable and possibly simple MLSLS models and let $CS \subseteq \mathbb{I}$. Then*

$$(M_1 \boxbar CS) \boxplus (M_2 \boxbar CS) = (M_1 \boxplus M_2) \boxbar CS \text{ and} \tag{3.14}$$

$$(M_1 \boxminus CS) \boxplus (M_2 \boxminus CS) = (M_1 \boxplus M_2) \boxminus CS \ . \tag{3.15}$$

Proof. We first consider Equation (3.14) and the case that M_1 is proper and that M_2 is simple.

Case 1 (M_1 is proper and M_2 is simple). Let $M_1 = (CS_1, TS_1, \Omega_1, V_1, \nu_1)$ with $V_1 = (L, X, E)$ and $M_2 = (CS_2, TS_2, \Omega_2, \nu_2)$. We make a case distinction on whether $E \in CS$.

Subcase 1.1 ($E \in CS$). For the third step we point out that M_1 and M_2 are composable. Then

$$\begin{aligned}
&(M_1 \boxbar CS) \boxplus (M_2 \boxbar CS)\\
&= (CS_1 \cap CS, CS \lhd TS_1, (CS \times \mathbb{TS}) \lhd \Omega_1, V_1, \nu_1 \rhd CS)\\
&\qquad \boxplus (CS_2 \cap CS, CS \lhd TS_2, (CS \times \mathbb{TS}) \lhd \Omega_2, \nu_2 \rhd CS)\\
&= ((CS_1 \cap CS) \uplus (CS_2 \cap CS), (CS \lhd TS_1) \uplus (CS \lhd TS_2),\\
&\qquad ((CS \times \mathbb{TS}) \lhd \Omega_1)) \uplus ((CS \times \mathbb{TS}) \lhd \Omega_2), V_1,\\
&\qquad (\nu_1 \rhd CS) \uplus (\nu_2 \rhd CS))\\
&= ((CS_1 \uplus CS_2) \cap CS, CS \lhd (TS_1 \uplus TS_2),\\
&\qquad (CS \times \mathbb{TS}) \lhd (\Omega_1 \uplus \Omega_2), V_1, (\nu_1 \uplus \nu_2) \rhd CS)\\
&= (CS_1 \uplus CS_2, TS_1 \uplus TS_2, \Omega_1 \uplus \Omega_2, V_1, \nu_1 \uplus \nu_2) \boxbar CS\\
&= (M_1 \boxplus M_2) \boxbar CS \ .
\end{aligned}$$

Subcase 1.2 ($E \notin CS$). For this case we point out that

$$(\mathsf{CVar} \lhd (\nu_1 \rhd CS)) \uplus (\nu_2 \rhd CS) = \mathsf{CVar} \lhd ((\nu_1 \rhd CS) \uplus (\nu_2 \rhd CS)) \ ,$$

which is true because M_2 is simple and hence $\mathsf{dom}\, \nu_2 \subseteq \mathsf{CVar}$. Additionally, as in the previous case we point out that M_1 and M_2 are composable. We use both properties in the last step in the equations below. Then

$$\begin{aligned}
&(M_1 \boxbar CS) \boxplus (M_2 \boxbar CS)\\
&= (CS_1 \cap CS, CS \lhd TS_1, (CS \times \mathbb{TS}) \lhd \Omega_1, \mathsf{CVar} \lhd (\nu_1 \rhd CS))\\
&\qquad \boxplus (CS_2 \cap CS, CS \lhd TS_2, (CS \times \mathbb{TS}) \lhd \Omega_2, \nu_2 \rhd CS)
\end{aligned}$$

$$\begin{aligned}
&= ((CS_1 \cap CS) \uplus (CS_2 \cap CS), (CS \lhd TS_1) \uplus (CS \lhd TS_2),\\
&\qquad ((CS \times \mathbb{TS}) \lhd \Omega_1)) \uplus ((CS \times \mathbb{TS}) \lhd \Omega_2),\\
&\qquad (\mathsf{CVar} \lhd (\nu_1 \rhd CS)) \uplus (\nu_2 \rhd CS))\\
&= ((CS_1 \uplus CS_2) \cap CS, CS \lhd (TS_1 \uplus TS_2),\\
&\qquad (CS \times \mathbb{TS}) \lhd (\Omega_1 \uplus \Omega_2), \mathsf{CVar} \lhd ((\nu_1 \uplus \nu_2) \rhd CS)) \ .
\end{aligned}$$

We point out that because $E \notin CS$ the view is removed by the restriction operator. Hence, continuing the equations above we can simply introduce a view. We get

$$\begin{aligned}
&= (CS_1 \uplus CS_2, TS_1 \uplus TS_2, \Omega_1 \uplus \Omega_2, V_1, \nu_1 \uplus \nu_2) \mathbin{\boxed{\upharpoonleft}} CS\\
&= (M_1 \boxplus M_2) \mathbin{\boxed{\upharpoonleft}} CS \ .
\end{aligned}$$

Case 2 (M_1 is simple and M_2 is proper). By commutativity of $\boxplus$ this case is analogous to the previous case.

Case 3 (M_1 is simple and M_2 is simple). Similar to before, only that the case distinctions on whether $E \in CS$ are not needed.

$\square$

Next, we show that the order in which we apply restriction and anti-restriction does not affect the result.

Lemma 3.2.16. *Let M_1 be a possibly simple MLSLS model and let $CS, CS' \subseteq \mathbb{I}$ be sets of car identifiers. Then*

$$(M_1 \mathbin{\boxed{\upharpoonleft}} CS) \boxminus CS' = (M_1 \mathbin{\boxed{\upharpoonleft}} (CS \setminus CS')) = (M_1 \boxminus CS') \mathbin{\boxed{\upharpoonleft}} CS \ .$$

Proof. We use Lemma 3.2.13 to combine and split the restriction operator. We show the first equation. Then

$$\begin{aligned}
&(M_1 \mathbin{\boxed{\upharpoonleft}} CS) \boxminus CS' && \text{def. } \boxminus\\
&= (M_1 \mathbin{\boxed{\upharpoonleft}} CS) \mathbin{\boxed{\upharpoonleft}} (\mathbb{I} \setminus CS') && \text{Lemma 3.2.13}\\
&= M_1 \mathbin{\boxed{\upharpoonleft}} (CS \cap (\mathbb{I} \setminus CS'))\\
&= (M_1 \mathbin{\boxed{\upharpoonleft}} CS \setminus CS') \ .
\end{aligned}$$

We continue with the second equation. Then

$$\begin{aligned}
& M_1 \boxed{\upharpoonright} (CS \cap (\mathbb{I} \setminus CS')) && \text{Lemma 3.2.13} \\
={}& (M_1 \boxed{\upharpoonright} (\mathbb{I} \setminus CS')) \boxed{\upharpoonright} CS && \text{def. } \boxminus \\
={}& (M_1 \boxminus CS') \boxed{\upharpoonright} CS \ .
\end{aligned}$$

□

Discussion of Operations on MLSLS Models

We briefly discuss our choice of operations on MLSLS. Our reasoning is that if a car is represented in a model it is represented completely therein. This idea is contained in our definition of composition, where we require composable models to be completely disjoint, and in restriction (resp. anti-restriction), where all data about the cars are removed (resp. kept). The choice that a car is completely represented in the model stems from our decision procedure for MLSLS (Section 3.3). There, a tuple of arithmetic variables contains all information about the cars the tuple represents. For the same reason, i.e. because a model has all information about a car, the restriction operators take a set of car identifiers as second parameter, instead of another MLSLS model.

We choose to introduce simple MLSLS models for two reasons: one reason is that it allows for easier definitions of the operations. At first we defined composition of proper MLSLS models, and required that all information regarding the view and its owner is equal. As MLSLS models are complicated, this requirement also becomes complicated. The other reason is that with simple MLSLS models we can define MLSLS transitions for each car independently because there is no view to move along as time passes.

The following proposition holds for the sanity conditions on traffic snapshots because we do not change the values of a car and similarly for the view, if the resulting model has a view.

Proposition 3.2.17 (Sanity Preservation)**.** *For all composable models M_1, M_2 and sets $CS \subseteq \mathbb{I}$ the models $M_1 \boxplus M_2$, $M_1 \boxminus CS$ and $M_1 \boxed{\upharpoonright} CS$ are sane (cf. Definitions 3.2.4, 3.2.5 and 2.3.10).*

Notice that a model M with scope $\mathbb{I}$ is a model of MLSL in the sense of [HLO+11; LH15].

3.2.2 The Logic

MLSLS extends MLSL with *scope formulas* of the form $cs : \phi$ (read ϕ under variable scope cs) for finite sets of car variables $cs \subseteq \mathsf{CVar}$. The idea is that ϕ is evaluated considering only cars denoted by variables in cs.

Definition 3.2.18 (MLSLS Syntax)**.** The *syntax of an MLSLS formula* ϕ is given as

$$\phi ::= \gamma = \gamma' \mid \mathsf{free} \mid \mathsf{re}(\gamma) \mid \mathsf{cl}(\gamma) \mid \ell = k \mid \neg\phi \mid \phi \wedge \phi \mid \exists c.\, \phi \mid \phi \frown \phi \mid \begin{matrix}\phi\\ \phi\end{matrix} \mid cs : \phi\,,$$

where $c \in \mathsf{CVar}$, $k \in \mathbb{R}$, $\gamma, \gamma' \in \mathsf{CVar} \cup \{\mathsf{ego}\}$, $cs \subseteq \mathsf{CVar}$ and ℓ is a special symbol denoting the length of the lanes' extension. With Φ we denote the set of MLSLS formulas. △

We define the set of free variables in an MLSLS formula. The definition is similar to the definition of free variables in first-order logic. The idea is that a variable c is free if it occurs outside a formula $\exists c.\, \phi$. Note that ego is a constant, and thus not a (free) variable.

Definition 3.2.19 (Free Variables)**.** Let $c \in \mathsf{CVar}$, $k \in \mathbb{R}, \gamma, \gamma' \in \mathsf{CVar} \cup \{\mathsf{ego}\}$ and $CS \subseteq \mathbb{I}$. Then we define the *free variables* in an MLSLS formula inductively as

$$\begin{aligned}
\mathsf{freeVar}(\gamma = \gamma') &= \{\gamma, \gamma'\} \setminus \{\mathsf{ego}\}\ ,\\
\mathsf{freeVar}(\mathsf{re}(\gamma)) &= \mathsf{freeVar}(\mathsf{cl}(\gamma)) = \{\gamma\} \setminus \{\mathsf{ego}\}\ ,\\
\mathsf{freeVar}(\mathsf{free}) &= \mathsf{freeVar}(\ell = k) = \emptyset\ ,\\
\mathsf{freeVar}(\neg\phi_1) &= \mathsf{freeVar}(\phi_1)\ ,\\
\mathsf{freeVar}(cs : \phi_1) &= cs \cup \mathsf{freeVar}(\phi_1)\ ,\\
\mathsf{freeVar}(\phi_1 \frown \phi_2) &= \mathsf{freeVar}(\begin{matrix}\phi_2\\ \phi_1\end{matrix}) = \mathsf{freeVar}(\phi_1 \wedge \phi_2)\\
&= \mathsf{freeVar}(\phi_1) \cup \mathsf{freeVar}(\phi_2)\ ,\\
\mathsf{freeVar}(\exists c.\, \phi_1) &= \mathsf{freeVar}(\phi_1) \setminus \{c\}\ .
\end{aligned}$$

△

We give the semantics of MLSLS and point out that we choose a semantics that is less abstract than the original one [HLO+11]. This is to make it easier to see the similarity of our semantics for MLSLS and of our decision procedure that we define later for it.

Definition 3.2.20 (Semantics)**.** Let $c \in \mathsf{CVar}$, $k \in \mathbb{R}_{\geq 0}$ and $\gamma, \gamma' \in \mathsf{CVar} \cup \{\mathsf{ego}\}$. Further, let $CS \subseteq \mathbb{I}$ be a scope, TS a traffic snapshot, $V = (L, [r, r'], E)$ with $L = [l, l']$ a view, Ω a sensor function and ν with $\nu(\mathsf{ego}) = E$ a valuation. We define the *satisfaction* of a formula by a proper model $M = (CS, TS, \Omega, V, \nu)$ as follows:

$$
\begin{array}{lll}
M \models \gamma = \gamma' & \text{iff} & \nu(\gamma) = \nu(\gamma') \\
M \models \mathsf{free} & \text{iff} & (l \notin \mathsf{res}(C) \cup \mathsf{clm}(C) \text{ or } \mathsf{se}(C, TS, \Omega) \cap (r, r') = \emptyset) \\
& & \text{for every } C \in CS, \text{ and } l = l' \text{ and } r < r' \\
M \models \mathsf{re}(\gamma) & \text{iff} & l \in \mathsf{res}(\nu(\gamma)) \text{ and } [r, r'] \subseteq \mathsf{se}(\nu(\gamma), TS, \Omega) \text{ and} \\
& & \quad l = l' \text{ and } r < r' \\
M \models \mathsf{cl}(\gamma) & \text{iff} & l \in \mathsf{clm}(\nu(\gamma)) \text{ and } [r, r'] \subseteq \mathsf{se}(\nu(\gamma), TS, \Omega) \text{ and} \\
& & \quad l = l' \text{ and } r < r' \\
M \models \ell = k & \text{iff} & r' - r = k \\
M \models cs : \phi & \text{iff} & (\{\nu(c) \mid c \in cs\}, TS, \Omega, V, \nu) \models \phi \\
M \models \neg\phi & \text{iff} & M \not\models \phi \\
M \models \phi_1 \land \phi_2 & \text{iff} & M \models \phi_1 \text{ and } M \models \phi_2 \\
M \models \exists c.\, \phi & \text{iff} & (CS, TS, \Omega, V, \nu \oplus \{c \mapsto C\}) \models \phi, \text{ for some } C \text{ in } CS \\
M \models \phi_1 \frown \phi_2 & \text{iff} & (CS, TS, \Omega, V_1, \nu) \models \phi_1 \text{ and } (CS, TS, \Omega, V_2, \nu) \models \phi_2 \\
& & \text{with } V_1 = V_{[r, r'']} \text{ and } V_2 = V_{[r'', r']}, \text{ for some } r'' \in [r, r'] \\
M \models \begin{matrix} \phi_2 \\ \phi_1 \end{matrix} & \text{iff} & L \neq \emptyset \text{ implies} \\
& & \quad (CS, TS, \Omega, V_1, \nu) \models \phi_1 \text{ and } (CS, TS, \Omega, V_2, \nu) \models \phi_2 \\
& & \quad \text{with } V_1 = V^{[l, l'']} \text{ and } V_2 = V^{[l''+1, l']}, \\
& & \quad \text{for some } l'' \in [l-1, l'], \text{ and} \\
& & L = \emptyset \text{ implies } M \models \phi_1 \text{ and } M \models \phi_2
\end{array}
\qquad \triangle
$$

In the definition of the semantics of the vertical chop operator $\begin{pmatrix} \phi_2 \\ \phi_1 \end{pmatrix}$, we deviate from the classical semantics and distinguish two cases. If the current view contains at least one lane we split the view into a lower and an upper subview and evaluate ϕ_1 on the lower subview and ϕ_2 on the upper subview.

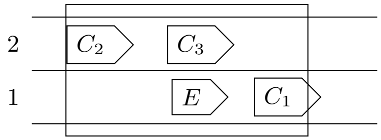


Figure 3.4: Visualisation of a model satisfying the formula from Example 3.2.21. The scope and the valuation are not shown.

Otherwise, when the view is empty, we do not chop the view and instead evaluate both formulas on the same view. The intuition here is that all subviews of an empty view are empty and we cannot distinguish different empty views with MLSLS. This special handling is necessary, because if we chop along a lane into a lower and an upper subview the lanes of the two subviews should be disjoint. However, for horizontal chops the endpoint of the left subview and the start point of the right subview are shared.

The scope CS of a model (CS, TS, Ω, V, ν) is used in the semantics for the formulas free and $\exists c.\, \phi$. The formula free holds if no car from the scope CS occupies a part of the lane under consideration, and $\exists c.\, \phi$ holds if ϕ holds for some car C in the scope CS.

Example 3.2.21. Consider the example formula

$$\phi \equiv \{c_1, c_2\} : \left(\left\langle \begin{matrix} \mathsf{free} \\ \mathsf{re}(\mathsf{ego}) \end{matrix} \right\rangle \frown \exists c.\, \langle \mathsf{re}(c) \rangle \right) \ .$$

The formula states that horizontally right of ego there is the reservation of either c_1 or c_2 and that vertically above ego there is neither a reservation from c_1 nor from c_2. A model satisfying ϕ is shown in Figure 3.4. The valuation is $\nu = \{\mathsf{ego} \mapsto E, c_1 \mapsto C_1, c_2 \mapsto C_2\}$. Note that the scope (which is not shown in the figure) does not affect whether the formula is satisfied, because the scope operator overrides the scope. △

As for the original logic we define two forms of satisfiability for our extension.

Definition 3.2.22. Given a *infinite* set of lanes $\mathbb{L}$ and an MLSL formula ϕ we say that ϕ is *lane-unboundedly satisfiable* iff there exists a model M such that $M \models \phi$. △

Definition 3.2.23. Given a *finite* set of lanes $\mathbb{L}$ and an MLSL formula ϕ we say that ϕ is *lane-boundedly satisfiable* iff there exists a model M such that $M \models \phi$. $\triangle$

As for the original logic MLSL, we point out that in this work we only consider the unbounded version of MLSLS satisfiability.

Basic Properties of MLSLS

We prove some basic properties of our extension of MLSL. We introduced a satisfaction relation for MLSLS that is less abstract than the one given for MLSL in Chapter 2. The goal of defining a less abstract satisfaction relation is to make our semantics for MLSLS and our decision procedure that we define later for it more similar. In the following lemma we prove that we could have extended the abstract semantics from Definition 2.3.13 and added a scope component, and that both satisfaction relations would be equivalent. For an example consider the formula $\phi \equiv \begin{matrix}\phi_2\\ \phi_1\end{matrix}$, an MLSLS model $M = (CS, TS, \Omega, V, \nu)$ and let $\models_{\mathsf{c}}$ be the satisfaction relation of MLSLS. In the semantics of MLSLS we defined that to check $M \models_{\mathsf{c}} \phi$, we make a case distinction on whether the view V is empty. If V is empty, we check ϕ_1 and ϕ_2 on V. Otherwise, we chop V into a lower and an upper subview and check ϕ_1 and ϕ_2 on these subviews. Instead, we could have defined a satisfaction relation $\models_{\mathsf{a}}$ for MLSLS that essentially takes the definition from Definition 2.3.13 and adds the scope, i.e.

$$\begin{gathered} M \models_{\mathsf{a}} \phi \\ \text{iff} \\ \exists V_1, V_2 .\, V = V_1 \ominus V_2 \;\text{ and }\; (CS, TS, \Omega, V_i, \nu) \models_{\mathsf{a}} \phi_i \text{ with } i \in \{1, 2\} \;\; . \end{gathered}$$

Then $M \models_{\mathsf{a}} \phi$ iff $M \models_{\mathsf{c}} \phi$.

Lemma 3.2.24. *Let $\models_{\mathsf{c}}$ be the concrete semantics of MLSLS from Definition 3.2.20 and let $\models_{\mathsf{a}}$ be the abstract semantics from above. Then, for all MLSLS models M and formulas ϕ we have*

$$M \models_{\mathsf{c}} \phi \;\text{ iff }\; M \models_{\mathsf{a}} \phi \;\; .$$

Proof. We proceed by induction on the structure of the formula ϕ.
Induction base.

For the base cases this is fairly clear.
Induction hypothesis.
For all M_i and ϕ_i with $i \in \{1,2\}$ we have $M \models_{\mathsf{c}} \phi$ iff $M \models_{\mathsf{a}} \phi$.
Induction step.
We consider the case $\phi \equiv \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$. Let $M = (CS, TS, \Omega, V, \nu)$ with $V = (L, X, E)$.

Case 1 (if). Assume $M \models_{\mathsf{a}} \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$. Then there are views V_1, V_2 with $V = V_1 \ominus V_2$ such that $(CS, TS, \Omega, V_i, \nu) \models_{\mathsf{a}} \phi_i$. From the IH it follows that $(CS, TS, \Omega, V_i, \nu) \models_{\mathsf{c}} \phi_i$.

We make a case distinction on whether $L = \emptyset$. If $L = \emptyset$, then the definition of $\ominus$ implies that $V_1 = V_2 = V$, which means $M \models_{\mathsf{c}} \phi$.

Assume $L \neq \emptyset$ and let $L = [l, l']$. We have to show that we can choose a chop point $l'' \in [l-1, l']$ that produces from V the lane intervals L_1 and L_2. We choose $l'' = l + |L_1| - 1$. Then, independent of whether $L_1 = \emptyset$, $L_2 = \emptyset$ or neither interval is empty we have $L_1 = [l, l'']$ and $L_2 = [l''+1, l']$. Thus, we can always choose an appropriate chop point and conclude $M \models_{\mathsf{c}} \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$.

Case 2 (only if). Assume $M \models_{\mathsf{c}} \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$. Then $(CS, TS, \Omega, V_i, \nu) \models_{\mathsf{c}} \phi_i$ and the views can have two forms, depending on whether $L = \emptyset$: if $L = \emptyset$, then we can see in the definition of $\models_{\mathsf{c}}$ that $V_1 = V_2 = V$. As then we have $V = V_1 \ominus V_2$ we can use the IH to conclude $M \models_{\mathsf{a}} \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$. If however, we have $L \neq \emptyset$, then there is $l'' \in [l-1, l']$ such that $(CS, TS, \Omega, V_i, \nu) \models_{\mathsf{c}} \phi_i$ with $V_1 = V^{[l,l'']}$ and $V_2 = V^{[l''+1,l']}$ and $i \in \{1,2\}$. Again, we have $V = V_1 \ominus V_2$, which implies with the IH that $M \models_{\mathsf{a}} \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$.

$\square$

From the previous lemma it follows that if we disregard formulas of the form $cs : \phi$ and use $\mathbb{I}$ as scope in models, then MLSLS is a conservative extension of MLSL.

We prove that the scope operator is distributive with the binary operators and commutes with negation. Note that we use the original semantics from

[LH15] (cf. $\models_{\mathsf{a}}$ on Page 65) as it is more concise, especially for the vertical chop.

Lemma 3.2.25. *For all $M = (CS, TS, \Omega, V, \nu)$, MLSLS formulas ϕ, ϕ_1, ϕ_2, sets of car variables cs, cs_1, cs_2 and binary operators $\odot \in \{\land, \frown, \smile\}$, where $\smile$ represents the vertical chop operator, we have*

$$M \models cs : \neg\phi \text{ iff } M \models \neg(cs : \phi) \ , \tag{3.16}$$
$$M \models cs : (\phi_1 \land \phi_2) \text{ iff } M \models (cs : \phi_1) \land (cs : \phi_2) \ , \tag{3.17}$$
$$M \models cs : (\phi_1 \frown \phi_2) \text{ iff } M \models (cs : \phi_1) \frown (cs : \phi_2) \ , \tag{3.18}$$
$$M \models cs : \begin{pmatrix}\phi_2\\ \phi_1\end{pmatrix} \text{ iff } M \models \begin{matrix} cs : \phi_2 \\ cs : \phi_1 \end{matrix} \ , \tag{3.19}$$
$$M \models cs_1 : (cs_2 : \phi_1) \text{ iff } M \models cs_2 : \phi_1 \tag{3.20}$$
$$M \models cs_2 : ((cs_1 : \phi_1) \odot \phi_2) \text{ iff } M \models (cs_1 : \phi_1) \odot (cs_2 : \phi_2) \ , \tag{3.21}$$
$$M \models (cs_1 : \phi_1) \odot (cs_2 : \phi_2) \text{ iff } M \models cs_1 : (\phi_1 \odot (cs_2 : \phi_2)) \ . \tag{3.22}$$

Proof. The idea of the proof is to unfold the semantics, introduce first the scope operator and then the other operator and then fold the semantics again. Let $CS' = \{\nu(c) \mid c \in cs\}$ and $M' = (CS', TS, \Omega, V, \nu)$. We first prove Equation (3.16). We have

$$\begin{aligned} M \models cs : \neg\phi \ &\text{iff} \ M' \models \neg\phi \\ &\text{iff} \ M' \not\models \phi \\ &\text{iff} \ M \not\models cs : \phi \\ &\text{iff} \ M \models \neg cs : \phi \ . \end{aligned}$$

We proceed to Equation (3.17) and get

$$\begin{aligned} M \models cs : \phi_1 \land \phi_2 \ &\text{iff} \ M' \models \phi_1 \land \phi_2 \\ &\text{iff} \ M' \models \phi_1 \ \text{and} \ M' \models \phi_2 \\ &\text{iff} \ M \models cs : \phi_1 \ \text{and} \ M \models cs : \phi_2 \\ &\text{iff} \ M \models cs : \phi_1 \land cs : \phi_2 \ . \end{aligned}$$

For Equation (3.18) let $M_i' = (CS', TS, \Omega, V_i, \nu)$ and $M_i = (CS, TS, \Omega, V_i, \nu)$

with $i \in \{1,2\}$. Then

$$\begin{aligned}
M \models cs : \phi_1 \frown \phi_2 \text{ iff } & M' \models \phi_1 \frown \phi_2 \\
\text{iff } & \exists V_1, V_2 .\, V_1 \oplus V_2 = V \text{ and } M_1' \models \phi_1 \text{ and } M_2' \models \phi_2 \\
\text{iff } & \exists V_1, V_2 .\, V_1 \oplus V_2 = V \text{ and } M_1 \models cs : \phi_1 \text{ and } \\
& M_2 \models cs : \phi_2 \\
\text{iff } & M \models cs : \phi_1 \frown cs : \phi_2 \ .
\end{aligned}$$

For Equation (3.19) we have

$$\begin{aligned}
M \models cs : \begin{pmatrix} \phi_2 \\ \phi_1 \end{pmatrix} \text{ iff } & M' \models \begin{array}{c} \phi_2 \\ \phi_1 \end{array} \\
\text{iff } & \exists V_1, V_2 .\, V_1 \ominus V_2 = V \text{ and } M_1' \models \phi_1 \text{ and } M_2' \models \phi_2 \\
\text{iff } & \exists V_1, V_2 .\, V_1 \ominus V_2 = V \text{ and } M_1 \models cs : \phi_1 \text{ and } M_2 \models cs : \phi_2 \\
\text{iff } & M \models \begin{array}{c} cs : \phi_2 \\ cs : \phi_1 \end{array} ,
\end{aligned}$$

where $M_i' = (CS', TS, \Omega, V_i, \nu)$ and $M_i = (CS, TS, \Omega, V_i, \nu)$ with $i \in \{1,2\}$.

For the remaining cases let $M_i = (\{\nu(c) \mid c \in cs_i\}, TS, \Omega, V, \nu)$ with $i \in \{1,2\}$. We come to Equation (3.20) and show that for directly nested scope operators only the innermost scope affects the satisfaction. We have

$$M \models cs_1 : (cs_2 : \phi_1) \text{ iff } M_1 \models cs_2 : \phi_1 \text{ iff } M_2 \models \phi_1 \text{ iff } M \models cs_2 : \phi_1 \ .$$

We use the previous cases to prove Equations (3.21) to (3.22). We have

$$\begin{aligned}
& M \models cs_2 : ((cs_1 : \phi_1) \odot \phi_2) && \text{Equations (3.17) to (3.19)} \\
\text{iff } & M \models (cs_2 : (cs_1 : \phi_1)) \odot (cs_2 : \phi_2) && \text{Equation (3.20)} \\
\text{iff } & M \models (cs_1 : \phi_1) \odot (cs_2 : \phi_2) && \text{Equation (3.20)} \\
\text{iff } & M \models (cs_1 : \phi_1) \odot (cs_1 : (cs_2 : \phi_2)) && \text{Equations (3.17) to (3.19)} \\
\text{iff } & M \models cs_1 : (\phi_1 \odot (cs_2 : \phi_2)) \ . &&
\end{aligned}$$

□

For the rest of this section we investigate which models MLSLS can distinguish. The intuition is that MLSLS can distinguish spatial differences, but not differences in the dynamics. We define that two models are weakly equal iff for a certain set of cars the traffic snapshot is equal and the sensor function

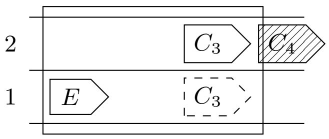


Figure 3.5: Model from Figure 3.3. The scope contains the cars E and C_3.

for that traffic snapshot and that set of cars are equal. The reasoning is that we consider only a single data point of the sensor function and not a function assigning for each traffic snapshot a value.

Definition 3.2.26 (Weak Equality of MLSLS Models)**.** Let $M = (CS, TS, \Omega, \nu)$, $M' = (CS', TS', \Omega', \nu')$ be two simple MLSLS models. Then we define that M and M' are *weakly equal* (denoted $M \eqsim_{\mathsf{m}} M'$) as

$$
\begin{aligned}
M \eqsim_{\mathsf{m}} M' \text{ iff}& \\
& CS = CS' \land \nu = \nu' \\
\land \quad & ((CS \cup \operatorname{ran} \nu) \lhd TS) = ((CS' \cup \operatorname{ran} \nu') \lhd TS') \\
\land \quad & (((CS \cup \operatorname{ran} \nu) \times \{TS\})) \lhd \Omega = (((CS' \cup \operatorname{ran} \nu') \times \{TS'\})) \lhd \Omega' \ .
\end{aligned}
$$

We lift this to proper models by additionally requiring that their views are equal. △

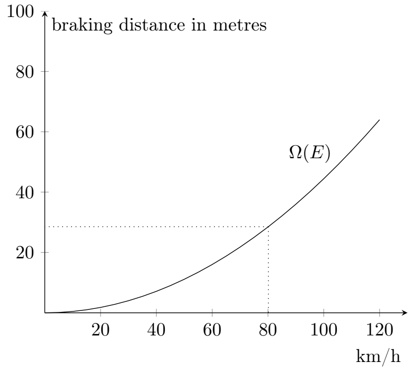
Note that a simple MLSLS model and a proper one are never weakly equal.

Example 3.2.27. Consider the MLSLS model $M = (CS, TS, \Omega, V, \nu)$ from Figure 3.3 shown again in Figure 3.5. Further, let the speed of E in M be 80 and let the sensor function Ω for E be as shown in Figure 3.6 on the left. We assume that $C_4 \notin \operatorname{ran} \nu$, i.e. that there is no variable mapping to C_4.

We introduce $M' = (CS, TS', \Omega', V, \nu)$ which we define to be quite similar to M. That is, $TS' = \{C_4\} \lessdot TS$, for C_3 the sensor function Ω' is equal to Ω, for E we show Ω' in Figure 3.6 on the right and Ω' does not assign values for C_4. We see that Ω' for E is a sampled version of Ω for E.

Now, M' does not contain any information about C_4 and Ω' in M' has less information than Ω in M. Nevertheless, we have $M \eqsim_{\mathsf{m}} M'$ because weak equality only considers the spatial aspects of the given situation models and ignores irrelevant cars. △

It can be easily seen that $\eqsim_{\mathsf{m}}$ is an equivalence relation, i.e. that it is reflexive, symmetric and transitive.



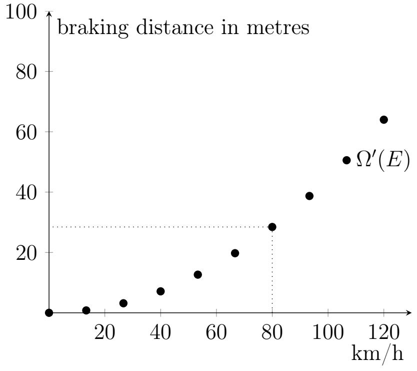


Figure 3.6: On the left we show a sensor function Ω for E. On the right we show a sampled version Ω' of the sensor function for E on the left.

Proposition 3.2.28. $\approx_{\mathsf{m}}$ *is an equivalence relation.*

The following lemma states that weak equality of MLSLS models is preserved by model composition.

Lemma 3.2.29. *For all composable possibly simple MLSLS models M_1, M_1' and M_2, M_2' we have*

$$(M_1 \approx_{\mathsf{m}} M_1' \text{ and } M_2 \approx_{\mathsf{m}} M_2') \text{ implies } M_1 \boxplus M_2 \approx_{\mathsf{m}} M_1' \boxplus M_2' \ .$$

Proof. We first consider the case that $M_1 \boxplus M_2$ and $M_1' \boxplus M_2'$ are proper models. Let $M_1 = (CS_1, TS_1, \Omega_1, V_1, \nu_1)$ and $M_1' = (CS_1', TS_1', \Omega_1', V_1', \nu_1')$ be proper models and let $M_2 = (CS_2, TS_2, \Omega_2, V_2, \nu_2)$ and $M_2' = (CS_2', TS_2', \Omega_2', V_2', \nu_2')$ be simple models such that M_1 and M_2 (resp. M_1' and M_2') are composable. Further, let $M = M_1 \boxplus M_2 = (CS, TS, \Omega, V, \nu)$ and $M' = M_1' \boxplus M_2' = (CS', TS', \Omega', V', \nu')$. We consider the requirements of weak equality separately.

Case 1 ($CS = CS'$). From $M_i \approx_{\mathsf{m}} M_i'$ we know $CS_i = CS_i'$, which together with the compositionality of the models implies $CS_1 \uplus CS_2 = CS_1' \uplus CS_2'$.

Case 2 ($\nu = \nu'$). From $M_i \approx_{\mathsf{m}} M_i'$ we know $\nu_i = \nu_i'$. With compositionality of the models this implies $\nu_1 \uplus \nu_1 = \nu_2' \uplus \nu_2'$.

Case 3 ($(((CS \cup \mathsf{ran}\,\nu) \times \{TS\})) \lhd \Omega = (((CS' \cup \mathsf{ran}\,\nu') \times \{TS'\})) \lhd \Omega'$). From the definition of M we get

$$\begin{aligned}
&((CS \cup \mathsf{ran}\,\nu) \times \{TS\}) \lhd \Omega \\
&= (((CS_1 \uplus CS_2) \cup (\mathsf{ran}\,\nu_1 \uplus \mathsf{ran}\,\nu_2)) \times \{TS_1 \uplus TS_2\}) \lhd (\Omega_1 \uplus \Omega_2) \ .
\end{aligned}$$

Now, $TS_1 \uplus TS_2$ is a single traffic snapshot and not a set of snapshots. Because M_1 and M_2 are composable we can restrict Ω_1 and Ω_2 to smaller domains separately and then join them. Additionally, from Assumption 3.2.6 we know that cars not in $\mathsf{cars}\,\Omega_i$ with $i \in \{1,2\}$ do not affect the function. Thus, continuing the equations above we have

$$\begin{aligned}
&= \quad (((CS_1 \cup \mathsf{ran}\,\nu_1) \times \{TS_1\}) \lhd \Omega_1) \cup (((CS_2 \cup \mathsf{ran}\,\nu_2) \times \{TS_2\}) \lhd \Omega_2) \\
&\overset{M_i \eqsim_{\mathsf{m}} M_i'}{=} (((CS_1' \cup \mathsf{ran}\,\nu_1') \times \{TS_1'\}) \lhd \Omega_1') \cup (((CS_2' \cup \mathsf{ran}\,\nu_2') \times \{TS_2'\}) \lhd \Omega_2') \ .
\end{aligned}$$

Using the same arguments as before we can undo the previous steps for M_1' and M_2' and conclude

$$= (((CS' \cup \mathsf{ran}\,\nu') \times \{TS'\})) \lhd \Omega' \ .$$

Case 4 ($(CS \cup \mathsf{ran}\,\nu) \lhd TS = (CS' \cup \mathsf{ran}\,\nu') \lhd TS'$). We apply a similar reasoning as in the previous case: because of composability we can separate and combine the restricted traffic snapshots (second and fourth step). We have

$$\begin{aligned}
&(CS \cup \mathsf{ran}\,\nu) \lhd TS && \text{def. of } M \\
&= ((CS_1 \uplus CS_2) \cup (\mathsf{ran}\,\nu_1 \uplus \mathsf{ran}\,\nu_2)) \lhd (TS_1 \uplus TS_2)) && M_1, M_2 \text{ are composable} \\
&= ((CS_1 \cup \mathsf{ran}\,\nu_1) \lhd TS_1) \uplus ((CS_2 \cup \mathsf{ran}\,\nu_2) \lhd TS_2) && M_i \eqsim_{\mathsf{m}} M_i' \\
&= ((CS_1' \cup \mathsf{ran}\,\nu_1) \lhd TS_1) \uplus ((CS_2' \cup \mathsf{ran}\,\nu_2') \lhd TS_2') && M_1', M_2' \text{ are composable} \\
&= ((CS_1' \uplus CS_2') \cup (\mathsf{ran}\,\nu_1' \uplus \mathsf{ran}\,\nu_2')) \lhd (TS_1' \uplus TS_2')) && \text{def. of } M' \\
&= (CS' \cup \mathsf{ran}\,\nu') \lhd TS'
\end{aligned}$$

Case 5 ($V = V'$). W.l.o.g. we assume that M_1 and M_1' are proper MLSLS models. From $M_1 \eqsim_{\mathsf{m}} M_1'$ we know $V_1 = V_1'$, which implies $V = V'$.

We proofed the case that $M_1 \boxplus M_2$ and $M_1' \boxplus M_2'$ are proper models. The case that both are simple models is analogous. □

The following lemma states that weak equality of MLSLS models is a sufficient condition for being indistinguishable with MLSLS formulas.

Lemma 3.2.30. *Let $M = (CS, TS, \Omega, V, \nu)$, $M' = (CS', TS', \Omega', V', \nu')$ be two proper MLSLS models with $M \approx_{\mathsf{m}} M'$. Then for all MLSLS formulas ϕ the following holds:*

$$M \models \phi \text{ iff } M' \models \phi \ .$$

Proof. The lemma holds because the parts of the model the semantics of the logic checks are equal. We proceed by induction on the structure of ϕ. Note that we only show one direction of the equivalence. The other direction follows because $\approx_{\mathsf{m}}$ is symmetric. In the following let $k \in \mathbb{R}, \gamma, \gamma' \in \mathsf{CVar} \cup \{\mathsf{ego}\}$. Additionally, we assume $M \models \phi$ and $M \approx_{\mathsf{m}} M'$.

Induction base.

Case 1 ($\phi \equiv \mathsf{re}(\gamma)$). From $M \approx_{\mathsf{m}} M'$ we know $\nu = \nu'$, which means $\nu(\gamma) = \nu'(\gamma)$. Further, we know that for $C = \nu(\gamma)$ the spatial properties of C in M and M' and the respective views are equal. That is, we know $\mathsf{se}(C, TS, \Omega) = \mathsf{se}(C, TS', \Omega')$ and $\{C\} \lhd \mathsf{res} = \{C\} \lhd \mathsf{res}'$. It follows that $M \models \mathsf{re}(\gamma)$ implies $M' \models \mathsf{re}(\gamma)$.

Case 2 ($\phi \equiv \mathsf{cl}(\gamma)$). Analogously to $\phi \equiv \mathsf{re}(\gamma)$.

Case 3 ($\phi \equiv \mathsf{free}$). We have $CS = CS'$. Thus, as $M \models \mathsf{free}$, $V = V'$, $CS \lhd TS = CS' \lhd TS'$ and $(CS \times \{TS\}) \lhd \Omega = (CS' \times \{TS'\}) \lhd \Omega'$ it follows that $M' \models \mathsf{free}$. That is, for the cars in CS the spatial aspects in M and M' are equal.

Case 4 ($\phi \equiv \ell = k$). From $V = V'$ and $M \models \ell = k$ it follows that $M' \models \ell = k$.

Case 5 ($\phi \equiv \gamma = \gamma'$). From $\nu = \nu'$ and $M \models \gamma = \gamma'$ it follows that $M' \models \gamma = \gamma'$.

As we pointed out before, the other direction follows because $\approx_{\mathsf{m}}$ is symmetric.

Induction hypothesis.

For the MLSLS formulas ϕ_i with $i \in \{1, 2\}$ and all MLSLS models M_i, M_i' with $M_i \approx_{\mathsf{m}} M_i'$ we have

$$M_i \models \phi_i \text{ iff } M_i' \models \phi_i \ .$$

Induction step.

Case 1 ($\phi \equiv cs : \phi_1$). As $M \models cs : \phi_1$ we have $M_1 \models \phi_1$ with $M_1 = (CS_1, TS, \Omega, V, \nu)$ and $CS_1 = \{\nu(c) \mid c \in cs\}$. As $\nu = \nu'$ and $M \approx_{\mathsf{m}} M'$ we have $M_1 \approx_{\mathsf{m}} M_1'$ with $M_1' = (CS_1, TS', \Omega', V', \nu')$. From the IH we conclude $M_1' \models \phi_1$, which implies $M' \models cs : \phi_1$.

Case 2 ($\phi \equiv \neg\phi_1$). $M \models \neg\phi_1$ implies $M \not\models \phi_1$. From the IH we conclude $M' \not\models \phi_1$, which implies $M' \models \neg\phi_1$.

Case 3 ($\phi \equiv \phi_1 \wedge \phi_2$). $M \models \phi_1 \wedge \phi_2$ implies $M \models \phi_1$ and $M \models \phi_2$. From the IH it follows that $M' \models \phi_1$ and $M' \models \phi_2$, which implies $M' \models \phi_1 \wedge \phi_2$.

Case 4 ($\phi \equiv \exists c.\, \phi_1$). As $M \models \exists c.\, \phi_1$, there is a car $C \in \mathit{CS}$ such that for $M_1 = (\mathit{CS}, \mathit{TS}, \Omega, V, \nu \oplus \{c \mapsto C\})$ we have $M_1 \models \phi_1$. As $M \approx_{\mathsf{m}} M'$ we have for $M_1' = (\mathit{CS}', \mathit{TS}', \Omega', V', \nu' \oplus \{c \mapsto C\})$ that $M_1 \approx_{\mathsf{m}} M_1'$. We conclude from the IH that $M_1' \models \phi_1$, which implies $M' \models \exists c.\, \phi_1$.

Case 5 ($\phi \equiv \phi_1 \frown \phi_2$). As $M \models \phi_1 \frown \phi_2$, there exists a value $r_1 \in [r, r']$ with $M_1 = (\mathit{CS}, \mathit{TS}, \Omega, V_{[r,r_1]}, \nu)$ and $M_1 \models \phi_1$ and $M_2 = (\mathit{CS}, \mathit{TS}, \Omega, V_{[r_1,r']}, \nu) \models \phi_2$. We can mimic the chop in M' such that for $M_1' = (\mathit{CS}', \mathit{TS}', \Omega', V'_{[r,r_1]}, \nu')$ and $M_2' = (\mathit{CS}', \mathit{TS}', \Omega', V'_{[r_1,r']}, \nu')$ we have $M_1 \approx_{\mathsf{m}} M_1'$ and $M_2 \approx_{\mathsf{m}} M_2'$. From the IH it follows that $M_1' \models \phi_1$ and $M_2' \models \phi_2$. As the start point of the extension of M_2' is the endpoint of the extension of M_1' it follows that $M' \models \phi_1 \frown \phi_2$.

Case 6 ($\phi \equiv \begin{smallmatrix}\phi_2\\ \phi_1\end{smallmatrix}$). Assume $M \models \begin{smallmatrix}\phi_2\\ \phi_1\end{smallmatrix}$. We use the semantics from [LH15], which is equivalent to ours. Then there are views V_1, V_2 with $V = V_1 \ominus V_2$ and $M_i \models \phi_i$, where $M_i = (\mathit{CS}, \mathit{TS}, \Omega, V_i, \nu)$. Now, for $M_i' = (\mathit{CS}', \mathit{TS}', \Omega', V_i, \nu')$ we have $M_i \approx_{\mathsf{m}} M_i'$. From the IH we conclude $M_i' \models \phi_i$, which implies that we can join the views again and get $M' \models \begin{smallmatrix}\phi_2\\ \phi_1\end{smallmatrix}$.

As in the base cases, the other direction follows because $\approx_{\mathsf{m}}$ is symmetric. $\square$

3.3 Satisfiability of MLSLS

In this section, we give a decision procedure for lane-unbounded satisfiability for a subset of MLSLS. To do so, we transform formulas to constraints belonging to the first-order theory of mixed linear arithmetic (FOMLA), for which the satisfiability problem is decidable [Wei99; Mon08]. Note that "mixed" indicates that the theory allows for mixing constraints involving real-valued and integer-valued variables. In the considered fragment, scoped formulas are used to enforce that there is a fixed bound on the number of cars that need consideration. In particular, it is required that the formulas free and $\exists c.\, \phi$ occur only inside a scoped formula. Such formulas are called *well-scoped formulas.*

Definition 3.3.1 (Well-scoped MLSLS)**.** The set of *well-scoped* MLSLS formulas is generated by the following grammar:

$$\begin{aligned}
\phi &::= A \mid \neg\phi \mid \phi \wedge \phi \mid \phi \frown \phi \mid \begin{matrix}\phi\\ \phi\end{matrix} \mid cs : \phi' \, , \\
A &::= \ell = k \mid \gamma = \gamma' \mid \mathsf{re}(\gamma) \mid \mathsf{cl}(\gamma) \\
\phi' &::= \mathsf{free} \mid \exists c.\, \phi' \mid A \mid \neg\phi' \mid \phi' \wedge \phi' \mid \phi' \frown \phi' \mid \begin{matrix}\phi'\\ \phi'\end{matrix} \mid cs : \phi' \, ,
\end{aligned}$$

where $c \in \mathsf{CVar}$, $cs \subseteq \mathsf{CVar}$ is finite, $k \in \mathbb{R}_{\geq 0}$ and $\gamma, \gamma' \in \mathsf{CVar} \cup \{\mathsf{ego}\}$. △

We prove that the scope of a model does not affect whether a model satisfies a well-scoped MLSLS formula.

Lemma 3.3.2. *For all CS, CS', all well-scoped formulas ϕ, traffic snapshots TS, sensor functions Ω, views V and valuations ν we have*

$$(CS, TS, \Omega, V, \nu) \models \phi \text{ iff } (CS', TS, \Omega, V, \nu) \models \phi$$

Proof. As we can see in the EBNF of well-scoped formulas, before the scope of a model is applied in the semantics of MLSLS ($\mathsf{free}, \exists c$), a scope operator is encountered. Such a scope operator is evaluated independently of the current scope of the model and replaces the current scope. Hence, the initial scope is never used. □

As an example for a well-scoped MLSLS formula consider the MLSLS formula

$$\{c_1, c_2\} : \left(\left\langle \begin{matrix}\mathsf{free}\\ \mathsf{re}(\mathsf{ego})\end{matrix} \right\rangle \frown \exists c.\, \langle \mathsf{re}(c) \rangle\right)$$

from Example 3.2.21. The formula first restricts the scope to at most two cars. Then it states that ego has a reservation and above of ego there is some space where neither of those two cars has a reservation and in front of ego one of those cars has a reservation. Note that it is possible that c_1 or c_2 and ego may point to the same car.

3.3.1 Transforming MLSLS to Arithmetic Constraints

Now we reduce the satisfiability problem for well-scoped formulas ϕ to the satisfiability of FOMLA formulas. To this end, we introduce FOMLA variables representing the various components of a model. Then the translation function "mimics" the definition of the semantics relation $\models$ in Definition 3.2.20. Note that we use a special type of variables DVar ranging over the set of car identifiers, which we call *car identifier variables.* The set DVar can be represented, for example by variables ranging over the natural numbers. However, we present them as a different type to have a clearer presentation.

The decision procedure in this section is taken from [FHO15]. However, we added a proof of correctness and we adapted the presentation. Here, we collect the variables representing data of a car in a new structure. With this we avoid the global variable naming scheme used in [FHO15]. This, we believe, helps to be more precise and clear.

We collect the car identifier variables under consideration in a set $\mathbb{D} \subseteq \mathsf{DVar}$. The information for each of these variables $D \in \mathbb{D}$ is represented by four real-valued FOMLA variables v_{p}, $v_{\Omega}, v_{\mathsf{s}}$ and v_{a} and three natural number-valued FOMLA variables v_{r1}, v_{r2} and v_{c}. We call these variables *data variables.* We collect these data variables in a tuple $s \in \mathsf{RVar}^4 \times \mathsf{NVar}^3$, where RVar (resp. NVar) is the set of variables ranging over the real numbers (resp. natural numbers). For the tuple s and its primed version s' we refer to the variables therein with v_{r1} and v'_{r1} and so on. For a set of car identifier variables $\mathbb{D}$ we collect the data variables for all cars in a structure $\mathcal{S} : \mathsf{DVar} \to \mathsf{RVar}^4 \times \mathsf{NVar}^3$. For the variables v_{r1}, v_{r2} and v_{c} we assume an additional special value $\bullet$ to indicate that the variable is unassigned. If $\mathcal{S}$ is clear from the context, we shall abbreviate $\mathcal{S}(D)(v_{\mathsf{p}})$ as $D.v_{\mathsf{p}}$ and similar for the other data variables, where $D \in \mathbb{D}$.

We define a data type such that elements of this type contain the information necessary to represent MLSLS models in FOMLA.

Definition 3.3.3 (Proper FOMLA (Data) Tuples)**.** Let

$$\mathrm{T}_{\mathsf{p}}^{\mathsf{mla}} = \mathcal{P}(\mathsf{DVar}) \times \mathsf{NTerm}^2 \times \mathsf{RTerm}^2 \times ((\mathsf{CVar} \cup \{\mathsf{ego}\}) \to \mathsf{DVar}) \times \\ \mathcal{P}(\mathsf{DVar}) \times (\mathsf{DVar} \to \mathsf{RVar}^4 \times \mathsf{NVar}^3) \times \mathsf{DVar} \ ,$$

where NTerm (resp. RTerm) is the set of natural number-valued (resp. real-valued) FOMLA terms. We call $\Upsilon \in \mathrm{T}_{\mathsf{p}}^{\mathsf{mla}}$ with $\Upsilon = (DS, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ a *proper FOMLA data tuple*, where

- $DS \subseteq \mathsf{DVar}$ represents the scope of an MLSLS model,
- $\alpha, \alpha' \in \mathsf{NTerm}$ are natural number-valued terms representing the lanes of the view,

- $\beta, \beta' \in \mathsf{RTerm}$ are real-valued terms representing the extension of the view,
- $f : (\mathsf{CVar} \cup \{\mathsf{ego}\}) \to \mathsf{DVar}$ represents the MLSLS variable valuation (usually ν),
- $\mathbb{D} \subseteq \mathsf{DVar}$ is the set of all car identifier variables in the model,
- $\mathcal{S} : \mathsf{DVar} \to \mathsf{RVar}^4 \times \mathsf{NVar}^3$ contains the set of data variables,
- $D_E \in \mathsf{DVar}$ represents the owner of the view and
- we assume $DS, \mathsf{ran}\, f \subseteq \mathbb{D}, \mathsf{dom}\, \mathcal{S} = \mathbb{D}, D_E \in \mathbb{D}$ and $f(\mathsf{ego}) = D_E$.

For a proper FOMLA data tuple Υ and an assignment κ assigning values to the variables in Υ we refer to (κ, Υ) as a *proper FOMLA tuple.* $\triangle$

We wish to establish a connection between FOMLA tuples and MLSLS models. To achieve this, we define FOMLA formulas representing the sanity conditions for MLSLS models (cf. Assumption 3.2.6). We start to give FOMLA formulas imposing the sanity conditions on traffic snapshots and the sensor function. Note that the upper bound on the number of reservations imposed by the sanity conditions is satisfied because we use two variables to represent the set of reservations, and similarly for the restriction of having at most one claim. Further, the condition that only a finite number of cars have a claim or multiple reservations set is satisfied because of our assumption that we only consider finitely many different cars.

Definition 3.3.4 (FOMLA Sanity Conditions)**.** Let s be the data variables of a car and let $\mathcal{S}$ be the data variables of multiple cars. Then we define

$$\begin{aligned}
\mathsf{sanity}^{\mathsf{mla}}_{\mathbb{D}}(\mathcal{S}) &\equiv \bigwedge_{D \in \mathbb{D}} \mathsf{sanity}^{\mathsf{mla}}(\mathcal{S}(D)) \ , \\
\mathsf{sanity}^{\mathsf{mla}}(s) &\equiv \bigwedge_{j \in \{1,\ldots,6\}} \mathsf{sanity}^{\mathsf{mla}}_j(s) \ , \\
\mathsf{sanity}^{\mathsf{mla}}_1(s) &\equiv v_{\mathsf{c}} \neq \bullet \implies v_{\mathsf{r1}} \neq v_{\mathsf{c}} \wedge v_{\mathsf{r2}} \neq v_{\mathsf{c}} \ , \\
\mathsf{sanity}^{\mathsf{mla}}_2(s) &\equiv v_{\mathsf{r1}} \neq \bullet \vee v_{\mathsf{r2}} \neq \bullet \ , \\
\mathsf{sanity}^{\mathsf{mla}}_3(s) &\equiv v_{\mathsf{r1}} \neq \bullet \wedge v_{\mathsf{r2}} \neq \bullet \implies (v_{\mathsf{r1}} = v_{\mathsf{r2}} + 1 \vee v_{\mathsf{r1}} = v_{\mathsf{r2}} - 1) \ , \\
\mathsf{sanity}^{\mathsf{mla}}_4(s) &\equiv v_{\mathsf{c}} \neq \bullet \implies \left(\begin{array}{rl} & v_{\mathsf{c}} = v_{\mathsf{r1}} + 1 \vee v_{\mathsf{c}} = v_{\mathsf{r1}} - 1 \\ \vee & v_{\mathsf{c}} = v_{\mathsf{r2}} + 1 \vee v_{\mathsf{c}} = v_{\mathsf{r2}} - 1 \end{array} \right) \ ,
\end{aligned}$$

$$\mathsf{sanity}_5^{\mathsf{mla}}(s) \equiv v_{\mathsf{r1}} = \bullet \lor v_{\mathsf{r2}} = \bullet \lor v_{\mathsf{c}} = \bullet \enspace ,$$
$$\mathsf{sanity}_6^{\mathsf{mla}}(s) \equiv v_{\Omega} > 0 \enspace . \qquad \triangle$$

We briefly explain the numbered formulas above. The first requires that claims and reservations are disjoint. The second that at least one reservation is set. The third ensures that if two reservations are set, then they are adjacent. The fourth ensures the same for claims, i.e. if a claim is set, then it is adjacent to a reservation. The fifth requires that a car does not have a claim and two reservations. And the last ensures that the sensor function is strictly positive. We see that $\bullet$ should be represented by a value that is not adjacent to any possible lane to ensure that, e.g. $v_{\mathsf{c}} = v_{\mathsf{r1}} + 1$ is never true when v_{c} or v_{r1} is set to $\bullet$.

Additionally, if two distinct variables $D, D' \in \mathbb{D}$ point to the same car, then the data variables for D and D' must agree. Such properties can be formulated in FOMLA, for example as

$$\mathsf{id\text{-}eq}(\mathcal{S}) \equiv \bigwedge_{D,D' \in \mathbb{D}} D = D' \implies \mathsf{id\text{-}eq}(\mathcal{S}(D), \mathcal{S}(D')) \enspace ,$$
$$\mathsf{id\text{-}eq}(s, s') \equiv \left(\begin{array}{cc} & v_{\mathsf{p}} = v'_{\mathsf{p}} \land v_{\Omega} = v'_{\Omega} \land v_{\mathsf{s}} = v'_{\mathsf{s}} \land v_{\mathsf{a}} = v'_{\mathsf{a}} \\ \land & \{v_{\mathsf{r1}}, v_{\mathsf{r2}}\} = \{v'_{\mathsf{r1}}, v'_{\mathsf{r2}}\} \land v_{\mathsf{c}} = v'_{\mathsf{c}} \end{array} \right) \enspace .$$

Now, given a FOMLA data tuple $\Upsilon = (DS, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ we can define a formula that ensures that any satisfying assignment represents a sane MLSLS model. For this we point out that in MLSLS we allow empty sets of lanes, but not an empty extension. Hence, we require $\beta \leq \beta'$ to ensure that the extension is at least a point interval. We define

$$\mathsf{sane}(\Upsilon) \equiv \mathsf{sanity}(\mathcal{S}) \land \mathsf{id\text{-}eq}(\mathcal{S}) \land \beta \leq \beta' \enspace .$$

We call a FOMLA tuple (κ, Υ) *sane* if $\kappa \models \mathsf{sane}(\Upsilon)$ holds.

For a data tuple $\Upsilon = (DS, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ and $D \in \mathbb{D}$ we use the abbreviation $D.\mathsf{se} \equiv [\mathcal{S}(D)(v_{\mathsf{p}}), \mathcal{S}(D)(v_{\mathsf{p}}) + \mathcal{S}(D)(v_{\Omega})]$. Now we can define our transformation, which works inductively over the structure of MLSLS formulas. In the following we define a function. Still, we use $\equiv$ to define the function to clearly demarcate input and output of the function.

Definition 3.3.5 (Transformation)**.** The *transformation* is given by a function

$$\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}} : \mathrm{T}_{\mathsf{p}}^{\mathsf{mla}} \times \Phi \to \Psi \enspace ,$$

where Φ is the set of MLSLS formulas and Ψ is the set of FOMLA formulas. Let $\Upsilon = (DS, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E) \in \mathrm{T}$, $k \in \mathbb{R}_{\geq 0}$, $\gamma, \gamma' \in \mathsf{CVar} \cup \{\mathsf{ego}\}$, $c \in \mathsf{CVar}$ and $cs \subseteq \mathsf{CVar}$. Then the transformation is given as:

$$\begin{aligned}
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \mathsf{re}(\gamma)) \equiv\ & \beta' > \beta \wedge [\beta, \beta'] \subseteq f(\gamma).\mathsf{se} \wedge \\
& \alpha = \alpha' \wedge (\alpha = f(\gamma).v_{\mathsf{r1}} \vee \alpha = f(\gamma).v_{\mathsf{r2}}) \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \mathsf{cl}(\gamma)) \equiv\ & \beta' > \beta \wedge [\beta, \beta'] \subseteq f(\gamma).\mathsf{se} \wedge \alpha = \alpha' \wedge \alpha = f(\gamma).v_\mathsf{c} \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \mathsf{free}) \equiv\ & \alpha = \alpha' \wedge \beta' > \beta \wedge \bigwedge_{D \in DS} \left(\begin{array}{l} \alpha \notin \{D.v_{\mathsf{r1}}, D.v_{\mathsf{r2}}, D.v_\mathsf{c}\} \\ \vee \\ (\beta, \beta') \cap D.\mathsf{se} = \emptyset \end{array} \right) \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \ell = k) \equiv\ & \beta' - \beta = k \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \gamma = \gamma') \equiv\ & f(\gamma) = f(\gamma') \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, cs : \phi) \equiv\ & \mathsf{tr}_\mathsf{f}^\mathsf{mla}((\{f(c) \mid c \in cs\}, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E), \phi) \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi_1 \wedge \phi_2) \equiv\ & \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi_1) \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi_2) \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \neg\phi) \equiv\ & \neg\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi) \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \exists c.\, \phi) \equiv\ & \bigvee_{D \in DS} \mathsf{tr}_\mathsf{f}^\mathsf{mla}((DS, \alpha, \alpha', \beta, \beta', f \oplus \{c \mapsto D\}, \mathbb{D}, \mathcal{S}, D_E), \phi) \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi_1 \frown \phi_2) \equiv\ & \exists x'' \in \mathbb{R}.\, \beta \leq x'' \leq \beta' \wedge \\
& \qquad \mathsf{tr}_\mathsf{f}^\mathsf{mla}((DS, \alpha, \alpha', \beta, x'', f, \mathbb{D}, \mathcal{S}, D_E), \phi_1) \wedge \\
& \qquad \mathsf{tr}_\mathsf{f}^\mathsf{mla}((DS, \alpha, \alpha', x'', \beta', f, \mathbb{D}, \mathcal{S}, D_E), \phi_2) \\
& \text{where } x'' \text{ is a fresh FOMLA variable} \\
\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \tfrac{\phi_2}{\phi_1}) \equiv\ & \alpha \leq \alpha' \implies \exists y'' \in \mathbb{N}. \\
& \qquad (\alpha - 1 \leq y'' \leq \alpha' \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_2)) \\
& \wedge \\
& \alpha > \alpha' \implies (\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi_1) \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi_2)) \\
& \text{where } \Upsilon_1 = (DS, \alpha, y'', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E), \\
& \qquad \Upsilon_2 = (DS, y'' + 1, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E) \text{ and} \\
& \qquad y'' \text{ is a fresh FOMLA variable} \qquad \triangle
\end{aligned}$$

Notice that the transformation is a direct reflection of the definition of the semantic relation $\models$. On Page 79 we show an example of the transformation.

```
1  $y > y' \implies$
2      $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}((DS_1, y, y', x, x', f, \mathbb{D}, \mathcal{S}, D_E), \mathsf{true})$
3      $\wedge$
4      $\exists x_1 \in [x, x'].$
           /* $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}((DS_1, y, y', x, x_1, f, \mathbb{D}, \mathcal{S}, D_E), \mathsf{re}(\mathsf{ego}))$ */
5          $x_1 > x \wedge [x, x_1] \subseteq D_E.\mathsf{se} \wedge y = y' \wedge y \in \{D_E.v_{\mathsf{r1}}, D_E.v_{\mathsf{r2}}\}$
6          $\wedge$
7          $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}((DS_1, y, y', x_1, x', f, \mathbb{D}, \mathcal{S}, D_E), \mathsf{free} \frown \exists c.\, \mathsf{re}(c))$
8  $\vee$
9  $y \leq y' \implies$
10     $\exists y_1 \in [y-1, y'].$
11         $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}((DS_1, y_1 + 1, y', x, x', f, \mathbb{D}, \mathcal{S}, D_E), \mathsf{true})$
12         $\wedge$
13         $\exists x_3 \in [x, x'].$
               /* $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}((DS_1, y, y_1, x, x_3, f, \mathbb{D}, \mathcal{S}, D_E), \mathsf{re}(\mathsf{ego}))$ */
14             $x_3 > x \wedge [x, x_3] \subseteq D_E.\mathsf{se} \wedge y = y_1 \wedge y \in \{D_E.v_{\mathsf{r1}}, D_E.v_{\mathsf{r2}}\})$
15             $\wedge$
16             $\exists x_4 \in [x_3, x'].$
                   /* $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}((DS_1, y, y_1, x_3, x_4, f, \mathbb{D}, \mathcal{S}, D_E), \mathsf{free})$ */
17                 $\bigwedge_{D \in DS}(y \notin \{D.v_{\mathsf{r1}}, D.v_{\mathsf{r2}}, D.v_{\mathsf{c}}\} \vee (x_3, x_4) \cap D.\mathsf{se} = \emptyset) \wedge y =$
                     $y_1 \wedge x_4 > x_3$
18                 $\wedge$
                   /* $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}((DS_1, y, y_1, x_4, x', f, \mathbb{D}, \mathcal{S}, D_E), \exists c.\, \mathsf{re}(c))$ */
19                 $(x' > x_4 \wedge [x_4, x'] \subseteq D_1.\mathsf{se} \wedge y = y_1 \wedge y \in \{D_1.v_{\mathsf{r1}}, D_1.v_{\mathsf{r2}}\}$
20                 $\vee$
21                 $x' > x_4 \wedge [x_4, x'] \subseteq D_2.\mathsf{se} \wedge y = y_1 \wedge y \in \{D_2.v_{\mathsf{r1}}, D_2.v_{\mathsf{r2}}\})$
```

Figure 3.7: Transformation of $\phi \equiv \{c_1, c_2\} : \left(\overset{\mathsf{true}}{\mathsf{re}(\mathsf{ego}) \frown \mathsf{free} \frown \exists c.\, \mathsf{re}(c)}\right)$ from Example 3.2.21 to FOMLA constraints. For the set of car identifier variables $\mathbb{D} = \{D_1, D_2, D_E\}$ and $f = \{c_1 \mapsto D_1, c_2 \mapsto D_2, \mathsf{ego} \mapsto D_E\}$ let $\Upsilon = (\emptyset, y, y', x, x', f, \mathbb{D}, \mathcal{S}, D_E)$. We want to use our transformation to check if ϕ is lane-unboundedly satisfiable and thus check $\psi \equiv \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \wedge \mathsf{sane}(\Upsilon)$ for satisfiability. As the scope operator does not result in actual constraints, but instead alters the data tuple we pass along, we have $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) = \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \overset{\mathsf{true}}{\phi_1})$ with $\Upsilon_1 = (DS_1, y, y', x, x', f, \mathbb{D}, \mathcal{S}, D_E)$, $DS_1 = \{f(c_1), f(c_2)\}$ and $\phi_1 \equiv \mathsf{re}(\mathsf{ego}) \frown \mathsf{free} \frown \exists c.\, \mathsf{re}(c)$. We continue the explanation on the next page.

Figure 3.7: The vertical chop makes a disjunction over whether $y > y'$ holds. We explore the case $y > y'$. In this case the subformulas are checked on the same data tuple. In Lines 4 to 7 we have the constraints for ϕ_1. For each horizontal chop a new quantified variable is introduced. However, here we only expand the left operator with the variable x_1. In Line 5 we see the constraints for $\mathsf{re}(\mathsf{ego})$. We see that $y > y'$ contradicts $y = y'$. Hence, this branch is unsatisfiable.

We explore $y \leq y'$. In Line 10 we search for a vertical chop point, represented by the variable y_1. From Lines 14 and 17 and Lines 19 to 21 we see that we should choose y_1 equal to y. Next we introduce a variable x_3 for the first horizontal chop operator. Note that we introduced two variables x_1 and x_3 to represent this chop point. The complete formula ψ is satisfiable, which implies that ϕ is lane-unboundedly satisfiable. In a satisfying assignment we represent at least two cars. We may assign values such that D_E and D_1 both have a reservation on the lane given by the value of y. Furthermore, the right end of the safety envelope of D_E has to be smaller than the position of D_1, i.e. $D_E.v_{\mathsf{p}} + D_E.v_{\Omega} < D_1.v_{\mathsf{p}}$ has to hold. To D_2 and the accompanying data variables we may assign the same values as to the variables of D_E or D_1 or we might use it to represent a third car on some lane above the lane of D_E and D_1. Further, the interval of lanes contains at least one lane ($y \leq y'$) and the extension has to overlap with the reservations of D_E and D_1.

Let us consider what happens if we assign D_1, D_2, D_E the same value. In this case $\mathsf{sane}(\Upsilon)$ requires that these variables represent the same car. Then the constraints in Line 17 contradict the combined constraints in Line 14 and Lines 19 to 21. More specifically, in Line 17 we require for $D \in \{D_1, D_2\}$ that $y \notin \{D.v_{\mathsf{r1}}, D.v_{\mathsf{r2}}, D.v_{\mathsf{c}}\}$ or $(x_3, x_4) \cap D.\mathsf{se} = \emptyset$. The first option is prevented by the constraint $y \in \{D_E.v_{\mathsf{r1}}, D_E.v_{\mathsf{r2}}\}$. The second option is prevented by the combination of $[x, x_3] \subseteq D_E.\mathsf{se}$, $[x_4, x'] \subseteq D.\mathsf{se}$, $x < x_3 < x_4 < x'$ and $D.\mathsf{se}$ being an interval.

3.3.2 Correctness of the Transformation

In this section we prove the correctness of our construction, by which we mean that properties of our construction translate to MLSLS properties and vice versa. On the intuitive side it seems clear that our construction is correct because we simply imitate the semantics of MLSLS within FOMLA. However, there are some subtleties. For example, on FOMLA side we represent cars as intermediate variables D, which take as values car identifiers $C \in \mathbb{I}$. Hence, we operate on these intermediate variables rather than on car identifiers. For this reason we may have cars that are both in *and* not in a scope, i.e. for (κ, Υ) with $\Upsilon = (DS, y, y', x, x', f, \mathbb{D}, \mathcal{S}, D_E)$ we may have $\kappa(D) = \kappa(D')$ and $D \in DS, D' \notin DS$. This seems undesirable, but we show that nevertheless our construction is correct.

For a well-scoped MLSLS formula ϕ we formalise correctness of the transformation as

$$\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \wedge \mathsf{sane}(\Upsilon) \text{ is satisfiable iff } \phi \text{ is lane-unboundedly satisfiable } . \tag{3.23}$$

To prove this statement an induction over the structure of ϕ seems natural. However, to prove both directions of the bi-implication we have to provide a satisfying MLSLS model in one direction and a satisfying assignment in the other direction; but neither is provided by the induction hypothesis. Hence, we have to strengthen the lemma and use concrete models and assignments. Thus, we need methods to transform models into FOMLA assignments and vice versa. We first define these transformations, then we prove some helpful properties and then we prove the strengthened version of Equation (3.23), which we use to prove Equation (3.23).

Before we define the transformations between MLSLS models and FOMLA assignments we prove some basic properties of our arithmetic representation of MLSLS. We point out that our arithmetic representation consists of variables to which we assign values via a variable assignment. This will complicate things somewhat.

Similar to MLSLS we define simple FOMLA tuples where information about the view and ego-car is not present.

Definition 3.3.6 (Simple FOMLA Tuple)**.** A *simple FOMLA data tuple* $\Upsilon = (DS, f, \mathbb{D}, \mathcal{S})$ has the type

$$\mathrm{T}_{\mathsf{s}}^{\mathsf{mla}} = \mathcal{P}(\mathsf{DVar}) \times (\mathsf{CVar} \to \mathsf{DVar}) \times \mathcal{P}(\mathsf{DVar}) \times (\mathsf{DVar} \to \mathsf{RVar}^4 \times \mathsf{NVar}^3) \ ,$$

where NVar and RVar are the set of natural number-valued and real-valued variables. We assume $DS, \mathsf{ran}\, f \subseteq \mathbb{D}$ and $\mathsf{dom}\, \mathcal{S} = \mathbb{D}$. For a simple FOMLA data tuple Υ and a FOMLA assignment κ that assigns values to the variables in Υ we refer to (κ, Υ) as a *simple FOMLA tuple.* $\triangle$

We denote the empty simple FOMLA tuple with $(\kappa_\emptyset, \Upsilon_\emptyset) = (\emptyset, \emptyset, \emptyset, \emptyset)$. We define a conversion of a proper FOMLA tuple to a simple one. The definition is more involved than for MLSLS because we work with variables rather than values.

Definition 3.3.7. Let (κ, Υ) with $\Upsilon = (DS, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ be a proper FOMLA tuple. Then we define

$$\begin{aligned} \mathsf{simple}(\kappa, \Upsilon) = (\kappa', \Upsilon') \text{ with} & \\ & \Upsilon' = (DS, \{\mathsf{ego}\} \mathbin{⩤} f, \mathbb{D}, \mathcal{S}) \text{ and} \\ & \kappa' = (\mathbb{D} \cup \bigcup_{D \in \mathbb{D}} \mathsf{set}(\mathcal{S}(D))) \lhd \kappa \ , \end{aligned}$$

where set transforms a tuple to a set (cf. Page 5). $\triangle$

In the definition above we use $(\mathbb{D} \cup \bigcup_{D \in \mathbb{D}} \mathsf{set}(\mathcal{S}(D)) \lhd \kappa$ to remove all variables from the assignment not representing a car or data of a car from $\mathbb{D}$. This is necessary because in general a data tuple has terms representing the view, rather than variables. We extend this definition such that $\mathsf{simple}(\kappa, \Upsilon) = (\kappa, \Upsilon)$ if κ, Υ are simple.

Similar to MLSLS models we define weak equality for simple FOMLA assignments and data tuples. However, for FOMLA the definition of weak equality is more involved than for MLSLS. We specify that two FOMLA tuples are weakly equal if their only difference is that some of the car identifier variables are “split”. We formalise this below. First we introduce

$$\begin{aligned} \mathsf{eq}(\kappa(s), \kappa'(s')) \equiv \{\kappa(v_{\mathsf{r1}}), \kappa(v_{\mathsf{r2}})\} = \{\kappa'(v_{\mathsf{r1}}), \kappa'(v'_{\mathsf{r2}})\} \land \kappa(v_{\mathsf{c}}) = \kappa'(v'_{\mathsf{c}}) \land \\ \kappa(v_{\mathsf{p}}) = \kappa'(v'_{\mathsf{p}}) \land \kappa(v_\Omega) = \kappa'(v'_\Omega) \land \kappa(v_{\mathsf{s}}) = \kappa'(v'_{\mathsf{s}}) \land \kappa(v_{\mathsf{a}}) = \kappa'(v'_{\mathsf{a}}) \end{aligned}$$

as an abbreviation for comparing the values assigned two sets of data variables by their assignments. Note that here we relate two different assignments. Hence, the formula $\mathsf{id\text{-}eq}$ from Page 77, which does something similar, does not apply. Further, note that we require the set of represented reservations to be equal, rather than comparing the reservation variables directly.

Definition 3.3.8 (FOMLA Weak Equality)**.** Let $\Upsilon_i = (DS_i, f_i, \mathbb{D}_i, \mathcal{S}_i)$ for $i \in \{1, 2\}$ be two simple FOMLA data tuples and let κ_1 and κ_2 be two assignments. Then we define that (κ_1, Υ_1) and (κ_2, Υ_2) are *weakly equal* (denoted $(\kappa_1, \Upsilon_1) \eqsim_{\mathsf{mla}} (\kappa_2, \Upsilon_2)$) as

$$(\kappa_1, \Upsilon_1) \eqsim_{\mathsf{mla}} (\kappa_2, \Upsilon_2) \text{ iff}$$
$$\left(\begin{array}{ll} & ((\mathsf{ran}\, f_1) \lhd \kappa_1) \circ f_1 = ((\mathsf{ran}\, f_2) \lhd \kappa_2) \circ f_2 \wedge \kappa_1(\!|DS_1|\!) = \kappa_2(\!|DS_2|\!) \\ \wedge & \forall D_1 \in \mathbb{D}_1 . \forall D_2 \in \mathbb{D}_2 . \\ & \qquad \kappa_1(D_1) = \kappa_2(D_2) \text{ implies } \mathsf{eq}(\kappa_1(\mathcal{S}_1(D_1)), \kappa_2(\mathcal{S}_2(D_2))) \end{array}\right)$$

We lift this to proper FOMLA tuples $\Upsilon_i = (DS_i, \alpha_i, \alpha'_i, \beta_i, \beta'_i, f_i, \mathbb{D}_i, \mathcal{S}_i, D_{Ei})$ with $i \in \{1, 2\}$. Then

$$(\kappa_1, \Upsilon_1) \eqsim_{\mathsf{mla}} (\kappa_2, \Upsilon_2) \text{ iff } \left(\begin{array}{ll} & \mathsf{simple}(\kappa_1, \Upsilon_1) \eqsim_{\mathsf{mla}} \mathsf{simple}(\kappa_1, \Upsilon_1) \\ \wedge & \kappa_1(D_{E1}) = \kappa_2(D_{E2}) \\ \wedge & \kappa_1(\alpha_1) = \kappa_2(\alpha_2) \wedge \kappa_1(\alpha'_1) = \kappa_2(\alpha'_2) \\ \wedge & \kappa_1(\beta_1) = \kappa_2(\beta_2) \wedge \kappa_1(\beta'_1) = \kappa_2(\beta'_2) \end{array}\right)$$

△

For simple FOMLA tuples we ensure with $((\mathsf{ran}\, f_1) \lhd \kappa_1) \circ f_1 = ((\mathsf{ran}\, f_2) \lhd \kappa_2) \circ f_2$ that the same car variables map through f_i and κ_i to the same car identifier, even if the intermediate car identifier variable D is different. Note that the type of the function is $(\mathsf{ran}\, f_i \lhd \kappa_i) \circ f_i : \mathsf{CVar} \to \mathbb{I}$ with $i \in \{1, 2\}$, i.e. the domain of κ and the range of f which both are DVar, is hidden within the function. With $\kappa_1(\!|DS_1|\!) = \kappa_2(\!|DS_2|\!)$ we ensure that the FOMLA representation of the scope is equal, i.e. that DS_1 and DS_2 represent the same set of car identifiers. Note that we only ensure equality of the values assigned and ignore the variable names used. And with the last line we ensure that if two car identifier variables point to the same car identifier, then their data variables should have equal values. For proper FOMLA tuples we additionally ensure that the view is equal, i.e. that the owner of the view, the extension and lanes in the view are equal.

We state a lemma that the FOMLA constraints for an MLSLS formula cannot distinguish weakly equal FOMLA assignments and data tuples.

Lemma 3.3.9. *For proper FOMLA tuples* $(\kappa_1, \Upsilon_1), (\kappa_2, \Upsilon_2)$ *with* $(\Upsilon_1, \kappa_1) \eqsim_{\mathsf{mla}} (\Upsilon_2, \kappa_2)$ *and all MLSLS formulas* ϕ *we have*

$$\kappa_1 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi) \text{ iff } \kappa_2 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2, \phi) \ .$$

Proof. We perform an induction over the structure of the MLSLS formula ϕ. Let $\Upsilon_i = (DS_i, \alpha_i, \alpha'_i, \beta_i, \beta'_i, f_i, \mathbb{D}_i, \mathcal{S}_i, D_{Ei})$ with $i \in \{1, 2\}$.
Induction base.
Case 1 ($\phi \equiv \mathsf{free}$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \mathsf{free})$. We know $\kappa_1(\!|DS_1|\!) = \kappa_2(\!|DS_2|\!)$. As the views represented in the assignments are equal and all cars represented in DS_i with $i \in \{1, 2\}$ have equal values assigned because $(\Upsilon_1, \kappa_1) \eqsim_\mathsf{mla} (\Upsilon_2, \kappa_2)$ holds, it follows that $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \mathsf{free})$.

Case 2 ($\phi \equiv \ell = k$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \ell = k)$. As $(\Upsilon_1, \kappa_1) \eqsim_\mathsf{mla} (\Upsilon_2, \kappa_2)$ ensures that the views represented in both assignments are equal we conclude $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \ell = k)$.

Case 3 ($\phi \equiv \mathsf{re}(\gamma)$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \mathsf{re}(\gamma))$. Because $(\Upsilon_1, \kappa_1) \eqsim_\mathsf{mla} (\Upsilon_2, \kappa_2)$ holds, we know $\kappa_1(f_1(\gamma)) = \kappa_2(f_2(\gamma))$. As the views represented in both assignments are equal and as the values assigned to the data variables are equal, we conclude $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \mathsf{re}(\gamma))$.

Case 4 ($\phi \equiv \mathsf{cl}(\gamma)$). Analogous to the case $\phi \equiv \mathsf{re}(\gamma)$.

Case 5 ($\phi \equiv \gamma = \gamma'$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \gamma = \gamma')$. From $((\mathsf{ran}\, f_1) \lhd \kappa_1) \circ f_1 = ((\mathsf{ran}\, f_2) \lhd \kappa_2) \circ f_2$ we conclude that γ and γ' are mapped through κ_i and f_i with $i \in \{1, 2\}$ to the same value, i.e. $\kappa_1(f_1(\gamma)) = \kappa_2(f_2(\gamma'))$. This implies $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \gamma = \gamma')$.

We have proven the lemma for one direction of the base cases. The other direction follows because the property is symmetric.
Induction hypothesis.
We have the following induction hypothesis: for ϕ_i and all FOMLA tuples (κ_i, Υ_i) with $\Upsilon_i = (DS_i, \alpha_i, \alpha'_i, \beta_i, \beta'_i, f_i, \mathbb{D}_i, \mathcal{S}_i, D_{Ei})$, $i \in \{1, 2\}$ and $(\Upsilon_1, \kappa_1) \eqsim_\mathsf{mla} (\Upsilon_2, \kappa_2)$ we have

$$\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_i) \text{ iff } \kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_i) \ .$$

Induction step.
Case 1 ($\phi \equiv cs{:}\phi_1$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, cs{:}\phi_1)$. Evaluating the scope operator we get $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon'_1, \phi_1)$, where $\Upsilon'_1 = (DS'_1, \alpha_1, \alpha'_1, \beta_1, \beta'_1, f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ and $DS'_1 = \{f_1(c) \mid c \in cs\}$. From $(\Upsilon_1, \kappa_1) \eqsim_\mathsf{mla} (\Upsilon_2, \kappa_2)$ we know that the car variables are mapped to the same car identifiers through κ_i, f_i, i.e. for $DS'_2 = \{f_2(c) \mid c \in cs\}$ we have $\kappa_1(\!|DS'_1|\!) = \kappa_2(\!|DS'_2|\!)$. Hence, $(\Upsilon'_1, \kappa_1) \eqsim_\mathsf{mla} (\Upsilon'_2, \kappa_2)$ and we can use the IH to conclude $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon'_2, \phi_1)$. At last, we can pull the scope variables out and get $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, cs : \phi_1)$.

Case 2 ($\phi \equiv \phi_1 \wedge \phi_2$). We assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1 \wedge \phi_2)$, which implies $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_i)$ with $i \in \{1,2\}$. We apply the IH to conclude $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_i)$, which in turn implies $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_1 \wedge \phi_2)$.

Case 3 ($\phi \equiv \neg\phi_1$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \neg\phi_1)$, which implies $\kappa_1 \not\models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1)$. We apply the IH to conclude $\kappa_2 \not\models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_1)$, which implies $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \neg\phi_1)$.

Case 4 ($\phi \equiv \exists c.\, \phi_1$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \exists c.\, \phi_1)$. Then $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1', \phi_1)$ with $\Upsilon_1' = (DS_1, \alpha_1, \alpha_1', \beta_1, \beta_1', f_1', \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ and $f_1' = f_1 \oplus \{c \mapsto D_1\}$ for some $D_1 \in \mathbb{D}_1$. From $(\Upsilon_1, \kappa_1) \eqsim_\mathsf{mla} (\Upsilon_2, \kappa_2)$ we know that there is $D_2 \in \mathbb{D}_2$ with $\kappa_1(D_1) = \kappa_2(D_2)$. Hence, for $f_2' = f_2 \oplus \{c \mapsto D_2\}$ and $\Upsilon_2' = (DS_2, \alpha_2, \alpha_2', \beta_2, \beta_2', f_2', \mathbb{D}_2, \mathcal{S}_2, D_{E2})$ we have $(\Upsilon_1', \kappa_1) \eqsim_\mathsf{mla} (\Upsilon_2', \kappa_2)$. Thus, we can apply the IH to conclude $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2', \phi_1)$, which implies that we can put c below a car quantifier again, i.e. $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \exists c.\, \phi_1)$.

Case 5 ($\phi \equiv \phi_1 \frown \phi_2$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1 \frown \phi_2)$. By evaluating the horizontal chop operator we can conclude that there is a value $r \in [\kappa_1(\beta_1), \kappa_1(\beta_1')]$ where we can chop the view into a left subview (resp. right subview) such that in this view ϕ_1 (resp. ϕ_2) holds. Thus, for an assignment κ_1' that extends κ_1 by mapping a fresh real-valued variable x_1'' to r, and two data tuples $\Upsilon_1' = (DS_1, \alpha_1, \alpha_1', \beta_1, x_1'', f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ and $\Upsilon_1'' = (DS_1, \alpha_1, \alpha_1', x_1'', \beta_1', f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ the new assignment κ_1' satisfies $\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1', \phi_1)$ and $\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1'', \phi_2)$. We mirror the introduction of the new variable x_1'' for Υ_2. W.l.o.g. we assume that x_1'' does not occur in Υ_2. Then, for $\Upsilon_2' = (DS_2, \alpha_2, \alpha_2', \beta_2, x_2'', f_2, \mathbb{D}_2, \mathcal{S}_2, D_{E2})$, $\Upsilon_2'' = (DS_2, \alpha_2, \alpha_2', x_2'', \beta_2', f_2, \mathbb{D}_2, \mathcal{S}_2, D_{E2})$, and $\kappa_2' = \kappa_2 \oplus \{x_2'' \mapsto r\}$ we have $(\Upsilon_1', \kappa_1') \eqsim_\mathsf{mla} (\Upsilon_2', \kappa_2')$ and $(\Upsilon_1'', \kappa_1') \eqsim_\mathsf{mla} (\Upsilon_2'', \kappa_2')$ because we introduced on both sides of the weak equalities a new variable representing a chop point and assigned it the same value in both assignments. Now we can apply the IH to conclude $\kappa_2' \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2', \phi_1)$ and $\kappa_2' \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2'', \phi_2)$. We reintroduce the quantifier over the chop point and get $\kappa_2 \models \exists x_2''.\, \beta_2 \leq x_2'' \leq \beta_2' \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2', \phi_1) \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2'', \phi_2)$, which implies $\kappa_2 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_1 \frown \phi_2)$.

Case 6 ($\phi \equiv \begin{smallmatrix}\phi_2\\\phi_1\end{smallmatrix}$). Assume $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \begin{smallmatrix}\phi_2\\\phi_1\end{smallmatrix})$. We make a case distinction on whether $\kappa_1(\alpha_1) > \kappa_1(\alpha_1')$.

If $\kappa_1(\alpha_1) > \kappa_1(\alpha_1')$, then $\kappa_1 \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_2)$. This case is analogues to the case $\phi \equiv \phi_1 \wedge \phi_2$.

If $\kappa_1(\alpha_1) \leq \kappa_1(\alpha_1')$, then this case is analogous to the case $\phi \equiv \phi_1 \frown \phi_2$.

We finished one direction of the induction step. Again, the other direction follows because the property is symmetric. $\square$

For a possibly simple FOMLA tuple (κ, Υ) we denote the set of cars represented in it by $\mathsf{cars}(\kappa, \Upsilon) = \kappa(\!|\mathbb{D}|\!)$ and the size as $|(\kappa, \Upsilon)| = |\,\mathsf{cars}(\kappa, \Upsilon)|$.

We define that two simple FOMLA tuples are composable if the MLSLS models they represent are composable.

Definition 3.3.10 (Composable FOMLA Tuples)**.** Let (κ_i, Υ_i) with $i \in \{1, 2\}$ and $\Upsilon_i = (DS_i, f_i, \mathbb{D}_i, \mathcal{S}_i)$ be two simple FOMLA tuples. Then we say that (κ_i, Υ_i) are *composable* iff $\kappa_1(\!|\mathbb{D}_1|\!) \cap \kappa_2(\!|\mathbb{D}_2|\!) = \emptyset$ and $\mathsf{dom}\, f_1 \cap \mathsf{dom}\, f_2 = \emptyset$.

We extend this to define that a simple FOMLA tuple (κ_1, Υ_1) and a proper one (κ_2, Υ_2) are composable if $\mathsf{simple}(\kappa_2, \Upsilon_2)$ and (κ_1, Υ_1) are composable. $\triangle$

For simple FOMLA tuples we have simple sanity conditions. Let

$$\mathsf{sane_s}(\Upsilon) \equiv \mathsf{sanity}(\mathcal{S}) \wedge \mathsf{id\text{-}eq}(\mathcal{S})$$

We call a simple FOMLA tuple (κ, Υ) *sane* if $\kappa \models \mathsf{sane_s}(\Upsilon)$ holds.

For the following definition we assume that common FOMLA variables are simply renamed so that the assignments use disjoint variables. The definition states that we combine two variable assignments and data tuples by combining all variables. As for MLSLS we allow at most one operand to be proper.

Definition 3.3.11 (Joining FOMLA Tuples)**.** For $i \in \{1, 2\}$ let (κ_i, Υ_i) with $\Upsilon_i = (DS_i, f_i, \mathbb{D}_i, \mathcal{S}_i)$ be two composable simple FOMLA tuples. Then we define the *disjoint union* of (κ_1, Υ_1) and (κ_2, Υ_2) (denoted $(\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)$) as

$$(\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2) = (\kappa, (DS, f, \mathbb{D}, \mathcal{S}))$$

with $\kappa = \kappa_1 \uplus \kappa_2$ $DS = DS_1 \uplus DS_2$, $f = f_1 \uplus f_2$, $\mathbb{D} = \mathbb{D}_1 \uplus \mathbb{D}_2$ and $\mathcal{S} = \mathcal{S}_1 \uplus \mathcal{S}_2$.

We extend this to the case that in $(\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)$ at most one tuple is a proper FOMLA tuple. Then the result also is proper and takes the view and D_E from the proper tuple. $\triangle$

As for MLSLS we define the *restriction* and *anti-restriction* of FOMLA tuples, denoted by $\oplus$ and $\ominus$.

Definition 3.3.12. Let (κ, Υ) with $\Upsilon = (DS, f, \mathbb{D}, \mathcal{S})$ be a simple FOMLA tuple, let $CS \subseteq \mathbb{I}$ and let $DS' = \mathsf{dom}(\kappa \rhd CS)$ be those variables from $\mathbb{D}$ that are mapped to identifiers in CS. Then we define

$$(\kappa,\Upsilon) \oplus\!\!\!\!\bigcirc CS = (\kappa',(DS \cap DS', f \rhd DS', \mathbb{D} \cap DS', DS' \lhd \mathcal{S})) \text{ with}$$
$$\kappa' = (DS' \cup \bigcup_{D\in DS'} \mathsf{set}(\mathcal{S}(D))) \lhd \kappa \ ,$$
$$(\kappa,\Upsilon) \ominus CS = (\kappa,\Upsilon) \oplus\!\!\!\!\bigcirc (\mathbb{I} \setminus CS) \ .$$

We extend both definitions to the case that (κ,Υ) is proper. For $(\kappa,\Upsilon) \oplus\!\!\!\!\bigcirc CS$ the result is proper if $\kappa(D_E) \in CS$ and otherwise it is simple. If the result is proper, then κ' also needs to retain the variables from the terms representing the view. △

The following proposition holds for the sanity conditions on FOMLA tuples because we do not change the values of a car and similarly for the view, if the resulting model has a view.

Proposition 3.3.13 (FOMLA Operations Preserve Sanity)**.** *For all sane, composable and possibly simple FOMLA tuples $(\kappa_1,\Upsilon_1),(\kappa_2,\Upsilon_2)$ and sets $CS \subseteq \mathbb{I}$ the FOMLA tuples $(\kappa_1,\Upsilon_1) \oplus (\kappa_2,\Upsilon_2)$, $(\kappa_1,\Upsilon_1) \ominus CS$ and $(\kappa_1,\Upsilon_1) \oplus\!\!\!\!\bigcirc CS$ satisfy the proper or simple sanity conditions, depending on whether the resulting FOMLA tuple is proper or simple.*

Just like for MLSLS, for our operations on FOMLA representations of MLSLS models properties like associativity are desirable. However, on FOMLA side we represent data as values assigned to variables and we do not constrain the naming of these variables. Hence, the algebraic properties involving two nonempty FOMLA tuples only hold up to renaming of FOMLA variable names. We denote this *equality up to renaming* with $=_\mathsf{r}$.

Example 3.3.14. Consider for $i \in \{1,2\}$ the simple FOMLA tuples (κ_i,Υ_i), where $\Upsilon_i = (\{D_i\},\{c \mapsto D_i\},\{D_i\},\{D_i \mapsto s\})$, $D_i \in \mathsf{DVar}, c \in \mathsf{CVar}$, $D_1 \neq D_2$, s is a tuple of data variables and $\kappa_i = \{D_i \mapsto C\} \uplus \kappa$ with $C \in \mathbb{I}$ and κ assigns values to the variables in s. Then, (κ_1,Υ_1) is not equal to (κ_2,Υ_2) because of $D_1 \neq D_2$. However, the two FOMLA tuples are equal up to FOMLA variable names (denoted $(\kappa_1,\Upsilon_1) =_\mathsf{r} (\kappa_2,\Upsilon_2)$), as the values assigned to the FOMLA variables is equal. Note that we do require equality w.r.t. the names of car variables (here c). △

For the following lemmas let (κ_i,Υ_i) with $i \in \{1,2,3\}$ be possibly simple composable FOMLA tuples and let $CS, CS' \subseteq \mathbb{I}$. The properties can be proven similarly to the corresponding properties on MLSLS side because the operations have very similar definitions. However, in FOMLA we have to go through variables, which makes the proofs more cumbersome. Thus, we do not provide the proofs here. Let us proceed to the properties. The operation $\oplus$ is associative and commutative, i.e.

$$(\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2) =_{\mathsf{r}} (\kappa_2, \Upsilon_2) \oplus (\kappa_1, \Upsilon_1) \tag{3.24}$$
$$((\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)) \oplus (\kappa_3, \Upsilon_3) =_{\mathsf{r}} (\kappa_1, \Upsilon_1) \oplus ((\kappa_2, \Upsilon_2) \oplus (\kappa_3, \Upsilon_3)) \tag{3.25}$$

Additionally, multiple $\obar$ and $\ominus$ operations can be joined by combining the set of car identifiers. That is,

$$((\kappa_1, \Upsilon_1) \obar CS) \obar CS' = (\kappa_1, \Upsilon_1) \obar (CS \cap CS') \ , \tag{3.26}$$
$$((\kappa_1, \Upsilon_1) \ominus CS) \ominus CS' = (\kappa_1, \Upsilon_1) \ominus (CS \cup CS') \ . \tag{3.27}$$

The empty FOMLA tuple is composable with every FOMLA tuple (including itself) and is neutral under all operations. That is,

$$(\kappa_1, \Upsilon_1) \oplus (\kappa_\emptyset, \Upsilon_\emptyset) = (\kappa_1, \Upsilon_1) \ , \tag{3.28}$$
$$(\kappa_\emptyset, \Upsilon_\emptyset) \ominus CS = (\kappa_\emptyset, \Upsilon_\emptyset) \ , \tag{3.29}$$
$$(\kappa_\emptyset, \Upsilon_\emptyset) \obar CS = (\kappa_\emptyset, \Upsilon_\emptyset) \ . \tag{3.30}$$

Furthermore, $\ominus$ and $\obar$ distribute over $\oplus$. However, this is not the classical distributive property, as $\ominus$ and $\obar$ operate on different domains than $\oplus$. Formally,

$$((\kappa_1, \Upsilon_1) \obar CS) \oplus ((\kappa_2, \Upsilon_2) \obar CS) =_{\mathsf{r}} ((\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)) \obar CS \ , \tag{3.31}$$
$$((\kappa_1, \Upsilon_1) \ominus CS) \oplus ((\kappa_2, \Upsilon_2) \ominus CS) =_{\mathsf{r}} ((\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)) \ominus CS \ . \tag{3.32}$$

At last, $\obar$ and $\ominus$ are associative with each other, i.e.

$$((\kappa_1, \Upsilon_1) \obar CS) \ominus CS' = ((\kappa_1, \Upsilon_1) \obar CS \setminus CS') = ((\kappa_1, \Upsilon_1) \ominus CS') \obar CS \ .$$

We first define how an MLSLS model can be extracted from a FOMLA tuple representing a model with a single car. Note that this definition is defined only for sane FOMLA tuples. Hence, we require that the assignment satisfies $\mathsf{sane}(\mathcal{S})$ (resp. $\mathsf{sane}_{\mathsf{s}}(\mathcal{S})$ if it is simple) from Pages 77 and 86. Further, note that our sensor function ignores its second input, the traffic snapshot. This is because we care only about static traffic configurations here. Note that we use the index E to denote an elementary (or base) version of a function that we extend later.

Definition 3.3.15 (FOMLA to MLSLS)**.** Let (κ, Υ) be a simple FOMLA tuple with $\Upsilon = (DS, f, \mathbb{D}, \mathcal{S})$ and $\kappa \models \mathsf{sane_s}(\mathcal{S})$ and $|(\kappa, \Upsilon)| = 1$ with $\mathsf{sane_s}$ as defined on Page 86. Then we define $\mathsf{mla2m_E}(\kappa, \Upsilon) = (CS, TS, \Omega, \nu)$, where

$$\begin{aligned}
CS &= \{\kappa(D) \mid D \in DS\} \ , \\
TS &= (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc}) \text{ where} \\
&\qquad \mathsf{res} = \{\kappa(D) \mapsto \{\kappa(D.v_{\mathsf{r1}}), \kappa(D.v_{\mathsf{r2}})\} \setminus \{\bullet\} \mid D \in \mathbb{D}\} \ , \\
&\qquad \mathsf{clm} = \{\kappa(D) \mapsto \{\kappa(T.v_{\mathsf{c}})\} \setminus \{\bullet\} \mid D \in \mathbb{D}\} \ , \\
&\qquad \mathsf{pos} = \{\kappa(D) \mapsto \kappa(D.v_{\mathsf{p}}) \mid D \in \mathbb{D}\} \ , \\
&\qquad \mathsf{spd} = \{\kappa(D) \mapsto \kappa(D.v_{\mathsf{s}}) \mid D \in \mathbb{D}\} \ , \\
&\qquad \mathsf{acc} = \{\kappa(D) \mapsto \kappa(D.v_{\mathsf{a}}) \mid D \in \mathbb{D}\} \ , \\
\Omega &= \{(\kappa(D), TS) \mapsto \kappa(D.v_{\Omega}) \mid D \in \mathbb{D}\} \ , \\
\nu &= \{c \mapsto \kappa(f(c)) \mid c \in \mathsf{dom}\, f\} \ .
\end{aligned}$$

We lift this definition to proper FOMLA tuples. Let (κ, Υ) be a proper FOMLA tuple with $\Upsilon = (DS, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ and $\kappa \models \mathsf{sane}(\mathcal{S})$ and $|(\kappa, \Upsilon)| = 1$. Then we define $\mathsf{mla2m_E}(\kappa, \Upsilon) = (CS, TS, \Omega, V, \nu)$, where CS, TS, Ω are defined as above and $V = ([\kappa(\alpha), \kappa(\alpha')], [\kappa(\beta), \kappa(\beta')], \kappa(D_E))$ and $\nu = \{\gamma \mapsto \kappa(f(\gamma)) \mid \gamma \in \mathsf{dom}\, f\}$ △

We lift the previous definition to FOMLA tuples representing multiple cars.

Definition 3.3.16 (FOMLA to MLSLS)**.** Let (κ, Υ) be a simple or a proper FOMLA tuple. Then we define

$$\mathsf{mla2m}(\kappa, \Upsilon) = \begin{cases} M_{\emptyset} & \text{if } |(\kappa, \Upsilon)| = 0 \\ (\mathsf{mla2m_E}((\kappa, \Upsilon) \oplus \{C\}) \boxplus & \text{if } |(\kappa, \Upsilon)| \geq 1 \\ \qquad \mathsf{mla2m}((\kappa, \Upsilon) \ominus \{C\}) & \text{where } C \in \kappa(\!|\mathbb{D}|\!)) \end{cases}$$

△

In the definition of $\mathsf{mla2m}$ we use the function $\boxplus$, which has requirements on its inputs, i.e. it is only defined when both MLSLS models are composable. However, it is not difficult to see that $(\kappa, \Upsilon) \oplus CS$ and $(\kappa, \Upsilon) \ominus CS$ are composable FOMLA tuples and that being composable carries over from FOMLA tuples through $\mathsf{mla2m}$ to MLSLS models.

To define the transformation of MLSLS models to FOMLA tuples we use the following assumption. The assumptions serves to simplify the presentation.

Assumption 3.3.17. Given a model $M = (CS, TS, \Omega, V, \nu)$, we assume for all cars $C \in \mathbb{I}$ that $\mathsf{res}(C) = \{l_1, l_2\}$ or $\mathsf{res}(C) = \{l_1, \bullet\}$ with $l_1, l_2 \in \mathbb{L}$ and $\bullet \notin \mathbb{L}$ and $\mathsf{clm}(C) = \{l_3\}$ or $\mathsf{clm}(C) = \{\bullet\}$. $\triangle$

We define how to transform an MLSLS model to a FOMLA tuple. The intuition of the definition is that we represent each variable in $\mathsf{dom}\,\nu$ with a distinct car identifier variable D. This means that even if for two different variables we have $\nu(c) = \nu(c')$, we represent them with different car identifier variables. Furthermore, we represent cars in the scope that have no variable mapping to it with additional car identifier variables. We first define the transformation for a simple MLSLS model with a single car.

Definition 3.3.18 (Simple Models MLSLS to FOMLA)**.** Given a simple MLSLS model $M = (CS, TS, \Omega, \nu)$ with $\mathsf{cars}\,M = \{C\}$, $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$, $\mathsf{res}(C) = \{\tilde{l_1}, \tilde{l_2}\}$, $\mathsf{clm}(C) = \{\tilde{l_3}\}$ and $\tilde{l_1}, \tilde{l_2}, \tilde{l_3} \in \mathbb{L} \cup \{\bullet\}$ let $\mathbb{D}$ be a set of car identifier variables such that $|\mathbb{D}| = \mathsf{max}(|\,\mathsf{dom}\,\nu|, |CS|)$. Further, if $\mathsf{dom}\,\nu \neq \emptyset$, we assume a bijective mapping between $D \in \mathbb{D}$ and $c \in \mathsf{dom}\,\nu$ indicated by D_c. Then we define

$$\begin{aligned}
\mathsf{m2mla_E}(M) = (\kappa_1 \uplus \kappa_2, (DS, f, \mathbb{D}, \mathcal{S})) \text{ where}& \\
&DS = \mathbb{D} \text{ if } CS = \mathsf{cars}\,M \text{ and } \emptyset \text{ otherwise} \ , \\
&f = \{c \mapsto D_c \mid c \in \mathsf{dom}\,\nu\} \ , \\
&\kappa_1 = \{D \mapsto C \mid D \in \mathbb{D}\} \ , \\
&\kappa_2 = \{D.v_{\mathsf{p}} \mapsto \mathsf{pos}(\kappa_1(D)), D.v_{\Omega} \mapsto \Omega(\kappa_1(D), TS), \\
&\qquad D.v_{\mathsf{s}} \mapsto \mathsf{spd}(\kappa_1(D)), D.v_{\mathsf{a}} \mapsto \mathsf{acc}(\kappa_1(D)), \\
&\qquad D.v_{\mathsf{c}} \mapsto \tilde{l_3}, D.v_{\mathsf{r1}} \mapsto \tilde{l_1}, D.v_{\mathsf{r2}} \mapsto \tilde{l_2} \mid D \in \mathbb{D}\} \ .
\end{aligned}$$
$\triangle$

The definition states that we have a unique car identifier variable D for each variable $c \in \mathsf{dom}\,\nu$ or a single car identifier variable if $|\,\mathsf{dom}\,\nu| = 0$ and $|CS| = 1$.

We can easily extend the definition above to proper MLSLS models where the domain only includes the view owner. We point out that for proper MLSLS models the domain of the valuation contains at least one element. Hence, the requirements from Definition 3.3.18 can be simplified as shown below.

Definition 3.3.19 (Proper Models MLSLS to FOMLA)**.** For a proper MLSLS model M with $M = (CS, TS, \Omega, V, \nu)$, $V = (L, X, E)$, $\mathsf{cars}\,M = \{E\}$, $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$, $\mathsf{res}(E) = \{\tilde{l_1}, \tilde{l_2}\}$, $\mathsf{clm}(E) = \{\tilde{l_3}\}$ and $\tilde{l_1}, \tilde{l_2}, \tilde{l_3} \in \mathbb{L} \cup \{\bullet\}$

let $|\mathbb{D}| = |\operatorname{\mathsf{dom}} \nu|$. Further, we assume a bijective mapping between $D \in \mathbb{D}$ and $\gamma \in \operatorname{\mathsf{dom}} \nu$ indicated by D_γ. Then we define

$$\begin{aligned}
\mathsf{m2mla}_{\mathsf{E}}(M) = (&\kappa_1 \uplus \kappa_2 \uplus \kappa_3, (DS, y, y', x, x', f, \mathbb{D}, \mathcal{S}, D_{\mathsf{ego}})) \text{ where}\\
&DS = \mathbb{D} \text{ if } CS = \operatorname{\mathsf{cars}} M \text{ and } \emptyset \text{ otherwise} ,\\
&f = \{\gamma \mapsto D_\gamma \mid \gamma \in \operatorname{\mathsf{dom}} \nu\} ,\\
&\kappa_1 = \{y \mapsto l, y' \mapsto l', x \mapsto r, x' \mapsto r'\} ,\\
&\kappa_2 = \{D \mapsto C \mid D \in \mathbb{D}\} ,\\
&\kappa_3 = \{D.v_{\mathsf{p}} \mapsto \mathsf{pos}(\kappa_1(D)), D.v_\Omega \mapsto \Omega(\kappa_1(D), TS),\\
&\qquad D.v_{\mathsf{s}} \mapsto \mathsf{spd}(\kappa_1(D)), D.v_{\mathsf{a}} \mapsto \mathsf{acc}(\kappa_1(D)),\\
&\qquad D.v_{\mathsf{c}} \mapsto \tilde{l_3}, D.v_{\mathsf{r1}} \mapsto \tilde{l_1}, D.v_{\mathsf{r2}} \mapsto \tilde{l_2} \mid D \in \mathbb{D}\}\} . \qquad \triangle
\end{aligned}$$

We give a recursive transformation of MLSLS models to FOMLA tuples. Note that by splitting the model into smaller parts we work with simple MLSLS models and FOMLA tuples and with proper ones.

Definition 3.3.20 (MLSLS to FOMLA)**.** Given a finite simple or proper MLSLS model M we define

$$\mathsf{m2mla}(M) = \begin{cases} (\mathsf{m2mla}_{\mathsf{E}}(M \boxbar \{C\}) \oplus \mathsf{m2mla}(M \boxminus \{C\})) & \text{if } |\operatorname{\mathsf{cars}} M| \geq 1 \\ \qquad \text{for some } C \in \operatorname{\mathsf{cars}} M & \\ (\kappa_\emptyset, \Upsilon_\emptyset) & \text{otherwise} . \end{cases}$$

$\triangle$

Note that always at most one input to $\oplus$ is a proper assignment and data tuple. We use the function $\oplus$, which has requirements on its inputs, i.e. it is only defined when both input FOMLA tuples are composable. However, it is clear that these requirements are satisfied, i.e. that for composable MLSLS models their FOMLA representation also is composable.

We defined operations on MLSLS models and FOMLA representations of MLSLS models along with operations to transform one into the other. We proceed to show properties relating these operations. The following six lemmas (Lemmas 3.3.21 to 3.3.26) relate the domain of FOMLA tuples and the domain of MLSLS models. We show that we can perform an operation and transform the result into the other domain, or equivalently transform first into the other domain and then perform the operation in the other domain.

Lemma 3.3.21. *For all composable finite and possibly simple MLSLS models M_1, M_2 we have*

$$\mathsf{m2mla}(M_1) \oplus \mathsf{m2mla}(M_2) =_{\mathsf{r}} \mathsf{m2mla}(M_1 \boxplus M_2) \ .$$

Proof. We proceed by induction on $n = |M_1 \boxplus M_2|$.
Induction base.
The models M_1 and M_2 may both be simple, or one of the models may be proper. Hence, we have the following three base cases: both models are simple, M_1 is proper or M_2 is proper.
Case 1 (M_1 and M_2 are simple). In this case we have $n = 0$, which means that both models are in fact the empty MLSLS model $M_\emptyset$. Taking the disjoint union of the empty model with itself gives the empty model. Further, transforming the empty model with m2mla gives the empty FOMLA tuple $(\kappa_\emptyset, \Upsilon_\emptyset)$. Finally, the disjoint union of two times the empty FOMLA tuple yields the empty FOMLA tuple. Hence, the lemma simplifies to $(\kappa_\emptyset, \Upsilon_\emptyset) \oplus (\kappa_\emptyset, \Upsilon_\emptyset) =_{\mathsf{r}} (\kappa_\emptyset, \Upsilon_\emptyset)$, which is true.

Case 2 (M_1 is proper and M_2 is simple). In this case we have $n = 1$. As M_1 is proper, it contains at least one car. Hence, M_2 has no cars and is equal to the empty MLSLS model $M_\emptyset$. With similar arguments as in the earlier case the lemma simplifies to $\mathsf{m2mla}(M_1) \oplus (\kappa_\emptyset, \Upsilon_\emptyset) =_{\mathsf{r}} \mathsf{m2mla}(M_1)$. Again, this equality up to renaming of FOMLA variables is true.

Case 3 (M_1 is simple and M_2 is proper). Because of commutativity of $\boxplus$ and $\oplus$ reducible to the previous case.

Induction hypothesis.
For all composable finite MLSLS models M_1, M_2 with $|M_1 \boxplus M_2| \leq n$ we have

$$\mathsf{m2mla}(M_1) \oplus \mathsf{m2mla}(M_2) =_{\mathsf{r}} \mathsf{m2mla}(M_1 \boxplus M_2) \ .$$

Induction step.
Let $|M_1 \boxplus M_2| = n + 1$, which means that at least one of the models is not empty. As $\boxplus$ and $\oplus$ are commutative assume w.l.o.g. $|\,\mathsf{cars}\, M_1| \geq 1$. In the following three steps we transform M_1 to reduce the size of $|M_1 \boxplus M_2|$ to apply the IH. In the first step we apply the definition of m2mla, in the second step we

use the associativity of $\oplus$ and in the third step we apply the IH:

$$\begin{aligned}
&\mathsf{m2mla}(M_1) \oplus \mathsf{m2mla}(M_2)\\
=\ &(\mathsf{m2mla}_{\mathsf{E}}(M_1 \boxbar \{C\}) \oplus \mathsf{m2mla}(M_1 \boxminus \{C\})) \oplus \mathsf{m2mla}(M_2)\\
=_{\mathsf{r}}\ &\mathsf{m2mla}_{\mathsf{E}}(M_1 \boxbar \{C\}) \oplus (\mathsf{m2mla}(M_1 \boxminus \{C\}) \oplus \mathsf{m2mla}(M_2))\\
=_{\mathsf{r}}\ &\mathsf{m2mla}_{\mathsf{E}}(M_1 \boxbar \{C\}) \oplus \mathsf{m2mla}((M_1 \boxminus \{C\}) \boxplus M_2)\ .
\end{aligned}$$

As M_1, M_2 are composable and as C is from M_1 it follows that C is not represented in M_2. Hence, we can pull the anti-restriction operator out and continue the steps from above:

$$\begin{aligned}
&\mathsf{m2mla}_{\mathsf{E}}(M_1 \boxbar \{C\}) \oplus \mathsf{m2mla}((M_1 \boxminus \{C\}) \boxplus M_2)\\
=_{\mathsf{r}}\ &\mathsf{m2mla}_{\mathsf{E}}(M_1 \boxbar \{C\}) \oplus \mathsf{m2mla}((M_1 \boxplus M_2) \boxminus \{C\}) \qquad \text{def. of } \mathsf{m2mla}\\
=\ &\mathsf{m2mla}(M_1 \boxplus M_2)\ . \qquad \square
\end{aligned}$$

We prove a similar lemma for the other direction. Note that the proof is very similar to the proof of the previous lemma.

Lemma 3.3.22. *For all composable and sane FOMLA tuples* $(\kappa_1, \Upsilon_1), (\kappa_2, \Upsilon_2)$ *we have*

$$\mathsf{mla2m}((\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)) = \mathsf{mla2m}(\kappa_1, \Upsilon_1) \boxplus \mathsf{mla2m}(\kappa_2, \Upsilon_2)\ .$$

Proof. For $i \in \{1, 2\}$ let $M_i = \mathsf{mla2m}(\kappa_i, \Upsilon_i)$ and $M = \mathsf{mla2m}((\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2))$. We proceed by induction on $n = |(\kappa_1, \Upsilon_1)| \oplus (\kappa_2, \Upsilon_2)|$.

Induction base.

We have the following three base cases: both tuples are simple, (κ_1, Υ_1) is proper or (κ_2, Υ_2) is proper.

Case 1 ((κ_1, Υ_1) and (κ_2, Υ_2) are simple). In this case we have $n = 0$, which means that both FOMLA tuples are equal to the empty FOMLA tuple $(\kappa_\emptyset, \Upsilon_\emptyset)$. Now, taking the disjoint union of the empty FOMLA tuple $(\kappa_\emptyset, \Upsilon_\emptyset)$ with itself gives the empty FOMLA tuple. Further, transforming the empty FOMLA tuple with $\mathsf{mla2m}$ to MLSLS returns the empty MLSLS model $M_\emptyset$. Finally, the disjoint union of the empty MLSLS model and itself yields the empty MLSLS model. Hence, the lemma simplifies to $M_\emptyset = M_\emptyset \boxplus M_\emptyset$. This equality is true.

Case 2 ((κ_1, Υ_1) is proper and (κ_2, Υ_2) is simple). In this case we have $n = 1$. As (κ_1, Υ_1) is proper it contains at least one car. Hence, as the two tuples are composable, (κ_2, Υ_2) has no cars and is equal to the empty FOMLA tuple $(\kappa_\emptyset, \Upsilon_\emptyset)$. With similar arguments as in the earlier case the lemma simplifies to $\mathsf{mla2m}(\kappa_1, \Upsilon_1) = \mathsf{mla2m}(\kappa_1, \Upsilon_1) \boxplus M_\emptyset$. Again, this equality is true.

Case 3 ((κ_1, Υ_1) is simple and (κ_2, Υ_2) is proper). Because of commutativity of $\boxplus$ and $\oplus$ reducible to the previous case.

Induction hypothesis.
For all composable sane FOMLA tuples $(\kappa_1, \Upsilon_1), (\kappa_2, \Upsilon_2)$ with $|(\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)| \leq n$ we have

$$\mathsf{mla2m}((\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)) = \mathsf{mla2m}(\kappa_1, \Upsilon_1) \boxplus \mathsf{mla2m}(\kappa_2, \Upsilon_2) \ .$$

Induction step.
Let $|(\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)| = n + 1$. Thus, at least one of the FOMLA tuples is not empty. As $\oplus$ and $\boxplus$ are commutative assume w.l.o.g. $|(\kappa_1, \Upsilon_1)| \geq 1$. Then, for some $C \in \kappa_1 \llparenthesis \mathbb{D} \rrparenthesis$ first we apply the definition of $\mathsf{mla2m}$, then we use associativity of $\boxplus$ and finally we apply the IH:

$$\begin{aligned}
&\mathsf{mla2m}(\kappa_1, \Upsilon_1) \boxplus \mathsf{mla2m}(\kappa_2, \Upsilon_2) \\
=&(\mathsf{mla2m}_{\mathsf{E}}((\kappa_1, \Upsilon_1) \obar \{C\}) \boxplus \mathsf{mla2m}((\kappa_1, \Upsilon_1) \ominus \{C\})) \boxplus \mathsf{mla2m}(\kappa_2, \Upsilon_2) \\
=& \mathsf{mla2m}_{\mathsf{E}}((\kappa_1, \Upsilon_1) \obar \{C\}) \boxplus (\mathsf{mla2m}((\kappa_1, \Upsilon_1) \ominus \{C\}) \boxplus \mathsf{mla2m}(\kappa_2, \Upsilon_2)) \\
=& \mathsf{mla2m}_{\mathsf{E}}((\kappa_1, \Upsilon_1) \obar \{C\}) \boxplus \mathsf{mla2m}(((\kappa_1, \Upsilon_1) \ominus \{C\}) \oplus (\kappa_2, \Upsilon_2)) \ .
\end{aligned}$$

As in the proof of Lemma 3.3.21, because $(\kappa_1, \Upsilon_1), (\kappa_2, \Upsilon_2)$ are composable and $C \in \mathsf{cars}(\kappa_1, \Upsilon_1)$ we conclude $C \notin \mathsf{cars}(\kappa_2, \Upsilon_2)$. Hence, we can pull the anti-restriction operator out and continue the steps from above:

$$\begin{aligned}
&\mathsf{mla2m}_{\mathsf{E}}((\kappa_1, \Upsilon_1) \obar \{C\}) \boxplus \mathsf{mla2m}(((\kappa_1, \Upsilon_1) \ominus \{C\}) \oplus (\kappa_2, \Upsilon_2)) \\
=& \mathsf{mla2m}_{\mathsf{E}}((\kappa_1, \Upsilon_1) \obar \{C\}) \boxplus \mathsf{mla2m}(((\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)) \ominus \{C\}) \quad \text{def. } \mathsf{mla2m} \\
=& \mathsf{mla2m}((\kappa_1, \Upsilon_1) \oplus (\kappa_2, \Upsilon_2)) \ .
\end{aligned}$$
□

We prove a similar lemma for anti-restriction.

Lemma 3.3.23. *For all finite and possibly simple MLSLS models M and sets $CS \subseteq \mathbb{I}$ we have*

$$\mathsf{m2mla}(M \boxminus CS) =_{\mathsf{r}} \mathsf{m2mla}(M) \ominus CS \ .$$

Proof. We proceed by induction on $n = |M|$.
Induction base.
We need to make a case distinction on whether M is simple or not.
Case 1 (M is simple). Let $|M| = 0$. Then $M = M_\emptyset$ and $M_\emptyset \boxminus CS = M_\emptyset$. Further, $\mathsf{m2mla}(M_\emptyset) = (\kappa_\emptyset, \Upsilon_\emptyset)$ and $(\kappa_\emptyset, \Upsilon_\emptyset) \ominus CS = (\kappa_\emptyset, \Upsilon_\emptyset)$.

Case 2 (M is proper). We make a case distinction on whether $\mathsf{cars}\, M \cap CS = \emptyset$.

Subcase 2.1 ($\mathsf{cars}\, M \cap CS \neq \emptyset$). We point out that $\mathsf{m2mla}$ preservers the car domain, i.e. for all MLSLS models M' we have $\mathsf{cars}\, M' = \mathsf{cars}\, \mathsf{m2mla}(M')$. Here that means that both $M \boxminus CS$ and $\mathsf{m2mla}(M) \ominus CS$ have an empty car domain, which implies that the lemma holds.

Subcase 2.2 ($\mathsf{cars}\, M \cap CS = \emptyset$). As in the previous case we point out that $\mathsf{m2mla}$ preservers the car domain. Hence, as $\mathsf{cars}\, M \cap CS = \emptyset$ we conclude $\mathsf{m2mla}(M \boxminus CS) = \mathsf{m2mla}(M) = \mathsf{m2mla}(M) \ominus CS$.

Induction hypothesis.
For all finite, possibly simple MLSLS models M with $|M| \leq n$ and all $CS \subseteq \mathbb{I}$ we have $\mathsf{m2mla}(M \boxminus CS) =_{\mathsf{r}} \mathsf{m2mla}(M) \ominus CS$.
Induction step.
Let $|M| = n + 1$. Then for some $C \in \mathsf{cars}\, M$ we have

$$\begin{aligned}
&\mathsf{m2mla}(M) \ominus CS && \text{def. of } \mathsf{m2mla}\\
=&(\mathsf{m2mla}_{\mathsf{E}}(M \boxbar \{C\}) \oplus \mathsf{m2mla}(M \boxminus \{C\})) \ominus CS && \text{distr. of } \ominus, \oplus\\
=_{\mathsf{r}}&(\mathsf{m2mla}_{\mathsf{E}}(M \boxbar \{C\}) \ominus CS) \oplus (\mathsf{m2mla}(M \boxminus \{C\}) \ominus CS) && \text{IH}\\
=_{\mathsf{r}}&(\mathsf{m2mla}_{\mathsf{E}}(M \boxbar \{C\}) \ominus CS) \oplus (\mathsf{m2mla}((M \boxminus \{C\}) \boxminus CS)) && \textit{Lemma } 3.2.13\\
=_{\mathsf{r}}&(\mathsf{m2mla}_{\mathsf{E}}(M \boxbar \{C\}) \ominus CS) \oplus (\mathsf{m2mla}(M \boxminus (\{C\} \cup CS)))
\end{aligned}$$

Now we make a case distinction on whether $C \in CS$. Note that in both cases we ensure that the constraints from $\mathsf{m2mla}_{\mathsf{E}}$ (that the model contains exactly one car) are satisfied.
Case 1 ($C \in CS$). As $C \in \mathsf{cars}\, M$ we know that $M \boxbar \{C\}$ contains a single car. Then, the FOMLA representation $\mathsf{m2mla}_{\mathsf{E}}(M \boxbar \{C\})$ also contains data of a single car C. Thus, if we remove this single car, we arrive at the empty FOMLA tuple, i.e. $\mathsf{m2mla}_{\mathsf{E}}(M \boxbar \{C\}) \ominus CS = (\kappa_\emptyset, \Upsilon_\emptyset)$. We use this and that $(\kappa_\emptyset, \Upsilon_\emptyset)$ is neutral in $\oplus$ to continue the steps from earlier and finish this case:

$$\begin{aligned}
&(\mathsf{m2mla}_{\mathsf{E}}(M \boxbar \{C\}) \ominus CS) \oplus \mathsf{m2mla}(M \boxminus (\{C\} \cup CS))\\
=&(\kappa_\emptyset, \Upsilon_\emptyset) \oplus \mathsf{m2mla}(M \boxminus (\{C\} \cup CS))\\
=&\,\mathsf{m2mla}(M \boxminus (\{C\} \cup CS))\\
=&\,\mathsf{m2mla}(M \boxminus CS) \ .
\end{aligned}$$

Case 2 ($C \notin CS$). We start as in the other case: we know $C \in \mathsf{cars}\, M$ and thus $\mathsf{m2mla}_{\mathsf{E}}(M \mathbin{\boxed{\upharpoonright}} \{C\})$ represents a single car C. The first equality follows because if $C \notin CS$ we do not change anything by leaving the removal of CS out. The second equality follows because we can perform the restriction to C in two steps: first we remove some cars CS, and then we remove all cars except C, i.e.

$$\mathsf{m2mla}_{\mathsf{E}}(M \mathbin{\boxed{\upharpoonright}} \{C\}) \ominus CS = \mathsf{m2mla}_{\mathsf{E}}(M \mathbin{\boxed{\upharpoonright}} \{C\}) =_{\mathsf{r}} \mathsf{m2mla}_{\mathsf{E}}((M \boxminus CS) \mathbin{\boxed{\upharpoonright}} \{C\}) \ . \tag{3.33}$$

We use this and continue the steps from before the case distinction:

$$\begin{aligned}
&(\mathsf{m2mla}_{\mathsf{E}}(M \mathbin{\boxed{\upharpoonright}} \{C\}) \ominus CS) \oplus \mathsf{m2mla}(M \boxminus (\{C\} \cup CS)) && \textit{Equation } (3.33)\\
&= \mathsf{m2mla}_{\mathsf{E}}((M \boxminus CS) \mathbin{\boxed{\upharpoonright}} \{C\}) \oplus \mathsf{m2mla}(M \boxminus (\{C\} \cup CS)) && \textit{Lemma } 3.2.13\\
&= \mathsf{m2mla}_{\mathsf{E}}((M \boxminus CS) \mathbin{\boxed{\upharpoonright}} \{C\}) \oplus \mathsf{m2mla}((M \boxminus CS) \boxminus \{C\}) && \text{def. of } \mathsf{m2mla}\\
&= \mathsf{m2mla}(M \ominus CS)
\end{aligned}$$

□

We proceed to show a similar lemma for the restriction operator.

Lemma 3.3.24. *For all finite and possibly simple MLSLS models M and sets $CS \subseteq \mathbb{I}$ we have*

$$\mathsf{m2mla}(M \mathbin{\boxed{\upharpoonright}} CS) =_{\mathsf{r}} \mathsf{m2mla}(M) \mathbin{\langle\!\upharpoonright\!\rangle} CS \ .$$

Proof. We have

$$\begin{aligned}
&\mathsf{m2mla}(M \mathbin{\boxed{\upharpoonright}} CS) && \text{def. of } \boxminus\\
&= \mathsf{m2mla}(M \boxminus (\mathbb{I} \setminus CS)) && \textit{Lemma } 3.3.23\\
&=_{\mathsf{r}} \mathsf{m2mla}(M) \ominus (\mathbb{I} \setminus CS) && \text{def. of } \ominus\\
&= \mathsf{m2mla}(M) \mathbin{\langle\!\upharpoonright\!\rangle} (\mathbb{I} \setminus (\mathbb{I} \setminus CS)) && \text{as } CS \subseteq \mathbb{I}\\
&= \mathsf{m2mla}(M) \mathbin{\langle\!\upharpoonright\!\rangle} CS
\end{aligned}$$

□

We prove the reverse direction of Lemma 3.3.23.

Lemma 3.3.25. *For all sane and possibly simple FOMLA tuples (κ, Υ) and sets $CS \subseteq \mathbb{I}$ we have*

$$\mathsf{mla2m}((\kappa, \Upsilon) \ominus CS) = \mathsf{mla2m}(\kappa, \Upsilon) \boxminus CS \ .$$

Proof. Note that we require (κ, Υ) to be sane to ensure that $\mathsf{mla2m}$ is defined for (κ, Υ). We proceed by induction on $n = |(\kappa, \Upsilon)|$.

Induction base.

If (κ, Υ) is simple then $|(\kappa, \Upsilon)| = 0$, and if (κ, Υ) is proper then $|(\kappa, \Upsilon)| = 1$. However, we consider both cases together and make a case distinction on whether the intersection of $\mathsf{cars}(\kappa, \Upsilon)$ and CS is empty or not.

Case 1 ($\mathsf{cars}(\kappa, \Upsilon) \cap CS = \emptyset$). This case is possible when (κ, Υ) is proper and when (κ, Υ) is simple. However, in both cases the operations $\ominus$ and $\boxminus$ have no effect, i.e. the result of both operations is their respective first argument. Hence, the claim of the lemma holds.

Case 2 ($\mathsf{cars}(\kappa, \Upsilon) \cap CS \neq \emptyset$). This case is only possible when (κ, Υ) is proper. If the intersection is not empty, then both sides of the equality in the lemma, that is $\mathsf{mla2m}((\kappa, \Upsilon) \ominus CS)$ and $\mathsf{mla2m}(\kappa, \Upsilon) \boxminus CS$, have an empty car domain. Hence, both sides in the claim are equal to $M_\emptyset$ and the claim holds.

Induction hypothesis.

For all sane and possibly simple FOMLA tuples (κ, Υ) with $|(\kappa, \Upsilon)| \leq n$ we have $\mathsf{mla2m}((\kappa, \Upsilon) \ominus CS) = \mathsf{mla2m}(\kappa, \Upsilon) \boxminus CS$.

Induction step.

Let $|(\kappa, \Upsilon)| = n + 1$ and $C \in \mathsf{cars}(\kappa, \Upsilon)$. We perform some steps to be able to apply the IH to move the removal of the cars in CS from the MLSLS domain to the FOMLA domain. That is, we apply the definition of $\mathsf{mla2m}$, then we use distributivity of $\boxminus$ over $\boxplus$ and finally we apply the IH:

$$\begin{aligned}
&\mathsf{mla2m}(\kappa, \Upsilon) \boxminus CS \\
=&\big(\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \boxplus \mathsf{mla2m}((\kappa, \Upsilon) \ominus \{C\})\big) \boxminus CS \\
=&(\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \boxminus CS) \boxplus (\mathsf{mla2m}((\kappa, \Upsilon) \ominus \{C\}) \boxminus CS) \\
=&(\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \boxminus CS) \boxplus \mathsf{mla2m}(((\kappa, \Upsilon) \ominus \{C\}) \ominus CS) \ .
\end{aligned}$$

Using Equation (3.27) this is equal to

$$(\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \boxminus CS) \boxplus \mathsf{mla2m}((\kappa, \Upsilon) \ominus (\{C\} \cup CS)) \ .$$

For the equations above we point out that we may apply the IH because $\ominus$ preserves $\mathsf{sane}(\Upsilon)$ (resp. $\mathsf{sane_s}(\Upsilon)$) from Pages 77 and 86.

Now we make a case distinction on whether $C \in CS$. Note that in both cases we ensure that the constraints from $\mathsf{mla2m}$ (that the model contains exactly one car) are satisfied.

Case 1 ($C \in CS$). As $C \in \mathsf{cars}(\kappa, \Upsilon)$ we know that $(\kappa, \Upsilon) \obar \{C\}$ contains a single car. Then, the MLSLS model $\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\})$ also contains data of a single car C. Thus, if we remove this single car, we arrive at the empty MLSLS model, i.e. $\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \boxminus CS = M_\emptyset$. We use this in the first step. In the second step we use that $M_\emptyset$ is neutral in $\boxplus$. And in the third step we use $C \in CS$, which finishes this case:

$$\begin{aligned}
&(\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \boxminus CS) \boxplus \mathsf{mla2m}((\kappa, \Upsilon) \ominus (\{C\} \cup CS)) \\
=&M_\emptyset \boxplus \mathsf{mla2m}((\kappa, \Upsilon) \ominus (\{C\} \cup CS)) \\
=&\,\mathsf{mla2m}((\kappa, \Upsilon) \ominus (\{C\} \cup CS)) \\
=&\,\mathsf{mla2m}((\kappa, \Upsilon) \ominus CS) \ .
\end{aligned}$$

Case 2 ($C \notin CS$). We start this case with some preparation. The first equality follows because if $C \notin CS$, then we do not change anything by leaving the removal of CS out. The second equality follows because we can perform the restriction to C in two steps. First we remove some cars CS, and then we remove all cars except C. Thus,

$$\begin{aligned}
&\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \boxminus CS \\
=&\,\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \\
=&\,\mathsf{mla2m_E}(((\kappa, \Upsilon) \ominus CS) \obar \{C\}) \ .
\end{aligned} \tag{3.34}$$

We continue the steps from before the case distinction. The first step follows from Equation (3.34), the second step from Equation (3.27) and in the third step we apply the definition of $\mathsf{mla2m}$:

$$\begin{aligned}
&(\mathsf{mla2m_E}((\kappa, \Upsilon) \obar \{C\}) \boxminus CS) \boxplus \mathsf{mla2m}((\kappa, \Upsilon) \ominus (\{C\} \cup CS)) \\
=&\ \mathsf{mla2m_E}(((\kappa, \Upsilon) \ominus CS) \obar \{C\}) \boxplus \mathsf{mla2m}((\kappa, \Upsilon) \ominus (\{C\} \cup CS)) \\
=&\ \mathsf{mla2m_E}(((\kappa, \Upsilon) \ominus CS) \obar \{C\}) \boxplus \mathsf{mla2m}(((\kappa, \Upsilon) \ominus CS) \ominus \{C\}) \\
=&\ \mathsf{mla2m}((\kappa, \Upsilon) \ominus CS) \ .
\end{aligned}$$

□

We proceed to show a similar lemma for the restriction operator.

Lemma 3.3.26. *For all sane and possibly simple FOMLA tuples (κ, Υ) and all sets $CS \subseteq \mathbb{I}$ we have*

$$\mathsf{mla2m}((\kappa, \Upsilon) \obar CS) = \mathsf{mla2m}(\kappa, \Upsilon) \boxbar CS \ .$$

Proof. We have

$$\begin{aligned}
&\mathsf{mla2m}((\kappa,\Upsilon) \obar CS) && \text{def. of } \ominus\\
=\,&\mathsf{mla2m}((\kappa,\Upsilon) \ominus (\mathbb{I} \setminus CS)) && \textit{Lemma } 3.3.25\\
=\,&\mathsf{mla2m}(\kappa,\Upsilon) \boxminus (\mathbb{I} \setminus CS) && \text{def. of } \boxminus\\
=\,&\mathsf{mla2m}(\kappa,\Upsilon) \boxbar (\mathbb{I} \setminus (\mathbb{I} \setminus CS)) &&\\
=\,&\mathsf{mla2m}(\kappa,\Upsilon) \boxbar CS && \square
\end{aligned}$$

The following proposition follows from Definition 3.3.20 and because MLSLS models are sane.

Proposition 3.3.27. *Let M_1 and M_2 be two finite MLSLS models, where M_1 is proper and M_2 is simple. Then for $(\kappa_1,\Upsilon_1) = \mathsf{m2mla}(M_1)$ and $(\kappa_2,\Upsilon_2) = \mathsf{m2mla}(M_2)$ we have*

$$\kappa_1 \models \mathsf{sane}(\Upsilon_1) \ ,$$
$$\kappa_2 \models \mathsf{sane_s}(\Upsilon_2) \ ,$$

where sane *and* $\mathsf{sane_s}$ *are defined on Pages 77 and 86.*

The following lemma formalises that $\mathsf{mla2m}$ is something similar of an inverse operation to $\mathsf{m2mla}$. Note that it is not a real inverse operation because $\mathsf{m2mla}$ loses information. That is, the sensor function is represented on FOMLA side with a single variable. The information about the evolution of the sensor value as the traffic changes is not represented on the FOMLA side. Hence, when recovering the MLSLS model from the FOMLA tuple the new MLSLS model only is weakly equal to the original model.

Lemma 3.3.28 (Inversion Lemma)**.** *For all finite possibly simple MLSLS models M we have*

$$M \eqsim_{\mathsf{m}} \mathsf{mla2m}(\mathsf{m2mla}(M)) \ .$$

Proof. We point out that according to Proposition 3.3.27 the FOMLA tuple $\mathsf{m2mla}(M)$ is sane. Hence, $\mathsf{mla2m}$ is defined for this tuple. Let $M' = \mathsf{mla2m}(\mathsf{m2mla}(M))$. Then, we proceed by induction on $n = |M|$.
Induction base.
Similar to the earlier proofs we distinguish to base cases: that M is proper and that M is simple.

Case 1 (M is simple). Let $|M| = 0$. Then we know that $M = M_\emptyset$. Hence, for $(\kappa, \Upsilon) = \mathsf{m2mla}(M)$ we have $(\kappa, \Upsilon) = (\kappa_\emptyset, \Upsilon_\emptyset)$, which implies that for $M' = \mathsf{mla2m}(\kappa, \Upsilon)$ we have $M' = M_\emptyset$. As $M = M'$, we conclude $M \approx_{\mathsf{m}} M'$.

Case 2 (M is proper). Now, M contains a single car. The requirements for two MLSLS models to be weakly equal (cf. Page 69) are that

- their scopes are equal,
- their variable valuations are equal,
- the traffic snapshots restricted to the cars in the scopes and in the variable valuations are equal, and
- the sensor function restricted to the same cars and traffic snapshots are equal.

For each of these points it is clear from the definitions of m2mla and mla2m that the equality holds.

Induction hypothesis.
For all finite possibly simple MLSLS models M with $|M| \leq n$ we have

$$M \approx_{\mathsf{m}} \mathsf{mla2m}(\mathsf{m2mla}(M)) \ .$$

Induction step.
Let $|M| = n+1$. For $i \in \{1, 2\}$ there are MLSLS models M_i with $|M_i| \leq n$ such that $M_1 \boxplus M_2 = M$. From the IH we conclude for $M'_i = \mathsf{mla2m}(\mathsf{m2mla}(M_i))$ that $M_i \approx_{\mathsf{m}} M'_i$. We have

$$\begin{aligned} M' &\overset{\text{def}}{=} \mathsf{mla2m}(\mathsf{m2mla}(M_1 \boxplus M_2)) && \textit{Lemma } 3.3.21 \\ &=_{\mathsf{r}} \mathsf{mla2m}(\mathsf{m2mla}(M_1) \oplus \mathsf{m2mla}(M_2)) && \textit{Lemma } 3.3.22 \\ &= \mathsf{mla2m}(\mathsf{m2mla}(M_1)) \boxplus \mathsf{mla2m}(\mathsf{m2mla}(M_2)) \ . \end{aligned}$$

Note that for the equalities above it causes no problem that Lemma 3.3.21 only ensures equality up to renaming because the FOMLA variable names used are ignored by mla2m. Hence, $M' = M'_1 \boxplus M'_2$. From Lemma 3.2.29 and $M_i \approx_{\mathsf{m}} M'_i$ we conclude $M_1 \boxplus M_2 \approx_{\mathsf{m}} M'_1 \boxplus M'_2$. This is equivalent to $M \approx_{\mathsf{m}} M'$. □

The following corollary follows from the previous lemma together with Lemma 3.2.30.

Corollary 3.3.29. *For all finite MLSLS models and all MLSLS formulas ϕ we have*

$$M \models \phi \text{ iff } \mathsf{mla2m}(\mathsf{m2mla}(M)) \models \phi \ .$$

We showed that $\mathsf{mla2m}$ is something similar to an inverse operation of $\mathsf{m2mla}$. Next, we prove an analogous result for the other direction. That is, we show that $\mathsf{m2mla}$ is something like an inverse operation of $\mathsf{mla2m}$. As for the other direction, we do not have a real inverse operation. On FOMLA side the valuation can be represented in multiple ways. That is, we could have two variables D, D' such that they represent the same car. When we go from FOMLA to MLSLS we lose this information because on MLSLS side, each car has an unique representation. Note that we use $\mathsf{sane}(\Upsilon_1)$ (resp. $\mathsf{sane_s}(\Upsilon_1)$) from Pages 77 and 86 to ensure that $\mathsf{mla2m}(\kappa_1, \Upsilon_1)$ is defined. Further, we point out that the conditions for weak equality of FOMLA tuples are more involved than for weak equality of MLSLS models. Hence, here we consider each condition individually.

Lemma 3.3.30 (Inversion Lemma)**.** *For all sane and possibly simple FOMLA tuples (κ_1, Υ_1) we have*

$$\begin{aligned}&(\kappa_1, \Upsilon_1) \eqsim_{\mathsf{mla}} (\kappa_2, \Upsilon_2) \ , \\ &\textit{where } (\kappa_2, \Upsilon_2) = \mathsf{m2mla}(\mathsf{mla2m}(\kappa_1, \Upsilon_1)) \ .\end{aligned}$$

Proof. We first consider the case that both FOMLA tuples are proper. The case that both tuples are simple works analogous, albeit simpler.

Let $M = (CS, TS, \Omega, V, \nu) = \mathsf{mla2m}(\kappa_1, \Upsilon_1)$ with $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$ and $\Upsilon_i = (DS_i, \alpha_i, \alpha'_i, \beta_i, \beta'_i, f_i, \mathbb{D}_i, \mathcal{S}_i, D_{Ei})$. We make a case distinction over the conditions of $(\kappa_1, \Upsilon_1) \eqsim_{\mathsf{mla}} (\kappa_2, \Upsilon_2)$ and show that they are satisfied.
Case 1 ($((\mathsf{ran}\, f_1) \lhd \kappa_1) \circ f_1 = ((\mathsf{ran}\, f_2) \lhd \kappa_2) \circ f_2$). We show equality of the functions by proving equality of their underlying relations. Let $R_1, R_2 \subseteq (\mathsf{CVar} \cup \{\mathsf{ego}\}) \times \mathbb{I}$ be these relations, i.e. $(\gamma, C) \in R_i \iff \kappa_i(f_i(\gamma)) = C$ for $i \in \{1, 2\}$.

We start by showing $R_1 \subseteq R_2$. Let $(\gamma, C) \in R_1$. Then $f_1(\gamma) = D$ for some $D \in \mathbb{D}_1$ and $\kappa_1(D) = C$. From the definition of $\mathsf{mla2m}$ we know $\nu = \{\gamma \mapsto \kappa_1(f_1(\gamma)) \mid \gamma \in \mathsf{dom}\, f_1\}$. This means that we have $\nu(\gamma) = C$. From the definition of $\mathsf{m2mla}$ we know that in κ_2 and Υ_2 we have for every $\gamma' \in \mathsf{dom}\, \nu$ a variable $D_{\gamma'}$ with $\kappa_2(D_{\gamma'}) = \nu(\gamma')$ and further $f_2(\gamma') = D_{\gamma'}$. Hence, $\kappa_2(f_2(\gamma)) = C$, which implies $(\gamma, C) \in R_2$.

We show $R_2 \subseteq R_1$. Let $(\gamma, C) \in R_2$. Then $f(\gamma) = D_\gamma$ for some $D_\gamma \in \mathbb{D}_2$, $\kappa_2(D_\gamma) = C$ and from the definition of $\mathsf{m2mla}$ we know $\nu(\gamma) = \kappa_2(f_2(\gamma))$. From definition of $\mathsf{mla2m}$ we know that $\kappa_1(f_1(\gamma)) = \nu(\gamma)$, which implies $(\gamma, C) \in R_1$.

Note that we have $f_1(\gamma) = D$ for some $D \in \mathbb{D}_1$, but $f_2(\gamma) = D_\gamma$. That is, f_1 might not be injective, while f_2 is injective. This is the main reason why the two assignments and data tuples are not equal in the usual sense.

Case 2 ($\kappa_1(\!| DS_1 |\!) = \kappa_2(\!| DS_2 |\!)$). From the definition of $\mathsf{mla2m}$ we know that $\kappa_1(\!| DS_1 |\!) = CS$. Next, in the definition of $\mathsf{m2mla}$ we see that DS_2 contains all variables from $\mathbb{D}_2$ that κ_2 maps to CS and no other variables. Also, we can see that $CS \subseteq \kappa_2(\!| \mathbb{D}_2 |\!)$. From these two observations we conclude $CS = \kappa_2(\!| DS_2 |\!)$. It follows that $\kappa_1(\!| DS_1 |\!) = \kappa_2(\!| DS_2 |\!)$.

Case 3 ($\kappa_1(D_{E1}) = \kappa_2(D_{E2})$). Follows from the definitions of $\mathsf{mla2m}$ and $\mathsf{m2mla}$.

Case 4 ($\kappa_1(\alpha_1) = \kappa_2(y_2) \wedge \kappa_1(\alpha_1') = \kappa_2(y_2') \wedge \kappa_1(\beta_1) = \kappa_2(x_2) \wedge \kappa_1(\beta_1') = \kappa_2(x_2')$). Follows from the definitions of $\mathsf{mla2m}$ and $\mathsf{m2mla}$.

Case 5 ($\forall D_1 \in \mathbb{D}_1.\, \forall D_2 \in \mathbb{D}_2.\, \kappa_1(D_1) = \kappa_2(D_2) \implies \kappa_1(\mathcal{S}_1(D_1)) = \kappa_2(\mathcal{S}_2(D_2))$). Let $D_1 \in \mathbb{D}_1, D_2 \in \mathbb{D}_2$ with $\kappa_1(D_1) = \kappa_2(D_2)$. We first consider reservations. That means, we show $\{\kappa(v_{\mathsf{r1}}), \kappa(v_{\mathsf{r2}})\} = \{\kappa'(v_{\mathsf{r1}}), \kappa'(v_{\mathsf{r2}}')\}$. In M we have $\mathsf{res}(\kappa_1(D_1)) = \{\kappa(D_1.v_{\mathsf{r1}}), \kappa(D_1.v_{\mathsf{r2}})\}$. Note that the set might contain $\bullet$ if a car has only one reservation. From the definition of $\mathsf{m2mla}$ it follows that κ_2 assigns to $D_2.v_{\mathsf{r1}}$ and $D_2.v_{\mathsf{r2}}$ the two values in $\mathsf{res}(\kappa_1(D_1))$. As we know that $\kappa_1(D_1) = \kappa_2(D_2)$ equality of the two sets follows. For the other data variables this works analogously. □

The following lemma follows from the previous lemma and from Lemma 3.3.9.

Lemma 3.3.31. *For all MLSLS formulas ϕ and all proper FOMLA tuples (κ, Υ) we have*

$$\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \wedge \mathsf{sane}(\Upsilon) \text{ implies } \kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon', \phi) \wedge \mathsf{sane}(\Upsilon') \textit{ where } (\kappa', \Upsilon') = \mathsf{m2mla}(\mathsf{mla2m}(\kappa, \Upsilon)) \ .$$

Proof. From Lemmas 3.3.9 and 3.3.30 we can see that $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi)$ implies $\kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon', \phi)$. Next, as $\kappa \models \mathsf{sane}(\Upsilon)$ we know that $\mathsf{mla2m}(\kappa, \Upsilon)$ is defined, i.e. for some proper MLSLS model M we have $M = \mathsf{mla2m}(\kappa, \Upsilon)$. We can see from Proposition 3.3.27 that for $(\kappa', \Upsilon') = \mathsf{m2mla}(M)$ we have $\kappa' \models \mathsf{sane}(\Upsilon')$. □

Now we can prove that our FOMLA encoding is correct, which we mentioned in the explanation surrounding Equation (3.23) on Page 81. More specifically, we show that satisfaction of an MLSLS formula by a model translates to satisfaction of the FOMLA constraints of that formula by the FOMLA representation of that model and similarly for the other direction. We split this statement into two statements expressing that satisfaction of an MLSLS formula corresponds to satisfaction of our FOMLA constraints and vice versa. For this we do not only consider well-scoped formulas, but MLSLS formulas in general. We prove both statements simultaneously via structural induction to use the induction hypothesis of the other statement during the proof. Note that we require that the FOMLA tuple we extract from the MLSLS model is weakly equal to the tuple we get through $\mathsf{m2mla}$ because we are not interested in the particular variable names used.

Lemma 3.3.32. *For all MLSLS formulas ϕ and all finite proper MLSLS models $M = (CS, TS, \Omega, V, \nu)$ with $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}\,\nu$ we have*

$$M \models \phi \text{ implies } \big(\textit{for all } (\kappa, \Upsilon) \textit{ with } (\kappa, \Upsilon) \eqsim_{\mathsf{mla}} \mathsf{m2mla}(M).\, \kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi)\big) \ . \tag{3.35}$$

For all proper FOMLA tuples (κ, Υ) with $\Upsilon = (DS, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$, all MLSLS formulas ϕ such that $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}(f)$ and $M = \mathsf{mla2m}(\kappa, \Upsilon)$ we have

$$\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \wedge \mathsf{sane}(\Upsilon) \text{ implies } M \models \phi \ . \tag{3.36}$$

Proof. We prove the lemma by induction on the structure of ϕ. We start with the base cases. Note that we group the cases first by statement to have a nicer exposition. We point out that for the cases of Equation (3.35) we often have to worry about the names of variables or the number of variables introduced by our construction. This complicates some cases.

Induction base.

Case 1 (Equation (3.35)). For this case let $M = (CS, TS, \Omega, V, \nu)$ with $V = ([l, l'], [r, r'], E)$ and $(\kappa, \Upsilon) = \mathsf{m2mla}(M)$ with $\Upsilon = (DS, x, x', y, y', f, \mathbb{D}, \mathcal{S}, D_E)$. We first show that for (κ, Υ) we have $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi)$. Afterwards, we conclude from Lemma 3.3.9 that for all FOMLA tuples (κ', Υ') with $(\kappa, \Upsilon) \eqsim_{\mathsf{mla}} (\kappa', \Upsilon')$ we have $\kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon', \phi)$.

Subcase 1.1 ($\phi \equiv \ell = k$). Assume that $M \models \ell = k$ holds, which implies $r' - r = k$. As $\kappa(x) = r, \kappa(x') = r'$, it follows that $\kappa \models x' - x = k$, which implies $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \ell = k)$.

Subcase 1.2 ($\phi \equiv \mathsf{free}$). Assume $M \models \mathsf{free}$. We know $r < r'$ and $l = l'$, which implies $\kappa(x') > \kappa(x)$ and $\kappa(y) = \kappa(y')$. Further, we have for all $C \in CS$ that either $l \notin \mathsf{res}(C) \cup \mathsf{clm}(C)$ or $(r, r') \cap \mathsf{se}(C, TS, \Omega) = \emptyset$. Let $\kappa^{\sim}$ be the inverse of κ (see also Page 6) and note that $\kappa^{\sim}$ may be a relation instead of a function. Then, as $DS = \bigcup_{C \in CS} \kappa^{\sim}(C)$ and as $\mathsf{m2mla}$ simply takes the values of the cars and assigns them to data variables of a car it, follows that for all $D \in DS$ the assignment κ assigns values such that $y \notin \{D.v_{\mathsf{r1}}, D.v_{\mathsf{r2}}, D.v_{\mathsf{c}}\}$ or $(x, x') \cap D.\mathsf{se} = \emptyset$ is satisfied. This implies that $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \mathsf{free})$.

Subcase 1.3 ($\phi \equiv \mathsf{re}(\gamma)$). Assume $M \models \mathsf{re}(\gamma)$. We know $r < r'$, $l = l'$, $l \in \mathsf{res}(\nu(\gamma))$ and $X \subseteq \mathsf{se}(\nu(\gamma), TS, \Omega)$. This implies that similar properties hold for κ. We have $\kappa(x') > \kappa(x)$, $\kappa(y) = \kappa(y')$, $[\kappa(x), \kappa(x')] \subseteq \kappa(f(\gamma).\mathsf{se})$ and either $\kappa(f(\gamma).v_{\mathsf{r1}}) = \kappa(y)$ or $\kappa(f(\gamma).v_{\mathsf{r2}}) = \kappa(y)$. These are exactly the constraints imposed by $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \mathsf{re}(\gamma))$, hence we have $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \mathsf{re}(\gamma))$.

Subcase 1.4 ($\phi \equiv \mathsf{cl}(\gamma)$). Analogous to the previous case.

Subcase 1.5 ($\phi \equiv \gamma = \gamma'$). Assume $M \models \gamma = \gamma'$, which implies $\nu(\gamma) = \nu(\gamma')$. From the definition of $\mathsf{m2mla}$ we know that for each $\gamma'' \in \mathsf{dom}\,\nu$ we have $\kappa(f(\gamma'')) = \nu(\gamma'')$, which implies $\kappa(f(\gamma)) = \kappa(f(\gamma'))$. Hence, we conclude $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \gamma = \gamma')$.

Case 2 (Equation (3.36)). For this case let $M = \mathsf{mla2m}(\kappa, \Upsilon)$ be an MLSLS model with $M = (CS, TS, \Omega, V, \nu)$ and $V = ([l, l'], [r, r'], E)$. Note that in the proof $\mathsf{sane}(\Upsilon)$ is only needed to ensure that $\mathsf{mla2m}(\kappa, \Upsilon)$ is defined.

Subcase 2.1 ($\phi \equiv \ell = k$). Assume $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \ell = k) \land \mathsf{sane}(\Upsilon)$, which implies $\kappa(\beta') - \kappa(\beta) = k$. As $X = [\kappa(\beta), \kappa(\beta')]$ we have $M \models \ell = k$.

Subcase 2.2 ($\phi \equiv \mathsf{free}$). Assume $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \mathsf{free}) \land \mathsf{sane}(\Upsilon)$. Then $\kappa(\beta') > \kappa(\beta)$, $\kappa(\alpha) = \kappa(\alpha')$ and for all $D \in DS$ it holds that κ assigns values to the FOMLA variables such that $\alpha \notin \{D.v_{\mathsf{r1}}, D.v_{\mathsf{r2}}, D.v_{\mathsf{c}}\}$ or $D.\mathsf{se} \cap (\beta, \beta') = \emptyset$ holds. This implies that similar properties hold for M, which means $M \models \mathsf{free}$.

Subcase 2.3 ($\phi \equiv \mathsf{re}(\gamma)$). Assume $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \mathsf{re}(\gamma)) \land \mathsf{sane}(\Upsilon)$. Then, the assignment κ ensures $\kappa(\beta') > \kappa(\beta)$, $\kappa(\alpha) = \kappa(\alpha')$, $[\kappa(\beta), \kappa(\beta')] \subseteq \kappa(f(\gamma).\mathsf{se})$ and either $\kappa(f(\gamma).v_{\mathsf{r1}}) = \kappa(\alpha)$ or $\kappa(f(\gamma).v_{\mathsf{r2}}) = \kappa(\alpha)$. This implies that similar properties hold for M, which means $M \models \mathsf{re}(\gamma)$.

Subcase 2.4 ($\phi \equiv \mathsf{cl}(\gamma)$). Analogous to the previous case.

Subcase 2.5 ($\phi \equiv \gamma = \gamma'$). Assume $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \gamma = \gamma') \wedge \mathsf{sane}(\Upsilon)$, which implies $\kappa(f(\gamma)) = \kappa(f(\gamma'))$. As $\nu = \{\gamma \mapsto \kappa(f(\gamma)) \mid \gamma \in \mathsf{dom}\, f\}$, i.e. $\nu(\cdot)$ assigns the same car identifiers as $\kappa(f(\cdot))$, we have $\nu(\gamma) = \nu(\gamma')$. This implies $M \models \gamma = \gamma'$.

Induction hypothesis.
We have finished proving the lemmas for the base cases and have the following two induction hypotheses.
Case 1 (Equation (3.35)). For an MLSLS formula ϕ and all finite proper MLSLS models $M = (CS, TS, \Omega, V, \nu)$ with $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}\,\nu$ we have

$$M \models \phi \text{ implies } \big(\text{for all } (\kappa, \Upsilon) \text{ with } (\kappa, \Upsilon) \eqsim_{\mathsf{mla}} \mathsf{m2mla}(M).\, \kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi)\big) \ . \tag{3.37}$$

Case 2 (Equation (3.36)). For an MLSLS formula ϕ and all proper FOMLA tuples (κ, Υ) with $\Upsilon = (DS, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ and $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}(f)$ we have

$$\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \wedge \mathsf{sane}(\Upsilon) \text{ implies } M \models \phi \ , \tag{3.38}$$

where $M = \mathsf{mla2m}(\kappa, \Upsilon)$.

Induction step.
We continue with the induction step.
Case 1 (Equation (3.35)). For $i \in \{1, 2\}$ let $(\kappa_i, \Upsilon_i) = \mathsf{m2mla}(M_i)$ with $\Upsilon_i = (DS_i, y_i, y_i', x_i, x_i', f_i, \mathbb{D}_i, \mathcal{S}_i, D_{Ei})$.

Subcase 1.1 ($\phi \equiv \phi_1 \wedge \phi_2$). Assume $M \models \phi_1 \wedge \phi_2$, which implies $M \models \phi_1$ and $M \models \phi_2$. As $\mathsf{freeVar}(\phi_i) \subseteq \mathsf{freeVar}(\phi)$ and $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}\,\nu$ we can apply the IH to conclude that $\kappa_i \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_i, \phi_i)$ with $i \in \{1, 2\}$. As we do not change the model M we have $(\kappa_1, \Upsilon_1) = (\kappa_2, \Upsilon_2) = (\kappa, \Upsilon)$. Thus, we conclude $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi_1 \wedge \phi_2)$.

Subcase 1.2 ($\phi \equiv \dfrac{\phi_2}{\phi_1}$). Assume $M \models \dfrac{\phi_2}{\phi_1}$. We do a case distinction on whether $l \leq l'$.

Subsubcase 1.2.1 (Case $l \leq l'$). There is an $l'' \in [l-1, l']$ such that for $i \in \{1, 2\}$ we have $M_i \models \phi_i$ where $M_1 = (CS, TS, \Omega, V^{[l,l'']}, \nu)$ and $M_2 = (CS, TS, \Omega, V^{[l''+1,l']}, \nu)$. We choose l'' such that the previous property holds. As $\mathsf{freeVar}(\phi_i) \subseteq \mathsf{dom}\,\nu$ we can apply the IH and get $\kappa_i \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_i, \phi_i)$. We can see that $\{y_1, y_1'\} \lhd \kappa_1$ and $\{y_2, y_2'\} \lhd \kappa_2$ are equal up to variable renaming. Further, we know $\kappa_1(y_1) = l, \kappa_1(y_1') = l''$ and $\kappa_2(y_2) = l'' + 1, \kappa_2(y_2') = l'$.

For the assignment $\kappa' = \kappa_1 \uplus (\{y_2, y_2'\} \lhd \kappa_2)$ and the data tuple $\Upsilon_2' = (DS_1, y_2, y_2', x_1, x_1', f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ we have $\kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1)$ and $\kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2', \phi_2)$, i.e. we can use mostly the variables from κ_1 and Υ_1. This implies $\kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2', \phi_2)$.

To be able to apply the FOMLA semantics of the vertical chop we have to combine κ_1 and κ_2 into a single assignment. Additionally, we have to combine y_1' and y_2 into a single chop point. As $\kappa'(y_2) = \kappa'(y_1') + 1$ we can replace y_2, i.e. for $\Upsilon_2'' = (DS_1, y_1' + 1, y_2', x_1, x_1', f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ and $\kappa'' = \kappa_1 \uplus (\{y_2'\} \lhd \kappa_2)$ we have $\kappa'' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2'', \phi_2)$. As $\kappa''(y_1') = l''$ and $l'' \in [l-1, l']$ we can bind y_1' by a quantifier, i.e. let $\kappa''' = (\kappa_1 \ntriangleleft \{y_1'\}) \uplus (\kappa_2 \lhd \{y_2'\})$ be an assignment that takes everything except y_1' from κ_1 and only y_2' from κ_2. Then

$$\begin{aligned} &\kappa'' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2'', \phi_2) \\ \text{iff } &\kappa''' \models \exists y_1' .\, y_1 - 1 \leq y_1' \leq y_2' \wedge \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1) \wedge \kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2'', \phi_2) \\ \text{iff } &\kappa''' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon', \begin{smallmatrix}\phi_2\\ \phi_1\end{smallmatrix}) \enspace , \end{aligned}$$

where $\Upsilon' = (DS_1, y_1, y_2', x_1, x_1', f_1, \mathcal{S}_1, \mathbb{D}_1, D_E)$. At last, (κ''', Υ') is equal to (κ, Υ) up to variable renaming, which implies $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \begin{smallmatrix}\phi_2\\ \phi_1\end{smallmatrix})$.

Subsubcase 1.2.2 ($l > l'$). We have $M \models \begin{smallmatrix}\phi_2\\ \phi_1\end{smallmatrix}$, which means that $M \models \phi_1$ and $M \models \phi_2$ holds. The proof is then analogous to the case $\phi \equiv \phi_1 \wedge \phi_2$.

Subcase 1.3 ($\phi \equiv \neg\phi_1$). We first reformulate the IH of 3.36, which states that for all FOMLA tuples (Υ', κ') with $\mathsf{freeVar}(\phi_1) \subseteq \mathsf{dom}\, f'$ we have

$$\kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon', \phi_1) \wedge \mathsf{sane}(\Upsilon') \text{ implies } \mathsf{mla2m}(\kappa', \Upsilon') \models \phi_1 \enspace .$$

Using contraposition we can reformulate this equivalently as

$$\mathsf{mla2m}(\kappa', \Upsilon') \not\models \phi_1 \text{ implies } \kappa' \not\models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon', \phi_1) \wedge \mathsf{sane}(\Upsilon') \enspace . \tag{3.39}$$

Assume $M \models \neg\phi_1$, which means $M \not\models \phi_1$. Let $(\kappa, \Upsilon) = \mathsf{m2mla}(M)$ and $M' = \mathsf{mla2m}(\mathsf{m2mla}(M)) = \mathsf{mla2m}(\kappa, \Upsilon)$. From $M \not\models \phi_1$ and from Corollary 3.3.29 we can conclude that $M' \not\models \phi_1$. Using Equation (3.39) we conclude $\kappa \not\models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi_1) \wedge \mathsf{sane}(\Upsilon)$. As we know from Proposition 3.3.27 that the FOMLA tuple we get with $\mathsf{m2mla}$ is sane, i.e. $\kappa \models \mathsf{sane}(\Upsilon)$, it follows that $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \neg\phi_1) \wedge \mathsf{sane}(\Upsilon)$.

Subcase 1.4 ($\phi \equiv cs : \phi_1$). For this case we point out that when applying $\mathsf{m2mla}$ the resulting FOMLA tuple has $|\operatorname{\mathsf{dom}} \nu| + |CS \setminus \operatorname{\mathsf{ran}} \nu|$ variables in $\mathbb{D}$. Thus, M and $M' = (\{\nu(c) \mid c \in cs\}, TS, \Omega, V, \nu)$ may have a different number of variables in their FOMLA representation. We circumvent this by separating M into two models: an essential part and an optional part that does not affect the satisfaction of the formula, but may affect the number of variables.

Assume $M \models cs : \phi_1$ and let $M_1 = M \boxed{\upharpoonright} \operatorname{\mathsf{ran}} \nu$ and $M_1' = M \boxminus \operatorname{\mathsf{ran}} \nu$ with $M_1 = (CS_1, TS_1, \Omega_1, V, \nu)$ and $M_1' = (CS_1', TS_1', \Omega_1', \emptyset)$. Note that M_1 is a proper model, as $E \in \operatorname{\mathsf{ran}} \nu$ and that $M = M_1 \boxplus M_1'$. As $M \models cs : \phi_1$ we know $M_1 \models cs : \phi_1$ because of the semantics of the scope operator. This implies for $M_{11} = (CS_{11}, TS_1, \Omega_1, V_1, \nu_1)$ with $CS_{11} = \{\nu(c) \mid c \in cs\}$ that $M_{11} \models \phi_1$. As we did not change the valuation, i.e. $\nu_1 = \nu$ we still have $\operatorname{\mathsf{dom}} \nu_1 \subseteq \mathsf{freeVar}(\phi_1)$. Thus we may apply the IH to conclude for $(\kappa_{11}, \Upsilon_{11}) = \mathsf{m2mla}(M_{11})$ that $\kappa_{11} \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_{11}, \phi_1)$, where $\Upsilon_{11} = (DS_{11}, y_{11}, y_{11}', x_{11}, x_{11}', f_{11}, \mathbb{D}_{11}, \mathcal{S}_{11}, D_{E11})$.

We undo the scope evaluation. Let Υ_{12} be equal to Υ_{11} except that we replace DS_{11} with an arbitrary $DS_{12} \subseteq \mathbb{D}_{11}$. We have $\kappa_{11} \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_{12}, cs{:}\phi_1)$ because DS_{12} is overwritten by the scope operator. We choose $DS_{12} = \mathsf{dom}(\kappa_{11} \rhd CS_1)$ and have $(\kappa_{12}, \Upsilon_{12}) =_{\mathsf{r}} (\kappa_1, \Upsilon_1)$ with $(\kappa_1, \Upsilon_1) = \mathsf{m2mla}(M_1)$, which implies $\kappa_1 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, cs : \phi_1)$.

For $(\kappa_1', \Upsilon_1') = \mathsf{m2mla}(M_1')$ we know that (κ_1, Υ_1) and (κ_1', Υ_1') are composable. Let $(\kappa', \Upsilon') = (\kappa_1, \Upsilon_1) \oplus (\kappa_1', \Upsilon_1')$. Then we have $\kappa' \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon', cs{:}\phi_1)$ because $\kappa_1 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, cs : \phi_1)$ and the variables from Υ_1' are not constrained in $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, cs : \phi_1)$. By applying Lemma 3.3.21 we conclude $(\kappa_1, \Upsilon_1) \oplus (\kappa_1', \Upsilon_1') =_{\mathsf{r}} \mathsf{m2mla}(M_1 \boxplus M_1')$. As $\mathsf{m2mla}(M_1 \boxplus M_1') = \mathsf{m2mla}(M)$, we conclude $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, cs : \phi_1)$.

Subcase 1.5 ($\phi \equiv \exists c.\, \phi_1$). Assume $M \models \exists c.\, \phi_1$, which implies that there is $C \in CS$ such that for $M_1 = (CS, TS, \Omega, V, \nu_1)$ with $\nu_1 = \nu \oplus \{c \mapsto C\}$ we have $M_1 \models \phi_1$. Note that we assume w.l.o.g. $c \notin \operatorname{\mathsf{dom}} \nu$. From $\mathsf{freeVar}(\exists c.\, \phi_1) \subseteq \operatorname{\mathsf{dom}} \nu$ we know that $\mathsf{freeVar}(\phi_1) \subseteq \operatorname{\mathsf{dom}} \nu_1$ and thus we can apply the IH to conclude $\kappa_1 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1)$. We introduce the quantifier again and have $\kappa_{11} \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_{11}, \exists c.\, \phi_1)$, where $\kappa_{11} = \kappa_1$, $\Upsilon_{11} = (DS_1, y_1, y_1', x_1, x_1', f_{11}, \mathbb{D}_1, \mathcal{S}_1, D_E)$ and $f_{11} = \{c\} \lhd f_1$. We would like to point out that $\mathsf{m2mla}$ introduces for each variable $c \in \operatorname{\mathsf{dom}} \nu$ a unique variable

$D \in \mathsf{DVar}$ into $\mathbb{D}$ and for each car identifier $C \in CS \setminus \mathsf{ran}\,\nu$ another unique variable, i.e. $|\mathbb{D}| = |\,\mathsf{dom}\,\nu| + |CS \setminus \mathsf{ran}\,\nu|$. Thus, if $C \in \mathsf{ran}\,\nu$ we have $|\mathbb{D}_1| = |\mathbb{D}| + 1$ and if $C \in CS \setminus \mathsf{ran}\,\nu$ then $|\mathbb{D}_1| = |\mathbb{D}|$. Nevertheless, in both cases we have $(\kappa, \Upsilon) \eqsim_{\mathsf{mla}} (\kappa_{11}, \Upsilon_{11})$ because $\mathsf{dom}\,f = \mathsf{dom}\,f_{11}$ and $\forall \gamma \in \mathsf{dom}\,f$ we have $\kappa(f(\gamma)) = \kappa_{11}(f_{11}(\gamma))$ and we did not change other parts. Thus, with Lemma 3.3.9 we conclude $\kappa \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon, \exists c.\, \phi_1)$.

Subcase 1.6 ($\phi \equiv \phi_1 \frown \phi_2$). Let $M \models \phi_1 \frown \phi_2$. Then there is $r'' \in [r, r']$ such that $M_i \models \phi_i$ with $M_i = (CS, TS, \Omega, V_i, \nu)$ for $i \in \{1, 2\}$, $V_1 = (L, [r, r''], E), V_2 = (L, [r'', r'], E)$. As $\mathsf{freeVar}(\phi_i) \subseteq \mathsf{dom}\,\nu_i$ we can apply the IH to conclude $\kappa_i \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon_i, \phi_i)$. Note that $\{x_1, x_1'\} \lhd \kappa_1$ and $\{x_2, x_2'\} \lhd \kappa_2$ are equal up to renaming and that $\kappa_1(x_1') = \kappa_2(x_2) = r''$. Thus, we can mostly use variables from (κ_1, Υ_1) in (κ_2, Υ_2), i.e. for $\Upsilon_{12} = (DS_1, y_1, y_1', x_1', x_2', f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ and $\kappa_{12} = \kappa_1 \oplus \{x_2' \mapsto \kappa_2(x_2')\}$ we have $\kappa_{12} \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon_1, \phi_1)$ and $\kappa_{12} \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon_{12}, \phi_2)$, which implies $\kappa_{12} \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon_{12}, \phi_2)$. We can put x_2' below a quantifier and have for $\kappa' = \{x_1'\} \lhd \kappa_{12}$ that $\kappa' \models \exists x_1'.\, x_1 \leq x_1' \leq x_2' \wedge \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon_{12}, \phi_2)$. This is the semantics of our FOMLA representation of horizontal chop. Thus, for $\Upsilon' = (DS_1, y_1, y_1', x_1, x_2', f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ we have $\kappa' \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon', \phi_1 \frown \phi_2)$. As (κ, Υ) and (κ', Υ') are equivalent up to renaming we conclude $\kappa \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon, \phi_1 \frown \phi_2)$.

Case 2 (Equation (3.36)). Our general approach (except for the case with negation) is that we assume $\kappa \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon, \phi) \wedge \mathsf{sane}(\Upsilon)$, where ϕ has a subformula ϕ_1 (or perhaps two subformulas ϕ_1, ϕ_2). We then simplify (κ, Υ) to (κ_1, Υ_1) such that $\kappa_1 \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon_1, \phi_1) \wedge \mathsf{sane}(\Upsilon)$, argue that $\kappa_1 \models \mathsf{sane}(\Upsilon_1)$ holds and thus have all conditions satisfied to be able to apply the IH to conclude for $M_1 = \mathsf{mla2m}(\kappa_1, \Upsilon_1)$ that $M_1 \models \phi_1$. Note that we need $\mathsf{sane}(\Upsilon)$ to ensure that $\mathsf{mla2m}(\kappa_1, \Upsilon_1)$ is defined. We then reintroduce the formerly removed operator, i.e. we create M' with $M' \models \phi$. At the end we show that $M' = M$, because we have to show for $M = \mathsf{mla2m}(\kappa, \Upsilon)$ that $M \models \phi$.

During the proof let $M = (CS, TS, \Omega, V, \nu) = \mathsf{mla2m}(\kappa, \Upsilon)$. We proceed with the induction step.

Subcase 2.1 ($\phi \equiv \phi_1 \wedge \phi_2$). Assume $\kappa \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon, \phi_1 \wedge \phi_2) \wedge \mathsf{sane}(\Upsilon)$ which implies $\kappa \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon, \phi_1) \wedge \mathsf{sane}(\Upsilon)$ and $\kappa \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon, \phi_2) \wedge \mathsf{sane}(\Upsilon)$. As $\mathsf{freeVar}(\phi_i) \subseteq \mathsf{dom}\,f$ we can apply the IH and get $M \models \phi_1$ and $M \models \phi_2$, which implies $M \models \phi_1 \wedge \phi_2$.

Subcase 2.2 ($\phi \equiv \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$). Assume $\kappa \models \mathsf{tr}^{\mathsf{mla}}_{\mathsf{f}}(\Upsilon, \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}) \wedge \mathsf{sane}(\Upsilon)$. We do a case distinction on whether $\kappa(\alpha) \leq \kappa(\alpha')$.

Subsubcase 2.2.1 ($\kappa(\alpha) \leq \kappa(\alpha')$). In this case we know that for $\Upsilon_1 = (DS, \alpha, y, \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$, $\Upsilon_2 = (DS, y+1, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ and a fresh integer variable y the assignment κ satisfies the FOMLA formula $\exists y \in [\alpha - 1, \alpha'].\, \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_2) \wedge \mathsf{sane}(\Upsilon)$. We extend κ to κ' by choosing a value for y that satisfies the formula. We have $\kappa' \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1) \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_2) \wedge \mathsf{sane}(\Upsilon)$, which implies $\kappa' \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1) \wedge \mathsf{sane}(\Upsilon)$ and $\kappa' \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_2, \phi_2) \wedge \mathsf{sane}(\Upsilon)$. As the formula $\mathsf{sane}(\Upsilon)$ ignores the terms representing lanes we conclude $\kappa' \models \mathsf{sane}(\Upsilon_1)$ and $\kappa' \models \mathsf{sane}(\Upsilon_2)$.

For $i \in \{1, 2\}$ we have $\mathsf{freeVar}(\phi_i) \subseteq \mathsf{dom}\, f$. Hence, we satisfy the conditions of the IH and apply it to get $M_1 \models \phi_1$ and $M_2 \models \phi_2$, where $M_i = (CS_i, TS_i, \Omega_i, V_i, \nu_i) = \mathsf{mla2m}(\kappa', \Upsilon_i)$ with $V_1 = ([\kappa'(\alpha), \kappa'(y)], X_1, E_1)$ and $V_2 = ([\kappa'(y)+1, \kappa'(\alpha')], X_2, E_2)$.

All elements of M_1, M_2 except the lanes are equal. We can combine the two views: let $L' = [\kappa'(\alpha), \kappa'(y)] \ominus [\kappa'(y)+1, \kappa'(\alpha')]$. Then we have for $M' = (CS_1, TS_1, \Omega_1, (L', X_1, E_1), \nu_1)$ that $M' \models \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$. As $\kappa'(y) \in [\kappa'(\alpha), \kappa'(\alpha')]$ and $L = [\kappa'(\alpha), \kappa'(\alpha')]$ it follows that $L = L_1 \ominus L_2$. As we did not change anything except the terms representing the extension it follows that $M' = M$, which implies $M \models \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}$.

Subsubcase 2.2.2 ($\kappa(\alpha) > \kappa(\alpha')$). In this case $\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \genfrac{}{}{0pt}{}{\phi_2}{\phi_1}) \wedge \mathsf{sane}(\Upsilon)$ evaluates to $\mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi_1) \wedge \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \phi_2) \wedge \mathsf{sane}(\Upsilon)$. The proof then is analogous to the case $\phi \equiv \phi_1 \wedge \phi_2$.

Subcase 2.3 ($\phi \equiv \neg\phi_1$). Assume $\kappa \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \neg\phi_1) \wedge \mathsf{sane}(\Upsilon)$. As $\mathsf{freeVar}(\phi_1) \subseteq \mathsf{dom}\, f$ we may apply the IH of 3.35 and conclude that for all finite MLSLS models M' we have

$$M' \models \phi_1 \text{ implies } \kappa' \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon', \phi_1)\ , \text{ where } (\Upsilon', \kappa') = \mathsf{m2mla}(M')\ .$$

By contraposition this is equivalent to

$$\kappa' \not\models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon', \phi_1) \text{ implies } M' \not\models \phi_1\ , \text{ where } (\Upsilon', \kappa') = \mathsf{m2mla}(M')\ . \tag{3.40}$$

We choose $M' = M = \mathsf{mla2m}(\kappa, \Upsilon)$ and get $(\Upsilon', \kappa') = \mathsf{m2mla}(\mathsf{mla2m}(\kappa, \Upsilon))$. From Lemma 3.3.31 and $\kappa \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \neg\phi_1) \wedge \mathsf{sane}(\Upsilon)$ we know $\kappa' \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon', \neg\phi_1)$, which implies $\kappa' \not\models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon', \phi_1)$. Thus, from Equation (3.40) we conclude $M' \not\models \phi_1$. As $M = M'$, it follows that $M \models \neg\phi_1$.

Subcase 2.4 ($\phi \equiv cs : \phi_1$). Assume $\kappa \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, cs : \phi_1) \wedge \mathsf{sane}(\Upsilon)$. Then $\kappa \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1) \wedge \mathsf{sane}(\Upsilon)$ with $\Upsilon_1 = (DS_1, \alpha, \alpha', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ and $DS_1 = \{f(c) \mid c \in cs\}$. As $\mathsf{sane}(\Upsilon)$ ignores the scope, which we changed, we conclude $\kappa \models \mathsf{sane}(\Upsilon_1)$. As $\mathsf{freeVar}(\phi_1) \subseteq \mathsf{dom}\, f$ we can use the IH to get $M_1 \models \phi_1$, where $M_1 = (CS_1, TS_1, \Omega_1, V_1, \nu_1) = \mathsf{mla2m}(\kappa, \Upsilon_1)$. Note that all elements of M_1 are equal to M except for CS_1.

We are left to show that evaluating the scope operator on MLSLS side results in CS_1, i.e. $CS_1 = \{\nu(c) \mid c \in cs\}$. We see

$$\begin{aligned} & CS_1 && \text{scope operator in FOMLA} \\ =\ & \{\kappa(f(c)) \mid c \in cs\} && \text{def. of } \mathsf{mla2m} \\ =\ & \{\nu_1(c) \mid c \in cs\} && \end{aligned}$$

This is the semantics of the scope operator. This means that for all sets $CS' \subseteq \mathbb{I}$ we have $(CS', TS, \Omega, V, \nu) \models cs : \phi_1$ because the car identifier scope is replaced by the scope operator. This implies $M \models cs : \phi_1$.

Subcase 2.5 ($\phi \equiv \exists c.\, \phi_1$). Assume $\kappa \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon, \exists c.\, \phi_1) \wedge \mathsf{sane}(\Upsilon)$, where we assume w.l.o.g. $c \notin \mathsf{dom}\, f$. Then $\kappa \models \bigvee_{D \in DS} \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1) \wedge \mathsf{sane}(\Upsilon)$, where $\Upsilon_1 = (DS, \alpha, \alpha', \beta, \beta', f_1, \mathbb{D}, \mathcal{S}, D_E)$ and $f_1 = f \oplus \{c \mapsto D\}$. We choose $D \in DS$ such that $\kappa \models \mathsf{tr}_\mathsf{f}^\mathsf{mla}(\Upsilon_1, \phi_1) \wedge \mathsf{sane}(\Upsilon)$. As in the other cases, the formula $\mathsf{sane}(\Upsilon)$ ignores our FOMLA representation of the valuation and thus $\kappa \models \mathsf{sane}(\Upsilon_1)$ holds. As $\mathsf{freeVar}(\phi_1) \subseteq \mathsf{dom}\, f$ we can apply the IH and we have $M_1 \models \phi_1$, where $M_1 = (CS_1, TS_1, \Omega_1, V_1, \nu_1) = \mathsf{mla2m}(\kappa, \Upsilon_1)$.

By the definition of $\mathsf{mla2m}$ we have $\nu(c) = C$ with $C \in CS_1$. Thus, we can reintroduce the quantifier and get $M' \models \exists c.\, \phi_1$, where M' is equal to M_1 except that $\nu' = \{c\} \lhd \nu_1$. We are left to show that $M' = M$. It is clear that $CS = CS'$, $TS = TS'$, $\Omega = \Omega'$ and $V = V'$ because we only changed f, which only affects the valuation ν. Further, we know from the definition of $\mathsf{mla2m}$ that

$$\begin{aligned} \nu_1 &= \{\gamma \mapsto \kappa(f_1(\gamma)) \mid \gamma \in \mathsf{dom}\, f_1\} \text{ and} \\ \nu &= \{\gamma \mapsto \kappa(f(\gamma)) \mid \gamma \in \mathsf{dom}\, f\} \ . \end{aligned}$$

As $c \notin \mathsf{dom}\, f$ and $f_1 = f \oplus \{c \mapsto D\}$ it follows that $\nu = \{c\} \lhd \nu_1$, which implies $\nu = \nu'$. Thus, $M = M'$ and $M \models \exists c.\, \phi_1$.

Subcase 2.6 ($\phi \equiv \phi_1 \frown \phi_2$). We assume $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi_1 \frown \phi_2) \land \mathsf{sane}(\Upsilon)$. Then $\kappa \models \exists x \in [\beta, \beta']. \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1) \land \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2, \phi_2) \land \mathsf{sane}(\Upsilon)$ follows, where $\Upsilon_1 = (DS, \alpha, \alpha', \beta, x, f, \mathbb{D}, \mathcal{S}, D_E)$, $\Upsilon_2 = (DS, \alpha, \alpha', x, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$. This means that we can define $\kappa_1 = \kappa \oplus \{x \mapsto r\}$ for some suitable real number $r \in [\kappa(\beta), \kappa(\beta')]$ such that $\kappa_1 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1) \land \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2, \phi_2) \land \mathsf{sane}(\Upsilon)$, which implies $\kappa_1 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi_1) \land \mathsf{sane}(\Upsilon)$ and $\kappa_1 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_2, \phi_2) \land \mathsf{sane}(\Upsilon)$. As $\kappa_1 \models \beta \leq x \leq \beta'$ we further know $\kappa_1 \models \mathsf{sane}(\Upsilon_1)$ and $\kappa_1 \models \mathsf{sane}(\Upsilon_2)$.

As for $i \in \{1, 2\}$ we have $\mathsf{freeVar}(\phi_i) \in \mathsf{dom}\, f_i$, we can apply the IH and get $M_i \models \phi_i$, where $M_i = (CS_i, TS_i, \Omega_i, V_i, \nu_i) = \mathsf{mla2m}(\kappa_i, \Upsilon_i)$ with $V_i = (L_i, X_i, E_i)$, $X_1 = [\kappa'(\beta), \kappa'(x)]$ and $X_2 = [\kappa'(x), \kappa'(\beta')]$. Note that all elements in M_1, M_2 except the extensions are equal. Thus for $M' = (CS_1, TS_1, \Omega_1, (L_1, X_1 \oplus X_2, E), \nu_1)$ we have $M' \models \phi_1 \frown \phi_2$. At last, as $X = [\kappa'(\beta), \kappa'(\beta')]$ we see that $X = X_1 \oplus X_2$. Further, all other elements in M are equal to their corresponding counter parts in M', which implies $M \models \phi_1 \frown \phi_2$.

We finished the structural induction for both cases of the lemma. Thus, the lemma is proven. □

Now we can prove that we can decide lane-unbounded satisfiability of well-scoped MLSLS by checking our FOMLA constraints for satisfiability. For this proof we use the previous lemma. The difficulty now lies in choosing an appropriate FOMLA data tuple. We point out that while the previous lemma holds for arbitrary MLSLS formulas, the following lemma only holds for well-scoped formulas. The reason for this is that in the previous lemma we worked with *finite* models. That is, if a well-scoped MLSLS formula is satisfiable, then there is a finite model satisfying the formula.

Lemma 3.3.33. *Given a well-scoped MLSLS formula ϕ we have that*

$$\phi \textit{ is satisfiable} \quad \textit{iff} \quad \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \land \mathsf{sane}(\Upsilon) \textit{ is satisfiable} \ ,$$

where $\Upsilon = (\emptyset, y, y', x, x', f \uplus \{\mathsf{ego} \mapsto D_E\}, \mathbb{D}, \mathcal{S}, D_E)$ with $|\mathbb{D}| = |\mathsf{freeVar}(\phi)| + 1$ and f maps each variable from $\mathsf{freeVar}(\phi)$ to a distinct element from $\mathbb{D} \setminus \{D_E\}$.

Proof.
Case 1 (if). Assume $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \land \mathsf{sane}(\Upsilon)$ is satisfiable. Then there is an assignment κ such that $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \land \mathsf{sane}(\Upsilon)$. Then by Lemma 3.3.32 we know for the MLSLS model $M = \mathsf{mla2m}(\kappa, \Upsilon)$ that $M \models \phi$.

Case 2 (only if). Assume that ϕ is satisfiable. Then there is a proper model $M = (CS, TS, \Omega, V, \nu)$ such that $M \models \phi$. From Lemma 3.3.2 we conclude for $M' = (\emptyset, TS, \Omega, V, \nu)$ that $M' \models \phi$. Furthermore, we can remove all mapped variables $c \in \mathsf{CVar}$ from ν that are not in $\mathsf{freeVar}(\phi)$ without affecting the satisfaction. Let $\nu' = (\mathsf{freeVar}(\phi) \cup \{\mathsf{ego}\}) \lhd \nu$ be this reduced valuation. Then for $M'' = (\emptyset, TS, \Omega, V, \nu')$ we have $M'' \models \phi$. Further, note that $|\operatorname{\mathsf{dom}} \nu'| = |\mathsf{freeVar}(\phi)| + 1$.

Let $(\kappa_1, \Upsilon_1) = \mathsf{m2mla}(M'')$. Then, from Lemma 3.3.32 it follows that for $\Upsilon_1 = (DS_1, y_1, y_1', x_1, x_1', f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ we have $\kappa_1 \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon_1, \phi)$. Note that $DS_1 = \emptyset$ and $|\mathbb{D}_1| = |\operatorname{\mathsf{dom}} \nu'| = |\mathsf{freeVar}(\phi)| + 1$ and f_1 maps each element (including ego) from $\operatorname{\mathsf{dom}} \nu'$ to a distinct variable from $\mathbb{D}_1$. Further, from Proposition 3.3.27 we know $\kappa_1 \models \mathsf{sane}(\Upsilon_1)$ Thus, as (κ_1, Υ_1) and (κ, Υ) are equal up to renaming it follows that they are equisatisfiable, which implies that $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi) \land \mathsf{sane}(\Upsilon)$ is satisfiable. □

As satisfiability of FOMLA is decidable [Wei99], the following theorem holds. Note that we did not investigate the complexity of our procedure. However, for the complexity of FOMLA we refer to Lemma 2.4.4.

Theorem 3.3.34 (Decidability of well-scoped MLSLS)**.** *Satisfiability and validity of well-scoped MLSLS are decidable.*

3.4 Decidability of Model Problem for MLSLS

In the previous section we gave an algorithm to determine if a well-scoped MLSLS formula is satisfiable. In this section we extend this algorithm to check if a given finite model satisfies an MLSLS formula. We point out that we do not restrict ourselves to well-scoped formulas.

However, for this we can restrict ourselves to the first-order logic of linear real arithmetic (FOLRA), which is a fragment of FOMLA. First we define FOLRA data tuples, which are a modification of FOMLA data tuples. In FOLRA data tuples we replace natural number-valued variables by natural numbers and real-valued variables. Then we adapt our transformation of MLSLS formulas to arithmetic constraints to not use integer quantification any more. And at last we bring it all together and show correctness of our approach to solving the model problem.

As for FOMLA we represent the data of a car in so called *data variables.* Here however, we have no integer variables for claims and reservations. Instead we use only variables ranging over the real numbers. Let v_{p}, $v_{\Omega}, v_{\mathsf{s}}$, v_{a}, v_{r1}, v_{r2} and v_{c} from RVar be our *data variables*. As for FOMLA we collect these variables in a set s, and for a set of car identifier variables we collect the sets of data variables in a structure $\mathcal{S} : \mathsf{DVar} \to \mathsf{RVar}^7$ and for a given $\mathcal{S}$ we sometimes abbreviate $\mathcal{S}(D)(v_{\mathsf{p}})$ with $D.v_{\mathsf{p}}$ for $D \in \mathbb{D}$.

Additionally, we introduce data tuples. Note that as in FOLRA we do not have natural number-valued variables, we represent the lanes of a view with natural numbers, i.e. values, rather than variables. The reason why we represent lanes of the view with values and claims and reservations as variables will become clear in Chapter 5. The set of all *proper FOLRA data tuples* is defined as

$$\mathrm{T}^{\mathsf{lra}}_{\mathsf{p}} = \mathcal{P}(\mathsf{DVar}) \times \mathbb{N}^2 \times \mathsf{RTerm}^2 \times (\mathsf{CVar} \cup \{\mathsf{ego}\} \to \mathsf{DVar}) \times \\ \mathcal{P}(\mathsf{DVar}) \times (\mathsf{DVar} \to \mathsf{RVar}^7) \times \mathsf{DVar} \ .$$

We denote a FOLRA data tuple together with a FOLRA assignment as a *FOLRA tuple*. We reuse the notions of *proper* and *simple* FOMLA data tuples (cf. Pages 75 and 81), the notion of *composable* FOMLA tuples (cf. Page 86) and the operations *disjoint union*, *restriction* and *anti-restriction* for FOMLA tuples (cf. Page 86) for FOLRA. As for FOMLA data tuples, we require for a FOLRA data tuple $p = (DS, l, l', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ that $DS, \mathsf{ran}\, f \subseteq \mathbb{D}, \mathsf{dom}\, \mathcal{S} = \mathbb{D}, D_E \in \mathbb{D}$ and $f(\mathsf{ego}) = D_E$ and similarly for simple FOLRA data tuples (see also Page 75).

We give the transformation of MLSLS formulas to FOLRA. Note that in the FOMLA transformation the only concepts not available in FOLRA is integer quantification, used to represent vertical chop. Thus, for the other MLSLS operators we reuse the FOMLA definition.

Definition 3.4.1. The *transformation* is given by a function

$$\mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}} : \mathrm{T}^{\mathsf{lra}}_{\mathsf{p}} \times \Phi \to \mathsf{FOLRAFormulas} \ .$$

Let $p = (DS, l, l', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E) \in \mathrm{T}^{\mathsf{lra}}_{\mathsf{p}}$. Then the transformation is given as:

$$\mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p, \phi) \equiv \begin{array}{l} l \leq l' \implies \bigvee\limits_{l'' \in [l-1, l']} \left(\begin{array}{rl} & \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}((DS, l, l'', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E), \phi_1) \\ \wedge & \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}((DS, l''+1, l', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E), \phi_2) \end{array} \right) \\ \wedge \\ l > l' \implies \left(\begin{array}{rl} & \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}((DS, l, l', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E), \phi_1) \\ \wedge & \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}((DS, l, l', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E), \phi_2) \end{array} \right) \end{array}$$

if $\phi \equiv \binom{\phi_2}{\phi_1}$ and $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(p, \phi)$ otherwise, where $\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}$ is the transformation from Definition 3.3.5. △

We define a transformation of MLSLS models to arithmetic constraints. Note that the difference to $\mathsf{m2mla}$ (Definition 3.3.20) is that here we do not create an assignment. To simplify things we use for a given MLSLS model M a precomputed *car identifier variable assignment* $g : \mathsf{DVar} \to \mathbb{I}$. Often we use for $\mathbb{D} \subseteq \mathsf{DVar}$ the type $g : \mathbb{D} \to \mathbb{I}$. Essentially, g is the restriction of a FOMLA assignment representing an MLSLS model to the car identifier variables. We define constraints on good choices for p and g for a given finite M by relating the three. Intuitively, p, g, M are well-chosen, which we shall call *sane*, if g can be extended with some assignment h such that $(g \uplus h, p)$ represents M.

Definition 3.4.2. For a simple MLSLS model $M = (CS, TS, \Omega, \nu)$, a simple FOLRA data tuple $p = (DS, f, \mathbb{D}, \mathcal{S})$ and a car identifier variable assignment $g : \mathbb{D} \to \mathbb{I}$ we call (p, g, M) *sane* if

$$g(\!|DS|\!) = CS \ ,$$
$$\mathsf{ran}\, g = \mathsf{cars}\, M \ ,$$
$$g \circ f = \nu \ .$$

We lift this definition to proper models and data tuples: For a proper MLSLS model $M = (CS, TS, \Omega, V, \nu)$ with $V = ([l, l'], [r, r'], E)$ and a proper FOLRA data tuple $p_1 = (DS_1, l_1, l_1', \beta_1, \beta_1', f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$ we additionally require $g(D_{E1}) = E$, $l = l_1, l' = l_1'$ and the FOLRA constraints $\beta_1 = r \wedge \beta_1' = r'$ being satisfiable. △

In the definition above we require that $\beta_1 = r$ is satisfiable because β_1 is a *term*, i.e. it may or may not contain variables. If β_1 does not contain variables then $\beta_1 = r$ is is only satisfiable, if β_1 evaluates to r. Similar reasoning applies for β_1'. Note that even if β_1 and β_1' both contain variables, $\beta_1 = r \wedge \beta_1' = r'$ still may not be satisfiable. This may be the case when β_1 and β_1' are not independent of each other because β_1 and β_1' share their variables.

We state that our operations preserve sanity.

Proposition 3.4.3. *For $i \in \{1, 2\}$ let M_i be finite and composable MLSLS models and (p_i, g_i) composable FOLRA tuples such that (p_i, g_i, M_i) is sane.*

Then, for a finite set of car identifiers $CS \subseteq \mathbb{I}$ we have

$$
\begin{aligned}
&(p, g, M_1 \boxplus M_2) \textit{ is sane, where } (p, g) = (p_1, g_1) \oplus (p_2, g_2) \enspace , \\
&(p, g, M_1 \boxminus CS) \textit{ is sane, where } (p, g) = (p_1, g_1) \ominus CS \enspace , \\
&(p, g, M_1 \boxbar CS) \textit{ is sane, where } (p, g) = (p_1, g_1) \obar CS \enspace .
\end{aligned}
$$

We first define the constraints for a model consisting of a single car. As for the functions transforming FOMLA tuples to MLSLS models and vice versa, we use E to denote an elementary (or base) version of a function that we extend later.

Definition 3.4.4. Let $M = (CS, TS, \Omega, \nu)$ with $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$ be a simple MLSLS model with $\mathsf{cars}\, M = \{C\}$ and $p = (DS, f, \mathbb{D}, \mathcal{S})$ a simple FOLRA data tuple. Then we define

$$
\begin{aligned}
\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p, M) \equiv \bigwedge_{D \in \mathbb{D}} \big(& D.v_{\mathsf{p}} = \mathsf{pos}(C) \wedge \{D.v_{\mathsf{r1}}, D.v_{\mathsf{r2}}\} = \mathsf{res}(C) \wedge \\
& \{D.v_{\mathsf{c}}\} = \mathsf{clm}(C) \wedge D.v_{\mathsf{s}} = \mathsf{spd}(C) \wedge D.v_{\mathsf{a}} = \mathsf{acc}(C) \wedge D.v_{\Omega} = \Omega(C, TS) \big) \enspace .
\end{aligned}
$$

We extend this to proper models. For an MLSLS model M with a view $V = (L, [r, r'], E)$ and $|\mathsf{cars}\, M| = 1$ and a proper FOLRA data tuple $p = (DS, l, l', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ we additionally require $\beta = r \wedge \beta' = r'$. $\triangle$

Note that in the definition above we do not represent the lane, the valuation or the scope. Later we constrain these by requiring that the FOMLA tuple and the model together are sane.

We extend the previous definition to models with more cars. Note that from Proposition 3.4.3 it follows that we use the formula only for inputs where the result is defined, i.e. only for sane inputs.

Definition 3.4.5. Let M be a possibly simple MLSLS model, p a possibly simple FOLRA data tuple and g a car identifier variable assignment such that all of them together are sane. Then we define

$$
\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M) \equiv
\begin{cases}
\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_1, M \boxbar \{C\}) \wedge \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p_2, g_2, M \boxminus \{C\}) \enspace , & \text{if } \mathsf{cars}\, M \neq \emptyset \\
\text{where } (p_1, g_1) = (p, g) \obar \{C\}, & \\
\qquad (p_2, g_2) = (p, g) \ominus \{C\} \text{ and } C \in \mathsf{cars}\, M & \\
\mathsf{true} & \text{otherwise} \enspace .
\end{cases}
$$

$\triangle$

The following lemma states how we can derive a FOLRA assignment that satisfies the arithmetic constraints representing an MLSLS model.

Lemma 3.4.6. *Let M be a finite and possibly simple MLSLS model, p a possibly simple FOLRA data tuple and $g : \mathsf{DVar} \to \mathbb{I}$ a car identifier variable assignment such that (p, g, M) is sane. Then there is an FOLRA assignment h such that*

$$g \subseteq h \text{ and } h \models \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M) \ .$$

Proof. First we consider the case that M is simple. For all $C \in \mathsf{cars}\, M$ we can simply assign the values in $M \mathbin{\boxed{\upharpoonright}} \{C\}$ to the data variables in $(g, p) \mathbin{\hexagon\!\!\upharpoonright} \{C\}$. The resulting assignment satisfies $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M)$.

If M is proper let $p = (DS_1, l_1, l'_1, \beta_1, \beta'_1, f_1, \mathbb{D}_1, \mathcal{S}_1, D_{E1})$. As before, we assign values to the data variables in p to get an assignment h. This assignment does not yet consider β_1 and β'_1. As p, g and M are sane we know that $\beta_1 = r \wedge \beta'_1 = r'$ is satisfiable. Let h' be such a satisfying assignment, where h' might be the empty function if β_1 and β'_1 do not contain free variables. Note that the terms representing the extension do not share any variables with $\mathbb{D}_1$ or $\mathcal{S}_1$. Hence, we can use $\uplus$ on the two assignments. Thus, $h \uplus h' \models \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M)$. $\square$

Now we can prove that our encoding of the model problem for MLSLS formulas is correct. With correct we mean here that the FOLRA constraints are valid iff the MLSLS model satisfies the MLSLS formula.

Lemma 3.4.7. *Let M be a finite and proper MLSLS model, $p \in \mathrm{T}^{\mathsf{lra}}_{\mathsf{p}}$ a FOLRA data tuple and $g : \mathsf{DVar} \to \mathbb{I}$ a car identifier variable assignment such that (p, g, M) is sane. Then, for all MLSLS formulas ϕ with $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}\,\nu$ we have*

$$\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M) \implies \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p, \phi) \textit{ is valid} \text{ iff } M \models \phi$$

Proof. For this proof let $p = (DS, l, l', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$.
Case 1 (only if). Assume $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M) \implies \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p, \phi)$ is valid. With Lemma 3.4.6 we can extend g to a FOLRA assignment h such that $h \models \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M)$. Because of the validity, it follows that the assignment satisfies the transformed MLSLS formula, i.e. $h \models \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p, \phi)$.

The assignment assigns all reservation and claim variables in p nonnegative integer values (ensured by $h \models \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M)$). Hence, we can create a FOMLA tuple (κ, Υ) from h and p by replacing for all $D \in \mathbb{D}$ the variables $\mathcal{S}(D)(v_{\mathsf{c}}), \mathcal{S}(D)(v_{\mathsf{r1}})$ and $\mathcal{S}(D)(v_{\mathsf{r2}})$ by natural number-valued variables, while keeping the assigned value.

Because the MLSLS model M is sane, the assignment h essentially is a copy of values in M and κ is a copy of h, we know that (κ, Υ) is also sane, i.e. $\kappa \models \mathsf{sane}(\Upsilon)$. Further, from $h \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p, \phi)$ we know $\kappa \models \mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}(\Upsilon, \phi)$.

From Lemma 3.3.32 we know that for $M' = \mathsf{mla2m}(\kappa, \Upsilon)$ we have $M' \models \phi$. At last, we conclude $M = M'$ because we did not change any values when going from FOLRA to FOMLA, which implies $M \models \phi$.

Case 2 (if). We proceed by contraposition. Assume $\mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p, g, M) \implies \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p, \phi)$ is not valid. Then there is an assignment h that satisfies the premise but not the conclusion of the implication. That is, we have $h \models \mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p, g, M) \land \neg\mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p, \phi)$, which is equivalent to $h \models \mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p, g, M) \land \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p, \neg\phi)$.

We apply the same steps as in the previous case to create (κ, Υ) from h and p. As before, we conclude for $M' = \mathsf{mla2m}(\kappa, \Upsilon)$ that we have $M' \models \neg\phi$, which implies $M' \not\models \phi$. As we did not change any values when going from FOLRA to FOMLA we conclude $M = M'$, which implies $M \not\models \phi$. $\square$

As FOLRA is a fragment of FOMLA, for which satisfiability is decidable [Wei99], the following theorem holds. We did not investigate the complexity of our procedure, but for the complexity of FOLRA we refer to Lemma 2.4.6.

Theorem 3.4.8 (Decidability of Model Problem for MLSLS)**.** *The model problem for MLSLS is decidable.*

3.5 Related Work

Interval Logics

One of the first interval logics used in computer science is Interval Temporal Logic (ITL) [Mos85]. It is used to specify and verify hardware circuits. To the best of our knowledge ITL is the first temporal logic featuring the chop operator.

Halpern-Shoham-logic (HS) [HS91] is an interval-based modal logic, where for each of Allen's interval relations [All83] (such as *before* or *overlaps*) there is a modal operator capturing the semantics of this relation. For HS and its fragments there are plenty of results for properties such as (un)decidability (cf. [BMG+08] for a survey). One insight is that interval logics quickly become undecidable.

Another well known interval logic is Duration Calculus (DC) [ZHR91]. The main feature of DC is the chop operator and the integral operator. With the integral operator we can express such as “in any interval of length ≥ 60, gas is leaking for at most 5 % of the time” [SRR90; RRH93]. Duration Calculus is decidable if we disallow the integral operator and instead allow the almost everywhere operator ($\lceil \cdot \rceil$) [ZH04], or if we only allow the chop operator to occur below an even number of negations [FH07].

Shape Calculus [Sch06] is an extension of DC to multiple dimensions. The author shows that the fragment $\mathrm{SC}_{\mathrm{fin}}$, which restricts models to one dense infinite dimension and multiple finite discrete dimensions, is decidable. These models are similar to the model of MLSL, which consists of one dense and one discrete dimension. However, unlike MLSL, $\mathrm{SC}_{\mathrm{fin}}$ does not allow for quantitative length measurements. If the infinite dimension in $\mathrm{SC}_{\mathrm{fin}}$ instead is discrete, quantitative measurements are possible. In our case we allow for quantitative statements and have a dense dimension.

Guarded First-Order Theory

For our decidability result we introduced the scope operator to restrict quantifiers over car variables to bounded domains. This is similar to guarded first-order logic of [ANB98], where the authors show that the fragment of first-order logic consisting of formulas of the type $\exists y.\, \alpha(x, y) \wedge \phi(x, y)$ and $\forall y.\, \alpha(x, y) \implies \phi(x, y)$, where α is an atom serving as a guard to restrict the quantified variables and ϕ is a formula of first-order logic. Their guard seems quite similar to our restriction of the quantifier over car variables. However, MLSL is a modal logic extended with first-order quantifiers. Thus, their approach is not directly applicable.

Spatial Logics

There exist other spatial logics. One of the most popular ones is the Region Connection Calculus (RCC) [RCC92]. It can be used to formalise and reason about topological properties of regions. For example, we can express that two regions are connected via a third region. However, RCC does not allow for quantitative properties and it is not suited to reason about traffic configurations because of the special significance of the road separated into adjacent and disjoint lanes.

4 Satisfiability of Spatial Properties with Imprecise Information

In this chapter we investigate whether satisfiability is decidable if we assume that spatial information is known only approximately. This is interesting because often it is unlimited precision that leads to undecidability [LH15; AFH96; Frä99]. Note that even though we did not use length measurement in the previous chapter, we still were able to express that one reservation starts exactly where another reservation ends. This is not possible in this chapter. Additionally, we adapt our encoding of the model problem from the previous chapter to the setting that information contains small errors.

The undecidability result is mostly taken from [Ody15b; Ody15a]. In this work we slightly adapt the construction from [Ody15b; Ody15a] such that the set of satisfying models for the complete formula forms an open set.[1] Furthermore, we adapt the proof of undecidability by connecting it to the undecidability proof of the previous chapter and we make the arguments from [Ody15a] clearer.

The idea for the procedure to decide the static model problem with imprecise information is taken from [Ody17]. However, there we jump straight to the dynamic case. In this thesis we explicitly state the easier static case and separately prove its correctness.

In Section 4.1 we adapt our undecidable construction from the previous chapter to the setting of imprecise information, which we prove undecidable in Section 4.2. In Section 4.3 we encode the model problem with imprecise information and in Section 4.4 we discuss related work.

[1]Intuitively, a set is open if it does not have a clear boundary. For example, the closed interval $[1, 2]$ is not an open set, while the open interval $(1, 2)$ forms an open set.

4.1 Undecidability of Robust Satisfiability

The undecidability result for MLSL by Hilscher and Linker relies on length measurement [LH15] and in their construction the authors rely on exact values. In the previous chapter we investigated whether the satisfiability problem of MLSL becomes decidable if we trade length measurement for an unbounded number of lanes. We extend this and investigate if satisfiability remains undecidable for an unbounded number of lanes, even if we perturb positional data. That is, even though in Chapter 3 we do not use length measurement for the undecidability result, we still rely on correct positional information. For example, we represented letters as nonoverlapping, successive reservations without free space in between, i.e. when the position of a single car is shifted by a small amount the resulting model does not satisfy the formula anymore.

Here we show that the lane-unbounded satisfiability problem of MLSL remains undecidable, even when the position of cars along the lanes are perturbed. We say that a model M *robustly* satisfies a formula iff there is some $\delta > 0$ such that all models differing by at most δ (w.r.t. to a certain *metric*) from M also satisfy the formula. A formula is robustly satisfiable iff there is a model that robustly satisfies the formula. In the following we adapt our construction from Section 3.1 to this definition of robust satisfiability.

Our definition of robust satisfiability is similar to the definition of tube acceptance from [GHJ97], where a robust timed automaton accepts a trajectory iff the automaton also accepts all similar trajectories.

4.1.1 Robust Satisfiability of MLSL

To formalise similarity usually metrics are introduced to define distances between elements. Here we need a metric to assign a distance to every two models. Then, similarity can be quantified with these metrics. To capture positional similarity of two models we assign a distance of ∞ if any other data than position, extension or sensor function differ. Otherwise, we assign the maximal difference of these values. For a car C, a sensor function Ω and a traffic snapshot $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$ we remind that $\mathsf{se}(C, TS, \Omega)$ is an abbreviation for the safety envelope of C. Further, $\overline{\mathsf{se}(C, TS, \Omega)}$ is the right end of this safety envelope, and $\underline{\mathsf{se}(C, TS, \Omega)}$, which is equal to $\mathsf{pos}(C)$, is the left end of the safety envelope. The following definition is adapted from [Ody15b].

Definition 4.1.1 (Metric on MLSL Models)**.** For $i \in \{1, 2\}$ let there be two MLSL models $M_i = (TS_i, \Omega_i, V_i, \nu_i)$ with $TS_i = (\mathsf{res}_i, \mathsf{clm}_i, \mathsf{pos}_i, \mathsf{spd}_i, \mathsf{acc}_i)$ and

$V_i = (L, [r_i, r'_i], E_i)$. We define

$$d_{\mathsf{m}}(M_1, M_2) = \begin{cases} \infty & \text{if} \quad (\mathsf{res}_1, \mathsf{clm}_1, \nu_1, L_1, E_1) \\ & \quad \neq (\mathsf{res}_2, \mathsf{clm}_2, \nu_2, L_2, E_2) \\ \mathsf{max}\big(|r_1 - r_2|, |r'_1 - r'_2|, & \\ \quad \mathsf{sup}_{C \in \mathbb{I}}(|\mathsf{pos}_1(C) - \mathsf{pos}_2(C)|, & \\ \quad\quad |\overline{\mathsf{se}(C, TS_1, \Omega_1)} - \overline{\mathsf{se}(C, TS_2, \Omega_2)}|)\big) & \text{otherwise} \ . \end{cases}$$

△

Since we only use absolute values, the distance between two models is always positive. Further, it is not difficult to show that the triangle inequality is satisfied. Thus, d_{m} is indeed a metric. This means that the distance of two models is infinite, if they disagree on discrete values. Otherwise the distance is the greatest difference of any dense value. For $\delta \in \mathbb{R}_{>0}$ we say that two models M, M' are *δ-similar*, if $d_{\mathsf{m}}(M, M') \leq \delta$.

We define that a model δ-robustly satisfies a formula if all δ-similar models also satisfy the formula.

Definition 4.1.2 (Robust Satisfaction of MLSL Formulas)**.** Given a model M, a desired error allowance $\delta \in \mathbb{R}_{>0}$ and a formula ϕ, we define that M *satisfies ϕ with robustness δ* as

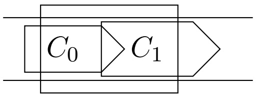
$$M \models^{\delta} \phi \text{ iff } \forall M'. (d_{\mathsf{m}}(M, M') \leq \delta \implies M' \models \phi) \ . \qquad △$$

We say that M *robustly satisfies* ϕ iff there is $\delta \in \mathbb{R}_{>0}$ such that $M \models^{\delta} \phi$. An MLSL formula ϕ is *robustly lane-unboundedly satisfiable* (resp. robustly lane-boundedly satisfiable) iff there is an infinite (resp. finite) set of lanes $\mathbb{L}$ and an MLSL model M such that M robustly satisfies ϕ.

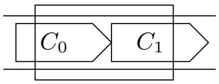
Example 4.1.3. Consider the following formulas:

$$\phi_0 \equiv c_0 \neq c_1 \wedge \langle \mathsf{re}(c_0) \frown \mathsf{re}(c_1) \rangle \ ,$$
$$\phi_1 \equiv \phi_0 \wedge \neg \exists c_2, c_3. (c_2 \neq c_3 \wedge \langle \mathsf{re}(c_2) \wedge \mathsf{re}(c_3) \rangle) \ .$$

The formula ϕ_0 requires that there are two successive reservations from different cars without free space in between. The model M_0, depicted in Figure 4.1a, robustly satisfies ϕ_0 because the positions of the cars can be perturbed by a



(a) The model M_0, which contains two overlapping reservations.



(b) The model M_1, which contains two reservations. The second reservation starts exactly where the first ends.

Figure 4.1: Depiction of two MLSL models to visualise the difference of classical and robust satisfaction.

small amount without affecting satisfaction by the model. Hence, ϕ_0 is robustly satisfiable. The model, M_1 (Figure 4.1b) does not robustly satisfy ϕ_0 because if the position of c_1 is increased (moved to the right) by an arbitrary small amount there is free space between the reservations, which violates ϕ_0.

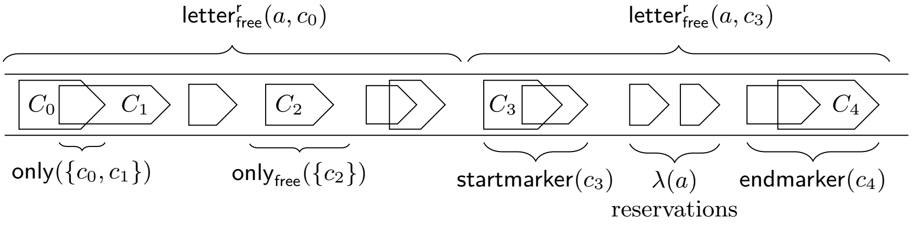
The formula ϕ_1 additionally requires that there are no overlapping reservations. Thus, M_0 does not satisfy the formula. While M_1 satisfies ϕ_1, moving a single car in any direction either creates overlapping reservations or free space, both of which violate the formula. In general ϕ_1 requires that the reservation of c_0 ends exactly where the reservation of c_1 starts. Naturally, this is not robustly satisfiable. △

4.1.2 Construction

We need to replace those parts of our construction that are affected by small perturbations of positions. For this we adapt our representation of letters, because they are represented as successive reservations starting exactly where another reservation ends. Additionally, we have to adapt the subseq-formulas, because they ensured perfect alignment of letters, which is not possible in a setting with perturbations. To distinguish the robust from the nonrobust formulas we mark formulas from our robust construction that occur with the same name in the nonrobust construction with r.

Letter

To represent letters we remove the assumption that there are no overlapping reservations. However, we still do not consider cars with multiple reservations.



$\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(a, c_0)$
$\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(a, c_3)$
C_0
C_1
C_2
C_3
C_4
$\mathsf{only}(\{c_0, c_1\})$
$\mathsf{only}_{\mathsf{free}}(\{c_2\})$
$\mathsf{startmarker}(c_3)$
$\lambda(a)$
reservations
$\mathsf{endmarker}(c_4)$

Figure 4.2: Visualisation of a model satisfying $\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(a, c_0) \frown \mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(a, c_3)$ with $\lambda(a) = 2$, where each variable c_j is mapped to C_j with $j \in \{0, \dots, 4\}$. The view is not depicted and different heights of the reservations are used for better visualisation.

For a finite set $\mathcal{C} \subseteq \mathsf{CVar}$ of car variables we define

$$\mathsf{only}(\mathcal{C}) \equiv \bigwedge_{c \in \mathcal{C}} \mathsf{re}(c) \land (\forall c'.\, (\mathsf{true} \frown \mathsf{re}(c') \frown \mathsf{true}) \implies \bigvee_{c'' \in \mathcal{C}} c'' = c') \enspace ,$$
$$\mathsf{only}_{\mathsf{free}}(\mathcal{C}) \equiv \mathsf{free} \frown \mathsf{only}(\mathcal{C}) \frown \mathsf{free}\,,$$

to ensure that the current view is filled by reservations from all car variables in $\mathcal{C}$, but does not contain reservations from any other cars. See Figure 4.2 for a visualisation. Now we can define our representation of letters as

$$\begin{aligned}
\mathsf{letter}^{\mathsf{r}}(\sigma, c) &\equiv \mathsf{startmarker}(c) \frown \exists c_0, \dots, c_{\lambda(\sigma)-1}.\, (\bigwedge_{i,j \in \{0,\dots,\lambda(\sigma)-1\}}^{i \neq j} c_i \neq c_j \land \\
&\qquad \mathsf{only}_{\mathsf{free}}(\{c_0\}) \frown \dots \frown \mathsf{only}_{\mathsf{free}}(c_{\lambda(\sigma)-1})) \frown \exists c'.\, \mathsf{endmarker}(c') \enspace , \\
\mathsf{startmarker}(c) &\equiv \mathsf{only}(\{c\}) \frown \exists c'.\, c \neq c' \land \mathsf{only}(\{c, c'\}) \frown \mathsf{only}(\{c'\}) \enspace , \\
\mathsf{endmarker}(c) &\equiv \exists c'.\, c \neq c' \land \mathsf{only}(\{c'\}) \frown \mathsf{only}(\{c, c'\}) \frown \mathsf{only}(\{c\}) \enspace ,
\end{aligned}$$

where σ is either a terminal or a nonterminal and c is a car variable. Our representation of letters begins and ends with two different markers. In between these markers the representation contains $\lambda(\sigma)$ reservations (see Page 35 for $\lambda(\sigma)$). This representation of letters does not depend on exact positions of cars. In Figure 4.2 we depict two adjacent letters with free space in between them.

Subsequence

Before coming to our subsequence formula we point out that we use a renaming function for terminals, as in Section 3.1. That is, given two context-free grammars $G_i = (\mathcal{N}_i, \mathcal{T}, \mathcal{R}_i, S_i)$ with $i \in \{\mathsf{D}, \mathsf{U}\}$, we assume two sets $\mathcal{T}_\mathsf{D}$, $\mathcal{T}_\mathsf{U}$ and two bijective functions $\pi_\mathsf{D} : \mathcal{T} \to \mathcal{T}_\mathsf{D}$ and $\pi_\mathsf{U} : \mathcal{T} \to \mathcal{T}_\mathsf{U}$ such that $\mathcal{T}_\mathsf{D}$, $\mathcal{N}_\mathsf{D}$, $\mathcal{T}_\mathsf{U}$ and $\mathcal{N}_\mathsf{U}$ are all disjoint. Further, for $i \in \{\mathsf{D}, \mathsf{U}\}$ and $\tau \in \mathcal{T}$ we use τ_i as abbreviation for $\pi_i(\tau)$.

In Section 3.1 we ensured that terminals at the same position in the derived words are horizontally aligned. With imprecise positions we cannot ensure such an alignment. We define the new subsequence formulas similar to their definition in Section 3.1. For the following formula we point out that $\mathsf{letter}^\mathsf{r}_\mathsf{free}$ is defined as $\mathsf{letter}_\mathsf{free}$ on Page 37, with the difference that $\mathsf{letter}^\mathsf{r}_\mathsf{free}$ uses $\mathsf{letter}^\mathsf{r}$ instead of letter. That is, $\mathsf{letter}^\mathsf{r}_\mathsf{free}(\sigma, c) \equiv \mathsf{free} \frown \mathsf{letter}^\mathsf{r}(\sigma, c) \frown \mathsf{free}$. Let c, c' be car variables, then we define

$$\phi(\tau, c, c') \equiv \left\langle \begin{pmatrix} \mathsf{free} \\ \mathsf{true} \\ \mathsf{free} \end{pmatrix} \frown \left(\begin{pmatrix} \mathsf{letter}^\mathsf{r}_\mathsf{free}(\tau_\mathsf{D}, c) \\ \mathsf{true} \\ \mathsf{letter}^\mathsf{r}(\tau_\mathsf{U}, c') \end{pmatrix} \vee \begin{pmatrix} \mathsf{letter}^\mathsf{r}(\tau_\mathsf{D}, c) \\ \mathsf{true} \\ \mathsf{letter}^\mathsf{r}_\mathsf{free}(\tau_\mathsf{U}, c') \end{pmatrix} \right) \frown \begin{pmatrix} \mathsf{free} \\ \mathsf{true} \\ \mathsf{free} \end{pmatrix} \right\rangle,$$

which requires that one representation of τ is horizontally strictly contained within the other representation. Horizontal containment is ensured by aligning $\mathsf{letter}^\mathsf{r}$ with $\mathsf{letter}^\mathsf{r}_\mathsf{free}$. The disjunction represents that it does not matter which representation is the larger one. Further, we define

$$\mathsf{subseq}^\mathsf{r}_\mathsf{D} \equiv \bigwedge_{\tau \in \mathcal{T}} \forall c.\, (\langle \mathsf{letter}^\mathsf{r}_\mathsf{free}(\tau_\mathsf{D}, c) \rangle \implies \exists c'.\, (\langle \mathsf{letter}^\mathsf{r}_\mathsf{free}(\tau_\mathsf{U}, c') \rangle \wedge \phi(\tau, c, c'))) \ ,$$

$$\mathsf{subseq}^\mathsf{r}_\mathsf{U} \equiv \bigwedge_{\tau \in \mathcal{T}} \forall c'.\, (\langle \mathsf{letter}^\mathsf{r}_\mathsf{free}(\tau_\mathsf{U}, c') \rangle \implies \exists c.\, (\langle \mathsf{letter}^\mathsf{r}_\mathsf{free}(\tau_\mathsf{D}, c) \rangle \wedge \phi(\tau, c, c'))) \ .$$

As in Section 3.1, the subformula $\phi(\tau, c, c')$ is the same in $\mathsf{subseq}^\mathsf{r}_\mathsf{D}$ and $\mathsf{subseq}^\mathsf{r}_\mathsf{U}$ and we swapped the car variable names and the subscripts D and U outside $\phi(\tau, c, c')$.

Final Formula

To make our new letter formula work we have to allow overlapping reservations. Hence, we remove mutex from our construction. For $i \in \{\mathsf{D}, \mathsf{U}\}$ the formulas $\mathsf{stepAll}^\mathsf{r}_i, \mathsf{start}^\mathsf{r}_i, \mathsf{letterNextToLetter}^\mathsf{r}_i, \mathsf{subseq}^\mathsf{r}_i, \mathsf{allResInLetter}^\mathsf{r}$ mostly remain as in

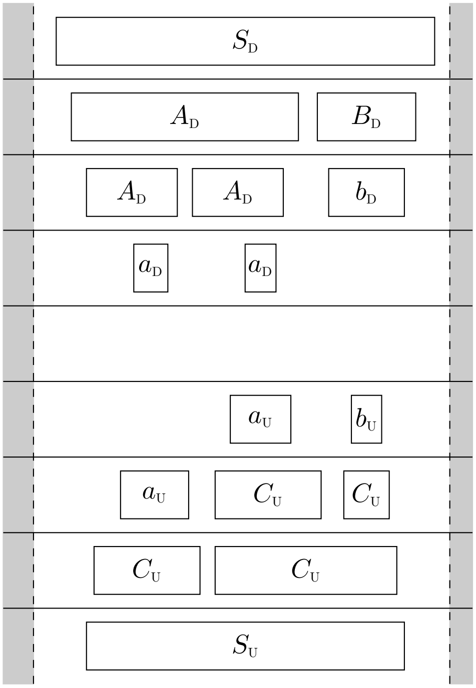


Figure 4.3: Visualisation of an MLSL model satisfying $F_{\mathrm{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ for the grammars from Example 2.1.6. We can see that the size of terminals in the two derivations may differ and that always one terminal is strictly contained in its counterpart of the other derivation or vice versa. The boxes corresponding to letters and reservations are not shown. The shaded area is the part where true holds, i.e. where we impose no restrictions.

Section 3.1 (cf. Pages 37 to 40). The only difference is that they use the new letter-formulas $\mathsf{letter}^{\mathsf{r}}$ and $\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}$. In the final formula we surround our formulas encoding the trees with true to handle perturbation of the view. We define

$$\begin{aligned}
\mathsf{asm}^{\mathsf{r}} &\equiv \mathsf{noClaims} \wedge \mathsf{noTwoRes} \ , \\
F_1 &\equiv \bigwedge_{i \in \{\mathsf{U},\mathsf{D}\}} \mathsf{stepAll}^{\mathsf{r}}_i \wedge \mathsf{start}^{\mathsf{r}}_i \wedge \mathsf{letterNextToLetter}^{\mathsf{r}}_i \wedge \mathsf{subseq}^{\mathsf{r}}_i \wedge \\
&\qquad \mathsf{freeLane} \wedge \mathsf{allResInLetter}^{\mathsf{r}} \wedge \mathsf{asm}^{\mathsf{r}} \ , \\
F_{\mathrm{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}}) &\equiv \mathsf{true} \frown F_1 \frown \mathsf{true} \ .
\end{aligned}$$

For a visualisation of a model satisfying $F_{\mathrm{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ for the grammars from Example 2.1.6 we refer to Figure 4.3.

4.2 Proving Undecidability of Robust Satisfiability of MLSL

In this section we prove for two context-free grammars $G_{\mathsf{D}}, G_{\mathsf{U}}$ in Chomsky normal form without ϵ-rules (CNF$^{-\epsilon}$) that our MLSL formula $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ from the previous section is robustly lane-unboundedly satisfiable iff the intersection of the two grammars is not empty. For this we first prove the robust formula and the nonrobust formula to be lane-unboundedly equisatisfiable. Then we show that the set of models satisfying $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ forms an open set with our metric on MLSL models. This means that we can perturb models satisfying $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ without affecting satisfaction.

We start by proving that for two context-free grammars $G_{\mathsf{D}}, G_{\mathsf{U}}$ in CNF$^{-\epsilon}$ the formulas $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ and $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ (see Pages 41 and 125) are equisatisfiable. To prove this, we need methods to transform robust representations of letters to nonrobust representations and vice versa. First, we define two functions that put nonoverlapping reservations from a given set of cars into a given view: the first function does not leave any free space between the reservations, and the second function surrounds each reservation with free space. Both functions ignore previous reservations. We remind that $\mathbb{M}$ is the set of all MLSL models and that $\mathbb{V}$ is the set of all views.

Definition 4.2.1. Let $\mathcal{C} = \{C_1, \dots, C_n\} \subseteq \mathbb{I}$ be a nonempty set of car identifiers, $M = (TS, \Omega, V, \nu)$ an MLSL model with $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$ and $V' = ([l', l'], X', E)$ a view with one lane. Further, let $\mathsf{res}' = \mathsf{res} \oplus \{C \mapsto \{l'\} \mid C \in \mathcal{C}\}$, $\mathsf{clm}' = \mathsf{clm} \oplus \{C \mapsto \emptyset \mid C \in \mathcal{C}\}$, $j_C \in \{1, \dots n\}$ such that $C = C_j$ and let TS'' be an arbitrary traffic snapshot. We define the function $f : \mathbb{M} \times \mathcal{P}(\mathbb{I}) \times \mathbb{V} \to \mathbb{M}$ to fill the given view completely with reservations of the given car identifiers as

$$f(M, \mathcal{C}, V') = ((\mathsf{res}', \mathsf{clm}', \mathsf{pos}', \mathsf{spd}, \mathsf{acc}), \Omega', V, \nu) \text{ with}$$

$$\mathsf{pos}'(C) = \begin{cases} \underline{X'} + \frac{\|X'\| * (j_C - 1)}{|\mathcal{C}|} & \text{if } C \in \mathcal{C} \ , \\ \mathsf{pos}(C) & \text{otherwise} \ , \end{cases}$$

$$\Omega'(C, TS'') = \begin{cases} \frac{\|X'\| * j_C}{|\mathcal{C}|} & \text{if } C \in \mathcal{C} \ , \\ \Omega(C, TS'') & \text{otherwise} \ . \end{cases}$$

Further, we define $g : \mathbb{M} \times \mathcal{P}(\mathbb{I}) \times \mathbb{V} \to \mathbb{M}$ to fill the given view with reservations padded with free space as

$$g(M, \mathcal{C}, V') = ((\mathsf{res}', \mathsf{clm}', \mathsf{pos}', \mathsf{spd}, \mathsf{acc}), \Omega', V, \nu) \text{ with}$$

$$\mathsf{pos}'(C) = \begin{cases} \underline{X'} + \frac{\|X'\| * (2j_C - 1)}{2|\mathcal{C}|+1} & \text{if } C \in \mathcal{C} \ , \\ \mathsf{pos}(C) & \text{otherwise} \ , \end{cases}$$

$$\Omega'(C, TS'') = \begin{cases} \frac{\|X'\| * 2j_C}{2|\mathcal{C}|+1} & \text{if } C \in \mathcal{C} \ , \\ \Omega(C, TS'') & \text{otherwise} \ . \end{cases}$$

△

In the definition above res' is equal to res, except that all reservations from cars in $\mathcal{C}$ are moved to lane l'. Similarly, clm' is equal to clm, except that the cars in $\mathcal{C}$ have their claims removed. Further, for f we split the extension into $|\mathcal{C}|$ parts of equal size and let each car in $\mathcal{C}$ have one part of the extension. In g we split the extension into $2|\mathcal{C}| + 1$ parts to ensure that their safety envelopes do not touch. Also we point out that the sensor function Ω' is defined for all traffic snapshots TS'' equally. We stress that we do not change the view of the model. Instead, we move the cars in $\mathcal{C}$ into the view V'.

Throughout this section we use for $i \in \{\mathsf{D}, \mathsf{U}\}$ and two context-free grammars $G_i = (\mathcal{N}_i, \mathcal{T}, \mathcal{R}_i, S_i)$ two bijective functions $\pi_i : \mathcal{T} \to \mathcal{T}_i$ such that $\mathcal{N}_i, \mathcal{T}, \mathcal{T}_i$ are all disjoint. Similar to Chapter 3 we define for an MLSL model $M = (TS, \Omega, V, \nu)$ with $V = (L, X, E)$, a letter $\sigma \in \mathcal{T}_\mathsf{D} \cup \mathcal{T}_\mathsf{U} \cup \mathcal{N}_\mathsf{D} \cup \mathcal{N}_\mathsf{U}$ and a view $V' = (L', X', E)$ with $L' = [l, l]$, $l \in L$, $X' \subset X$ and $\|X'\| > 0$ that (σ, V) represents a robust letter as

$$\begin{aligned} \mathsf{repr}^\mathsf{r}(M, \sigma, V') \text{ iff } & (TS, \Omega, (L', X', E), \nu) \models \exists c.\, \mathsf{letter}^\mathsf{r}(\sigma, c) && \text{and} \\ & \exists \delta \in \mathbb{R}_{>0}.\, (TS, \Omega, (L', [\underline{X'} - \delta, \underline{X'}], E), \nu) \models \mathsf{free} && \text{and} \\ & \exists \delta \in \mathbb{R}_{>0}.\, (TS, \Omega, (L', [\overline{X'}, \overline{X'} + \delta], E), \nu) \models \mathsf{free} \ . \end{aligned}$$

An *MLSL representation of a letter* inside a model M is given by a letter and a view $(\sigma, V) \in (\mathcal{T}_\mathsf{D} \cup \mathcal{N}_\mathsf{D} \cup \mathcal{T}_\mathsf{U} \cup \mathcal{N}_\mathsf{U}) \times \mathbb{V}$, where $\mathbb{V}$ is the set of all views, such that $\mathsf{repr}^\mathsf{r}(M, \sigma, V)$ is satisfied. When the distinction of terminals and nonterminals is not important we use Σ for some arbitrary finite alphabet.

Now we can define functions that transform nonrobust letters to robust letters and vice versa (cf. Figure 4.4 on Page 129 for an example).

Definition 4.2.2 (Transforming Nonrobust Letters to Robust Letters)**.** For some alphabet Σ let $M = (TS, \Omega, V, \nu)$ with $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$ be an

MLSL model, $S \subset \Sigma \times \mathbb{V}$ a set of representations of letters and $(\sigma, V') \in \Sigma \times \mathbb{V}$ with $V' = ([l', l'], X', E)$ a single representation. We define

$$h_1(M, \emptyset) = M \ ,$$
$$h_1(M, \{(\sigma, V')\} \cup S) = h_1(g(M', I^M_{V'}, V'_2), S \setminus \{(\sigma, V')\})$$

with $M' = ((\mathsf{res}', \mathsf{clm}', \mathsf{pos}', \mathsf{spd}, \mathsf{acc}), \Omega', V, \nu)$ and

$$\mathsf{pos}' = \mathsf{pos} \oplus \{C_1 \mapsto \underline{X'_1}, C_2 \mapsto \underline{X'_1} + \frac{\|X'_1\|}{3}, C_3 \mapsto \underline{X'_3}, C_4 \mapsto \underline{X'_3} + \frac{\|X'_3\|}{3}\} \ ,$$
$$\Omega' = \Omega \oplus \{(C_1, TS'') \mapsto \frac{2\|X'_1\|}{3}, (C_2, TS'') \mapsto \|X'_1\|,$$
$$(C_3, TS'') \mapsto \frac{2\|X'_3\|}{3}, (C_4, TS'') \mapsto \|X'_3\| \mid TS'' \in \mathbb{TS}\} \ ,$$

where $\mathcal{C} = \{C_1, C_2, C_3, C_4\} \subseteq \mathbb{I} \setminus I^M_V$ is a set of cars outside of V, $\mathsf{res}' = \mathsf{res} \oplus \{C \mapsto \{l'\} \mid C \in \mathcal{C}\}$, $\mathsf{clm}' = \mathsf{clm} \oplus \{C \mapsto \emptyset \mid C \in \mathcal{C}\}$ and for $i \in \{1, 2, 3\}$ $V'_i = (L', X'_i, E)$ is a view with a proper interval as extension s.t. $V' = V'_1 \ominus V'_2 \ominus V'_3$. $\triangle$

The function h_1 splits for each representation of a letter the view V' of that representation into three subviews V'_1, V'_2, V'_3 such that $V' = V'_1 \ominus V'_2 \ominus V'_3$ and each subview has a proper interval as extension. Then we place two cars from somewhere outside the model in V'_1 such that they overlap and another two cars in V'_3. These views are the start marker and the end marker of the letter. Next, the cars of the nonrobust representation are placed in V'_2, such that their reservations are a nonzero distance apart. We briefly explain some formulas of the previous definition. In M' we moved the cars C_1, C_2 into the view V'_1 and the cars C_3, C_4 into the view V'_3. We chose the positions and the value of the sensor function in a way that ensures that in V'_1 the first third of the extension is only covered by C_1, the second third by C_1 and C_2 and the last third by only C_2 and similarly for V'_3. All of these cars were outside the view of M before moving them. With $g(M', I^M_{V'}, V'_2)$ we move the cars in $I^M_{V'}$ (see Page 26 for $I^M_{V'}$), i.e. the cars that nonrobustly represented the letter σ in the view V' in the model M, into the view V'_2 with free space in between the reservations. In Figure 4.4 we provide an example.

With the function f from Definition 4.2.1 we can define a function that transforms every robust representation of a letter in a model to its nonrobust representation, without changing the extension or the lanes of the representation.

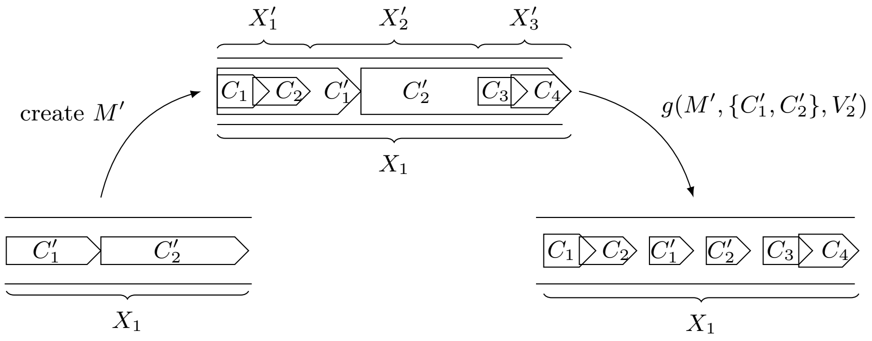


Figure 4.4: Transformation of a nonrobust MLSL representation of a letter σ to a robust MLSL representation of that letter. The representation uses two cars, i.e. $\lambda(\sigma) = 2$. We see the nonrobust representation in the model $M = (TS, \Omega, V, \nu)$ on the left, an intermediate model M' in the middle and the transformed robust representation on the right. We only depict the models cropped to the subview $V_1 = (L_1, X_1, E)$ of V, where the representation is located. When we transform a nonrobust representation to a robust one, we first chop the view V_1 into three subviews $V'_j = (L_1, X'_j, E)$ with $j \in \{1, 2, 3\}$ and fill V'_1 and V'_3 each with two overlapping reservations of fresh cars. Let this model be M'. Then, we spread the cars $I^M_{V_1} = \{C'_1, C'_2\}$ that initially represented σ with $g(M', \{C'_1, C'_2\}, V'_2)$ in the view V'_2.

Definition 4.2.3 (Transforming Robust Letters to Nonrobust Letters)**.** Let $M_1 = (TS_1, \Omega_1, V_1, \nu_1)$ be an MLSL model, $S \subset \Sigma \times \mathbb{V}$ a set of representations of letters, $(\sigma, V) \in \Sigma \times \mathbb{V}$ a single representation, $\mathcal{C} \subseteq \mathbb{I}$ the set of cars with overlapping reservations in V and let $V' = ([l', l'], X', E')$ be a view with a single lane and a proper interval as extension such that V' and V_1 do not overlap. We define

$$h_2(M_1, \emptyset) = M_1 \ ,$$
$$h_2(M_1, \{(\sigma, V)\} \cup S) = h_2(f(f(M_1, \mathcal{C}, V'), I_V^{M_1} \setminus \mathcal{C}, V), S \setminus \{(\sigma, V)\}) \ . \quad \triangle$$

In the definition above for any robust representation of a letter we first move with $f(M_1, \mathcal{C}, V')$ the overlapping cars of the start marker and the end marker to V', where the reservations have no effect. Then we take the resulting model $M' = f(M_1, \mathcal{C}, V')$ and spread the remaining $\lambda(\sigma)$ reservations out in V with $f(M', I_V^{M_1} \setminus \mathcal{C}, V)$, where we refer to Page 35 for $\lambda(\sigma)$. In Figure 4.5 we provide an example.

Now we have the nice property that we can use h_2 to change robust representations of letters to nonrobust representations without changing the view of the representations. Similarly, with h_1 we can transform nonrobust representations of letters to their robust counterparts without changing their views. This follows from the definitions.

We make the first step of showing that $F_{\text{robust}}(G_\mathsf{D}, G_\mathsf{U})$ and $F(G_\mathsf{D}, G_\mathsf{U})$ are lane-unboundedly equisatisfiable. For this, we first consider those subformulas that are quite similar in $F_{\text{robust}}(G_\mathsf{D}, G_\mathsf{U})$ and $F(G_\mathsf{D}, G_\mathsf{U})$. The formulas we consider now only differ in whether they use the robust or nonrobust representations of letters. Additionally, for the nonrobust letters we also consider the formula mutex, as the nonrobust representation of letters does not work otherwise.

Lemma 4.2.4. *For two context-free grammars $G_\mathsf{D}, G_\mathsf{U}$ in* $\text{CNF}^{-\epsilon}$ *let*

$$\phi \equiv \bigwedge_{i \in \{\mathsf{D},\mathsf{U}\}} \mathsf{stepAll}_i \wedge \mathsf{start}_i \wedge \mathsf{letterNextToLetter}_i \wedge \mathsf{freeLane} \wedge \mathsf{allResInLetter} \wedge \mathsf{asm} \ , \tag{4.1}$$

$$\phi^\mathsf{r} \equiv \bigwedge_{i \in \{\mathsf{U},\mathsf{D}\}} \mathsf{stepAll}_i^\mathsf{r} \wedge \mathsf{start}_i^\mathsf{r} \wedge \mathsf{letterNextToLetter}_i^\mathsf{r} \wedge \mathsf{freeLane}^\mathsf{r} \wedge \mathsf{allResInLetter}^\mathsf{r} \wedge \mathsf{asm}^\mathsf{r} \ . \tag{4.2}$$

Then, for all MLSL models M,

(i) if $M \models \phi^\mathsf{r}$ then $h_2(M, S) \models \phi$ and

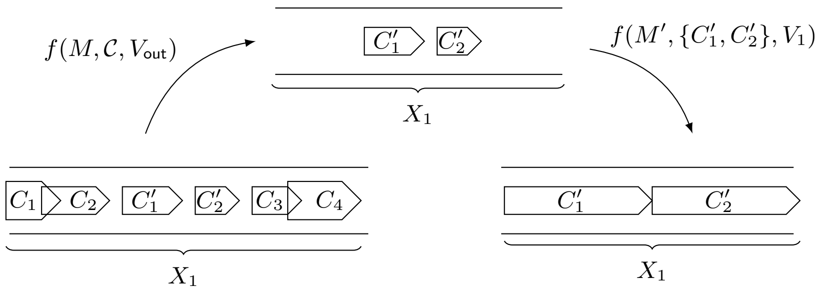


Figure 4.5: Transformation of a robust MLSL representation of a letter σ to a nonrobust MLSL representation of that letter, where $\lambda(\sigma) = 2$. We see the initial representation in the model $M = (TS, \Omega, V, \nu)$ on the left, the intermediate model $M' = f(M, \mathcal{C}, V')$ in the middle and the transformed representation on the right. We only depict the models cropped to the subview $V_1 = (L_1, X_1, E)$ of V, where the representation is located. To transform a robust representation to a nonrobust one, we first remove the cars representing the start marker and the end marker with $f(M, \mathcal{C}, V')$, where $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ and V' is a view outside of V. Then, we fill V_1 with the remaining cars in V_1, given as $I_{V_1}^M \setminus \mathcal{C} = \{C_1', C_2'\}$, with $f(M', \{C_1', C_2'\}, V_1)$.

(ii) if $M \models \phi$ then $h_1(M, S) \models \phi^{\mathsf{r}}$,

where S is the set of all (non)robust MLSL representations of letters in M.

Proof. For both parts of the lemma it is important that the functions h_2 and h_1 do not change the views of any of the MLSL representations of letters in M. For the definitions of the subformulas of ϕ and ϕ^{r} we refer to Pages 37 to 40 and Page 124. Further, for $i \in \{\mathsf{D}, \mathsf{U}\}$ let $G_i = (\mathcal{N}_i, \mathcal{T}, \mathcal{R}_i, S_i)$ and let $\pi_i : \mathcal{T} \to \mathcal{T}_i$ be a bijective terminal renaming function for some set $\mathcal{T}_i$ such that $\mathcal{T}_{\mathsf{D}}, \mathcal{T}_{\mathsf{U}}, \mathcal{N}_{\mathsf{U}}, \mathcal{N}_{\mathsf{D}}$ are disjoint (see also Page 35).

We start with Part (i). Let $M = (TS, \Omega, V, \nu)$ and $M' = h_2(M, S)$. As M satisfies $\mathsf{allResInLetter}^{\mathsf{r}}$ we know that there are no reservations outside of robust representations of letters in M. With the function h_2 we transform all representations in M into nonrobust representations in M' without changing the view of the representation. That is, for all $\sigma \in \mathcal{T}_{\mathsf{D}} \cup \mathcal{T}_{\mathsf{U}} \cup \mathcal{N}_{\mathsf{U}} \cup \mathcal{N}_{\mathsf{D}}$ and all subviews V_1 of V:

$$\text{if } \mathsf{repr}^{\mathsf{r}}(M, \sigma, V_1) \text{ holds, then also } \mathsf{repr}(M', \sigma, V_1) \text{ holds,} \tag{4.3}$$

where repr and $\mathsf{repr}^{\mathsf{r}}$ are defined on Pages 50 and 127. This follows from the definition of h_2. As we did not place any reservations outside representations of nonrobust letters in M', it follows that M' satisfies $\mathsf{allResInLetter}$.

We show that $M' \models \mathsf{stepAll}_{\mathsf{D}}$. To show this we consider M for which we know that it satisfies $\mathsf{stepAll}^{\mathsf{r}}_{\mathsf{D}}$. Hence, any view V_1 where $\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(N, c)$ holds for some $N \in \mathcal{N}_{\mathsf{D}}$ and some variable c mapped to some car C, there is a view V_2 where $\mathsf{step}^{\mathsf{r}}_{\mathsf{D}}(N, c)$ holds. Because we did not change the extensions or lanes of the representations of letters we know that in M' the view V_1 satisfies the nonrobust formula $\mathsf{letter}_{\mathsf{free}}(N, c)$, where c is mapped to some car C' with $C \neq C'$. Note that we have $C \neq C'$ because for $\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(N, c)$ the variable c is mapped to the first car of the start marker, while for $\mathsf{letter}_{\mathsf{free}}(N, c)$ it is mapped to the first of $\lambda(N)$ cars representing N (see Page 35 for λ). Again, as we did not change the views of the representations in V_2 the formula $\mathsf{step}_{\mathsf{D}}(N, c)$ is satisfied. Hence, M' satisfies $\mathsf{stepAll}_{\mathsf{D}}$.

With similar reasoning we conclude that $\mathsf{step}_{\mathsf{U}}$ and for $i \in \{\mathsf{U}, \mathsf{D}\}$ also start_i, $\mathsf{letterNextToLetter}_i$, $\mathsf{freeLane}$, $\mathsf{noClaims}$ and $\mathsf{noTwoRes}$ are satisfied by M'.

For mutex we remind the reader that in M each robust representation (σ, V_1) consists of exactly $\lambda(\sigma) + 4$ cars. The $+4$ is used for the start marker and the end marker. With the function h_2 we move the four cars from the start marker and the end marker somewhere outside the view of M', where they are not considered for evaluating mutex. The remaining $\lambda(\sigma)$ cars are placed without overlappings and without free space in between the reservations in V_1. Thus, M' satisfies the formula mutex.

We have shown that for $M' = h_2(M, S)$ we have $M' \models \phi$.

The argument is similar for Part (ii). It relies on the reverse direction of Equation (4.3). □

With h_1 we can transform nonrobust representations of letters into robust representations without changing their views. However, we require another function to change the size of a nonrobust representation of a letter. We define a function to move and resize the extension of a nonrobust representation of a letter to the extension of another nonrobust representation of a letter. For a nonempty set of tuples of representations of letters $T \subseteq (\Sigma \times \mathbb{V}) \times (\Sigma \times \mathbb{V})$, a tuple of representations of letters $((\sigma_1, V_1), (\sigma_2, V_2)) \in T$ with $V_1 = (L_1, X_1, E)$ and $V_2 = (L_2, X_2, E)$ and an MLSL model M we define $\mathsf{mv} : \mathbb{M} \times \mathcal{P}((\Sigma \times \mathbb{V}) \times (\Sigma \times \mathbb{V})) \to \mathbb{M}$ inductively as

$$\begin{aligned}
&\mathsf{mv}(M, \emptyset) = M \\
&\mathsf{mv}(M, T) = \mathsf{mv}(f(M, I^M_{V_1}, (L_1, X_2, E)), T \setminus \{((\sigma_1, V_1), (\sigma_2, V_2))\}) \ .
\end{aligned}$$

This means that we move the extension of the representation (σ_1, V_1) to the extension of the representation (σ_2, V_2). The lanes remain unchanged. More specifically, with $f(M, I^M_{V_1}, (L_1, X_2, E))$ we move the cars in the set $I^M_{V_1}$ representing σ_1 in the model M to the extension X_2. We ignore σ_2 and what might have been in the extension X_2 before. Afterwards, we continue recursively.

Using the previous definitions we prove the first direction of the equisatisfiability of our two big undecidable formulas.

Lemma 4.2.5. *Let $G_\mathsf{D}, G_\mathsf{U}$ be two context-free grammars in* $\mathrm{CNF}^{-\epsilon}$. *If the formula $F(G_\mathsf{D}, G_\mathsf{U})$ is satisfiable, then $F_{\text{robust}}(G_\mathsf{D}, G_\mathsf{U})$ also is lane-unboundedly satisfiable.*

Proof. In $F(G_\mathsf{D}, G_\mathsf{U})$ we encode two derivations of the same word from two different context-free grammars. The formula ensures that if two MLSL representations of terminals represent the same occurrence of that terminal in the derived word, then the extensions of these representations are equal. However, in $F_{\text{robust}}(G_\mathsf{D}, G_\mathsf{U})$ such representations of terminals do not have an equal extension. Instead, one extension is strictly contained in the other. Thus, when we transform a model satisfying $F(G_\mathsf{D}, G_\mathsf{U})$ into one satisfying $F_{\text{robust}}(G_\mathsf{D}, G_\mathsf{U})$

we first make in each pair of corresponding terminals one smaller, such that it is strictly contained in the other. Then, we transform every nonrobust letter to a robust letter.

For $i \in \{\mathsf{D}, \mathsf{U}\}$ let $G_i = (\mathcal{N}_i, \mathcal{T}, \mathcal{R}_i, S_i)$. Further, we remind that $\pi_i : \mathcal{T} \to \mathcal{T}_i$ is a bijective terminal renaming function for some set $\mathcal{T}_i$ such that $\mathcal{T}_\mathsf{D}, \mathcal{T}_\mathsf{U}, \mathcal{N}_\mathsf{U}, \mathcal{N}_\mathsf{D}$ are disjoint (see also Page 35). Assume that M satisfies $F(G_\mathsf{D}, G_\mathsf{U})$ and let S be the set of all nonrobust MLSL representations of letters in M. For the set $S_1 = \{(\tau_1, V_1), \ldots, (\tau_n, V_n)\}$ of all representations of terminals from $\mathcal{T}_\mathsf{D}$ in M let $S_1' = \{(\tau_1, V_1'), \ldots, (\tau_n, V_n')\}$ be the set of representations where all extensions are shrunk by $\delta \in \mathbb{R}$ with $0 < \delta < 1$. Further, let $T \subseteq ((\mathcal{T}_\mathsf{U} \cup \mathcal{T}_\mathsf{D}) \times \mathbb{V}) \times ((\mathcal{T}_\mathsf{U} \cup \mathcal{T}_\mathsf{D}) \times \mathbb{V})$ be the set of tuples of representations such that the first element in each tuple is the original representation and the second element is its shrunk version. That is, let

$$S_1' = \{(\tau_j, (L_j, [\underline{X_j} + \frac{\delta \|X_j\|}{2}, \overline{X_j} - \frac{\delta \|X_j\|}{2}], E)) \mid (\tau_j, (L_j, X_j, E)) \in S_1\} \ ,$$
$$T = \{((\tau_1, V_1), (\tau_1, V_1')), \ldots, ((\tau_n, V_n), (\tau_n, V_n'))\} \ .$$

Further, let $S' = (S \setminus S_1) \cup S_1'$ be the set of representations of letters where the terminals from $\mathcal{T}_\mathsf{D}$ are replaced by their downsized versions and

$$M_1 = \mathsf{mv}(M, T) \ ,$$
$$M_2 = h_1(M_1, S') \ .$$

In M_1 all representations of letters from $\mathcal{T}_\mathsf{D}$ have been made horizontally smaller. Then, in M_2 we transform the smaller representations and all unchanged representations into robust representations of letters without further changing the views.

We show $M_2 \models F_{\text{robust}}(G_\mathsf{D}, G_\mathsf{U})$. Let ϕ and ϕ^r be the formulas of the same name in Lemma 4.2.4. We point out that $M_1 \models \phi$ because the representations of terminals have no lower bound on the size of their extensions, except of being proper intervals. Furthermore, with Lemma 4.2.4 we can conclude that $M_2 \models \phi^\mathsf{r}$.

We show that $M_2 \models \mathsf{subseq}_i^\mathsf{r}$, with $i \in \{\mathsf{D}, \mathsf{U}\}$. First, we consider the case $i = \mathsf{D}$. Assume that the premise of $\mathsf{subseq}_\mathsf{D}^\mathsf{r}$ is satisfied, i.e. for some terminal $\tau \in \mathcal{T}$ and some view the formula $\mathsf{letter}_\mathsf{free}^\mathsf{r}(\pi_\mathsf{D}(\tau), c)$ is satisfied for some variable c. This means that this view has a subview $V_\mathsf{D}' = ([l_\mathsf{D}, l_\mathsf{D}], X_\mathsf{D}', E)$ such that $(\pi_\mathsf{D}(\tau), V_\mathsf{D}') \in S_1'$. Let $V_\mathsf{D} = ([l_\mathsf{D}, l_\mathsf{D}], X_\mathsf{D}, E)$ be the original view of this representation in S. Then, $(\pi_{\mathsf{D}}(\tau), V_{\mathsf{D}}) \in S$ and $X'_{\mathsf{D}} \subset X_{\mathsf{D}}$. Further, both $\mathsf{repr}^{\mathsf{r}}(M_2, \pi_{\mathsf{D}}(\tau), V'_{\mathsf{D}})$ and $\mathsf{repr}(M, \pi_{\mathsf{D}}(\tau), V_{\mathsf{D}})$ hold (see Page 127 for $\mathsf{repr}^{\mathsf{r}}$). As $M \models \mathsf{subseq}_{\mathsf{D}}$, there is a representation of $\pi_{\mathsf{U}}(\tau)$ on a lower lane that is horizontally aligned with the representation $(\pi_{\mathsf{D}}(\tau), V_{\mathsf{D}})$, i.e. there is a view $V_{\mathsf{U}} = ([l_{\mathsf{U}}, l_{\mathsf{U}}], X_{\mathsf{U}}, E)$ with $l_{\mathsf{U}} < l_{\mathsf{D}}$ and $X_{\mathsf{U}} = X_{\mathsf{D}}$ such that $\mathsf{repr}(M, \pi_{\mathsf{U}}(\tau), V_{\mathsf{U}})$ holds. Because, when changing the nonrobust representations in M into robust representations in M_2 we did not change the extension of any letter from $\mathcal{T}_{\mathsf{D}}$, for this view also $\mathsf{repr}^{\mathsf{r}}(M_2, \pi_{\mathsf{U}}(\tau), V_{\mathsf{U}})$ holds. We can slightly increase the extensions of V_{U} and V_{D} such that for $V = ([l_{\mathsf{U}}, l_{\mathsf{D}}], X, E)$ with $X_{\mathsf{D}} \subset X$ the conclusion of $\mathsf{subseq}^{\mathsf{r}}_{\mathsf{D}}$ is satisfied, i.e. in the view V the formula

$$\begin{pmatrix} \mathsf{free} \\ \mathsf{true} \\ \mathsf{free} \end{pmatrix} \frown \left(\begin{pmatrix} \mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(\tau_{\mathsf{D}}, c) \\ \mathsf{true} \\ \mathsf{letter}^{\mathsf{r}}(\tau_{\mathsf{U}}, c') \end{pmatrix} \right) \frown \begin{pmatrix} \mathsf{free} \\ \mathsf{true} \\ \mathsf{free} \end{pmatrix}$$

is satisfied for some variable c'. Hence, $M_2 \models \mathsf{subseq}^{\mathsf{r}}_{\mathsf{D}}$. The reasoning is symmetric for $i = \mathsf{U}$.

Note that we evaluate the outermost true-subformulas of $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ on point intervals, i.e. we evaluate the subformula in between the outermost horizontal chops on the full extension. □

We proceed with the other direction.

Lemma 4.2.6. *For two context-free grammars $G_{\mathsf{D}}, G_{\mathsf{U}}$ in* CNF$^{-\epsilon}$*, if the MLSL formula $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is lane-unboundedly satisfiable, then $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ also is lane-unboundedly satisfiable.*

Proof. Let ϕ and ϕ^{r} be the formulas of the same name in Lemma 4.2.4 and assume $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is satisfied by a model $M_1 = (TS_1, \Omega_1, V_1, \nu_1)$ with $V_1 = (L_1, X_1, E)$. Then there is a subview $V'_1 = (L_1, X', E)$ of V_1 where ϕ^{r} is satisfied, i.e. for $M'_1 = (TS_1, \Omega_1, V'_1, \nu_1)$ we have $M'_1 \models \phi^{\mathsf{r}}$.

We construct M'_2 from M'_1 such that $M'_2 \models F(G_{\mathsf{D}}, G_{\mathsf{U}})$. For this we first transform every robust MLSL representation of a letter to a nonrobust representation. Afterwards, we resize the nonrobust representations of terminals such that every terminal of one MLSL representation of a derivation has a counterpart with an equal extension in the other MLSL representation of a derivation.

For $i \in \{\mathsf{D}, \mathsf{U}\}$ let our grammars be $G_i = (\mathcal{N}_i, \mathcal{T}, \mathcal{R}_i, S_i)$ and let $\pi_i : \mathcal{T} \to \mathcal{T}_i$ be our bijective renaming function such that $\mathcal{T}_{\mathsf{D}}, \mathcal{T}_{\mathsf{U}}, \mathcal{N}_{\mathsf{U}}, \mathcal{N}_{\mathsf{D}}$ are disjoint (see also Page 35). Let $S_1 \subseteq (\mathcal{T}_{\mathsf{D}} \cup \mathcal{T}_{\mathsf{U}} \cup \mathcal{N}_{\mathsf{U}} \cup \mathcal{N}_{\mathsf{D}}) \times \mathbb{V}$ be the set of all robust

MLSL representations of letters in M'_1. Then we create $M_2 = h_2(M'_1, S_1)$. From Lemma 4.2.4 we can conclude that $M_2 \models \phi$.

Let $T \subseteq S_1 \times S_1$ be the set of all representations of terminals in M'_1 such that one terminal is from $\mathcal{T}_{\mathsf{D}}$ and the other from $\mathcal{T}_{\mathsf{U}}$, the terminals are equal modulo the inverses π_{D}^{-1} and π_{U}^{-1}, the second extension of every tuple is a strict subset of the first extension in the tuple and the representation of the terminal of $\mathcal{T}_{\mathsf{U}}$ is on a lower lane than the representation from $\mathcal{T}_{\mathsf{D}}$. If we do this for the model from Figure 4.3 we get the set $\{((a_{\mathsf{U}}, \widetilde{V}_1), (a_{\mathsf{D}}, \widetilde{V}_2)), ((a_{\mathsf{U}}, \widetilde{V}_3), (a_{\mathsf{D}}, \widetilde{V}_4)), ((b_{\mathsf{D}}, \widetilde{V}_5), (b_{\mathsf{U}}, \widetilde{V}_6))\}$ for suitable views $\widetilde{V}_j$ with $j \in \{1, \dots, 6\}$. Because M'_1 satisfies ϕ^{r} we know that despite the restrictions mentioned above, T contains every robust representation of a terminal in M'_1.

Now, let $M'_2 = \mathsf{mv}(M_2, T)$, i.e. we resize the extension of every larger representation in S_1 to the extension of its smaller partner. For M'_2 we still know $M'_2 \models \phi$ because the views of the new representations are subviews of the old representations and we only changed the representations of terminals. For those, there is no lower bound on the size of its extension, except that the extension must be a proper interval.

We define the set of tuples T' to be the set we get by first replacing in every tuple $((\tau_1, \widetilde{V}_1), (\tau_2, \widetilde{V}_2)) \in T$ the extension of $\widetilde{V}_1$ with the extension of $\widetilde{V}_2$ and then sorting each tuple such that the first terminal is from $\mathcal{T}_{\mathsf{D}}$ and the second terminal is from $\mathcal{T}_{\mathsf{U}}$.

We show $M'_2 \models \mathsf{subseq}_i$ with $i \in \{\mathsf{D}, \mathsf{U}\}$. First we consider the case $i = \mathsf{D}$. Assume we have a view and a terminal $\tau \in \mathcal{T}$ satisfying the premise $\mathsf{letter}_{\mathsf{free}}(c, \tau_{\mathsf{D}})$ of $\mathsf{subseq}_{\mathsf{D}}$, where $\tau_{\mathsf{D}} = \pi_{\mathsf{D}}(\tau)$. Then there is a view $V_{\mathsf{D}} = ([l_{\mathsf{D}}, l_{\mathsf{D}}], X_{\mathsf{D}}, E)$ such that $\mathsf{repr}(M'_2, \tau_{\mathsf{D}}, V_{\mathsf{D}})$ holds. As T' contains every representation of a terminal in M'_2 there is $((\tau_{\mathsf{D}}, V_{\mathsf{D}}), (\tau_{\mathsf{U}}, V_{\mathsf{U}})) \in T'$ with $V_{\mathsf{U}} = ([l_{\mathsf{U}}, l_{\mathsf{U}}], X_{\mathsf{U}}, E)$. Furthermore, we know $X_{\mathsf{D}} = X_{\mathsf{U}}$, $l_{\mathsf{D}} > l_{\mathsf{U}}$ and $\pi_{\mathsf{D}}^{-1}(\tau_{\mathsf{D}}) = \pi_{\mathsf{U}}^{-1}(\tau_{\mathsf{U}})$. Hence, the view $\widetilde{V}' = ([l_{\mathsf{U}}, l_{\mathsf{D}}], X_{\mathsf{D}}, E)$, which subsumes V_{U} and V_{D} and every lane in between, satisfies

$$\begin{pmatrix} \mathsf{letter}(\tau_{\mathsf{U}}, c_2) \\ \mathsf{true} \\ \mathsf{letter}(\tau_{\mathsf{D}}, c_1) \end{pmatrix},$$

for some variables c_1, c_2. As we know that $\mathsf{repr}(M'_2, \tau_i, V_i)$ with $i \in \{\mathsf{D}, \mathsf{U}\}$ holds, we know that we can extend the extension around the representations slightly such that the letters are surrounded by free space. That means with a view that has the same lanes as $\widetilde{V}'$ and an extension that is in both directions slightly larger than the extension of $\widetilde{V}'$ the formula

$$\begin{pmatrix}\mathsf{free}\\ \mathsf{true}\\ \mathsf{free}\end{pmatrix} \frown \left(\begin{pmatrix}\mathsf{letter}(\tau_{\mathsf{U}}, c_2)\\ \mathsf{true}\\ \mathsf{letter}(\tau_{\mathsf{D}}, c_1)\end{pmatrix}\right) \frown \begin{pmatrix}\mathsf{free}\\ \mathsf{true}\\ \mathsf{free}\end{pmatrix}$$

is satisfied. We conclude $M_2' \models \mathsf{subseq}_{\mathsf{D}}$. For $i = \mathsf{U}$ the reasoning is symmetric.
As $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ is equivalent to $\phi \wedge \bigwedge_{i\in\{\mathsf{D},\mathsf{U}\}} \mathsf{subseq}_i$, this finishes the proof. □

Let $\mathcal{M}$ be a set and let d be a metric on $\mathcal{M}$. Then $(\mathcal{M}, d)$ is a *metric space.*

Definition 4.2.7 (Open Set)**.** Given a metric space $(\mathcal{M}, d)$ and a set $S \subseteq \mathcal{M}$, then S is *open* if and only if

$$\forall x \in S.\, \exists \delta \in \mathbb{R}_{>0}.\, \forall y \in \mathcal{M}.\, (d(x, y) \leq \delta \text{ implies } y \in S) \ . \qquad \triangle$$

This means that a set is open iff for every element in the set, the set also contains the elements neighbourhood, where the metric is used to determine the elements in the neighbourhood.

For an MLSL formula ϕ let $\mathsf{sat}(\phi)$ be the set of MLSL models satisfying ϕ. We prove that the set of models satisfying $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is open, i.e. that we can perturb any satisfying model such that satisfaction of the formula is retained. We point out that for this to hold the outermost chop operators in $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ are needed because otherwise a reservation just outside the view could be perturbed to protrude into the view and then the formula would not be satisfied as it requires that all reservations are inside representations of letters. With the outermost chop operators the view where the representations of derivations are located is flexible.

Lemma 4.2.8. *For the metric d_{m} from Definition 4.1.1 and two context-free grammars $G_{\mathsf{D}}, G_{\mathsf{U}}$ in* CNF$^{-\epsilon}$ *the set* $\mathsf{sat}(F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}}))$ *is open.*

Proof. If $\mathsf{sat}(F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}}))$ is empty, i.e. if $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is not satisfiable, then the set trivially is open. Hence, we assume that $\mathsf{sat}(F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}}))$ is not empty and let $M_1 \in \mathsf{sat}(F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}}))$ with $M_1 = (TS_1, \Omega_1, V_1, \nu_1)$ and $V_1 = (L_1, X_1, E_1)$. Now we have to choose $\delta \in \mathbb{R}_{>0}$ such that for any M_2 that has a distance to M_1 of at most δ, the model M_2 is in the set $\mathsf{sat}(F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}}))$.

For $i \in \{\mathsf{D}, \mathsf{U}\}$ let our grammars be $G_i = (\mathcal{N}_i, \mathcal{T}, \mathcal{R}_i, S_i)$ and let $\pi_i : \mathcal{T} \to \mathcal{T}_i$ our renaming function for some set $\mathcal{T}_i$ such that $\mathcal{T}_{\mathsf{D}}, \mathcal{T}_{\mathsf{U}}, \mathcal{N}_{\mathsf{U}}, \mathcal{N}_{\mathsf{D}}$ are all mutually disjoint (see also Page 35). Let $M_1' = (TS_1, \Omega_1, V_1', \nu_1)$ with $V_1' =$

(L_1, X_1', E) be the model for which the subformula between the outermost chops of $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is satisfied, i.e. where F_1 from Page 125 holds. For $i \in \{\mathsf{D}, \mathsf{U}\}$ let $S_i \subseteq (\mathcal{T}_i \cup \mathcal{N}_i) \times \mathbb{V}$ be the set of all robust MLSL representations of letters in M_1' and let $S = S_{\mathsf{D}} \cup S_{\mathsf{U}}$. Further, let $S_i' \subseteq S_i$ be the set of all representations of terminals from $\mathcal{T}_i$ in M_1'. Now we can measure the distances between representations of letters and between reservations in M_1'. For this we define a function to measure the distance between the endpoints of two nonempty intervals $[r_j, r_j']$ with $r_j, r_j' \in \mathbb{R}$, $r_j \leq r_j'$ and $j \in \{1, 2\}$ as

$$dis([r_1, r_1'], [r_2, r_2']) = \mathsf{min}(|r_1 - r_2|, |r_1' - r_2|, |r_1 - r_2'|, |r_1' - r_2'|) \ .$$

For the rest of the proof, for the formulas $\mathsf{stepAll}_i^{\mathsf{r}}$, $\mathsf{subseq}_i^{\mathsf{r}}$, $\mathsf{letterNextToLetter}_i^{\mathsf{r}}$, $\mathsf{start}_i^{\mathsf{r}}$ and $\mathsf{allResInLetter}^{\mathsf{r}}$ with $i \in \{\mathsf{D}, \mathsf{U}\}$ we refer to Page 124. In the following we use Δ_1 to measure the distance of a representation related with $\mathsf{stepAll}_i^{\mathsf{r}}$ or $\mathsf{letterNextToLetter}_i^{\mathsf{r}}$, Δ_2 for the distance of representations related through $\mathsf{subseq}_i^{\mathsf{r}}$, Δ_3 for distances within a representation and Δ_4 for the distance from the part of the model where we encode our derivations to the outer part that we do not restrict. We define

$$\begin{aligned}
\Delta_1 &= \min_{i \in \{\mathsf{D}, \mathsf{U}\}} \Big(\min_{(\sigma_1, (L_1, X_1, E)), (\sigma_2, (L_2, X_2, E)) \in S_i} (dis(X_1, X_2)) \Big) \ , \\
\Delta_2 &= \min_{\substack{(\tau_{\mathsf{D}}, (L_{\mathsf{D}}, X_{\mathsf{D}}, E)) \neq (\tau_{\mathsf{U}}, (L_{\mathsf{U}}, X_{\mathsf{U}}, E)) \\ (\tau_{\mathsf{D}}, (L_{\mathsf{D}}, X_{\mathsf{D}}, E)) \in S_{\mathsf{D}}, (\tau_{\mathsf{U}}, (L_{\mathsf{U}}, X_{\mathsf{U}}, E)) \in S_{\mathsf{U}}}} (dis(X_{\mathsf{D}}, X_{\mathsf{U}})) \ , \\
\Delta_3 &= \min_{(\sigma, V) \in S} \Big(\min_{\substack{C \neq C' \\ C, C' \in I_V^{M_1'}}} (dis(\mathsf{se}(C, TS_1, \Omega_1), \mathsf{se}(C', TS_1, \Omega_1))) \Big) \ , \\
\Delta_4 &= \min_{(\sigma_1, (L_1, X_1, E)) \in S} (dis(X_1, X_1')) \ .
\end{aligned}$$

Each of these values is greater 0. For Δ_1 this holds because M_1' satisfies $\mathsf{stepAll}_i^{\mathsf{r}}$ and $\mathsf{letterNextToLetter}_i^{\mathsf{r}}$. For Δ_2 this follows from $\mathsf{subseq}_i^{\mathsf{r}}$. For Δ_3 from the formula $\mathsf{allResInLetter}^{\mathsf{r}}$ and for Δ_4 from $\mathsf{start}_i^{\mathsf{r}}$ and $\mathsf{letterNextToLetter}_i^{\mathsf{r}}$. We choose

$$\delta = \frac{\mathsf{min}(\Delta_1, \Delta_2, \Delta_3, \Delta_4)}{3}$$

and show that for an arbitrary MLSL model M_2 with $d_{\mathsf{m}}(M_1, M_2) \leq \delta$ we have $M_2 \in \mathsf{sat}(F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}}))$. The reasoning for our choice of δ is that we want to be able to perturb all representations of letters by δ and still have a distance of δ in between them. Let $M_2 = (TS_2, \Omega_2, V_2, \nu_2)$ with

$TS_2 = (\mathsf{res}_2, \mathsf{clm}_2, \mathsf{pos}_2, \mathsf{spd}_2, \mathsf{acc}_2)$ and $V_2 = (L_2, X_2, E_2)$ be arbitrary such that $d_{\mathsf{m}}(M_1, M_2) \leq \delta$. Then we choose $X_2' = [\underline{X_1'} + \delta, \overline{X_1'} - \delta]$ and argue that with the subview $V_2' = (L_2, X_2', E_2)$ of V_2 the model $M_2' = (TS_2, \Omega_2, V_2', \nu_2)$ satisfies F_1. The reasoning for our choice of X_2' is that any car that had a reservation in V_1' still has its reservation in V_2' after the perturbation and with space to the borders of X_2'. Similarly, any car with a reservation not in V_1' still has its reservation not in V_2' after perturbing the model.

Consider an arbitrary robust MLSL representation $(\sigma, V) \in S$ in M_1' with $V = (L, X, E)$ such that $\mathsf{repr}^{\mathsf{r}}(M_1', \sigma, V)$ from Page 127 holds. Then the view $V' = (L, [\underline{X} - 2\delta, \overline{X} + 2\delta], E)$ satisfies $\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(\sigma, c)$ in $M' = (TS_2, \Omega_2, V', \nu_2')$ for some c mapped to some car C. Note that this is the same car in M_1' and M'. This holds because in M_1' we need to perturb reservations of cars by at least Δ_3 to change the topological structure of reservations. Furthermore, the closest other representation of a letter is at least Δ_1 away.

We show that M_2' satisfies $\mathsf{allResInLetter}^{\mathsf{r}}$. Consider for M_2' some arbitrary view $\widetilde{V}_2 = (\widetilde{L}_2, \widetilde{X}_2, E)$ that is filled by the reservation of some car C, i.e. which satisfies $\mathsf{re}(c)$ and c is mapped to C. Now, consider in M_1' some view $\widetilde{V}_1 = (\widetilde{L}_1, \widetilde{X}_1, E)$ that is filled by the reservation of that same C. Note that $\widetilde{V}_1$ and $\widetilde{V}_2$ both contain the same single lane. The extensions of these views may or may not be overlapping, but the safety envelope of C in M_1' and M_2' differs by at most δ. As M_1' satisfies $\mathsf{allResInLetter}^{\mathsf{r}}$, there is some view $\widetilde{V}_1' = (\widetilde{L}_1, \widetilde{X}_1', E)$ with $\widetilde{X}_1 \subset \widetilde{X}_1'$ and a letter $\sigma \in \mathcal{T}_{\mathsf{D}} \cup \mathcal{T}_{\mathsf{U}} \cup \mathcal{N}_{\mathsf{D}} \cup \mathcal{N}_{\mathsf{U}}$ such that $\mathsf{repr}^{\mathsf{r}}(M_1', \sigma, \widetilde{V}_1')$ holds. This means that we can extend the extension of $\widetilde{V}_1'$ in both directions such that M_1' restricted to this extended view satisfies the formula $\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(\sigma, c')$ for some c'. As argued in the previous paragraph, in the view $\widetilde{V}_2''$, where the extension of $\widetilde{V}_1'$ is extended by 2δ in both directions, the formula $\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}(\sigma, c')$ is satisfied by M_2' restricted to $\widetilde{V}_2''$. Note that $\widetilde{V}_2$ is a subview of $\widetilde{V}_2''$. Thus, M_2' satisfies $\mathsf{allResInLetter}^{\mathsf{r}}$.

With a similar argumentation we conclude that the other formulas in F_1 are satisfied. Finally, as M_2' satisfies F_1 it follows that M_2 satisfies $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$. □

Now we can reduce the intersection problem for context-free grammars to the robust lane-unbounded satisfiability problem of MLSL.

Lemma 4.2.9. *For two context-free grammars $G_{\mathsf{D}}, G_{\mathsf{U}}$ in* CNF$^{-\epsilon}$ *the MLSL formula $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is robustly lane-unboundedly satisfiable iff $\mathcal{L}(G_{\mathsf{D}}) \cap \mathcal{L}(G_{\mathsf{U}}) \neq \emptyset$.*

Proof. We start with the "only if"-direction. Assume $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is robustly lane-unboundedly satisfiable, which implies that the formula also is (nonrobustly) lane-unboundedly satisfiable. From Lemma 4.2.6 we know that $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ is lane-unboundedly satisfiable. Then we can conclude from Lemma 3.1.7 that $\mathcal{L}(G_{\mathsf{D}}) \cap \mathcal{L}(G_{\mathsf{U}}) \neq \emptyset$.

We continue with the "if"-direction. Assume $\mathcal{L}(G_{\mathsf{D}}) \cap \mathcal{L}(G_{\mathsf{U}}) \neq \emptyset$. Then Lemma 3.1.6 implies that $F(G_{\mathsf{D}}, G_{\mathsf{U}})$ is lane-unboundedly satisfiable. Further, Lemma 4.2.5 implies that $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is lane-unboundedly satisfiable. Let M be such a satisfying model. As by Lemma 4.2.8 the set $\mathsf{sat}(F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}}))$ is open for the metric d_{m}, we conclude there is $\delta \in \mathbb{R}_{>0}$ such that

$$\forall M'.\, (d_{\mathsf{m}}(M, M') \leq \delta \text{ implies } M' \in \mathsf{sat}(F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})))\ .$$

This implies $M \models^{\delta} F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$, which in turn implies that $F_{\text{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$ is robustly lane-unboundedly satisfiable. □

We get the following theorem.

Theorem 4.2.10. *The robust lane-unbounded satisfiability problem of MLSL is undecidable.*

As we do not restrict the size of reservations in any way, the robust lane-unbounded satisfiability problem of MLSL remains undecidable, even if we impose an upper or a lower bound on the size of reservations or the view.

4.3 Decidability of Robust Model Problem for MLSLS

In this section we extend the approach to check the precise model problem from the previous chapter to solve the robust model problem for MLSLS. So far, for a FOLRA data tuple p that contains the FOLRA variables, an MLSLS model M and an MLSLS formula ϕ we introduced constraints to ensure that p represents M. Then we check if the FOLRA representation of M stored in p satisfies the FOLRA representation of ϕ by checking if the FOLRA formula $\mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p, \phi)$ is satisfied. Here, we introduce a second data tuple p', let p represent M and require that p' is similar to p. Then we check if $\mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p', \phi)$ is satisfied.

We extend our definition of a metric on MLSL models to a metric on models with a scope. The difference to the metric on MLSL models is that here we allow models without a view, and that the domain of the supremum is the car domain of the models.

Definition 4.3.1 (Metric on MLSLS Models)**.** Given two simple MLSLS models M_i with $M_i = (CS_i, TS_i, \Omega_i, \nu_i)$, $TS_i = (\mathsf{res}_i, \mathsf{clm}_i, \mathsf{pos}_i, \mathsf{spd}_i, \mathsf{acc}_i)$ and $i \in \{1, 2\}$ we define

$$d'_{\mathsf{m}}(M_1, M_2) = \begin{cases} \infty & \text{if} \quad (\mathsf{res}_1, \mathsf{clm}_1, \nu_1, CS_1) \\ & \qquad \neq (\mathsf{res}_2, \mathsf{clm}_2, \nu_2, CS_2) \\ \mathsf{max}\big(\mathsf{sup}_{C \in \mathsf{cars}\, M_1}(|\mathsf{pos}_1(C) - \mathsf{pos}_2(C)|, & \\ \qquad |\overline{\mathsf{se}(C, TS_1, \Omega_1)} - \overline{\mathsf{se}(C, TS_2, \Omega_2)}|)\big) & \text{otherwise} \ . \end{cases}$$

We extend this to proper models. For $i \in \{1, 2\}$ let $M_i = (CS_i, TS_i, \Omega_i, V_i, \nu_i)$ with TS_i as above, $V_i = (L_i, [r_i, r'_i], E_i)$ and let M'_i be the simple version of M_i (cf. Definition 3.2.7 on Page 55 for simple versions of models). We define

$$d_{\mathsf{m}}(M_1, M_2) = \begin{cases} \infty & \text{if} \quad (L_1, E_1) \\ & \qquad \neq (L_2, E_2) \\ \mathsf{max}(d'_{\mathsf{m}}(M'_1, M'_2), |r_1 - r_2|, |r'_1 - r'_2|) & \text{otherwise} \ . \end{cases}$$

Further, for a simple and a proper MLSLS model the distance is ∞. △

For the definition above we point out that if $\mathsf{res}_1 = \mathsf{res}_2$, then the car domain of M_1 and M_2 is equal.

We relate our operations on MLSLS models with our metric on MLSLS models.

Lemma 4.3.2. *Given two possibly simple MLSLS models M_1 and M_2 and a set of car identifiers $CS \subseteq \mathbb{I}$ we have*

$$d_{\mathsf{m}}(M_1, M_2) = \mathsf{max}(d_{\mathsf{m}}(M_1 \boxed{\upharpoonleft} CS, M_2 \boxed{\upharpoonleft} CS), d_{\mathsf{m}}(M_1 \boxminus CS, M_2 \boxminus CS)) \ .$$

Proof. The proof follows from the definition. First we need a case distinction on whether $d_{\mathsf{m}}(M_1, M_2)$ is infinite. If it is infinite it is clear that the right hand side also is infinite.

We continue with the case that the distance is finite. Starting from the right hand side, after unfolding the definitions we have two suprema with the domains

$\mathsf{cars}\, M_1 \cap CS$ and $\mathsf{cars}\, M_1 \setminus CS$. Note that the car domains of M_1 and M_2 are equal because the distance is finite. Taking the largest of these two suprema then is equivalent to taking the supremum of the combined domain $\mathsf{cars}\, M_1$, which is equivalent to $d_{\mathsf{m}}(M_1, M_2)$. $\square$

We define a formula to represent the shaking of data in a MLSLS model. The formula takes two possibly simple FOLRA data tuples, the desired perturbation and a relation between the car identifier variables of the two data tuples. We express this relation by requiring that there is an MLSLS model that is sane with both data tuples and car identifier variable assignments. We start with models and data tuples representing a single car. Similar to the case with precise information, we use E to denote an Elementary (or base) version of a function that we extend later.

Definition 4.3.3 (Shaking Data Variables)**.** For an MLSLS model M with $|M| = 1$, two proper FOLRA data tuples $p_i = (DS_i, l_i, l'_i, \beta_i, \beta'_i, f_i, \mathbb{D}_i, \mathcal{S}_i, D_{Ei})$ with $i \in \{1, 2\}$, a car identifier variable assignment g_i such that (g_i, p_i, M) is sane and a perturbation $\delta \in \mathbb{R}_{\geq 0}$ we define

$$
\begin{aligned}
&\mathsf{sim}_{\mathsf{m},\mathsf{E}}((p_1, g_1), (p_2, g_2), \delta) \equiv \\
&\qquad \exists x, x'.\, x, x' \in [-\delta, \delta] \land \beta_1 + x = \beta_2 \land \beta'_1 + x' = \beta'_2 \\
&\land \quad \exists v'_{\mathsf{p}}, v'_{\Omega}.\, v'_{\mathsf{p}} \in [-\delta, \delta] \land v'_{\mathsf{p}} + v'_{\Omega} \in [-\delta, \delta] \\
&\land \bigwedge_{D_1 \in \mathbb{D}_1, D_2 \in \mathbb{D}_2} \left(\begin{aligned} & \{D_1.v_{\mathsf{r1}}, D_1.v_{\mathsf{r2}}\} = \{D_2.v_{\mathsf{r1}}, D_2.v_{\mathsf{r2}}\} \\ \land\ & D_1.v_{\mathsf{p}} + v'_{\mathsf{p}} = D_2.v_{\mathsf{p}} \land D_1.v_{\mathsf{c}} = D_2.v_{\mathsf{c}} \\ \land\ & D_1.v_{\mathsf{p}} + D_1.v_{\Omega} + v'_{\mathsf{p}} + v'_{\Omega} = D_2.v_{\mathsf{p}} + D_2.v_{\Omega} \end{aligned} \right)
\end{aligned}
$$

where $D_i.v$ is an abbreviation for $\mathcal{S}_i(D_i)(v)$. For a simple MLSLS model and simple data tuples the constraints in the first line are left out. $\triangle$

The idea behind using $v'_{\mathsf{p}}, v'_{\Omega}$ is that if there are multiple D representing the same car we have to ensure that each car is perturbed by the same value. Otherwise, the resulting assignment would not represent an MLSLS model. Further, note that we use M only to ensure that the FOLRA tuples are sane with the same MLSLS model. This ensures that parts we do not constrain with $\mathsf{sim}_{\mathsf{m},\mathsf{E}}$ agree, such as the arithmetic representation of the scope DS_i or the lanes of the view and so on.

We extend the previous definition to multiple cars. This definition looks similar to previous definitions in that we define it recursively over car identifiers.

Definition 4.3.4. For a finite possibly simple MLSLS model M, $i \in \{1,2\}$, a possibly simple FOLRA data tuple p_i, a car identifier variable assignments g_i such that (p_i, g_i, M) is sane and a perturbation $\delta \in \mathbb{R}_{\geq 0}$ we define

$$\mathsf{sim_m}((p_1, g_1), (p_2, g_2), \delta) \equiv$$
$$\begin{cases} \mathsf{sim_{m,E}}((p_1, g_1) \obar \{C\}, (p_2, g_2) \obar \{C\}, \delta) & \\ \qquad \wedge\, \mathsf{sim_m}((p_1, g_1) \ominus \{C\}, (p_2, g_2) \ominus \{C\}, \delta) & \text{if } \mathsf{ran}\, g_1 \neq \emptyset \\ \qquad \text{where } C \in \mathsf{ran}\, g_1 & \\ \mathsf{true} & \text{otherwise} \end{cases}$$

△

We connect our MLSLS notion of similarity and our FOLRA approach to model this similarity with the following lemma. We point out that the part about “sanity of all combinations” in the following lemma is necessary to be able to use the similarity-formula from the previous definition.

Lemma 4.3.5. *For $i, j \in \{1,2\}$, a finite possibly simple MLSLS model M_j, a possibly simple FOLRA data tuple p_i, a car identifier variable assignment g_i and $\delta \in \mathbb{R}_{\geq 0}$ such that all combinations of (g_i, p_i, M_j) are sane and p_1 and p_2 contain no common FOLRA variables we have*

$\mathsf{tr^{lra}_m}(p_1, g_1, M_1) \wedge \mathsf{tr^{lra}_m}(p_2, g_2, M_2) \wedge \mathsf{sim_m}((p_1, g_1), (p_2, g_2), \delta)$ *is satisfiable*
iff
$d_\mathsf{m}(M_1, M_2) \leq \delta$.

Proof. As all combinations of (p_i, g_i, M_j) are sane we know that M_1 and M_2 have the same car domain. For the same reason we know that g_1 and g_2 have the same range. Hence, we can restrict the models to a single arbitrary common car, make a case distinction on whether the resulting model is simple or proper and then prove that the lemma holds for this car. □

For a proper FOLRA data tuple $p = (DS, l, l', \beta, \beta', f, \mathbb{D}, \mathcal{S}, D_E)$ we use the formula

$$\mathsf{sane^r}(p) \equiv \beta \leq \beta' \wedge \bigwedge_{D \in \mathbb{D}} \mathcal{S}(D)(v_\Omega) > 0$$

to express that the terms representing the extension of a view should form a nonempty interval and that the variable representing the value of the sensor function should be greater 0 for every car. We need this formula to ensure that with $\mathsf{sim_m}$ we only consider assignments that correspond to MLSLS models. We point out that $\mathsf{sane^r}$ alone is not sufficient to ensure that an assignment represents an MLSLS model.

We introduce some notational sugar. We denote the universal quantification over all data variables in $\mathcal{S}$ with $\forall\mathcal{S}$. Now, we can state that our FOLRA encoding of the robust model problem is correct.

Lemma 4.3.6. *Let M be a finite proper MLSLS model, p, p' two FOLRA data tuples with $p' = (DS', l'_1, l'_2, x'_1, x'_2, f', \mathbb{D}', \mathcal{S}', D'_E)$ and g, g' two car identifier variable assignments such that (p, g, M) and (p', g', M) are sane. Then, for all MLSLS formulas ϕ and all values $\delta \in \mathbb{R}_{>0}$ we have*

$$\psi_{\phi,\delta} \text{ is valid iff } M \models^{\delta} \phi \ ,$$

where the FOLRA formula $\psi_{\phi,\delta}$ is defined as

$$\psi_{\phi,\delta} \equiv \forall\mathcal{S}', x'_1, x'_2.\, (\mathsf{tr_m^{lra}}(p, g, M) \land \mathsf{sim_m}((p, g), (p', g'), \delta) \land \mathsf{sane^r}(p')) \implies \mathsf{tr_f^{lra}}(p', \phi) \ .$$

Proof. We start with the “if”-direction and proceed by contraposition. Assume the formula on the left side of the equivalence is not valid, i.e. there is an assignment h with $h \models \mathsf{tr_m^{lra}}(p, g, M) \land \mathsf{sim_m}((p, g), (p', g'), \delta) \land \mathsf{sane^r}(p') \land \neg\mathsf{tr_f^{lra}}(p', \phi)$. From the definition of the formulas $\mathsf{sim_m}$ and $\mathsf{sane^r}$ we see that the assignment h and the data tuple p' together represent some MLSLS model M'. We can create this model by first restricting h to the variables in p' and converting the restricted assignment and p' to a FOMLA tuple. This is possible because the formulas $\mathsf{sim_m}$ and $\mathsf{tr_m^{lra}}$ ensure that values representing lanes in h and p' are integers. Next, we transform the resulting FOMLA tuple with $\mathsf{mla2m}$ to an MLSLS model. For the resulting MLSLS model M' we know $h \models \mathsf{tr_m^{lra}}(p', g', M')$. Thus,

$$h \models \mathsf{tr_m^{lra}}(p, g, M) \land \mathsf{sim_m}((p, g), (p', g'), \delta) \land \mathsf{tr_m^{lra}}(p', g', M') \land \mathsf{sane^r}(p') \land \neg\mathsf{tr_f^{lra}}(p', \phi) \ . \quad (4.4)$$

From Lemma 4.3.5 we conclude $d_{\mathsf{m}}(M, M') \leq \delta$. Next, from Equation (4.4) and Lemma 3.4.7 we conclude $M' \not\models \phi$. Finally, $d_{\mathsf{m}}(M, M') \leq \delta$ and $M' \not\models \phi$ together imply $M \not\models^{\delta} \phi$.

We continue with the "only if"-direction and again proceed by contraposition. Assume $M \not\models^{\delta} \phi$. Then there is M' with $d_{\mathsf{m}}(M, M') \leq \delta$ and $M' \not\models \phi$, which implies $M' \models \neg\phi$. As (p, g, M) and (p', g', M) are sane and as the distance of M and M' is finite we know that also (p, g, M') and (p', g', M') are sane. Now we can use Lemma 4.3.5. Thus, from $d_{\mathsf{m}}(M, M') \leq \delta$ we know that

$$\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M) \wedge \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p', g', M') \wedge \mathsf{sim}_{\mathsf{m}}((p, g), (p', g'), \delta)$$

is satisfiable and let h be a satisfying assignment. As h satisfies $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p', g', M')$ it is easy to see that it also satisfies $\mathsf{sane}^{\mathsf{r}}(p')$. From $M' \models \neg\phi$ and Lemma 3.4.7 we know that the formula $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p', g', M') \implies \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p', \neg\phi)$ is valid. Hence, h also satisfies $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p', \neg\phi)$. We can pull the negation out and conclude that

$$\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p, g, M) \wedge \mathsf{sim}_{\mathsf{m}}((p, g), (p', g'), \delta) \wedge \mathsf{sane}^{\mathsf{r}}(p') \wedge \neg\mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p', \phi)$$

is satisfiable. We see that after binding the variables in $\mathcal{S}', x'_1$ and x'_2 with existential quantifiers, the formula above is equal to the negation of $\psi_{\phi,\delta}$ from this lemma. We see that the negation of $\psi_{\phi,\delta}$ is satisfiable and finally conclude that $\psi_{\phi,\delta}$ is not valid. $\square$

Now, using the previous lemma we conclude decidability of the robust model problem for MLSLS. While we did not investigate the complexity of our procedure, we refer to Lemma 2.4.6 for the complexity of FOLRA .

Theorem 4.3.7. *For $\delta \in \mathbb{R}_{>0}$ it is decidable whether a model satisfies an MLSLS formula δ-robustly.*

4.4 Related Work

Robustness of spatial logics received little attention so far because most spatial logics consider qualitative properties. However, as MLSL is inspired by temporal logics, robustness of timed systems is related.

In [GHJ97] the authors define and study robust timed automata. These automata accept tubes of trajectories. Such a tube is a set of similar trajectories, where similarity is defined with a metric on trajectories The authors show that these robust timed automata behave mostly like classical timed automata. That is, checking language emptiness is reducible to checking emptiness of classical timed automata. Furthermore, robust timed automata are not determinisable, like classical timed automata. The approach from [GHJ97] has been generalised to rectangular automata (which are a restricted form of hybrid automata). In [HR00] the authors show that also for this weakened form of robust rectangular automata, the reachability problem remains undecidable.

In [Frä99] the author considers hybrid automata, which are subject to noise. That is, for a given hybrid automaton A the actual behaviour (after adding noise) is given by a perturbed hybrid automaton A', where all state invariants, transition guards and flow constraints are perturbed. The author shows that for this model of robust hybrid automata the reachability problem is decidable, while it is undecidable for classical hybrid automata. The difference to [HR00] is that in [Frä99] the behaviour of the system is perturbed, while in [HR00] the set of trajectories of the original automaton are perturbed.

In [Pur00] the author considers reachability for timed automata with skewed (or drifting) clocks. This means that clocks may rise with different speed and that errors can accumulate. This work is similar to [Frä99] in that the behaviour of timed automata is affected by the perturbation. The author shows that the reachability problem remains decidable for timed automata with drifting clocks.

In [FH05] the authors define a robust interpretation for Duration Calculus (DC). To this end, the authors develop a real-valued semantics. A truth value greater zero means that the formula is satisfied and a truth value smaller than zero means that the formula is not satisfied. As an example we might have a formula ϕ that says

$$\text{the state expression } P \text{ holds for a duration of least four time units} \ .$$

If in a trajectory the state expression P holds for a duration of six time units, then evaluating ϕ on this trajectory yields a truth value of $6 - 4 = 2$. We can create a robust version of the original formula by "shaking the constants of the formula". That is, a robust version of the formula above might require that P holds for at least five time units. The resulting truth value would be $6 - 5 = 1$. At first we tried to find such a real-valued semantics for MLSL. However, we did not manage to find such an interpretation because the semantics of the atoms of MLSL is complex. That is, for the formula $\mathsf{re}(c)$ the current view should be contained in the safety envelope of c and the view should be nonempty. We did not find a way to measure strength of (dis)satisfaction for these atoms, and thus did not continue in this direction.

In [Koy90] Metric Temporal Logic (MTL) has been introduced. This logic has been shown to be undecidable [AFH96]. The proof heavily relies on formulas of the form $\Box\,(P \implies \Diamond_{[1,1]}Q)$, which specifies that whenever P holds, then exactly one time unit later Q will hold. These formulas cannot be checked by timed automata because every occurrence of P would require its own clock. As the number of these occurrences within a bounded interval is unbounded, the resulting automaton would need infinitely many clocks. The authors also show that by forbidding point intervals the resulting fragment, called Metric Interval Temporal Logic (MITL), becomes decidable. They follow a similar approach as the tableau based method for LTL, where eventuality formulas introduce obligations to satisfy subformulas. Consider the MITL formula $\Box (P \implies \Diamond_{[0,1]} Q)$, which allows Q to occur in a proper interval and the run $\langle (0.1, P), (0.2, P), \ldots, (1, Q) \rangle$, where many P occur before Q occurs. Then only the first obligation for the first P needs to be tracked. The obligations of the later P are satisfied if the first is satisfied. Building on this the authors reduce the satisfiability problem of MITL to the emptiness problem of timed automata.

In [FP09] the authors define for a given signal, continuous in time and space, and an MTL formula the spatial robustness degree. That is, for a signal s and a formula ϕ they define the maximal spatial perturbation δ, such that perturbing s by δ does not affect satisfaction of ϕ. Additionally, they develop a real-valued semantics for MTL that allows to under-approximate the robustness degree. That means, if the truth value of s on ϕ is 1, then we can perturb s by 1 and be sure that the resulting signal still satisfies ϕ. We might be able to perturb by more than 1 without affecting satisfaction. Hence, this is an under-approximation.

5 Monitoring of Spatio-Temporal Properties with Precise Information

We extend our approach to check whether a given static traffic configuration satisfies a given MLSLS formula to also consider time. That is, we give a procedure to check if an MLSLS transition sequence satisfies a spatio-temporal property. We view such a sequence as a behaviour. For other temporal logics, such as metric temporal logic (MTL), checking whether a single behaviour satisfies as temporal property is called *monitoring*.[1]

In monitoring for MTL the original signal to monitor is sampled to produce a time-stamped sequence of measurements. The values at times in between measurements are linearly interpolated to obtain a usable (and finitely representable) approximation of the original signal. In original MLSL the evolution of traffic is defined with a labelled transition system [HLO+11]. In our approach to monitoring we attempt to stay close to the original definition of evolution of traffic in [HLO+11]. Hence, we do not take a sequence of measurements as input, but rather a sequence of actions and then (mostly) follow the definition of the original labelled transition system to derive the behaviour.

In our procedure we use the first-order theory of real closed fields (FORCF), which is decidable [Tar51] (there called *elementary algebra*). We point out that FORCF is more powerful that FOLRA, which we used in Chapters 3 and 4 to solve the static model problem. Furthermore, the expressible properties of FORCF and FOMLA (used in Chapter 3 to solve the satisfiability problem) are incomparable, i.e. in both directions there are properties that can be expressed in one logic but not in the other.

The ideas of this chapter are taken from [Ody17]. In this thesis we additionally give proofs of correctness and we substantiate the underlying theory.

[1]If we perform this check for a potentially infinite set of behaviours, given for example as a transition system, it is called model checking.

In Section 5.1 we introduce timed words as the basis of MLSLS transition sequences. In Section 5.2 we define the spatio-temporal properties we consider, i.e. we define what it means in our context for a spatial property to hold globally in the temporal sense. Additionally, we give an algorithm to check whether a transition sequence satisfies such a property. In Section 5.3 we prove our algorithm to be correct and in Section 5.4 we discuss related work.

5.1 Timed Words and MLSLS

In this section we define timed words as basis for MLSLS transition sequences. For this, we first define the semantics of MLSLS transitions, then we define some operations on timed words and then we define timed words as the basis of MLSLS transition sequences.

In Chapter 3 (cf. Page 54) we introduced the assumption that for each car the value of the sensor function Ω is independent of the other cars. In this chapter we use FORCF to reason about MLSLS transition sequences. To this end, we introduce the assumption that the change of position, speed and the sensor function can be represented in FORCF. Simplifying, we assume that the position, the speed and the sensor function behave according to classical mechanics. For the change of position and speed this already is encoded in the transitions of classical MLSL. For the current speed v and deceleration a the braking distance d is given as

$$d = \frac{v^2}{2a} \enspace .$$

For our sensor function we thus get the following assumption.

Assumption 5.1.1. Let $M = (CS, TS', \Omega, V, \nu)$ be an arbitrary MLSLS model. Then we assume that the sensor function Ω is computed according to classical mechanics. That is, for all $C \in \mathsf{cars}\, M$, all $TS \in \mathbb{TS}$ with $TS = (\mathsf{res}, \mathsf{clm}, \mathsf{pos}, \mathsf{spd}, \mathsf{acc})$, the maximal deceleration $\mathsf{dec}_{\mathsf{max}}(C)$ and the car length $\mathsf{length}(C)$ we have

$$\Omega(C, TS) = \frac{\mathsf{spd}(C) * \mathsf{spd}(C)}{2 * \mathsf{dec}_{\mathsf{max}}(C)} + \mathsf{length}(C) \enspace ,$$

which is the *braking distance* plus the physical size of the car. △

Note that for position and speed this assumption already is made in the definition of MLSL transitions.

So far we considered only the static aspects of MLSLS. We now define the dynamics of MLSLS, i.e. how actions of the cars may change the model. In Chapter 2 on Page 23 we introduced MLSL transitions based on [HLO+11; LH15; Lin15]. We defined Act to be the set of all discrete actions cars may execute. Here, we partition this set by cars performing the actions. We collect the actions a car may perform in a set. For a car $C \in \mathbb{I}$ let $\mathsf{Act}_C = \{\mathrm{c}(C,n), \mathrm{r}(C), \mathrm{wd\ c}(C), \mathrm{wd\ r}(C,n), \mathrm{a}(C,a) \mid n \in \mathbb{L}, a \in \mathbb{R}\}$ be the actions of C. Then for a set of car identifiers $CS \subseteq \mathbb{I}$ we define $\mathsf{Act}_{CS} = \bigcup_{C \in CS} \mathsf{Act}_C$.

We define the semantics of transitions for MLSLS in terms of the classical transitions for MLSL. For this we first extend the classical transitions to also allow simple MLSL models. We call an MLSL model simple if it does not have a view and the valuation function does not map the special ego constant (and otherwise proper). That means, $M = (TS, \Omega, \nu)$ with $\mathsf{ego} \notin \mathsf{dom}\,\nu$ is simple. Then for a simple MLSL model we only update the traffic snapshot (cf. Definition 2.3.4). Let $\rightsquigarrow$ be the resulting transition relation. Note that we use a different arrow than in Definition 2.3.4 because we use $\rightarrow$ in this chapter for MLSLS transitions (cf. Definition 5.1.2).

For MLSLS we define that the transitions are labelled by sets of actions, or delays. Thus, if two actions occur at the same time the transition is labelled by a set containing these two actions. This has the advantage that our semantics easily can model truly concurrent behaviour, rather than interleaving behaviour. For a discussion of true concurrency and interleaving semantics within MLSL see [BHL+17]. We use the standard semantics for transitions with single actions and lift this to sets of actions. To avoid problems with dependent actions, that is, sets of actions where the model reached depends on the evaluation order we forbid such sets of actions. That is, we disallow sets of actions such as $\{\mathrm{wd\ c}(C,n), \mathrm{r}(C)\}$. While we could allow for example $\{\mathrm{wd\ c}(C,n), \mathrm{a}(C,a)\}$, as the model reached is independent of the order in which the actions are resolved, we take the easy route and simply disallow sets of actions containing multiple actions from a single car. The semantics of transitions is defined using the original semantics of MLSL transitions.

Definition 5.1.2 (MLSLS Transitions)**.** For a (possibly simple) MLSLS model M let $f(M)$ be the classical MLSL model that is equal to M, except that $f(M)$ does not have a scope. That means, for $M = (CS, TS, \Omega, V, \nu)$ (resp. $M = (CS, TS, \Omega, \nu)$ if M is simple) we define $f(M) = (TS, \Omega, V, \nu)$ (resp. $f(M) = (TS, \Omega, \nu)$ if M is simple). Let M_1, M_2, M_3 be three MLSLS models such that all are either simple or proper and have equal scopes. For a finite set of actions $A \subseteq \mathsf{Act}$ such that for all $C \in \mathbb{I}$ we have $|A \cap \mathsf{Act}_C| \leq 1$ and a delay $t \in \mathbb{T}$ we define

$$M_1 \xrightarrow{A} M_2 \text{ iff } \begin{cases} M_1 = M_2 & \text{if } A = \emptyset \ , \\ f(M_1) \overset{a}{\rightsquigarrow} f(M_3) \text{ and } M_3 \xrightarrow{A \setminus \{a\}} M_2 & \\ \qquad \text{for some } M_3 \text{ with } a \in A & \text{otherwise} \ , \end{cases}$$

$$M_1 \xrightarrow{t} M_2 \text{ iff } f(M_1) \overset{t}{\rightsquigarrow} f(M_2) \ ,$$

where $\rightsquigarrow$ is the transition relation for original MLSL defined above (cf. Page 151).
△

We proceed to define operations on timed words and point out that here the operations may change the elements occurring in a timed word. For timed words we define a composition operator. The idea behind the definition is that for two timed words we compare the time stamps of the last sets of actions in both timed words. Then we either append the actions with the later time stamp or join the sets if their time stamps are equal. Then we continue recursively. We point out that while we do not consider the empty timed word in general, in this definition we consider the empty timed word to ease the exposition. We indicate this by the data type $(\mathcal{P}(\Sigma) \times \mathbb{T})^*$.

Definition 5.1.3. For $i \in \{1, 2\}$ let $\varrho'_i \in (\mathcal{P}(\Sigma) \times \mathbb{T})^+$ be a timed word and let $\varrho'_i = \varrho_i :: (A_i, t_i)$, which means $\varrho_i \in (\mathcal{P}(\Sigma) \times \mathbb{T})^*$ and $(A_i, t_i) \in (\mathcal{P}(\Sigma) \times \mathbb{T})$. We define the *parallel composition* of ϱ'_1 and ϱ'_2 (denoted $\varrho'_2 \parallel \varrho'_1$) as

$$\varrho_1 \parallel \langle\rangle = \langle\rangle \parallel \varrho_1 = \varrho_1 \ ,$$

$$(\varrho_1 :: (A_1, t_1)) \parallel (\varrho_2 :: (A_2, t_2)) = \begin{cases} (\varrho_1 \parallel (\varrho_2 :: (A_2, t_2))) :: (A_1, t_1) & \text{if } t_1 > t_2 \ , \\ ((\varrho_1 :: (A_1, t_1)) \parallel \varrho_2) :: (A_2, t_2) & \text{if } t_1 < t_2 \ , \\ (\varrho_1 \parallel \varrho_2) :: (A_1 \cup A_2, t_1) & \text{if } t_1 = t_2 \ . \end{cases}$$

△

Note that the definition above can easily be reversed to instead prepend at the start, rather than append at the end. As for static MLSLS models we define (anti-) restriction operators for timed words. Note that these operators leave the span of the timed word unchanged. In Example 5.1.10 on Page 155 we give an example of the parallel composition defined above, and the (anti-)restriction defined below.

Definition 5.1.4. Let $\varrho \in (\mathcal{P}(\Sigma) \times \mathbb{T})^+$ with $\mathsf{last}\,\varrho = (A, t)$. For a set $\Sigma' \subseteq \Sigma$ we define the *restriction* to Σ' (denoted by $\varrho \restriction \Sigma'$) as

$$\begin{aligned}\langle\rangle \restriction \Sigma' &= \langle\rangle \ , \\ \varrho \restriction \Sigma' &= ((\mathsf{front}\,\varrho) \restriction \Sigma') :: (A \cap \Sigma', t) \ .\end{aligned}$$

We define the *anti-restriction* of ϱ to Σ' (denoted $\varrho \setminus \Sigma'$) as

$$\varrho \setminus \Sigma' = \varrho \restriction (\Sigma \setminus \Sigma') \ . \qquad \triangle$$

To restrict a timed word to the action alphabet of a set of cars $CS \subseteq \mathbb{I}$, we use the abbreviations $\varrho \restriction CS = \varrho \restriction \mathsf{Act}_{CS}$ and $\varrho \setminus CS = \varrho \restriction \mathsf{Act}_{CS}$.

Note that for our operations like restriction to work intuitively it seems important to work with absolute time stamps. This is because with absolute time stamps we can leave the time stamps as they are in our operations.

We state some basic properties of concatenation of timed words and the newly defined operators. The following equality directly follows fro the definitions. For all timed words ϱ and all sets Σ we have

$$\varrho = (\varrho \restriction \Sigma) \parallel (\varrho \setminus \Sigma) \ .$$

The following observation also follows from the definition of $\parallel$.

Lemma 5.1.5 (Algebraic Properties of Parallel Composition of Timed Words)**.** *The operation $\parallel$ is commutative, associative and idempotent.*

That taking the parallel composition of two timed words is commutative and associative clearly is desirable. As for idempotence, this is not clearly desirable. However, in the setting of MLSLS this is not a problem because all MLSLS actions are idempotent. Consider some arbitrary model M, a car $C \in \mathbb{I}$ and the transition sequence

$$M \xrightarrow{\{\mathrm{r}(C)\}} M' \xrightarrow{\{\mathrm{r}(C)\}} M'' \ .$$

Then $M' = M''$ because a possibly existing claim is turned into a reservation, and if C does not have a claim, then the action is without effect. This works similarly for the other actions. Note that this does not hold for delays.

For two timed words $\varrho_i = (w_i, \tau_i)$ with $i \in \{1, 2\}$ we say that ϱ_1 is *earlier* than ϱ_2 if $\mathsf{last}\,\tau_1 \leq \tau_2(1)$. Similarly, we say that ϱ_2 is *later* than ϱ_1 if ϱ_1 is earlier than ϱ_2.

Let $\varrho = (w, \tau)$ be a timed word of length n. We allow appending sets of actions occurring at the same time as the last set of actions in ϱ. Then we consider $\varrho \cdot \langle (A, \mathsf{tail}\,\tau \rangle$ to be equal to $((w[..n-1]) :: (w(n) \cup A), \tau)$. That is, if a timed word has two sets of actions with equal time stamps, we silently merge those sets of actions.

There are more basic properties. For timed words where concatenation is defined, it commutes with parallel composition.

Lemma 5.1.6. *For all timed words $\varrho_i = (w_i, \tau_i)$ with $i \in \{1,2,3,4\}$ such that ϱ_1, ϱ_2 are earlier than ϱ_3, ϱ_4 we have*

$$\begin{gathered} \varrho_1 \parallel (\varrho_2 \cdot \varrho_3) = (\varrho_1 \parallel \varrho_2) \cdot \varrho_3 = (\varrho_1 \cdot \varrho_3) \parallel \varrho_2 \ , \\ \varrho_1 \parallel \varrho_3 = \varrho_1 \cdot \varrho_3 \ , \\ (\varrho_1 \parallel \varrho_2) \cdot (\varrho_3 \parallel \varrho_4) = (\varrho_1 \cdot \varrho_3) \parallel (\varrho_2 \cdot \varrho_4) \ . \end{gathered}$$

Proof. The first property can be shown via induction on ϱ_3 and the second follows from the definition the parallel composition of timed words. The last property follows from the first two. □

The properties of the previous lemma look similar to properties between layer composition, which introduces causal dependency, and parallel composition [Jan94, p. 46]. There, these operators are part of a language to define processes.

We consider two timed words ϱ, ϱ' to be *equal up to interior empty sets of actions* (denoted by $\varrho \approx \varrho'$) iff they have an equal time span and both have the same actions at the same point in time. That means, $\varrho \approx \varrho'$ iff $\mathsf{span}\,\varrho = \mathsf{span}\,\varrho'$ and for all (A, t) in ϱ with $a \in A$ there is (A', t) in ϱ' such that $a \in A'$ and analogously for all (A, t) in ϱ'. Now, the operations $\parallel, \setminus, \upharpoonright$ and $\cdot$ respect $\approx$.

Lemma 5.1.7. *Let ϱ_i, ϱ'_i with $i \in \{1,2\}$ be four timed words with $\varrho_i \approx \varrho'_i$ and ϱ_1 before ϱ_2 and ϱ'_1 before ϱ'_2. Further, let ϱ be an arbitrary timed word and let Σ be an arbitrary set. Then we have*

$$\begin{gathered} \varrho \parallel \varrho_1 \approx \varrho \parallel \varrho'_1 \\ \varrho_1 \cdot \varrho_2 \approx \varrho'_1 \cdot \varrho'_2 \\ \varrho_1 \setminus \Sigma \approx \varrho'_1 \setminus \Sigma \\ \varrho_1 \upharpoonright \Sigma \approx \varrho'_1 \upharpoonright \Sigma \end{gathered}$$

Proof. The proof follows quickly by first assuming that the property does not hold. Then the contradiction is immediate. □

We define the *time-bounded prefix* of a timed word.

Definition 5.1.8 (Time-Bounded Prefix of Timed Words)**.** For a timed word $\varrho = (w, \tau)$ and $t \in \mathbb{T}$ let $i \in \{0, \dots n\}$ be the largest index such that $\tau(i) \leq t$ where $\tau(0) = 0$. We define the *time-bounded prefix* $\mathsf{pre}(\varrho, t)$ as

$$\mathsf{pre}(\varrho, t) = \langle (w(1), \tau(1)), \dots, (w(i), \tau(i)), (\emptyset, \mathsf{min}(t, \overline{\mathsf{span}\,\varrho})) \rangle \ ,$$

where $\overline{\mathsf{span}\,\varrho}$ is the right (larger) border of the interval $\mathsf{span}\,\varrho$. △

We refer to Example 5.1.10 on Page 155 for an example. We proceed to show some basic properties of the timed prefix operator.

Lemma 5.1.9. *Let ϱ_1, ϱ_2 be two timed words, let Σ be an arbitrary set and $t \in \mathbb{T}$. Further, let ϱ be a timed word such that $\varrho \approx \varrho_1$. Then*

$$\begin{aligned}
\mathsf{pre}(\varrho_1, t) \parallel \mathsf{pre}(\varrho_2, t) &= \mathsf{pre}(\varrho_1 \parallel \varrho_2, t) \\
\mathsf{pre}(\varrho_1, t) \upharpoonright \Sigma &= \mathsf{pre}(\varrho_1 \upharpoonright \Sigma, t) \\
\mathsf{pre}(\varrho_1, t) \setminus \Sigma &= \mathsf{pre}(\varrho_1 \setminus \Sigma, t) \\
\mathsf{pre}(\varrho_1, t) &\approx \mathsf{pre}(\varrho_1', t)
\end{aligned}$$

Proof. For $i \in \{1, 2\}$ let ϱ_i' be the suffix of ϱ_i that happens after time t and note that ϱ_i' might be the empty sequence $\langle\rangle$. Then $\mathsf{pre}(\varrho_i, t) \cdot \varrho_i' = \varrho_i$. While $\langle\rangle$ is not a timed word, appending $\langle\rangle$ to a timed word leaves it unchanged.

For the first property we have

$$\begin{aligned}
\mathsf{pre}(\varrho_1, t) \parallel \mathsf{pre}(\varrho_2, t) &= \mathsf{pre}(\mathsf{pre}(\varrho_1, t) \parallel \mathsf{pre}(\varrho_2, t), t) \\
&= \mathsf{pre}((\mathsf{pre}(\varrho_1, t) \cdot \varrho_1') \parallel (\mathsf{pre}(\varrho_2, t) \cdot \varrho_2'), t) \\
&= \mathsf{pre}(\varrho_1 \parallel \varrho_2, t) \ .
\end{aligned}$$

In the first step we add the prefix operator. Taking the prefix until time t is equal to taking that prefix multiple times. Then we add the suffix, which the outermost prefix operator cuts away again. Hence, adding the suffix does not change anything. At last, we use the equality mentioned initially. For the second and third property a similar approach works.

The last property follows by assuming an counter example and deriving a contradiction. □

Example 5.1.10. Consider the timed words $\varrho_1, \varrho_2, \varrho_3$ and their parallel compositions:

$$\begin{aligned}
\varrho_1 &= \langle(\{b\}, 6.1)\rangle \ , & \varrho_1 \parallel \varrho_3 &= \langle(\{c\}, 1), (\{c\}, 5.1), (\{b\}, 6.1)\rangle \ , \\
\varrho_2 &= \langle(\{a\}, 1), (\{b\}, 6.1)\rangle \ , & \varrho_1 \parallel \varrho_2 &= \langle(\{a\}, 1), (\{b\}, 6.1)\rangle \ , \\
\varrho_3 &= \langle(\{c\}, 1), (\{c\}, 5.1)\rangle \ , & \varrho_2 \parallel \varrho_3 &= \langle(\{a, c\}, 1), (\{c\}, 5.1), (\{b\}, 6.1)\rangle \ .
\end{aligned}$$

As an example for the properties in Lemma 5.1.7 we see that $\varrho_3 \cdot \varrho_1 = \varrho_3 \parallel \varrho_1$, i.e. because the event in ϱ_1 happens after all events in ϱ_3 their parallel composition is equal to appending the later to the earlier. We define the two timed words

$$\begin{aligned}
\varrho &= (\varrho_2 \parallel \varrho_3) \restriction \{a, b\} = \langle(\{a\}, 1), (\emptyset, 5.1), (\{b\}, 6.1)\rangle \ , \\
\varrho' &= \langle(\{a\}, 1), (\{b\}, 6.1)\rangle
\end{aligned}$$

These timed words are equal up to interior empty sets of actions, i.e. we have $\varrho \approx \varrho'$. If we now consider the timed words $\varrho \setminus \{b\} = \langle(\{a\}, 1), (\emptyset, 5.1), (\emptyset, 6.1)\rangle$ and $\varrho' \setminus \{b\} = \langle(\{a\}, 1), (\emptyset, 6.1)\rangle$ we see that they also are equal up to interior empty sets of actions. Note that $\varrho \setminus \{b\}$ and $\langle(\{a\}, 1)\rangle$ are *not* equal up to interior empty sets of actions because the span of both timed words is different.

Next, we consider the time-bounded prefix. In the definition we see that an empty set of actions is inserted either at the time-bound of the prefix operation, or at the end of the time span. As an example we have

$$\begin{aligned}
\mathsf{pre}(\varrho_1, 7) = \mathsf{pre}(\varrho_1, 6.1) &= \langle(\{b\}, 6.1), (\emptyset, 6.1)\rangle \ , \\
\mathsf{pre}(\varrho_1, 6) &= \langle(\emptyset, 6)\rangle \ .
\end{aligned}$$

Further, it makes no difference, whether we first take the time-bounded prefix and then the parallel composition, or the other way around. Thus,

$$\begin{aligned}
\mathsf{pre}(\varrho_2, 6) &= \langle(\{a\}, 1), (\emptyset, 6)\rangle \ , \\
\mathsf{pre}(\varrho_3, 6) &= \langle(\{c\}, 1), (\{c\}, 5.1), (\emptyset, 6)\rangle \ , \\
\mathsf{pre}(\varrho_2 \parallel \varrho_3, 6) = \mathsf{pre}(\varrho_2, 6) \parallel \mathsf{pre}(\varrho_3, 6) &= \langle(\{a, c\}, 1), (\{c\}, 5.1), (\emptyset, 6)\rangle \ . \qquad \triangle
\end{aligned}$$

Now we define timed words as the basis of MLSLS transition sequences. We define that the application of a timed word to a possibly simple MLSLS model gives an MLSLS transition sequence. The idea is that we first let time advance to the i-th time stamp and then perform the i-th set of actions. This definition is similar to how we get *run* for a timed automaton from an initial configuration and a timed word. Note that we require that the timed word only contains actions of cars that are represented in the MLSLS model.

Definition 5.1.11 (From Timed Words to Transition Sequences)**.** For a possibly simple model M_0 and a timed word $\varrho = (w, \tau) \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\, M_0}) \times \mathbb{T})^+$ we define the transition sequence of ϱ from M_0 as

$$\begin{aligned}\mathcal{I}(M_0, \varrho) = M_0 \xrightarrow{\tau(1)} M_1 \xrightarrow{w(1)} M_2 \xrightarrow{\tau(2)-\tau(1)} \ldots \xrightarrow{\tau(n-1)-\tau(n-2)} M_{2n-3} \\ \xrightarrow{w(n-1)} TS_{2n-2} \xrightarrow{\tau(n)-\tau(n-1)} M_{2n-1} \xrightarrow{w(n)} M_{2n} \ .\end{aligned}$$

We use the letter ϖ to refer to transition sequences and we refer to the last model in ϖ with $\mathsf{last}\,\varpi$.[2] For $t \in \mathbb{T}$ we define the *time-bounded transition sequence until* t as $\mathcal{I}(M_0, \mathsf{pre}(\varrho, t))$. For a transition sequence $\varpi = \mathcal{I}(M_0, \varrho)$ we define $\varpi@t = \mathsf{last}\,\mathcal{I}(M_0, \mathsf{pre}(\varrho, t))$, i.e. $\varpi@t$ is the model at time t in ϖ. △

In this thesis we only consider transition sequences resulting from timed words and MLSLS models. Hence, we often write $\varrho(M)$ instead of ϖ.

We lift the time span for timed words (cf. Page 8) to transition sequences, which we denote as $\mathsf{span}\,\varpi$. Furthermore, we lift the equality of timed words up to interior empty sets of actions to equality of transition sequences up to interior empty sets of actions, which we denote with $\varpi \approx \varpi'$. As interior empty sets of actions do not change the state we make the following observation. Let ϖ, ϖ' be two transition sequences with $\varpi \approx \varpi'$. Then, at all points in time the models at that time in the two transition sequences are equal, i.e. we have

$$\forall t.\, (\varpi@t = \varpi'@t) \ .$$

For an easier exposition we make the following assumption.

Assumption 5.1.12. For a given (possibly simple) MLSLS model M we assume that all timed words we consider result in transition sequences with only legal transitions and that all transitions with $\mathrm{r}(C), \mathrm{wd\ c}(C), \mathrm{wd\ r}(C, n)$ change the state. That is, a car only makes a reservation if it has a claim, it only withdraws a claim when it has a claim and it only withdraws a reservation if it has two reservations. △

We give an example of a timed word and how we create a transition sequence from it. Note that we do not give units in our examples. However, we assume all units to match.

[2]Note that ϖ is a variant of ω (omega) that we use to set it apart from the Latin letter w.

Example 5.1.13. Let us assume that the global, maximal deceleration constant is given as $\mathsf{dec_{max}} = 12$ and that each car has a physical length of 3. Consider the timed word

$$\varrho = \langle(\{\mathrm{wd\ r}(E,3)\},1),(\{\mathrm{r}(C_2)\},1.1),(\{\mathrm{wd\ r}(C_2,2),\mathrm{c}(E,2)\},5.1),(\emptyset,6.1)\rangle$$

and a traffic snapshot $TS_0 = (\mathsf{res},\mathsf{clm},\mathsf{pos},\mathsf{spd},\mathsf{acc})$ defined as

$$\begin{array}{ll}
\mathsf{res} = \{C_1 \mapsto \{2\}, E \mapsto \{2,3\}, C_2 \mapsto \{1\}\} & \mathsf{spd} = \{C_1 \mapsto 6, E \mapsto 18, C_2 \mapsto 12\} \\
\mathsf{clm} = \{C_1 \mapsto \{3\}, E \mapsto \emptyset, C_2 \mapsto \{2\}\} & \mathsf{acc} = \{C_1 \mapsto 0, E \mapsto 0, E \mapsto 0\} \\
\mathsf{pos} = \{C_1 \mapsto 60, E \mapsto 16, C_2 \mapsto 12\} &
\end{array}$$

We assume for all cars C that $\mathsf{dec_{max}}(C) = 12$ and $\mathsf{length}(C) = 3$. According to Assumption 5.1.1 the sensor function gives the values

$$\Omega = \{(C_1, TS) \mapsto 4.5, (E, TS) \mapsto 16.5, (C_2, TS) \mapsto 9\} \ .$$

Note that TS is a formalisation of the traffic snapshot from Figure 2.4 on Page 21 (however, we use a different view here). Let $M_0 = (\{C_1, C_2, E\}, TS, \Omega, V, \nu)$ with $V = ([1,3],[0,90],E)$ and $\nu = \{\mathsf{ego} \mapsto E\}$. By applying ϱ to M_0, we get the transition sequence

$$\varrho(M_0) = M_0 \xrightarrow{1} M_1 \xrightarrow{\{\mathrm{wd\ r}(E,3)\}} M_2 \xrightarrow{0.1} M_3 \xrightarrow{\{\mathrm{r}(C_2)\}}$$
$$M_4 \xrightarrow{4} M_5 \xrightarrow{\substack{\{\mathrm{wd\ r}(C_2,2),\\ \mathrm{c}(E,2)\}}} M_6 \xrightarrow{1} M_7 \xrightarrow{\emptyset} M_7$$

depicted in Figure 5.1. The model within this sequence at time 5.1 is defined as $\varrho(M_0)@5.1 = \mathsf{last}\,\mathcal{I}(M_0, \mathsf{pre}(\varrho, 5.1)) = M_6$. We can also look at models for which there is no explicit time stamp in our timed word. For example, $\varrho(M_0)@4$ is temporally in between M_4 and M_5, i.e. we have $M_4 \xrightarrow{2.9} \varrho(M_0)@4$. △

Now we can define operations on transition sequences. Note that our operations on transition sequences translate to operations on the underlying timed words and starting model.

Definition 5.1.14. Let M be a possibly simple MLSLS model, $CS \subseteq \mathbb{I}$ and $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M}) \times \mathbb{T})^+$. We define the *restriction* and *anti-restriction* as

$$\begin{aligned}
\varrho(M) \upharpoonright CS &= (\varrho \upharpoonright \mathsf{Act}_{CS})(M \boxed{\upharpoonright} CS) \ , \\
\varrho(M) \setminus CS &= \varrho(M) \upharpoonright (\mathbb{I} \setminus CS) \ .
\end{aligned}$$

△

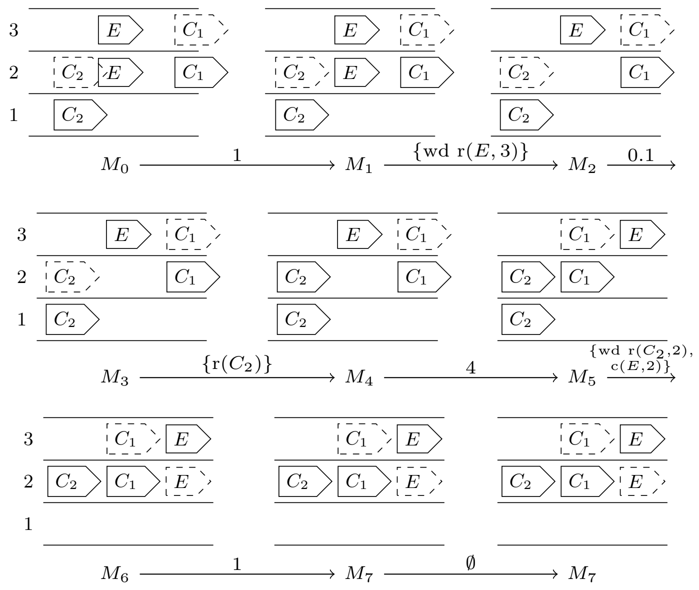


Figure 5.1: Visualisation of the transition sequence in Example 5.1.13. Claims are shown with dashed and reservations with solid lines. The view is not explicitly indicated, but contains the extension and the lanes shown.

We define the parallel composition of MLSLS transition sequences using the parallel composition of timed words and the disjoint union of MLSLS models.

Definition 5.1.15. Let M_1, M_2 be possibly simple, composable MLSLS models and let $\varrho_i \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M_i}) \times \mathbb{T})^+$ for $i \in \{1, 2\}$. We define the *parallel composition* of $\varrho_1(M_1)$ and $\varrho_2(M_2)$ as

$$\varrho_1(M_1) \mathbin{\boxed{|||}} \varrho_2(M_2) = \varrho_1 \parallel \varrho_2(M_1 \boxplus M_2) \ . \qquad \triangle$$

We call two transition sequences *composable* if their underlying starting models are composable.

Now, there is the question of what properties our operators on transition sequences have. However, it seems that our requirement for the disjoint union of MLSLS models, that the participating models need to represent disjoint sets of cars, limits interesting properties. For example, we might be tempted to relate $(\varrho \parallel \varrho')(M)$ to $\varrho(M) \mathbin{\boxed{|||}} \varrho'(M)$. However, M is not composable with itself (unless $M = M_\emptyset$). Thus, $\varrho(M) \mathbin{\boxed{|||}} \varrho'(M)$ is not defined. Similarly, for $\varrho(M \boxplus M')$. We might be tempted to relate this to $\varrho(M) \mathbin{\boxed{|||}} \varrho(M')$. But again, this is (in most cases) not defined as we require $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M \cup \mathsf{cars}\,M'}) \times \mathbb{T})^+$ and $\mathsf{cars}\,M \cap \mathsf{cars}\,M' = \emptyset$. Thus, we can relate $\varrho(M \boxplus M')$ and $\varrho(M) \mathbin{\boxed{|||}} \varrho(M')$ only if $\varrho \in (\mathcal{P}(\mathsf{Act}_\emptyset) \times \mathbb{T})^+$.

What we *can* relate is $\varrho(M)$ and $(\varrho(M) \upharpoonright CS) \mathbin{\boxed{|||}} (\varrho(M) \setminus CS)$. We will use the following proposition in our arithmetic encoding of transition sequences as it allows to break a transition sequence for multiple cars down into multiple sequences for a single car.

Proposition 5.1.16. *For all possibly simple MLSLS models M and all timed words $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M}) \times \mathbb{T})^+$ we know*

$$\varrho(M) = (\varrho(M) \upharpoonright CS) \mathbin{\boxed{|||}} (\varrho(M) \setminus CS) \ .$$

We introduce more properties relating transition sequences and our operators. Note that these properties relate operators on transition sequences with operators on static MLSLS models.

Lemma 5.1.17. *Let ϖ_1, ϖ_2 be two composable transition sequences with equal span and let $CS \subseteq \mathbb{I}$ be a set of car identifiers. Then*

$$\begin{aligned} \mathsf{last}(\varpi_1 \upharpoonright CS) &= (\mathsf{last}\,\varpi_1) \mathbin{\boxed{\upharpoonright}} CS \ , \\ \mathsf{last}(\varpi_1 \setminus CS) &= (\mathsf{last}\,\varpi_1) \boxminus CS \ , \\ \mathsf{last}(\varpi_1 \mathbin{\boxed{|||}} \varpi_2) &= (\mathsf{last}\,\varpi_1) \boxplus (\mathsf{last}\,\varpi_2) \ . \end{aligned}$$

Proof. Let $\varpi_1 = M_0 \xrightarrow{t_1} M_1 \xrightarrow{A_1} \dots \xrightarrow{t_n} M_{2n-1} \xrightarrow{A_n} M_{2n}$. For the first property let $\varpi_1 \upharpoonright CS = M'_0 \xrightarrow{t_1} M'_1 \xrightarrow{A_1 \cap \mathsf{Act}_{CS}} \dots \xrightarrow{t_n} M_{2n-1} \xrightarrow{A_n \cap \mathsf{Act}_{CS}} M'_{2n}$. Then we have $M'_{2n} = M_{2n} \boxed{\upharpoonright} CS$ because the state of the cars in CS is not affected by the cars not in CS. A similar argumentation works for the second property.

For the last property let $CS' \subseteq \mathbb{I}$ be such that $\varpi'_1 = (\varpi_1 \boxed{|||}\, \varpi_2) \upharpoonright CS'$ and $\varpi'_2 = (\varpi_1 \boxed{|||}\, \varpi_2) \setminus CS'$ with $\varpi'_1 \approx \varpi_1$ and $\varpi'_2 \approx \varpi_2$. Such a set CS exists because ϖ_1 and ϖ_2 are composable and because their time span is equal. Now

$$\begin{aligned}
\mathsf{last}(\varpi_1 \boxed{|||}\, \varpi_2) &= ((\mathsf{last}(\varpi_1 \boxed{|||}\, \varpi_2)) \boxed{\upharpoonright} CS') \boxplus ((\mathsf{last}(\varpi_1 \boxed{|||}\, \varpi_2)) \boxminus CS') \\
&= (\mathsf{last}((\varpi_1 \boxed{|||}\, \varpi_2) \upharpoonright CS')) \boxplus ((\mathsf{last}(\varpi_1 \boxed{|||}\, \varpi_2)) \setminus CS') \\
&= (\mathsf{last}\,\varpi'_1) \boxplus (\mathsf{last}\,\varpi'_2) \\
&= (\mathsf{last}\,\varpi_1) \boxplus (\mathsf{last}\,\varpi_2) \enspace .
\end{aligned}$$

The first step holds by definition of the operations on static MLSLS models. The second step uses the two first properties of this lemma. The third step substitutes the transition sequences we defined earlier. The last step holds because ϖ'_1 and ϖ'_2 merely differ from ϖ_1 and ϖ_2 by empty sets of actions. $\square$

5.2 Monitoring Global Properties

In [Lin15] the author extends MLSL with temporal modalities. However, the globally operator **G** defined there requires that *all* possible successor models satisfy the subformula. Hence, **G** is a branching time temporal operator. As branching time modalities are not suited for monitoring, we define a linear time globally modality, which is satisfied if the subformula is satisfied at every point in time. In this section we first formalise for an MLSLS model M, a timed word ϱ and an MLSLS formula ϕ what the statement "ϕ holds globally in $\varrho(M)$" means. The intuition is that we check for every point in time t, whether the model in the transition sequence $\varrho(M)$ at time t satisfies ϕ, which in symbols is $\varrho(M)@t \models \phi$. Afterwards, we define a transformation that takes as inputs ϱ, M and ϕ, and creates a formula from the first-order theory of real-closed fields [Tar51]. This formula is valid iff the MLSLS formula ϕ holds throughout $\varrho(M)$.

Definition 5.2.1 (Global Satisfaction)**.** A transition sequence $\varrho(M)$ globally satisfies a spatial property ϕ (denoted as $\varrho(M) \models_{\mathsf{seq}} \Box\, \phi$) iff at every point in time t the formula ϕ is satisfied. Formally,

$$\varrho(M) \models_{\mathsf{seq}} \Box\,\phi \text{ iff } \forall t.\, \varrho(M)@t \models \phi \ . \qquad \triangle$$

5.2.1 Encoding Global Properties in FORCF

In the previous chapters we used FOLRA data tuples to encode the model problem of MLSLS. In this chapter we use FORCF data tuples, which are very similar to FOLRA data tuples. The only difference is that here terms may contain multiplication, whereas in FOLRA data tuple they contained addition but not multiplication. The set of all proper (resp. simple) FORCF data tuples is defined as

$$\begin{aligned}
\mathrm{T}^{\mathsf{rcf}}_{\mathsf{p}} &= \mathcal{P}(\mathsf{DVar}) \times \mathbb{N}^2 \times \mathsf{RTerm}^2 \times (\mathsf{CVar} \cup \{\mathsf{ego}\} \to \mathsf{DVar}) \times \\
&\qquad\qquad \mathcal{P}(\mathsf{DVar}) \times (\mathsf{DVar} \to \mathsf{RVar}^7) \times \mathsf{DVar} \ , \\
\mathrm{T}^{\mathsf{rcf}}_{\mathsf{s}} &= \mathcal{P}(\mathsf{DVar}) \times \mathsf{CVar} \to \mathsf{DVar}) \times \mathcal{P}(\mathsf{DVar}) \times (\mathsf{DVar} \to \mathsf{RVar}^7) \ .
\end{aligned}$$

Note that this is the same type as the set of all FOLRA data tuples. The reason for this is that in our real-valued terms we do not distinguish whether the terms use multiplication or not. As for FOMLA and FOLRA data tuples we require for a FORCF data tuple $p = (DS, l_1, l_2, \beta_1, \beta_2, f, \mathbb{D}, \mathcal{S}, D_E)$ that $DS, \mathsf{ran}\, f \subseteq \mathbb{D}, \mathsf{dom}\, \mathcal{S} = \mathbb{D}, D_E \in \mathbb{D}$ and $f(\mathsf{ego}) = D_E$ and similarly for simple FORCF data tuples (see also Page 75). Further, we denote the set of all possibly simple FORCF data tuples as $\mathrm{T}^{\mathsf{rcf}}$.

To simplify our encoding we generally assume that the car identifier variable assignment is bijective. That is, for an MLSLS model M we assume a set of car identifier variables $\mathbb{D} \subseteq \mathsf{DVar}$ and a function $g : \mathbb{D} \to \mathsf{cars}\, M$ such that $|\mathbb{D}| = |\,\mathsf{cars}\, M|$. In the following definition, transforming MLSLS actions to arithmetic constraints, this bijectivity allows us to consider a single car identifier variable. Note that in the previous chapters $g : \mathbb{D} \to \mathsf{cars}\, M$ only was constrained to be surjective.

We assume that a single car will not perform multiple actions simultaneously (cf. Section 5.1). Additionally, we assume that all actions have an effect and actually are enabled, when they are performed (cf. Assumption 5.1.12). The goal of this assumption is to simplify our encoding but can likely lifted. We point out that the disjunction for the reservations are needed because we have not fixed a rule which reservation variable should take which lane, or which variable should be set to $\bullet$. Similarly, we use $\{v'_{\mathsf{r1}}, v'_{\mathsf{r2}}\} = \{v_{\mathsf{r1}}, v_{\mathsf{r2}}\}$ instead of simply

$\mathsf{id}(v_{\mathsf{r1}}, v_{\mathsf{r2}})$ because we want $\{v_{\mathsf{r1}}, v_{\mathsf{r2}}\}$ to stay the same. This will be helpful in proving correctness later. Our formula takes into account that we do not know which of the variables v'_{r1} and v'_{r2} had $\bullet$ assigned and which had a lane assigned. Further, for two sets of variables $S = \{v_1, \ldots, v_n\}, S' = \{v'_1, \ldots, v'_n\}$ we use the abbreviation $\mathsf{id}(S) \equiv \bigwedge_{i\in\{1,\ldots,n\}} v_i = v'_i$. Note that the set S' is implicitly known in the context. We point out that for real-valued terms $\beta_1, \beta_2, \beta_3$ we can express the equality $\beta_1 = \frac{\beta_2}{\beta_3}$ in FORCF as $\exists x.\, \beta_1 = \beta_2 * x \wedge \beta_3 * x = 1$.

Definition 5.2.2 (Transforming Actions)**.** Let $p = (DS, f, \mathbb{D}, \mathcal{S})$ and $p' = (DS', f', \mathbb{D}', \mathcal{S}')$ be two simple FORCF data tuples with $DS = DS'$, $f = f'$, $\mathbb{D} = \mathbb{D}'$ and $\mathbb{D} = \{D\}$ for some $D \in \mathsf{DVar}$. For some car $C \in \mathbb{I}$, a car identifier variable assignment g mapping D to C, a set of actions $A \subseteq \mathsf{Act}_C$ with $|A| \leq 1$, a lane $l \in \mathbb{L}$, an acceleration $\alpha \in \mathbb{R}$ and a FORCF term θ indicating a delay we encode performing an action and letting time pass as

$$\mathsf{tr}'_{\mathsf{a}}(A, p, p', g) \equiv \begin{cases} v'_{\mathsf{c}} = l \wedge \{v'_{\mathsf{r1}}, v'_{\mathsf{r2}}\} = \{v_{\mathsf{r1}}, v_{\mathsf{r2}}\} \wedge \mathsf{id}(v_{\mathsf{a}}) & \text{if } A = \{\mathrm{c}(C, l)\} \\ \mathsf{id}(v_{\mathsf{a}}) \wedge v'_{\mathsf{c}} = \bullet \wedge & \\ \qquad \{v'_{\mathsf{r1}}, v'_{\mathsf{r2}}\} = \{v_{\mathsf{r1}}, v_{\mathsf{r2}}, v_{\mathsf{c}}\} \setminus \{\bullet\} & \text{if } A = \{\mathrm{r}(C)\} \\ \{v'_{\mathsf{r1}}, v'_{\mathsf{r2}}\} = \{l, \bullet\} \wedge \mathsf{id}(v_{\mathsf{c}}, v_{\mathsf{a}}) & \text{if } A = \{\mathrm{wd\ r}(C, l)\} \\ v'_{\mathsf{c}} = \bullet \wedge \{v'_{\mathsf{r1}}, v'_{\mathsf{r2}}\} = \{v_{\mathsf{r1}}, v_{\mathsf{r2}}\} \wedge \mathsf{id}(v_{\mathsf{a}}) & \text{if } A = \{\mathrm{wd\ c}(C)\} \\ v'_{\mathsf{a}} = \alpha \wedge \{v'_{\mathsf{r1}}, v'_{\mathsf{r2}}\} = \{v_{\mathsf{r1}}, v_{\mathsf{r2}}\} \wedge \mathsf{id}(v_{\mathsf{c}}) & \text{if } A = \{\mathrm{a}(C, \alpha)\} \\ \{v'_{\mathsf{r1}}, v'_{\mathsf{r2}}\} = \{v_{\mathsf{r1}}, v_{\mathsf{r2}}\} \wedge \mathsf{id}(v_{\mathsf{c}}, v_{\mathsf{a}}) & \text{if } A = \emptyset \end{cases}$$

$$\mathsf{tr}''_{\mathsf{a}}(\theta, p, p', g) \equiv v'_{\mathsf{p}} = v_{\mathsf{p}} + v_{\mathsf{s}} * \theta + \frac{1}{2} v_{\mathsf{a}} * \theta^2 \wedge$$
$$v'_{\mathsf{s}} = v_{\mathsf{a}} * \theta + v_{\mathsf{s}} \wedge v'_{\Omega} = \frac{(v_{\mathsf{s}} + v_{\mathsf{a}} * \theta)^2}{2 * \mathsf{dec}_{\mathsf{max}}(C)} + \mathsf{length}(C)$$

For two proper FORCF data tuples $p = (DS, l_1, l_2, \beta_1, \beta_2, f, \mathbb{D}, \mathcal{S}, D_E)$ and $p' = (DS', l'_1, l'_2, \beta'_1, \beta'_2, f', \mathbb{D}', \mathcal{S}', D'_E)$ with $DS = DS'$, $f = f'$, $\mathbb{D} = \mathbb{D}'$ and $D_E = D'_E$ the formula $\mathsf{tr}'_{\mathsf{a}}(A, p, p', g)$ remains unchanged and $\mathsf{tr}'_{\mathsf{a}}(\theta, p, p', g)$ is extended with

$$\beta'_1 = \beta_1 + \Delta_E \wedge \beta'_2 = \beta_2 + \Delta_E$$

where Δ_E is an abbreviation for $(\mathcal{S}'(D'_E)(v'_{\mathsf{p}}) - \mathcal{S}(D_E)(v_{\mathsf{p}}))$. Now, performing an action after a delay is defined as

$$\mathsf{tr}_{\mathsf{a}}(\theta, A, p, p', g) \equiv \mathsf{tr}'_{\mathsf{a}}(A, p, p', g) \wedge \mathsf{tr}''_{\mathsf{a}}(\theta, p, p', g) \ . \qquad \triangle$$

We represent MLSLS transition sequences with FORCF variables. To this end, we introduce sequences of data tuples.

Definition 5.2.3 (Sequences of FORCF Data Tuples)**.** We call a sequence $\pi = \langle p_1, \dots, p_n \rangle$ with $p_i = (DS_i, l_i, l'_i, \beta_i, \beta'_i, f_i, \mathbb{D}_i, \mathcal{S}_i, D_{E,i})$ and $i \in \{1, \dots, n\}$ a *sequence of proper FORCF data tuples.* For such sequences we require for $j \in \{1, \dots, n-1\}$ that $DS_i = DS_{i+1}$, $l_i = l_{i+1}$, $l'_i = l'_{i+1}$, $f_i = f_{i+1}$, $\mathbb{D}_i = \mathbb{D}_{i+1}$ and $D_{E,i} = D_{E,i+1}$, i.e. that the car identifier variables and the lane values in the sequence are equal. Further, for simplicity we require $|\mathbb{D}_i| = 1$. Similarly, a *sequence of simple FORCF data tuples* is a sequence $\pi = \langle p_1, \dots, p_n \rangle$ with $p_i = (DS_i, f_i, \mathbb{D}_i, \mathcal{S}_i)$ and $i \in \{1, \dots, n\}$ such that for $j \in \{1, \dots, n-1\}$ we have $DS_i = DS_{i+1}$ $f_i = f_{i+1}$ and $\mathbb{D}_i = \mathbb{D}_{i+1}$. △

We extend our notion of sanity from static representations of MLSLS models to sequences of MLSLS transitions for a single car. Let π be a sequence of FORCF data tuples representing a single car, M an MLSLS model, $g : \mathsf{DVar} \to \mathbb{I}$ a car identifier variable assignment, and ϱ a timed word. Then we call the tuple (π, g, M, ϱ) *sane* if for all $i \in \mathsf{dom}\,\pi$ the tuple $(\pi(i), M, g)$ satisfies the static sanity condition (cf. Definition 3.4.2), $\#\pi = \#\varrho + 1$ and $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M}) \times \mathbb{T})^+$.

Below we define a formula that represents an MLSLS transition sequence $\varrho(M)$ of a single car. For this we use sequences of FORCF data tuples π and sequences of real FORCF terms ζ. Then, $\zeta(i)$ represents the point in time when D performs the i-th action and $\pi(i)$ represents the state after performing that action and moving according to the speed and acceleration of the car until that time. We introduce constraints for the terms in ζ later. In general we do not allow the empty sequence as timed word. However, here it is convenient to let ϱ be either a normal timed word or the empty sequence. Note that we use the FOLRA representation of an MLSLS model. As FOLRA is a fragment of FORCF this is no problem. Note that we keep the real-valued terms containing multiplication outside of the FOLRA formula.

Definition 5.2.4 (Transforming Transition Sequences)**.** Let M be an MLSLS model, $\varrho = (w, \tau)$ a timed word or the empty sequence, g a car identifier variable assignment, π a sequence of FORCF data tuples such that (π, g, M, ϱ) is sane and $\zeta \in \mathsf{Seq}\,\mathsf{RTerm}$ a sequence of real terms with $\#\zeta = \#\pi$. We define

$$\mathsf{tr_w}(\varrho, M, \zeta, \pi, g) \equiv \mathsf{tr^{lra}_{m,E}}(\pi(1), M) \land \bigwedge_{i \in \mathsf{dom}\,\varrho} \mathsf{tr_a}(\zeta(i+1) - \zeta(i), w(i), \pi(i), \pi(i+1), g) \ . \qquad \triangle$$

Now we define a transformation for a transition sequence until θ, where θ is a FORCF term. To achieve this we ignore all changes after time θ, and let the variables just keep their values. For this we perform a case distinction on the value of θ: if θ is greater equal than the latest time stamp we simply encode the complete word. Otherwise, we encode the timed word until the latest time stamp smaller than θ and then let time pass without executing actions until θ. Further, for a sequence of FORCF data tuples π and a car identifier variable assignment g we define an abbreviation that freezes the state of the car as

$$F(\pi, g) \equiv \bigwedge_{i \in \{1,\dots,\#\pi-1\}} \mathsf{tr_a}(0, \emptyset, \pi(i), \pi(i+1), g) \ .$$

As for the static case, we use E to denote an Elementary (or base) version of a function that we extend later.

Definition 5.2.5 (Transforming Timed-Bounded Transition Prefixes). Let M be an MLSLS model, $\varrho = (w, \tau)$ a timed word, g a car identifier variable assignment and π a sequence of FORCF data tuples such that (π, g, M, ϱ) is sane. Further, let $\zeta \in \mathsf{Seq\,RTerm}$ be a sequence of real terms with $\#\zeta = \#\pi$ and let θ be a FORCF term. Then we define

$$\begin{aligned} \mathsf{tr}^{\mathsf{E}}_{\mathsf{U}}(\varrho, M, \theta, \zeta, \pi, g) \equiv{} & (\theta \geq \zeta(n) \implies \mathsf{tr_w}(\varrho, M, \zeta, \pi, g)) \land \\ \bigwedge_{i \in \{1,\dots,\#\varrho\}} & \big(\zeta(i) \leq \theta < \zeta(i+1) \implies \mathsf{tr_w}(\varrho[..i-1], M, \zeta[..i], \pi[..i], g) \land \\ & \mathsf{tr_a}(\theta - \zeta(i), \emptyset, \pi(i), \pi(i+1), g) \land F(\pi[i+1..], g)\big) \ . \qquad \triangle \end{aligned}$$

For the definition above we point out that $\varrho[..i-1]$ may be a timed word or the empty sequence. In the left-hand side of each implication we check the value of θ. In the right-hand side we transform all actions until θ. Then we let the traffic evolve from $\zeta(i)$ until θ and discard all actions after θ. Note that in the definition above always exactly one of the implications is satisfied.

So far we always considered a single car. We extend the definition above to multiple cars. We use sequences of FORCF data tuples to encode sequences of MLSLS transitions for multiple cars. That is, for the set of all possibly simple FORCF data tuples $\mathrm{T}^{\mathsf{rcf}}$ we use the functions $\Pi : \mathsf{DVar} \to \mathsf{Seq}\,\mathrm{T}^{\mathsf{rcf}}$, $Z : \mathsf{DVar} \to \mathsf{Seq\,RTerm}$ that give us for a car identifier variable a sequence of FORCF data tuples, respectively a sequence of terms. We will call Π an *assignment of sequences of data tuples* and Z an *assignment of sequences of* *real-valued terms.* Then, $Z(D)(i)$ represents the point in time when D performs the i-th action and $\Pi(D)(i)$ represents the state after performing that action and moving according to the speed and acceleration of the car until that time. For simplicity, we assume that the set $\mathbb{D}$ in each data tuple in the sequence $\Pi(D)$ contains the single element D. Similar to FORCF sequences for single cars we need conditions for these extended variable structures, which we again call sanity conditions.

Definition 5.2.6 (Sanity of FORCF Sequences)**.** Let M be a possibly simple MLSLS model, g a car identifier variable assignment, $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M}) \times \mathbb{T})^+$ a timed word, an assignment of sequences of real-valued terms $Z : \mathsf{DVar} \to \mathsf{Seq}\,\mathsf{RTerm}$ and an assignment of sequences of data tuples $\Pi : \mathsf{DVar} \to \mathsf{Seq}\,\mathrm{T}^{\mathsf{rcf}}$. Then we call (Π, g, M, ϱ) *sane* if

- $\mathsf{dom}\, g = \mathsf{dom}\, Z = \mathsf{dom}\, \Pi$,
- $\mathsf{ran}\, g = \mathsf{cars}\, M$,
- for all different $D, D' \in \mathsf{dom}\, g$ the sequences $\Pi(D)$ and $\Pi(D')$ do not share any variables and
- for all $D \in \mathsf{dom}\, g$ the tuple $(\Pi(D), g \rhd \{D\}, M \boxed{\upharpoonright} \{g(D)\}, \varrho_{g(D)})$ is sane, where $\varrho_{g(D)} \approx (\varrho \upharpoonright \mathsf{Act}_{g(D)})$ and $\varrho_{g(D)}$ contains no interior empty sets of action. $\triangle$

We instantiate the data structures for a small example and show sanity of our instantiation.

Example 5.2.7. Consider the timed word ϱ and the model M with

$$\varrho = \varrho_0 \upharpoonright \{E, C_1\} = \langle(\{\mathrm{wd\ r}(E,3)\}, 1\mathrm{s}), (\emptyset, 1.1\mathrm{s}), (\{\mathrm{c}(E,2)\}, 5.1\mathrm{s}), (\emptyset, 6.1\mathrm{s})\rangle \tag{5.1}$$

$$M = M_0 \boxed{\upharpoonright} \{E, C_1\} \tag{5.2}$$

where ϱ_0 and M_0 are taken from Example 5.1.13. Let $\mathbb{D} = \{D_E, D_1\}$ and $g = \{D_E \mapsto E, D_1 \mapsto C_1\}$. Further, let

$$\varrho_E = \langle(\{\mathrm{wd\ r}(E,3)\}, 1\mathrm{s}), (\{\mathrm{c}(E,2)\}, 5.1\mathrm{s}), (\emptyset, 6.1\mathrm{s})\rangle\ ,$$
$$\varrho_{C_1} = \langle(\emptyset, 6.1\mathrm{s})\rangle\ .$$

Then $\varrho_E \approx \varrho \restriction \{E\}$ and $\varrho_{C_1} \approx \varrho \restriction \{C_1\}$, i.e. the timed words are equal to their counterparts defined with the restriction operator, except that they do not contain any interior empty sets of actions. Next, we choose $\Pi(D_E) = \langle p_1^{D_E}, p_2^{D_E}, p_3^{D_E}, p_4^{D_E}\rangle$, where for $i \in \{1, \ldots, 4\}$ the data tuples are $p_i^{D_E} = (\{D_E\}, 1, 3, x_i, x_i', \{\mathsf{ego} \mapsto D_E\}, \{D_E\}, \mathcal{S}_i^{D_E}, D_E)$ and each $\mathcal{S}_i^{D_E}$ contains seven unique variables $D_E.v_{\mathsf{c}}^i$, $D_E.v_{\mathsf{r1}}^i$, $D_E.v_{\mathsf{r2}}^i$, $D_E.v_{\mathsf{s}}^i$, $D_E.v_{\mathsf{a}}^i$, $D_E.v_{\mathsf{p}}^i$ and $D_E.v_{\Omega}^i$. Similarly, we define $\Pi(D_1) = \langle p_1^{D_1}, p_2^{D_1}\rangle$ with $p_j^{D_1} = (\{D_1\}, \{c_1 \mapsto D_1\}, \{D_1\}, \mathcal{S}_j^{D_1})$, where $j \in \{1, 2\}$ and each $\mathcal{S}_j^{D_1}$ again has seven unique variables.

We see that the tuple (M, g, Π, ϱ) is sane because the domains of g and Π are equal, $\mathsf{ran}\, g$ and $\mathsf{cars}\, M$ are equal, $\Pi(D_E)$ and $\Pi(D_1)$ do not share any variables and for $D \in \{D_E, D_1\}$ the length of $\Pi(D)$ is equal to $\#\varrho_{g(D)} + 1$. △

Now we define how to encode sequences of MLSLS transitions for multiple cars. As a preparation for the next chapter we restrict a timed word for multiple cars first to a single car. Then we remove all internal empty sets of actions and only then we use $\mathsf{tr}_{\mathsf{U}}^{\mathsf{E}}$ from our earlier definition.

Definition 5.2.8. Let M be an MLSLS model, ϱ a timed word, g a car identifier variable assignment and $\Pi : \mathsf{DVar} \to \mathsf{Seq}\, \mathrm{T}^{\mathsf{rcf}}$ an assignment of sequences of data tuples such that (Π, g, M, ϱ) is sane. Further, let $Z : \mathsf{dom}\, g \to \mathsf{Seq}\, \mathsf{RTerm}$ be an assignment of sequences of real-valued terms with $\#Z(D) = \#\Pi(D)$ and let θ be a FORCF term. Then we define

$$\mathsf{tr}_{\mathsf{U}}(\varrho, M, \theta, Z, \Pi, g) \equiv$$
$$\begin{cases} \mathsf{tr}_{\mathsf{U}}^{\mathsf{E}}(\varrho_{g(D)}, M \boxdot g(D), \theta, Z(D), \Pi(D)) \wedge & \\ \quad \mathsf{tr}_{\mathsf{U}}(\varrho \setminus g(D), M \boxminus g(D), \theta, \{D\} \lhd Z, \{D\} \lhd \Pi, \{D\} \lhd g) & \text{if } \mathsf{dom}\, g \neq \emptyset \\ \mathsf{true} & \text{otherwise} \end{cases}$$

where $D \in \mathsf{dom}\, g$ and $\varrho_{g(D)} \approx (\varrho \restriction \mathsf{Act}_{g(D)})$ and $\varrho_{g(D)}$ contains no interior empty sets of action. △

Now we can define our transformation to check global properties. However, before we present our transformation, we show for reasons of formatting an example of our transformation on the next two pages. The definition of the transformation is on Page 170.

Example 5.2.9. We consider the MLSLS formula $\Box\,\mathsf{npc} \equiv \Box \forall c, c'.\,(c \neq c' \implies \neg\langle(\mathsf{re}(c) \vee \mathsf{cl}(c)) \wedge \mathsf{cl}(c')\rangle)$, expressing “always no potential collision”. This formula states that all cars never have a claim that overlaps with the claim or reservation of another car. The FORCF formula $\mathsf{tr}_\Box\,(\varrho, M, \mathsf{npc})$ checks if this property holds throughout the transition sequence $\varrho(M)$, where M and ϱ are as defined in Example 5.2.7.

We show $\mathsf{tr}_\Box\,(\varrho, M, \mathsf{npc})$ in expanded form on the next page. We reuse the data structures and variable names from Example 5.2.7. For simplicity we assume for all cars $C \in \mathbb{I}$ that $\mathsf{dec}_{\mathsf{max}}(C) = 12$ and $\mathsf{length}(C) = 3$. Further,

$$\begin{aligned}
p_{\mathsf{f}} &= (\{D_E, D_{C_1}\}, 1, 3, x_{\mathsf{l}}^4, x_{\mathsf{r}}^4, \{\mathsf{ego} \mapsto D_E\}, \{D_E, D_{C_1}\}, \mathcal{S}_4^{D_E} \uplus \mathcal{S}_2^{D_{C_1}}, D_E) \ , \\
\Pi(D_E)(4) &= (\{D_E\}, 1, 3, x_{\mathsf{l}}^4, x_{\mathsf{r}}^4, \{\mathsf{ego} \mapsto D_E\}, \{D_E\}, \mathcal{S}_4^{D_E}, D_E)) \ , \\
\Pi(D_{C_1})(2) &= (\{D_{C_1}\}, f_\emptyset, \{D_{C_1}\}, \mathcal{S}_2^{D_{C_1}})) \ ,
\end{aligned}$$

where $f_\emptyset$ denotes the empty function. Let us denote exponentiation with $**$. Then we use for $D \in \{D_E, D_{C_1}\}$ the abbreviations $\xi_\Omega^D(i, \theta) \equiv \frac{(D.v_{\mathsf{s}}^i + D.v_{\mathsf{a}}^i * \theta) ** 2}{2 * 12} + 3$, which represents the value of the sensor function after waiting θ time after the i-th step, and $\xi_{\mathrm{pos}}^D(i, \theta) \equiv D.v_{\mathsf{p}}^i + D.v_{\mathsf{s}}^i * \theta + \frac{1}{2} * D.v_{\mathsf{a}}^i * (\theta ** 2)$ to represent the position after waiting θ time after the i-th step. Additionally, we use $\Delta^i \equiv D_E.v_{\mathsf{p}}^i - D_E.v_{\mathsf{p}}^{i-1}$ for the distance covered by the ego-car.

In Lines 2 to 11 we see the constraints for the car E. In Lines 3 to 7 we show the unfolded constraints for the case that the variable z takes a value from the interval $[1, 5.1)$. The other cases for E are not unfolded. Line 2 encodes the case that the freeze time is before E executes its first action and Lines 8 to 9 (resp. Lines 10 to 11) encode the case that the freeze time is after the second action but before the end of the timed word (resp. after the end of the timed word). Similarly, we show the unfolded constraints for car C_1 for the case where z takes a value smaller (resp. greater equal) than the point in time where the timed word ends in Lines 12 to 14 (resp. Line 15).

We show that $\mathsf{tr}_\Box\,(\varrho, M, \mathsf{npc})$ is not valid. For this consider an assignment h with $h(z) = 4$. For the relevant variables we show their resulting values assigned by h:

$$\begin{aligned}
D_E.v_{\mathsf{p}}^4, D_E.v_\Omega^4, D_E.v_{\mathsf{c}}^4, \{D_E.v_{\mathsf{r1}}^4, D_E.v_{\mathsf{r2}}^4\} &\mapsto 88, 16.5, \bullet, \{3, \bullet\} \\
D_{C_1}.v_{\mathsf{p}}^2, D_{C_1}.v_\Omega^2, D_{C_1}.v_{\mathsf{c}}^2, \{D_{C_1}.v_{\mathsf{r1}}^2, D_{C_1}.v_{\mathsf{r2}}^2\} &\mapsto 84, 4.5, 3, \{2, \bullet\}
\end{aligned}$$

We point out that the sensor function, speed and acceleration remain unchanged in this example. Furthermore, $h(x_{\mathsf{l}}^4) = 72$, $h(x_{\mathsf{r}}^4) = 162$. We see that $\mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \mathsf{npc})$ evaluates to false for this valuation. To see this we can choose D_E for the car variable c and D_{C_1} for the variable c'. Then the claim of c' overlaps with the reservation of c on lane 3 on the interval $[88, 88.5]$, which is inside the view. △

```
1  $\forall z.\,\big($
       /* Constraints for car $E$ */
2      $\big((0 \le z < 1 \implies \mathsf{tr}_{\mathsf{w}}(\langle\rangle, M_E, \langle 0\rangle, \langle p_1^E\rangle, g_E) \wedge \mathsf{tr}_{\mathsf{a}}(z-0, \emptyset, p_1^E, p_2^E, g_E) \wedge F(\langle p_2^E, p_3^E, p_4^E\rangle, g_E))$
3      $\wedge\, 1 \le z < 5.1 \implies$
           /* $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_1^E, M_E)$ within $\mathsf{tr}_{\mathsf{w}}(\langle(\{\mathrm{wd}\ \mathrm{r}(E,3)\},1)\rangle, M_E, \langle 0,1\rangle, \langle p_1^E, p_2^E\rangle, g_E)$ */
4          $D_E.v_{\mathsf{c}}^1 = \bullet \wedge \{D_E.v_{\mathsf{r1}}^1, D_E.v_{\mathsf{r2}}^1\} = \{2,3\} \wedge D_E.v_{\mathsf{p}}^1 = 16 \wedge D_E.v_\Omega^1 = 30 \wedge D_E.v_{\mathsf{s}}^1 = 18 \wedge D_E.v_{\mathsf{a}}^1 = 0 \wedge x_{\mathrm{l}}^1 = 0 \wedge x_{\mathrm{r}}^1 = 90$

           /* $\mathsf{tr}_{\mathsf{a}}(1-0, \{\mathrm{wd}\ \mathrm{r}(E,3)\}, p_1^E, p_2^E, g_E)$ within $\mathsf{tr}_{\mathsf{w}}(\langle(\{\mathrm{wd}\ \mathrm{r}(E,3)\},1)\rangle, M_E, \langle 0,1\rangle, \langle p_1^E, p_2^E\rangle, g_E)$ */
5          $\wedge\{D_E.v_{\mathsf{r1}}^2, D_E.v_{\mathsf{r2}}^2\} = \{\bullet, 3\} \wedge \mathsf{id}(D_E.v_{\mathsf{c}}^1, D_E.v_{\mathsf{a}}^1) \wedge D_E.v_{\mathsf{p}}^2 = \xi_{\mathrm{pos}}^{D_E}(1, 1-0) \wedge D_E.v_{\mathsf{s}}^2 =$
             $D_E.v_{\mathsf{s}}^1 + D_E.v_{\mathsf{a}}^1 * 1 \wedge D_E.v_\Omega^2 = \xi_\Omega^{D_E}(1, 1-0) \wedge x_{\mathrm{l}}^2 = x_{\mathrm{l}}^1 + \Delta_1^E \wedge x_{\mathrm{r}}^2 = x_{\mathrm{r}}^1 + \Delta_1^E$
           /* $\mathsf{tr}_{\mathsf{a}}(z-1, \emptyset, p_2^E, p_3^E, g_E)$ */
6          $\wedge\{D_E.v_{\mathsf{r1}}^3, D_E.v_{\mathsf{r2}}^3\} = \{D_E.v_{\mathsf{r1}}^2, D_E.v_{\mathsf{r2}}^2\} \wedge \mathsf{id}(D_E.v_{\mathsf{c}}^2, D_E.v_{\mathsf{a}}^2) \wedge D_E.v_{\mathsf{p}}^3 = \xi_{\mathrm{pos}}^{D_E}(2, z-1) \wedge D_E.v_{\mathsf{s}}^3 =$
             $D_E.v_{\mathsf{s}}^2 + D_E.v_{\mathsf{a}}^2 * (z-1) \wedge D_E.v_\Omega^3 = \xi_\Omega^{D_E}(2, z-1) \wedge x_{\mathrm{l}}^3 = x_{\mathrm{l}}^2 + \Delta_2^E \wedge x_{\mathrm{r}}^3 = x_{\mathrm{r}}^2 + \Delta_2^E$
           /* $F(\langle p_3^E, p_4^E\rangle, g_E)$ */
7          $\wedge\{D_E.v_{\mathsf{r1}}^4, D_E.v_{\mathsf{r2}}^4\} = \{D_E.v_{\mathsf{r1}}^3, D_E.v_{\mathsf{r2}}^3\} \wedge \mathsf{id}(D_E.v_{\mathsf{c}}^3, D_E.v_{\mathsf{a}}^3) \wedge D_E.v_{\mathsf{p}}^4 = \xi_{\mathrm{pos}}^{D_E}(3, 0) \wedge D_E.v_{\mathsf{s}}^4 =$
             $D_E.v_{\mathsf{s}}^3 + D_E.v_{\mathsf{a}}^3 * 0 \wedge D_E.v_\Omega^4 = \xi_\Omega^{D_E}(3, 0) \wedge x_{\mathrm{l}}^4 = x_{\mathrm{l}}^3 + \Delta_3^E \wedge x_{\mathrm{r}}^4 = x_{\mathrm{r}}^3 + \Delta_3^E$
8      $\wedge\, 5.1 \le z < 6.1 \implies$
9          $\mathsf{tr}_{\mathsf{w}}(\langle(\{\mathrm{wd}\ \mathrm{r}(E,3)\},1), (\{\mathrm{c}(E,2)\},5.1)\rangle, M_E, \langle 0,1,5.1\rangle, \langle p_1^E, p_2^E, p_3^E\rangle, g_E) \wedge \mathsf{tr}_{\mathsf{a}}(z-5.1, \emptyset, p_3^E, p_4^E, g_E) \wedge$
             $F(\langle p_4^E\rangle, g_E))$
10     $\wedge\, (6.1 \le z \implies$
11         $\mathsf{tr}_{\mathsf{w}}(\varrho_E, M_E, 0 :: \tau_E, \pi^E, g_E) \wedge \mathsf{tr}_{\mathsf{a}}(z-0, \emptyset, p_1^E, p_2^E, g_E) \wedge F(\langle p_2^E, p_3^E, p_4^E\rangle, g_E))$
       /* Constraints for car $C_1$ */
12     $\wedge\, 0 \le z < 6.1 \implies$
           /* $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_1^{C_1}, M_{C_1})$ within $\mathsf{tr}_{\mathsf{w}}(\langle\rangle, M_{C_1}, \langle 0\rangle, \langle p_1^{C_1}\rangle, g_{C_1})$ */
13         $D_{C_1}.v_{\mathsf{c}}^1 = 3 \wedge \{D_{C_1}.v_{\mathsf{r1}}^1, D_{C_1}.v_{\mathsf{r2}}^1\} = \{2, \bullet\} \wedge D_{C_1}.v_{\mathsf{p}}^1 = 60 \wedge D_{C_1}.v_\Omega^1 = 6 \wedge D_{C_1}.v_{\mathsf{s}}^1 = 6 \wedge D_{C_1}.v_{\mathsf{a}}^1 = 0$
           /* $\mathsf{tr}_{\mathsf{a}}(z-0, \emptyset, p_1^{C_1}, p_2^{C_1}, g_{C_1})$ */
14         $\wedge\, \{D_{C_1}.v_{\mathsf{r1}}^1, D_{C_1}.v_{\mathsf{r2}}^1\} = \{D_{C_1}.v_{\mathsf{r1}}^2, D_{C_1}.v_{\mathsf{r2}}^2\} \wedge \mathsf{id}(D_{C_1}.v_{\mathsf{c}}^1, D_{C_1}.v_{\mathsf{a}}^1) \wedge D_{C_1}.v_{\mathsf{p}}^2 = \xi_{\mathrm{pos}}^{D_E}(1, z-0) \wedge D_{C_1}.v_{\mathsf{s}}^2 =$
             $D_{C_1}.v_{\mathsf{s}}^1 + D_{C_1}.v_{\mathsf{a}}^1 * (z-0) \wedge D_{C_1}.v_\Omega^2 = \xi_\Omega^{D_E}(1, z-0)$
15     $\wedge\, (6.1 \le z \implies \mathsf{tr}_{\mathsf{w}}(\varrho_{C_1}, M_{C_1}, 0 :: \tau_{C_1}, \pi^{C_1}, g_{C_1})))$
16     $\implies \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathrm{f}}, \mathrm{npc})\big)$
```

Figure 5.2: Unfolding of $\mathsf{tr}_\Box(\varrho, M, \mathrm{npc})$ for Example 5.2.9.

The intuition of the transformation is that it checks if we can stop the evolution of $\varrho(M)$ at all points in time such that the transformed MLSLS formula is satisfied.

Definition 5.2.10 (Transforming Global Properties)**.** Given a proper finite model M and a timed word $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\, M}) \times \mathbb{T})^{+}$ let $\mathbb{D}$ and g be a car identifier variable assignment such that $|\mathbb{D}| = |\mathsf{cars}\, M|$ and $g : \mathbb{D} \to \mathsf{cars}\, M$ is bijective. Further, let $Z : \mathbb{D} \to \mathsf{Seq}\,\mathsf{RTerm}$ be such that for all $D \in \mathbb{D}$ we have $Z(D) = \langle 0 \rangle \cdot \tau_{g(D)}$, where $\varrho_{g(D)} \approx (\varrho \restriction \mathsf{Act}_{g(D)})$ and $\varrho_{g(D)} = (w_{g(D)}, \tau_{g(D)})$ contains no interior empty sets of actions. Then we can easily create an assignment of sequences of data tuples $\Pi : \mathbb{D} \to \mathsf{Seq}\, \mathrm{T}^{\mathsf{rcf}}$ such that (Π, g, M, ϱ) is sane and let p_{f} be the combination of all data tuples $\mathsf{last}\, \Pi(D)$ with $D \in \mathbb{D}$. Then for an MLSLS formula ϕ we define

$$\mathsf{tr}_{\Box}(\varrho, M, \phi) \equiv \forall z.\, (\mathsf{tr}_{\mathsf{U}}(\varrho, M, z, Z, \Pi, g) \implies \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p_{\mathsf{f}}, \phi)) \ . \qquad \triangle$$

5.3 Correctness of our Encoding

In this section we show that our construction to check if an MLSLS formula holds globally in a transition sequence is correct. We first prove several lemmas which state that for known values (rather than terms involving variables) our formulas are equivalent to a formula that directly encodes the behaviour as a conjunction of equality constraints. From such a conjunction of equality constraints it is then clear that it is satisfiable and we can extract a unique satisfying assignment.

Lemma 5.3.1. *Let C be a car identifier, $t \in \mathbb{T}, A \in \mathcal{P}(\mathsf{Act}_C)$ with $|A| \leq 1$ and M_1, M, M_2 be possibly simple MLSLS models with car domain $\{C\}$ and $M_1 \xrightarrow{t} M \xrightarrow{A} M_2$. Further, let p_1, p_2 be two data tuples and let g be a car identifier variable assignment such that $(\langle p_1, p_2 \rangle, g, M_1, \langle (A, t) \rangle)$ is sane. Then the FORCF formula*

$$\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_1, M_1) \wedge \mathsf{tr}_{\mathsf{a}}(A, t, p_1, p_2, g) \iff \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_1, M_1) \wedge \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_2, M_2)$$

is valid.

Proof. We start with the case that the MLSLS models are proper. For the index $i \in \{1, 2\}$ let $M_i = (CS_i, TS_i, \Omega_i, V_i, \nu_i)$ with $TS_i = (\mathsf{res}_i, \mathsf{clm}_i, \mathsf{pos}_i, \mathsf{spd}_i, \mathsf{acc}_i)$ and $p_i = (DS_i, l_i, l'_i, \beta_i, \beta'_i, f_i, \mathbb{D}_i, \mathcal{S}_i, D_{E,i})$.

The formula $\mathsf{tr_a}(A, t, p_1, p_2, g)$ constraints the variables in p_2 relative to the variables in p_1. We point out that $\mathsf{tr_a}$ and $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}$ both result in conjunctions of disjunctions (the disjunctions are hidden in the set equalities). Let us consider the position first. The formula $\mathsf{tr_a}(A, t, p_1, p_2, g)$ has the subformula $v^2_{\mathsf{p}} = v^1_{\mathsf{p}} + v^1_{\mathsf{s}} * t + \frac{1}{2} v^1_{\mathsf{a}} * t * t$ and $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_2, M_2)$ has the subformula $v^2_{\mathsf{p}} = \mathsf{pos}_2(C)$. Both formulas have no other constraints on the variable v^2_{p}. Thus, we have to show equality of these two constraints. From $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_1, M_1)$ we know that the variables in p_1 are constrained to take the values in M_1. We know from the definition of MLSLS transitions that $\mathsf{pos}_2(C) = \mathsf{pos}_1(C) + \mathsf{spd}_1(C) * t + \frac{1}{2} \mathsf{acc}_1(C) * t * t$. Hence, by substituting the variables with the values we conclude that one direction of the equality holds and for the other direction we substitute values by variables. For the other variables the reasoning works analogously.

Next we consider the parts of the model affected by discrete actions. For this we do a case distinction on the action in A. We start with $A = \{\mathrm{r}(C)\}$. As we assume that all actions change the state (cf. Assumption 5.1.12), we know that C has a claim that can be turned into a reservation, i.e. $\mathsf{clm}(C) \neq \{\bullet\}$. Now, $\mathsf{tr_a}(\{\mathrm{r}(C)\}, t, p_1, p_2, g)$ becomes $\{v^2_{\mathsf{r1}}, v^2_{\mathsf{r2}}\} = \{v^1_{\mathsf{r1}}, v^1_{\mathsf{r2}}, v^1_{\mathsf{c}}\} \wedge v^2_{\mathsf{r1}} \neq \bullet \wedge v^2_{\mathsf{r2}} \neq \bullet \wedge \mathsf{id}(v^1_{\mathsf{a}}) \wedge v^2_{\mathsf{c}} = \bullet$. This formula ensures that the variables $v^2_{\mathsf{r1}}, v^2_{\mathsf{r2}}$ take their old lane value (which may be in v^1_{r1} or in v^1_{r2}) and the lane value in v^1_{c} but not $\bullet$. In $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_2, M_2)$ we have for claims and reservations the constraint $\{v^2_{\mathsf{r1}}, v^2_{\mathsf{r2}}\} = \mathsf{res}_2(C) \wedge v^2_{\mathsf{c}} = \mathsf{clm}_2(C)$, where we know from the transition taken that $\mathsf{clm}_2(C) = \{\bullet\}$. The other cases work similarly.

The case of simple models is analogously. □

We lift the previous lemma to timed words.

Lemma 5.3.2. *Let C be a car identifier, M_0 be a possibly simple MLSLS model with $\mathsf{cars}\, M_0 = \{C\}$, $\varrho = (w, \tau) \in (\mathcal{P}(\mathsf{Act}_C) \times \mathbb{T})^+$ with $\#\varrho = n$, and $\varrho(M_0) = M_0 \xrightarrow{\tau(1)} M_1 \xrightarrow{w(1)} M_2 \ldots \xrightarrow{\tau(n)-\tau(n-1)} M_{2n-1} \xrightarrow{w(n)} M_{2n}$. Further, let $\pi = \langle p_0, \ldots, p_n \rangle$ be a sequence of data tuples such that (π, g, M, ϱ) is sane. Then the FORCF formula*

$$\mathsf{tr_w}(\varrho, M_0, \pi, 0 :: \tau, g) \iff \bigwedge_{i \in \{0,\ldots,n\}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p_i, M_i)$$

is valid

Proof. By induction on the length of the timed word with the help of Lemma 5.3.1. □

With $\mathsf{tr}_{\mathsf{U}}^{\mathsf{E}}$ we defined a transformation that represents an MLSLS transition sequence until a time θ, where θ is a FOLRA term. We prove that for all possible values t for θ, our encoding of until is correct. To achieve this, we show the equivalence of two different encodings of until; an incremental encoding using $\mathsf{tr}_{\mathsf{U}}^{\mathsf{E}}$ that is not dependent on the value of t, and an nonincremental encoding which depends on the value of t.

Lemma 5.3.3. *Let C be a car identifier, M_0 be a possibly simple MLSLS model with $\mathsf{cars}\, M_0 = \{C\}$, $\varrho = (w, \tau) \in (\mathcal{P}(\mathsf{Act}_C) \times \mathbb{T})^+$ with $\#\varrho = n$, $\varrho(M) = M_0 \xrightarrow{\tau(1)} M_1 \xrightarrow{w(1)} M_2 \dots \xrightarrow{\tau(n)-\tau(n-1)} M_{2n-1} \xrightarrow{w(n)} M_{2n}$ and π a sequence of data tuples such that (π, g, M, ϱ) is sane. Then for all $t \in \mathbb{T}$ we have*

$$\mathsf{tr}_{\mathsf{U}}^{\mathsf{E}}(\varrho, M, t, 0 :: \tau, \pi, g)$$
$$\Longleftrightarrow$$
$$\bigwedge_{i\in\{0,\dots,k\}} \mathsf{tr}_{\mathsf{m,E}}^{\mathsf{lra}}(\pi(i+1), M_{2i}) \wedge \bigwedge_{j\in\{k+1,\dots,n\}} \mathsf{tr}_{\mathsf{m,E}}^{\mathsf{lra}}(\pi(j+1), \varrho(M)@t)$$

is valid, where either $k \in \{0, \dots, n-1\}$ is the largest index such that $\tau(k) \le t < \tau(k+1)$ with $\tau(0) = 0$ or $k = n$ and $t \ge \tau(n)$.

Proof. Let $\tau' = 0 :: \tau$. We do a case distinction on whether $t = t_n$ and start by assuming $t \ge \tau(n)$. Now, the second big conjunction of the right side of the equivalence is empty. From Lemma 5.3.2 we know that the remaining conjunction is equivalent to $\mathsf{tr}_{\mathsf{w}}(\varrho, M, \pi, \tau', g)$. Finally, with $t \ge \tau(n)$ it is easy to see that $\mathsf{tr}_{\mathsf{w}}(\varrho, M, \pi, \tau', g)$ is equivalent to $\mathsf{tr}_{\mathsf{U}}^{\mathsf{E}}(\varrho, M, t, \tau', \pi, g)$ because only the first premise of the implications in $\mathsf{tr}_{\mathsf{U}}^{\mathsf{E}}(\varrho, M, t, \tau', \pi, g)$ is satisfied with $t \ge \tau(n)$.

We proceed to consider $t < \tau(n)$. We see that in the second big conjunction the model within $\mathsf{tr}_{\mathsf{m,E}}^{\mathsf{lra}}$ remains unchanged, which means that the constraints on the variables also are equal. We can ensure that the variables retain their values with the formula F (cf. Page 165). Thus, the right side of the equivalence in the lemma is equivalent to

$$\bigwedge_{i\in\{0,\dots,k\}} \mathsf{tr}_{\mathsf{m,E}}^{\mathsf{lra}}(\pi(i+1), M_{2i}) \wedge \mathsf{tr}_{\mathsf{m,E}}^{\mathsf{lra}}(\pi(k+2), \varrho(M)@t) \wedge F(\pi[k+2..], g)\ . \quad (5.3)$$

As $\varrho(M)@t$ is equal to the model we get after letting $t - \tau'(k+1)$ time pass from M_{2k-2} (i.e. $M_{2k-2} \xrightarrow{t-\tau'(k+1)} \varrho(M)@t$, where $+1$ is need due to the leading 0 in τ') Equation (5.3) is equivalent to

$$\bigwedge_{i\in\{0,\dots,k\}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\pi(i+1), M_{2i}) \wedge \\ \mathsf{tr}_{\mathsf{a}}(t-\tau'(k+1), \emptyset, \pi(k+1), \pi(k+2), g) \wedge F(\pi[k+2..], g) \ . \quad (5.4)$$

From Lemma 5.3.2 we know that instead of constraining the variables in $\pi[..k+1]$ by absolute values we can constrain them incrementally. Thus, Equation (5.4) is equivalent to

$$\mathsf{tr}_{\mathsf{w}}(\varrho[..k], M, \tau'[..k+1], \pi[..k+1], g) \\ \wedge \mathsf{tr}_{\mathsf{a}}(t-\tau'(k+1), \emptyset, \pi(k+1), \pi(k+2), g) \wedge F(\pi[k+2..], g) \ , \quad (5.5)$$

where $\varrho[..k]$ might be the empty sequence. We point out that the premise of exactly one implication is satisfied in $\mathsf{tr}^{\mathsf{E}}_{\mathsf{U}}(\varrho, M, t, \tau', \pi, g)$ (the one with the premise $\tau'(k) \le t < \tau'(k+1)$). Thus, equivalence of Equation (5.5) to $\mathsf{tr}^{\mathsf{E}}_{\mathsf{U}}(\varrho, M, t, 0 :: \tau, \pi, g)$ follows. □

Let M be an MLSLS model, ϱ a timed word, θ a real-valued term, Z an assignment of sequences of real-valued terms and Π an assignment of sequences of data tuples. Then, with $\mathsf{tr}_{\mathsf{U}}(\varrho, M, \theta, Z, \Pi, g)$ we represent the transition sequence $\varrho(M)$ (or alternatively $\mathcal{I}(M, \varrho)$) and freeze it at the value of θ. If θ is given as a constant t we can create equivalent constraints by transforming for each car each model in $\mathcal{I}(M, \mathsf{pre}(\varrho, t))$ individually. To state this equivalence as a lemma we introduce an abbreviation. For sane (Π, g, M, ϱ) and $D \in \mathsf{dom}\, g$ let $Z(D) = 0 :: \tau_{g(D)}$, where $\varrho_{g(D)} = (w_{g(D)}, \tau_{g(D)}) \approx \varrho \restriction \{g(D)\}$ and $\varrho_{g(D)}$ has no interior empty sets of actions. Further, for $C \in \mathsf{cars}\, M$ let $\varrho_C(M_C) = M_{C,0} \xrightarrow{\tau_C(1)} M_{C,1} \xrightarrow{w_C(1)} M_{C,2} \dots \xrightarrow{\tau_C(n)-\tau_C(n-1)} M_{C,2n-1} \xrightarrow{w_C(n)} M_{C,2n}$, where $n = \#\varrho_C$. Then we use the abbreviation

$$\psi(\varrho, M, t, \Pi, g) \equiv \bigwedge_{D\in\mathsf{dom}\, g} \Bigg(\bigwedge_{i\in\{0,\dots,k_{g(D)}\}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\Pi(D)(i+1), M_{g(D),2i}) \\ \wedge \bigwedge_{j\in\{k_{g(D)+1},\dots,n_{g(D)}\}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\Pi(D)(j+1), \varrho_{g(D)}(M_{g(D)})@t) \ , \Bigg) \quad (5.6)$$

where $k_{g(D)} \in \{0, \dots, n_{g(D)} - 1\}$ is the largest index such that t is in between the corresponding time stamps of $g(D)$, i.e. $\tau_{g(D)}(k_{g(D)}) \le t < \tau_{g(D)}(k_{g(D)} + 1)$ or $k_{g(D)} = n_{g(D)}$ and $\mathsf{last}\, \tau_{g(D)} \le t$.

We lift the previous lemma to multiple cars. For this we point out that we do not consider interior empty sets of actions. Furthermore, note that our assumption that for each car we have exactly one car identifier variable D is ensured by g being bijective.

Lemma 5.3.4. *Let M be a possibly simple MLSLS model, $\varrho = (w, \tau) \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M}) \times \mathbb{T})^+$, Π an assignment of sequences of data tuples and g a car identifier variable assignment such that (Π, g, M, ϱ) is sane. Further, for $C \in \mathsf{cars}\,M$ let $\varrho_C = (w_C, \tau_C)$ such that ϱ_C is equal to $\varrho \restriction \{C\}$ except that ϱ_C has no interior empty sets of actions, and $Z : \mathsf{dom}\,g \to \mathsf{Seq}\,\mathsf{RTerm}$ with $Z(D) = 0 :: \tau_{g(D)}$. Then for all $t \in \mathbb{T}$ the formula*

$$\mathsf{tr}_{\mathsf{U}}(\varrho, M, t, Z, \Pi, g) \iff \psi(\varrho, M, t, \Pi, g)$$

is valid (cf. Equation (5.6) for ψ).

Proof. Proof by induction on $n = |\,\mathsf{dom}\,g|$ using Lemma 5.3.3. □

In $\psi(\varrho, M, t, \Pi, g)$ we have for each car and each action of a car constraints to represent the state of that car after performing its action. In $\mathsf{tr}_{\Box}(\varrho, M, \phi)$ we use $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p_{\mathsf{f}}, \phi)$ to check the arithmetic representation of ϕ on the state represented in p_{f}. We show that the global state of all cars represented in p_{f} and the representation of each car individually in $\psi(\varrho, M, t, \Pi, g)$ are equivalent. To this end we only consider the constraints $\psi(\varrho, M, t, \Pi, g)$ constraining the variables in p_{f}. We show that constraining the data tuple of each car independently to represent the model at time t using a timed word that only has actions of this car (and no interior empty sets of actions) is equivalent to constraining the data tuples of all cars simultaneously to represent the model at time t using the combined timed words of all cars.

Lemma 5.3.5. *Let (Π, g, M, ϱ) be sane. Then for $t \in \mathbb{T}$ the FORCF formula*

$$\bigwedge_{D \in \mathsf{dom}\,g} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\mathsf{last}\,\Pi(D), \varrho_{g(D)}(M_{g(D)})@t) \iff \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p_{\mathsf{f}}, g, \varrho(M)@t)$$

is valid, where p_{f} is the combination of all last data tuples in Π and $\varrho_{\{g(D)\}} \approx \varrho \restriction \{g(D)\}$ and $\varrho_{\{g(D)\}}$ has no interior empty sets of actions.

Proof. The following five formulas are equivalent:

$$\begin{aligned}
&\bigwedge_{D\in\mathbb{D}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\mathsf{last}\,\Pi(D), \varrho_{g(D)}(M_{g(D)})@t)\\
\iff &\bigwedge_{D\in\mathbb{D}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\mathsf{last}\,\Pi(D), \mathsf{last}\,\mathcal{I}(M_{g(D)}, \mathsf{pre}(\varrho_{g(D)}, t)))\\
\iff &\bigwedge_{D\in\mathbb{D}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\mathsf{last}\,\Pi(D), \mathsf{last}\,\mathcal{I}(M_{g(D)}, \mathsf{pre}(\varrho \restriction \{g(D)\}, t)))\\
\iff &\bigwedge_{D\in\mathbb{D}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\mathsf{last}\,\Pi(D), (\mathsf{last}\,\mathcal{I}(M, \mathsf{pre}(\varrho, t))) \boxed{\restriction} \{g(D)\})\\
\iff &\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p_{\mathsf{f}}, g, \varrho(M)@t)
\end{aligned}$$

In the first step we remove our abbreviations and insert the definition of the @-operator. In the second step we replace $\varrho_{g(D)}$ with $\varrho \restriction \{g(D)\}$. This does not change the model for which we generate constraints because the two timed words are equal up to empty sets of actions, which do not change the MLSLS model. In the third step we pull the restriction operator out. For this we use Lemma 5.1.9 and Lemma 5.1.17. In the last step we use the definition of $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}$, which is a conjunction over all cars in the domain of g. Note that $(p_{\mathsf{f}}, g, \varrho(M)@t)$ is sane because (π, g, M, ϱ) is sane. $\square$

Lemma 5.3.6. *For all finite proper MLSLS models $M = (CS, TS, \Omega, V, \nu)$, all timed words $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M})\times\mathbb{T})^+$ and all MLSLS formulas ϕ with $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}\,\nu$ we have*

$$\mathsf{tr}_{\Box}(\varrho, M, \phi) \textit{ is valid} \quad \text{iff} \quad \varrho(M) \models_{\mathsf{seq}} \Box\,\phi \ .$$

Proof. For $C \in \mathsf{cars}\,M$ let $\varrho_C = (w_C, \tau_C)$ be the timed words such that $\varrho_C \approx \varrho \restriction \{C\}$ and ϱ_C has no interior empty sets of actions. Further, let Z, Π, g be defined as in the definition of $\mathsf{tr}_{\Box}$, i.e. for some set $\mathbb{D} \subseteq \mathsf{DVar}$ with $|\mathbb{D}| = |\,\mathsf{cars}\,M|$, $g : \mathbb{D} \to \mathsf{cars}\,M$ is bijective, (Π, g, M, ϱ) is sane and $Z : \mathbb{D} \to \mathsf{Seq}\,\mathsf{RTerm}$ is a function mapping car identifier variables to sequences of real FORCF terms with $Z(D) = 0 :: \tau_{g(D)}$.

We start with the "if"-direction and proceed by contraposition. This means that we assume that $\mathsf{tr}_{\Box}(\varrho, M, \phi)$ is not valid and derive an argument why $\varrho(M) \not\models_{\mathsf{seq}} \Box\,\phi$ holds. The overall idea is that we transform the incremental (or relative) constraints on the variables in p_{f} into absolute constraints of the form $\mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, \varrho(M)@t)$. Then we conclude that at time t the formula ϕ is not satisfied, which proofs this direction.

As $\mathsf{tr}_{\Box}(\varrho, M, \phi)$ is not valid the negated formula is satisfiable. Hence, we can choose a value t such that there is a satisfying assignment for

$$\mathsf{tr}_{\mathsf{U}}(\varrho, M, t, Z, \Pi, g) \wedge \neg \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \phi)) \ .$$

From Lemma 5.3.4 it follows that the equation above is equivalent to the formula $\psi(\varrho, M, t, \Pi, g) \wedge \neg \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \phi)$. As $\bigwedge_{D \in \mathbb{D}} \mathsf{tr}_{\mathsf{m,E}}^{\mathsf{lra}}(\mathsf{last}\, \Pi(D), \varrho_{g(D)}(M_{g(D)})@t)$ is a subformula of $\psi(\varrho, M, t, \Pi, g)$ we conclude from Lemma 5.3.5 that the formula above implies

$$\mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, \varrho(M)@t) \wedge \neg \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \phi) \ .$$

As $\mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, \varrho(M)@t) \wedge \neg \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \phi)$ is satisfiable we conclude from Lemma 3.4.7 that $\varrho(M)@t \not\models \phi$ holds. This implies $\exists t.\, \varrho(M)@t \models \neg\phi$, which implies $\varrho(M) \not\models_{\mathsf{seq}} \Box\, \phi$.

We continue with the "only if"-direction. We proceed by contraposition and assume that $\varrho(M) \models_{\mathsf{seq}} \Box\, \phi$ does not hold, i.e. that there is a value t such that $\varrho(M)@t \not\models \phi$, which is equivalent to $\varrho(M)@t \models \neg\phi$. Next, we know from Lemma 3.4.6 that the constraints we get by transforming an MLSLS model are satisfiable. Furthermore, by restricting different variables we know that conjunctions of such constraints are satisfiable. Thus, the formula $\psi(\varrho, M, t, \Pi, g)$ is satisfiable.

As $\bigwedge_{D \in \mathbb{D}} \mathsf{tr}_{\mathsf{m,E}}^{\mathsf{lra}}(\mathsf{last}\, \Pi(D), \varrho_{g(D)}(M_{g(D)})@t)$ is a subformula of $\psi(\varrho, M, t, \Pi, g)$ we know from Lemma 5.3.5 that $\psi(\varrho, M, t, \Pi, g) \wedge \mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, \varrho(M)@t)$ is satisfiable. From $\varrho(M)@t \models \neg\phi$, together with Lemma 3.4.7, it follows that $\mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, \varrho(M)@t) \implies \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \neg\phi)$ is valid. That is, any assignment satisfying $\mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, \varrho(M)@t)$ also satisfies $\mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \neg\phi)$. Hence, $\psi(\varrho, M, t, \Pi, g) \wedge \mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, \varrho(M)@t) \wedge \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \neg\phi)$ is satisfiable. As the formula $\psi(\varrho, M, t, \Pi, g)$ is equivalent to $\mathsf{tr}_{\mathsf{U}}(\varrho, M, t, Z, \Pi, g)$ (cf. Lemma 5.3.4) we conclude that the conjunction $\mathsf{tr}_{\mathsf{U}}(\varrho, M, t, Z, \Pi, g) \wedge \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \neg\phi)$ is satisfiable. Finally, this means that $\forall z.\, \mathsf{tr}_{\mathsf{U}}(\varrho, M, z, Z, \Pi, g) \implies \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p_{\mathsf{f}}, \phi)$ is not valid. $\Box$

With the previous lemma and from decidability of FORCF [Tar51], we get the following theorem. For the complexity of FORCF we refer to Lemma 2.4.2.

Theorem 5.3.7. *It is decidable whether an MLSLS formula holds globally in an MLSLS transition sequence.*

5.4 Related Work

In [LPR+95] the authors investigate distributed transition systems, where transitions are labelled by sets of actions, rather than single actions. Further, they develop a modal logic for this model and investigate questions such as satisfiability. Additionally, they show that distributed transition systems are a generalisation of other concurrent models, like event structures [Win86] and Petri nets. In our work we also allow simultaneous actions, as opposed to the interleaving semantics considered in original MLSL [HLO+11]. Our reasoning is that otherwise some of the concepts, for example claims, seem superfluous without simultaneous actions. This has been discussed in [BHL+17]. The labelled transition system we use to define the semantics of these sets of actions is a distributed labelled transition system, as defined in [LPR+95]. However, we only became aware of this towards the end of our work. The reason we define the semantics of simultaneous actions as it is, and not for example using event structures, is that we wanted to stay as close as possible to the original semantics of MLSL. However, we did not investigate the effects of allowing simultaneous actions. This has been considered in [BHL+17].

In [Lin15; LH15] the authors define a spatio-temporal semantics for MLSL, together with a labelled deduction system to reason about formulas of their extension of MLSL. In [Lin15] the author defines constraints for a distance controller and a lane change controller in his logic. The author then proves with this labelled deduction system that if all cars are equipped with controllers satisfying these constraints there will be no collisions.

In [LPN11] the authors prove safety of a controller for cars using a logic called quantified differential dynamic logic ($Qd\mathcal{L}$), designed to model and reason about hybrid systems. Differential dynamic logic ($d\mathcal{L}$) [Pla10a] is an extension of dynamic logic [HKT01] and features hybrid actions and differential equations. Using these hybrid actions one can create a hybrid program α and require that every execution of α should satisfy a formula ϕ. This can be formalised as "$[\alpha]\phi$ is valid". Quantified differential dynamic logic [Pla10b] is an extension of $d\mathcal{L}$ using quantifiers over typed variables to model a evolving system, i.e. a system where participants can dynamically leave and join. However, $Qd\mathcal{L}$ is not tailored towards modelling and reasoning about cars. Thus, it is difficult to validate the model and the property, i.e. to be sure that the model and the property are appropriate.

A traffic sequence chart (TSC) is a formalism to graphically specify formal properties for sets of test scenarios of highly autonomous traffic manoeuvres

[DMP+18]. The test scenarios are given as hybrid input/output automata, and the semantics of a TSC is defined in terms of a multi-sorted real-time logic, similar to metric temporal logic (MTL) [Koy90]. As in $Qd\mathcal{L}$, the different sorts allow for expressing the existence of an object such as a car. In terms of intended use, TSC is very similar to how we use MLSLS in this chapter. However, for TSC questions of decidability or robustness (cf. next chapter) have not been investigated (yet).

In [MN04] the authors develop an offline monitoring algorithm for a fragment of MTL. In their approach they combine Boolean signals of atomic formulas to Boolean signals of complex formulas. For this they propagate future values from the end of the recorded signal backward in time. Additionally, they implement their approach and showed feasibility. An example formula of MTL is $\Box\,(\mathit{water} > 3)$, which specifies that the water level should always be higher than 3 metres. We see that the atomic predicate $\mathit{water} > 3$ is a state expression, i.e. it defines properties about the current state of the system. This is the approach we have taken in our work. For an MLSLS formula ϕ, the spatio-temporal formula $\Box\,\phi$ evaluates ϕ at all points in time. In [BLS11] the authors define an online monitoring algorithm for a timed linear temporal logic, which is a real-time extension of LTL. However, this logic is used to measure temporal distances between events, while our focus of interest are spatial properties of the current traffic configuration.

There are few spatio-temporal logics allowing quantitative statements. The closest is Shape Calculus (SC) [Sch04]. With SC we can represent MLSLS models as long as we restrict ourselves to a bounded number of cars. Then we require two spatial dimensions (one discrete and one dense) and another dense dimension for time.

6 Monitoring of Spatio-Temporal Properties with Imprecise Information

In the previous chapter we defined what it means for an MLSL formula to hold globally (in the temporal sense) in a transition sequence. However, in the previous chapter we assumed spatio-temporal data to be known precisely. In this chapter first we weaken this assumption and allow small errors in spatial and temporal information. Then we extend our procedure from Chapter 5 which checks whether an MLSLS formula holds throughout an MLSLS transition sequence, to also consider imprecise information. This chapter is based on the work published in [Ody17].

6.1 Monitoring Global Properties with Imprecise Information

In this section we extend our transformation to check whether an MLSLS formula holds globally in a transition sequence with a temporal robustness of ϵ and a spatial robustness of δ. This allows us to check if, e.g., a behaviour given as a transition sequence is *barely safe* or if it is *robustly safe*.

Here, we consider errors in positional data and imprecisions of when reservations and claims are set and withdrawn. Similarity on timed words has originally been defined in [GHJ97]. However, usually the requirement is imposed that the order of events is equal in similar words. For distributed systems this requirement seems too strong. Here, we weaken this requirement and allow the order of independent actions to change in similar words.

Definition 6.1.1 (Independence Relation, [Maz86]). An *independence relation* on an alphabet Σ is a symmetric and irreflexive relation $I \subseteq \Sigma \times \Sigma$. △

We allow for multiple simultaneous actions, i.e. at a given point in time a set of actions may be executed. As we consider sets of actions there may be a set containing dependent actions. An example would be that a car simultaneously removes a claim and sets a reservation. In this work we do not consider such sets.

Assumption 6.1.2 (Independent Sets of Actions)**.** For an alphabet Σ and an independence relation $I \subseteq \Sigma \times \Sigma$ we assume for all words $w \in \mathcal{P}(\Sigma)^*$, all indices $i \in \{1, \dots, \#w\}$ and all actions $a, a' \in w(i)$ with $a \neq a'$ that $(a, a') \in I$. $\triangle$

We define that two timed words are *causally congruent* iff their untimed words are in the same *equivalence class*. These are strongly inspired by Mazurkiewicz traces [Maz86]. However, there the author considers words with letters, while here we consider words with sets of actions. The difference is that in [Maz86] two words belong to the same equivalence class iff we can create one word from the other by repeatedly swapping adjacent independent letters. Here, two words are in the same equivalence class iff we can create one word from the other by splitting and joining sets of actions that are adjacent.

Definition 6.1.3 (Causal Congruence)**.** Let $I \subseteq \Sigma \times \Sigma$ be an independence relation. For two words $w, w' \in \mathcal{P}(\Sigma)^*$ with $w = \langle A_1 \dots A_n \rangle$ we define that w and w' are *causally congruent* (denoted by $w \leftrightharpoons w'$) recursively as

$$\begin{aligned}
&w \leftrightharpoons w' \text{ iff } w = w' \text{ or } \Big(\exists w'' \in \Sigma^*, i \in \{1, \dots, n\} : w'' \leftrightharpoons w' \text{ and} \\
&\big((\langle A_1, \dots, A_{i-1}, A_i \cup A_{i+1}, A_{i+2}, \dots, A_n \rangle = w'' \text{ and } (A_i \times A_{i+1}) \subseteq I) \text{ or} \\
&\qquad (\langle A_1, \dots, A_{i-1}, A_i \setminus A', A', A_{i+1}, \dots, A_n \rangle = w'' \text{ and } A' \subseteq A_i)\big)\Big) .
\end{aligned}$$

Two timed words are *causally congruent* iff their untimed words are causally congruent. $\triangle$

We point out that for a word w satisfying Assumption 6.1.2 all words causally congruent to w also satisfy Assumption 6.1.2.

In order to better understand our new relation we show some basic properties. We start by showing that it is an equivalence relation. Then we relate our congruence relation to the classical congruence relation by Mazurkiewicz. And finally we relate the relation to our operators on timed words from Chapter 5.

Lemma 6.1.4. *Our causal congruence relation (cf. Definition 6.1.3) is an equivalence relation.*

Proof. An equivalence relation is reflexive, symmetric and transitive. Reflexivity and transitivity can be easily seen in the definition of the relation (cf. Definition 6.1.3).

For symmetry we argue that we can undo a joining step by splitting and vice versa. This assumes that in the original word each set of actions respectively only contains actions that are independent of each other (cf. Assumption 6.1.2). □

Our definition of causal congruence is a generalisation of the classical causal congruence from [Maz86]. We show this by proving that our definition can simulate the original definition.

Lemma 6.1.5. *For an alphabet Σ consider two words $w = \langle \{\sigma_1\}, \ldots, \{\sigma_n\} \rangle$ and $w' = \langle \{\sigma_1\}', \ldots, \{\sigma'_n\} \rangle$ with $\sigma_i, \sigma'_i \in \Sigma$ for $i \in \{1, \ldots, n\}$, where each letter in the word is in a singleton set. Further, let $\widetilde{w} = \langle \sigma_1, \ldots, \sigma_n \rangle$ and $\widetilde{w}' = \langle \sigma'_1, \ldots, \sigma'_n \rangle$ be two words, where the sets of w and w' are replaced by their contents. Then*

$$w \leftrightharpoons w' \text{ iff } \widetilde{w} \mathrel{\widetilde{\leftrightharpoons}} \widetilde{w}'$$

for some independence relation $I \subseteq \Sigma \times \Sigma$, where $\widetilde{\leftrightharpoons}$ denotes classical causal congruence from [Maz86].

Proof. We point out that all words $w, w', \widetilde{w}, \widetilde{w}'$ have equal length. The general idea is that for the words consisting of singleton sets we consider in this lemma both causal congruences, our causal congruence and the causal congruence from [Maz86], can simulate each other. Note that to simulate one step from $\widetilde{\in}$ we need two steps from our definition.

For the "only if"-direction we argue that for every swapped letters σ_i and σ_{i+1} we have $(\sigma_i, \sigma_{i+1}) \in I$, which implies $(\{\sigma_i\} \times \{\sigma_{i+1}\}) \subseteq I$. Hence, we can first join the adjacent letters and then split them apart in swapped order.

For the "if"-direction the reasoning is symmetric. □

In this thesis we consider independence of MLSLS actions. As our criterion what constitutes independent actions we take the view that two actions are independent if we can execute them in any order and arrive at the same result. For our purposes this is sufficient. To further simplify things, we only consider actions from different cars as independent.

Definition 6.1.6 (Independence Relation for MLSLS Actions)**.** Let Act_C be the actions of car C. We define the independence relation for MLSLS actions as

$$I_{\mathsf{Act}} = \bigcup_{C,C' \in \mathbb{I}}^{C \neq C'} (\mathsf{Act}_C \times \mathsf{Act}_{C'}) \cup (\mathsf{Act}_{C'} \times \mathsf{Act}_C) \ . \qquad \triangle$$

Note that Assumption 6.1.2 is ensured by our assumption that each car only performs at most one action at a time (cf. Page 151).

From now on we consider congruence w.r.t. the congruence relation I_{Act} from Definition 6.1.6.

Example 6.1.7. Consider w from Example 5.1.13 and w_1, w_2, w_3 shown below:

$$\begin{aligned} w &= \langle \{\mathrm{wd\ r}(E,3)\}, \{\mathrm{r}(C_2)\}, \{\mathrm{wd\ r}(C_2,2), \mathrm{c}(E,2)\}, \emptyset \rangle \ , \\ w_1 &= \langle \{\mathrm{wd\ r}(E,3)\}, \{\mathrm{wd\ r}(C_2,2)\}, \{\mathrm{r}(C_2)\}, \{\mathrm{c}(E,2)\}, \emptyset \rangle \ , \\ w_2 &= \langle \{\mathrm{r}(C_2), \mathrm{wd\ r}(E,3)\}, \{\mathrm{wd\ r}(C_2,2), \mathrm{c}(E,2)\}, \emptyset \rangle \ , \\ w_3 &= \langle \{\mathrm{r}(C_2)\}, \{\mathrm{wd\ r}(E,3)\}, \{\mathrm{wd\ r}(C_2,2)\}, \{\mathrm{c}(E,2)\} \rangle \ . \end{aligned}$$

For the independence relation I_{Act} we have $w_1 \not\leftrightharpoons w$ because we cannot switch the order of dependent actions, i.e. actions of the same car. We have $w_2 \leftrightharpoons w$ because we joined actions from different cars. Further, $w_3 \leftrightharpoons w_2$ because we can always split sets of actions up into any order. At last, from $w_2 \leftrightharpoons w$, $w_3 \leftrightharpoons w_2$ and transitivity (see the next lemma) we conclude $w_3 \leftrightharpoons w$. Also note that the empty set of actions is ignored for the purposes of causal congruence. That is, we can arbitrarily introduce and remove empty sets of actions. $\triangle$

With the following two lemmas we relate our congruence relation to our operators on timed words.

Lemma 6.1.8. *For all timed words ϱ, ϱ' and sets of car identifiers $CS \subseteq \mathbb{I}$ we have*

$$\begin{aligned} \varrho \leftrightharpoons \varrho' &\text{ implies } (\varrho \upharpoonright CS) \leftrightharpoons \varrho' \upharpoonright CS \ , \\ \varrho \leftrightharpoons \varrho' &\text{ implies } (\varrho \setminus CS) \leftrightharpoons \varrho' \setminus CS \ . \end{aligned}$$

Proof. The properties hold because our operators do not affect the order of the remaining actions (see also [Maz86, Eq. 8]). In other words, if any two words w and w' are causally congruent, then also for each car C the projection to Act_C are equal up to empty sets of actions. $\square$

Lemma 6.1.9. *For $i \in \{1,2\}$, sets of car identifiers $CS, CS' \subseteq \mathbb{I}$ with $CS \cap CS' = \emptyset$ and all timed words $\varrho_i \in (\mathcal{P}(\mathsf{Act}_{CS}) \times \mathbb{T})^+$ and $\varrho'_i \in (\mathcal{P}(\mathsf{Act}_{CS'}) \times \mathbb{T})^+$ we have*

$$\varrho_1 \leftrightharpoons \varrho_2 \text{ and } \varrho'_1 \leftrightharpoons \varrho'_2 \text{ implies } (\varrho_1 \parallel \varrho'_1) \leftrightharpoons (\varrho_2 \parallel \varrho'_2) \ .$$

Proof. Any actions occurring in ϱ_1, ϱ_2 are independent of all actions occurring in ϱ'_1, ϱ'_2 and vice versa. Hence, the parallel composition is causally congruent. $\square$

We define a metric on timed words to quantify temporal similarity. We assume that between two similar timed words the time stamps of all acceleration actions are equal. The reason for this restriction is that if we allow acceleration time stamps to differ, perturbations may accumulate. In this thesis we do not consider such issues. The intuition is that two timed words are ϵ-similar if any two corresponding actions are at most ϵ time units apart. More specifically, for all actions $a \in \mathsf{Act}$ we measure the maximal temporal distance of the i-th a-action in both timed words. Note that a is a single action, and not a set of actions. For the following definition we remind that $\approx$ denotes equality up to interior empty sets of actions and $\upharpoonright$ denotes restriction for timed words (cf. Pages 153 and 154).

Definition 6.1.10 (Metric on Timed Words)**.** Let $\Sigma_{\mathsf{acc}} = \{\mathrm{a}(C, x) \mid C \in \mathbb{I}, x \in \mathbb{R}\}$. Given two timed words ϱ, ϱ' with $\mathsf{span}\, \varrho = [0, r]$, $\mathsf{span}'\, \varrho = [0, r']$ and $r, r' \in \mathbb{T}$, we define $d_{\mathsf{t}}(\varrho, \varrho') = \infty$ if they are not causally congruent, or if $\varrho \upharpoonright \Sigma_{\mathsf{acc}} \not\approx \varrho' \upharpoonright \Sigma_{\mathsf{acc}}$ holds. Otherwise, we have

$$d_{\mathsf{t}}(\varrho, \varrho') = \mathsf{max}(|r - r'|, \sup_{a \in \mathsf{Act}} (\max_{i \in \{1, \ldots, \# \tau_a\}} (|\tau_a(i) - \tau'_a(i)|))) \ ,$$

where τ_a (resp. τ'_a) is the sequence of the time points when in ϱ (resp. ϱ') an action a occurs. That is, $t \in \tau_a$ iff $\exists j \in \mathsf{dom}\, w.\, a \in w(j)$ and $\tau(j) = t$ (and similarly for ϱ'). $\triangle$

For the definition above we point out that causal congruence ensures that ϱ and ϱ' contain for each $a \in \mathsf{Act}$ the same number of occurrences of a. In other words, τ_a and τ'_a have the same length.

For two timed words ϱ, ϱ' we say that ϱ and ϱ' are *ϵ-similar* if $d_{\mathsf{t}}(\varrho, \varrho') \leq \epsilon$ and we call two timed words *similar* if there is $\epsilon \in \mathbb{R}_{>0}$ such that they are ϵ-similar. We make the following observation: For all similar timed words the order of actions for each car individually is equal. Formally, let ϱ, ϱ' be two similar timed words, $(w_C, \tau'_C) = \varrho \upharpoonright \{C\}$ and $(w'_C, \tau'_C) = \varrho' \upharpoonright \{C\}$. Then, the untimed words w_C and w'_C are equal up to empty sets of actions. This holds because similar words are causally congruent.

We show that our function measuring distances of timed words is a pseudo metric.

Lemma 6.1.11. *The tuple* $((\mathcal{P}(\mathsf{Act}) \times \mathbb{T})^+, d_\mathsf{t})$ *is a pseudo metric space.*

Proof. We recapitulate the properties that $((\mathcal{P}(\mathsf{Act}) \times \mathbb{T})^+, d_\mathsf{t})$ has to satisfy. For all timed words $\varrho, \varrho', \varrho'' \in (\mathcal{P}(\mathsf{Act}) \times \mathbb{T})^+$ we have

$$\begin{aligned} d_\mathsf{t}(\varrho, \varrho) &= 0 \ , \\ d_\mathsf{t}(\varrho, \varrho') &= d_\mathsf{t}(\varrho', \varrho) \ , \\ d_\mathsf{t}(\varrho, \varrho'') &\leq d_\mathsf{t}(\varrho, \varrho') + d_\mathsf{t}(\varrho', \varrho'') \ . \end{aligned}$$

It is clear that $d_\mathsf{t}(\varrho, \varrho) = 0$ and $d_\mathsf{t}(\varrho, \varrho') = d_\mathsf{t}(\varrho', \varrho)$.

For the last property we make a case distinction on whether $d_\mathsf{t}(\varrho, \varrho'') = \infty$. If $d_\mathsf{t}(\varrho, \varrho'') = \infty$, then either the time stamps of acceleration actions differ ($\varrho \upharpoonright \Sigma_\mathsf{acc} \not\approx \varrho'' \upharpoonright \Sigma_\mathsf{acc}$), or the two timed words are not causally congruent. For both reasons, the same applies for ϱ and ϱ' or for ϱ' and ϱ''. If this was not the case, then by transitivity of $\approx$ and causal congruence that reason would also apply for ϱ and ϱ'', which contradicts our assumption.

If $d_\mathsf{t}(\varrho, \varrho'') \neq \infty$, it is easy to see that the supremum of absolute differences satisfies the triangle inequality. □

We point out that d_t is not a metric on our timed words. If d_t was a metric, then it would also satisfy:

$$\text{if } \varrho \neq \varrho' \text{ then } d_\mathsf{t}(\varrho, \varrho') > 0 \ . \tag{6.1}$$

This does not hold. As a counter example we can use the timed words $\varrho = \langle(\emptyset, 1), (\emptyset, 2)\rangle$ and $\varrho' = \langle(\emptyset, 2)\rangle$, for which we have $d_\mathsf{t}(\varrho, \varrho') = 0$ even though they are different. The idea of Equation (6.1) is to ensure that indiscriminability of elements is reflected in the metric. If we accept that equality up to interior empty sets of actions represents indiscriminability of timed words, then d_t is a metric on timed words. That is, we have

$$\text{if } \varrho \not\approx \varrho' \text{ then } d_\mathsf{t}(\varrho, \varrho') > 0 \ .$$

The property holds because if $\varrho \not\approx \varrho'$ then either their span is different, which means $d_\mathsf{t}(\varrho, \varrho') > 0$, or one of the timed words has an action at a time t and the other does not. In this case we again have $d_\mathsf{t}(\varrho, \varrho') > 0$.

We use our metric on timed words and our notion of robust static satisfaction to define what it means for a transition sequence to robustly satisfy globally ϕ.

Definition 6.1.12 (Robust Global Satisfaction)**.** Let M be a proper finite MLSLS model, $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M}) \times \mathbb{T})^+$ a timed word, ϕ an MLSLS formula and $\delta \in \mathbb{R}_{>0}$ an allowed spatial error. Then we define that ϕ holds globally in $\varrho(M)$ with spatial robustness δ (denoted $\varrho(M) \models^{\delta}_{\mathsf{seq}} \Box\,\phi$) as

$$\varrho(M) \models^{\delta}_{\mathsf{seq}} \Box\,\phi \ \text{ iff } \ \forall t.\, \varrho(M)@t \models^{\delta} \phi \ ,$$

where $\models^{\delta}$ is our robust static satisfaction relation (Page 121). Further, for the aforementioned parameters and an allowed temporal error $\epsilon \in \mathbb{R}_{>0}$ we define that ϕ holds globally in the transition sequence $\varrho(M)$ with spatial robustness δ and temporal robustness ϵ (denoted $\varrho(M) \models^{\epsilon,\delta}_{\mathsf{seq}} \Box\,\phi$) as

$$\varrho(M) \models^{\epsilon,\delta}_{\mathsf{seq}} \Box\,\phi \ \text{ iff } \ \forall \varrho'.\, (d_{\mathsf{t}}(\varrho, \varrho') \le \epsilon \implies \varrho'(M) \models^{\delta}_{\mathsf{seq}} \Box\,\phi) \ . \qquad \triangle$$

We extend our construction to check $\Box\,\phi$, where ϕ is a well-scoped MLSLS formula, to also check if all similar behaviours also satisfy $\Box\,\phi$. To this end, we define a formula $\mathsf{sim}_{\mathsf{t}}$ that we use to generate for a given timed word an ϵ-similar timed word. We split this formula into an elementary version considering timed words from a single car, and an extended version considering timed words from multiple cars.

For technical reasons we require that on the FORCF side words and also timed words contain an empty set as their last set of actions. We call such (timed) words *$\emptyset$-terminated*. If a timed word $\varrho = (w, \tau)$ is not $\emptyset$-terminated, we can create an $\emptyset$-terminated version of it as $\varrho' = (w \cdot \emptyset, \tau \cdot (\mathsf{last}\,\tau))$. Note that ϱ' is not a timed word as the time stamps are not strictly increasing. However, we can easily extract the original timed word. Nevertheless, for ease of presentation we shall refer to such a structure as a timed word.

Definition 6.1.13 (Shaking Temporal Variables for a Single Car)**.** For some car identifier $C \in \mathbb{I}$ let $w \in \mathcal{P}(\mathsf{Act}_C)^+$ be a $\emptyset$-terminated word and $\zeta, \zeta' \in \mathsf{Seq}\,\mathsf{RTerm}$ two sequences of real FORCF terms with $\#\zeta = \#\zeta' = \#w{+}1$. Then, for $\epsilon \in \mathbb{R}_{>0}$

and $n = \#\zeta$ we define

$$\mathsf{sim}_{\mathsf{t,E}}(w, \zeta, \zeta', \epsilon) \equiv \psi(\zeta) \wedge \psi(\zeta') \wedge$$
$$\bigwedge_{j \in \mathsf{dom}\, w} \left(\begin{array}{l} w(j) \subseteq \overline{\Sigma_{\mathsf{acc}}} \implies \exists z_j \in [-\epsilon, \epsilon].\, \zeta(j+1) = \zeta'(j+1) + z_j \\ \wedge \\ w(j) \not\subseteq \overline{\Sigma_{\mathsf{acc}}} \implies \zeta(j+1) = \zeta'(j+1) \end{array} \right)$$
$$\psi(\zeta) \equiv \zeta(1) \leq \zeta(2) \wedge \zeta(1) = 0 \wedge \zeta(n-1) \leq \zeta(n) \wedge \bigwedge_{j \in \{2, \dots, n-2\}} \zeta(j) < \zeta(j+1)$$

△

The intuition of the definition above is that the i-th entry of ζ (resp. ζ') is the time stamp of the i-th set of actions in w. For a single car two timed words are similar iff their untimed words are equal up to empty sets of actions. Hence, in $\psi(\zeta)$ we use $\zeta(j) < \zeta(j+1)$ (resp. $\zeta'(j) < \zeta'(j+1)$) to ensure that the order of time stamps are equal. Here, we take special care of the first and the last time stamp, which both have special meaning. The first time stamp is fixed to be 0 and does not represent a set of actions. The last time stamp represents an empty set of actions and is required to be at the end. With the convention that the i-th time stamp represents the i-th set of actions it follows that both untimed words are equal. The two ϵ-similar timed words we get from a satisfying assignment h are

$$\varrho = (w, \tau) \text{ with } \tau = \langle h(\zeta(2)), \dots, h(\mathsf{last}\, \zeta) \rangle \text{ and}$$
$$\varrho' = (w, \tau') \text{ with } \tau' = \langle h(\zeta'(2)), \dots, h(\mathsf{last}\, \zeta') \rangle \; .$$

Additionally, we distinguish the cases $w(j) \subseteq \overline{\Sigma_{\mathsf{acc}}}$ and $w(j) \not\subseteq \overline{\Sigma_{\mathsf{acc}}}$ to ensure that for the element $\mathsf{last}\, w$, which is equal to $\emptyset$, the first case holds.

We briefly explain how the restriction to $\emptyset$-terminated timed words is helpful. For an action $a \in \mathsf{Act}$ consider the timed words $\varrho = \langle (\{a\}, 1) \rangle$ and $\varrho' = \langle (\{a\}, 0.9), (\emptyset, 1.1) \rangle$. We want to use the formula $\mathsf{sim}_{\mathsf{t,E}}$ to recognise that ϱ and ϱ' are 0.1-similar. If we instantiate the sequences of terms ζ and ζ' with the sequences of time stamps of ϱ and ϱ', then ζ and ζ' have different length and we do not know which elements to compare. Hence, we ensure that ζ and ζ' have equal length. To achieve this, we restrict ourselves to $\emptyset$-terminated timed words. That is, we change ϱ to $\varrho'' = \langle (\{a\}, 1), (\emptyset, 1) \rangle$ and then instantiate $\mathsf{sim}_{\mathsf{t,E}}$. Note that this change does not affect temporal similarity.

We lift our formula to check similarity of timed words to multiple cars. Similar to Chapter 5 we represent timed words on the FORCF side for each car individually without interior empty sets of actions (for an example see Example 5.2.7 on Page 166).

Definition 6.1.14 (Shaking Temporal Variables)**.** For a finite set of car identifiers $CS \subseteq \mathbb{I}$, two sets of car identifier variables $\mathbb{D}_i \subseteq \mathsf{DVar}$ with $i \in \{1, 2\}$ and $|\mathbb{D}_i| = |CS|$ let $w \in \mathcal{P}(\mathsf{Act}_{CS})^+$ be a $\emptyset$-terminated word, $g_i : \mathbb{D}_i \to CS$ two bijective car identifier assignments and $Z_i : \mathbb{D}_i \to \mathsf{Seq}\,\mathsf{RTerm}$ two functions assigning sequences of real terms to car identifier variables such that for all $D_i \in \mathbb{D}_i$ we have $\#Z_i(D_i) = \#w_C + 1$, where w_C with $C \in CS$ is the word that is equal to $w \upharpoonright \mathsf{Act}_C$, except that w_C has no interior empty sets of actions. Then, for $\epsilon \in \mathbb{R}_{>0}$ we define

$$\mathsf{sim_t}(w, (Z_1, g_1), (Z_2, g_2), \epsilon) \equiv \bigwedge_{i \in \{1,2\}} \mathsf{span\text{-}eq}(Z_i) \wedge f(w, (Z_1, g_1), (Z_2, g_2), \epsilon) \ ,$$

$$f(w, (Z_1, g_1), (Z_2, g_2), \epsilon) \equiv$$
$$\begin{cases} \mathsf{sim_{t,E}}(w_C, Z_1(D_1), Z_2(D_2), \epsilon) \wedge & \\ \quad f(w \setminus \{C\}, \{D_1\} \lhd (Z_1, g_1), \{D_2\} \lhd (Z_2, g_2), \epsilon) & \\ \quad \text{where } D_i = g_i^{-1}(C), C \in CS & \text{if } CS \neq \emptyset \ , \\ \mathsf{true} & \text{otherwise} \ , \end{cases}$$

$$\mathsf{span\text{-}eq}(Z) \equiv \bigwedge_{D, D' \in \mathsf{dom}\, Z} \mathsf{last}(Z(D)) = \mathsf{last}(Z(D')) \ . \qquad \triangle$$

We prove that our encoding of temporal similarity is appropriate. We split this proof into two properties: That we can use $\mathsf{sim_t}$ to recognise ϵ-similarity of timed words and to generate ϵ-similar timed words.

Lemma 6.1.15. *For all finite nonempty sets of car identifiers CS, $i \in \{1, 2\}$, timed words $\varrho_i = (w_i, \tau_i) \in (\mathcal{P}(\mathsf{Act}_{CS}) \times \mathbb{T})^+$ and $\epsilon \in \mathbb{R}_{>0}$ the following holds:*

$$d_{\mathsf{t}}(\varrho_1, \varrho_2) \leq \epsilon \text{ implies } \mathsf{sim_t}(w, (Z_1, g_1), (Z_2, g_2), \epsilon) \textit{ is satisfiable,}$$

where either $w = w_1$ or $w = w_2$, the car identifier variable assignment $g_i : \mathbb{D}_i \to CS$ for some set $\mathbb{D}_i \subseteq \mathsf{DVar}$ is bijective and $Z_i = \{D_i \mapsto 0 :: \tau^i_{g_i(D_i)} \mid D_i \in \mathbb{D}_i\}$ and the timed word (w^i_C, τ^i_C) with $(w^i_C, \tau^i_C) \approx (\varrho_i \upharpoonright \{C\})$ contains no interior empty sets of actions and is $\emptyset$-terminated.

Proof. The formula $\mathsf{sim}_{\mathsf{t}}(w,(Z_1,g_1),(Z_2,g_2),\epsilon)$ unfolds to a conjunction of the form

$$\mathsf{span\text{-}eq}(Z_1) \wedge \mathsf{span\text{-}eq}(Z_2) \wedge \bigwedge_{C \in CS} \mathsf{sim}_{\mathsf{t,E}}(w_C, Z_1(D_1), Z_2(D_2), \epsilon) \ ,$$

where D_i is the inverse of g_i for C, i.e. for $i \in \{1,2\}$ we have $D_i = g(C)_i^{-1}$ which is defined as g_i is bijective.

As the last entry of Z_i is equal for all car identifier variables $D_i \in \mathsf{dom}\, g_i$, it is clear that $\mathsf{span\text{-}eq}(Z_i)$ is satisfied.

Next, consider the formula $\mathsf{sim}_{\mathsf{t,E}}(w_C, Z_1(D_1), Z_2(D_2), \epsilon)$. To create w_C we removed all internal empty sets of actions and ensured that the word is $\emptyset$-terminated. Thus, because of causal congruence of ϱ_1 and ϱ_2 we know that w_C is not affected by creating it from w_1 instead of w_2, or vice versa. Hence, the j-th entry in $Z_1(D_1)$ and $Z_2(D_2)$ both are the time stamps of the same action.

If we look at the definition of $\mathsf{sim}_{\mathsf{t,E}}$ we see that $\psi(Z_i(D_i))$ is satisfied. At last, consider the big conjunction in $\mathsf{sim}_{\mathsf{t,E}}$. For $j \in \mathsf{dom}\, w_C$, if $w_C(j)$ is a set containing a single acceleration action ($w_C(j) \not\subseteq \overline{\Sigma_{\mathsf{acc}}}$), then $d_{\mathsf{t}}(\varrho_1, \varrho_2) \leq \epsilon$ ensures that the time stamps of acceleration actions are equal. This means that $Z_1(D_1)(j+1) = Z_2(D_2)(j+1)$ is satisfied. Note that the $+1$ is necessary because the first entry in $Z_i(D_i)$ is fixed to be 0 and that we argued earlier that $Z_1(D_1)(j+1)$ and $Z_2(D_2)(j+1)$ represents time stamps of the same action in different timed words. On the other hand, if $w_C(j)$ does not represent a set containing a single acceleration action ($w_C(j) \subseteq \overline{\Sigma_{\mathsf{acc}}}$), then ϵ-similarity of ϱ_1 and ϱ_2 ensures that $\exists z_j \in [-\epsilon, \epsilon].\, Z_1(D_1)(j+1) = Z_2(D_2)(j+1) + z_j$ is satisfiable. $\square$

Next, we show that we can use $\mathsf{sim}_{\mathsf{t}}$ to generate ϵ-similar timed words. We briefly explain the intuition of the lemma. We have a timed word $\varrho = (w, \tau) \in (\mathcal{P}(\mathsf{Act}_{CS}) \times \mathbb{T})^+$ and represent it in a FORCF structure. This structure consists of a bijective car identifier variable assignment from a subset of DVar to CS and a function that assigns for each car a sequence such that the j-th entry in the sequence is the time stamp of the j-th action of that car. For some $i \in \{1,2\}$ let this structure be (Z_i, g_i). Further, we have another FORCF structure that is structurally equal to the first structure but only contains independent free variables instead of time stamps. Let this structure be $(Z_{\bar{i}}, g_{\bar{i}})$ with $\bar{i} \in \{1,2\} \setminus \{i\}$. Then, from a satisfying assignment for the formula $\mathsf{sim}_{\mathsf{t}}(w,(Z_1,g_1),(Z_2,g_2),\epsilon)$ we can extract a timed word that is ϵ-similar to ϱ. Now, also in $Z_{\bar{i}}$ for each car the j-th entry is the time stamp of the j-th action of that car. That means that for each car individually the time stamps are perturbed but the order of actions is unchanged. We can create this ϵ-similar timed word by creating the parallel composition of the perturbed timed words for each car.

Lemma 6.1.16. *For a finite nonempty set of car identifiers CS and a timed word $\varrho = (w, \tau) \in (\mathcal{P}(\mathsf{Act}_{CS}) \times \mathbb{T})^+$ let the car identifier variable assignments $g_1 : \mathbb{D}_1 \to CS$ and $g_2 : \mathbb{D}_2 \to CS$ for sets $\mathbb{D}_1, \mathbb{D}_2 \subseteq \mathsf{DVar}$ be bijective. For $i \in \{1,2\}$ let $Z_i = \{D \mapsto 0 :: \tau_{g_i(D)} \mid D \in \mathbb{D}_i\}$, where the timed word (w_C, τ_C) with $(w_C, \tau_C) \approx \varrho \upharpoonright \{C\}$ contains no interior empty sets of actions and is $\emptyset$-terminated. Further, for $\bar{i} \in \{1,2\}$ with $i \neq \bar{i}$ let $Z_{\bar{i}} = \{D \mapsto \zeta_D \mid D \in \mathbb{D}_{\bar{i}}, \zeta_D \in \mathsf{Seq\,RVar}, \#\zeta_D = \#w_{g_{\bar{i}}(D)} + 1\}$ with each variable in $\mathsf{ran}\, Z_{\bar{i}}$ being unique. Then, for $\epsilon \in \mathbb{R}_{>0}$ and all assignments h the following holds:*

$$h \models \mathsf{sim_t}(w, (Z_1, g_1), (Z_2, g_2), \epsilon) \text{ implies } d_{\mathsf{t}}(\varrho, \varrho') \leq \epsilon \ ,$$

$$\textit{where } \varrho' = \mathop{\Big\|}_{D \in \mathsf{dom}\, g_{\bar{i}}} (w_{g(D)}, \tau'_D) \ \text{ and } \ \tau'_D = \langle h(Z_{\bar{i}}(D)(2)), \ldots, h_{\bar{i}}(\mathsf{last}(Z_{\bar{i}}(D)))\rangle \ .$$

Proof. Let $g_{\bar{i}}^{-1}$ be the inverse of $g_{\bar{i}}$, which is defined as $g_{\bar{i}}$ is bijective. For $C \in CS$ let $\varrho_C = (w_C, \tau_C)$ and $\varrho'_C = (w_C, \varrho'_D)$ with $D = g_i^{-1}(C)$ as defined in the lemma. Looking at the definition of $\mathsf{sim_{t,E}}$ we see that $\psi(Z_{\bar{i}}(D))$ ensures that $\mathsf{tail}\,\tau'_D$ is a strictly increasing sequence of time stamps, except for the last time stamp, which is only greater equal than the one before. Thus, it is clear that ϱ'_C is a timed word in the broader sense of this chapter (see the paragraph on $\emptyset$-terminated timed words on Page 185). Further, we see that ϱ_C and ϱ'_C are ϵ-similar.

Next, let us make a brief observation. For this let CS_1, CS_2 be disjoint sets of car identifiers and let $\varrho_1, \varrho'_1 \in (\mathcal{P}(\mathsf{Act}_{CS_1}) \times \mathbb{T})^+, \varrho_2, \varrho'_2 \in (\mathcal{P}(\mathsf{Act}_{CS_2}) \times \mathbb{T})^+$ be timed words, where all four timed words have an equal span. Then we observe

$$\mathsf{max}(d_{\mathsf{t}}(\varrho_1, \varrho'_1), d_{\mathsf{t}}(\varrho_2, \varrho'_2)) = d_{\mathsf{t}}(\varrho_1 \parallel \varrho_2, \varrho'_1 \parallel \varrho'_2) \ . \tag{6.2}$$

This holds because the metric considers actions independently, i.e. for two different actions a, a' our metric considers all occurrences of a independent of the occurrences of a'. Note that this property does not hold if the span may differ because then the parallel composition can hide this difference.

For the timed word $\varrho' = \|_{D \in \mathsf{dom}\, g_{\bar{i}}} (w_{g(D)}, \tau'_D)$ the formula $\mathsf{sim_t}$ ensures with $\mathsf{span\text{-}eq}(Z_{\bar{i}})$ that the individual timed words have equal span. And for $\varrho'' =$

$\|_{C \in CS} \varrho_C$ by construction the individual timed words have equal span. Thus, using Equation (6.2) and that the corresponding individual timed words are ϵ-similar, we conclude that the timed words ϱ' and ϱ'' are ϵ-similar, i.e. $d_{\mathsf{t}}(\varrho', \varrho'') \leq \epsilon$. As ϱ'' is equal to ϱ up to empty sets of actions, their distance is 0, i.e. $d_{\mathsf{t}}(\varrho'', \varrho) = 0$. From the triangle inequality (cf. Lemma 6.1.11) we conclude $d_{\mathsf{t}}(\varrho', \varrho) \leq \epsilon$, which means that the lemma holds. $\square$

For a structure of variables $\mathcal{S}$ (resp. Z) we denote with $\forall \mathcal{S}$ (resp. $\forall Z$) universal quantification for all data variables in $\mathcal{S}$ (resp. temporal variables in Z).

For the definition below we remind that $\Pi : \mathbb{D} \to \mathsf{Seq}$ with $\mathbb{D} \subseteq \mathsf{DVar}$ is a function that assigns for each car identifier variable $D \in \mathbb{D}$ a sequence of FORCF data tuples. Let $g : \mathbb{D} \to \mathbb{I}$ be a car identifier variable assignment, M an MLSLS model and ϱ a timed word. Then, by requiring that (Π, g, M, ϱ) is sane we essentially require that there is a FORCF assignment assigning values to the variables in Π such that these sequences of values represent the transition sequence $\varrho(M)$ (cf. Definition 5.2.6 on Page 166). This works similarly for a FOLRA data tuple p'. Then (p', g, M) is sane if there is a FOLRA assignment such that the values assigned to the variables in p' represent M (cf. Definition 3.4.2 on Page 114). Further, note that FOLRA is a fragment of FORCF, which means that it is fine to mingle both formalisms, as we do here.

Definition 6.1.17 (Transforming Global Properties)**.** For a finite proper model $M = (CS, TS, \Omega, V, \nu)$ and an $\emptyset$-terminated timed word $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\, M}) \times \mathbb{T})^+$ let $\mathbb{D} \subseteq \mathsf{DVar}$ and g be such that $|\mathbb{D}| = |\,\mathsf{cars}\, M|$ and $g : \mathbb{D} \to \mathsf{cars}\, M$ is bijective. Further, let $Z : \mathbb{D} \to \mathsf{Seq\, RTerm}$ be an assignment of sequences of real-valued terms such that for all $D \in \mathbb{D}$ we have $Z(D) = \langle 0 \rangle \cdot \tau_{g(D)}$, where $\varrho_{g(D)} \approx (\varrho \upharpoonright \mathsf{Act}_{g(D)})$ and $\varrho_{g(D)} = (w_{g(D)}, \tau_{g(D)})$ contains no interior empty sets of actions and let $Z' : \mathbb{D} \to \mathsf{Seq\, RVar}$ be an assignment of sequences of unused FORCF variables such that for all $D \in \mathbb{D}$ we have $\#Z(D) = \#Z'(D)$. Then we can easily create an assignment of sequences of data tuples $\Pi : \mathbb{D} \to \mathsf{Seq}\, \mathrm{T}^{\mathsf{rcf}}$ such that (Π, g, M, ϱ) is sane. Let p_{f} be the combination of all data tuples $\mathsf{last}\, \Pi(D)$ with $D \in \mathbb{D}$ and let $p' = (DS, l_1, l_2, x_1', x_2', f, \mathbb{D}, \mathcal{S}', D_E)$ be a FOLRA data tuple such that (p', g, M) is sane. Then for an MLSLS formula ϕ with $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}\, \nu$ and $\epsilon, \delta \in \mathbb{R}_{>0}$ we define

$$\begin{aligned} \mathsf{tr}_{\Box}^{\epsilon,\delta}(\varrho, M, \phi) \equiv \forall z, Z', \mathcal{S}', x_1', x_2'. (((&\mathsf{tr}_{\mathsf{U}}(\varrho, M, z, Z', \Pi, g) \land \mathsf{sane}^{\mathsf{r}}(p') \land \\ &\mathsf{sim}_{\mathsf{m}}((p_{\mathsf{f}}, g), (p', g), \delta) \land \mathsf{sim}_{\mathsf{t}}(w, (Z, g), (Z', g), \epsilon)) \implies \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p', \phi)) \enspace . \end{aligned}$$

△

The intuition is that we create an ϵ-similar timed word and store it in the variables in Z' with $\mathsf{sim_t}(w,(Z,g),(Z',g),\epsilon)$. Then we encode the transition sequence with the perturbed timed word until the freeze time with $\mathsf{tr_U}(\varrho, M, z, Z', \Pi, g)$. Note that in the previous formula we use ϱ only to encode the discrete actions, for which for each car individually order is equivalent in the original timed word and the perturbed one. We store the model M_t at the freeze time in the variables p_f. Then we use $\mathsf{sim_m}((p_\mathsf{f}, g),(p',g),\delta) \wedge \mathsf{sane^r}(p')$ to create a model that is δ-similar to M_t and store it in the variables in p'. At last, we check if this model satisfies the MLSLS formula with $\mathsf{tr_f^{lra}}(p',\phi)$.

We consider a small example.

Example 6.1.18. Consider the transition sequence $\varrho_0(M_0) = \varrho(M) \restriction \{C_2, E\}$, where ϱ and M are taken from Example 5.1.13 and the formula

$$\mathsf{safe} \equiv \forall c, c'.\, (c \neq c' \implies \neg\langle \mathsf{re}(c) \wedge \mathsf{re}(c')\rangle)$$

from [HLO+11], which states that there do not exist two different cars with overlapping reservations. In the following let the desired temporal robustness be $\epsilon = 0.1$ and the desired spatial robustness $\delta = 1$. To determine that $\Box\,\mathsf{safe}$ does *not* hold ϵ-δ-robustly in $\varrho_0(M_0)$, we give a satisfying assignment for the formula $\neg\mathsf{tr}_{\Box}^{0.1,1}(\varrho_0, M_0, \mathsf{safe})$. This formula evaluates to

$$\exists z, \widetilde{Z}, \widetilde{\mathcal{S}}, \widetilde{x_1}, \widetilde{x_2}.\, (\mathsf{tr_U}(\varrho, M, z, \widetilde{Z}, \Pi, g) \wedge \mathsf{sim_t}(w,(Z,g),(\widetilde{Z},g),\epsilon) \\ \wedge\, \mathsf{sim_m}((p_\mathsf{f},g),(\widetilde{p},g),\delta) \wedge \mathsf{sane^r}(\widetilde{p}) \wedge \neg\mathsf{tr_f^{lra}}(\widetilde{p},\phi)) \quad , \qquad (6.3)$$

where $\mathbb{D} = \{D, D_E\}$, $g = \{D \mapsto C_2, D_E \mapsto E\}$, $p_\mathsf{f} = (\mathbb{D}, 1, 3, x, x', f, \mathbb{D}, \mathcal{S}, D_E)$, $\widetilde{p} = (\mathbb{D}, 1, 3, \widetilde{x}, \widetilde{x}', f, \mathbb{D}, \widetilde{\mathcal{S}}, D_E)$, $\mathcal{S} = \{D \mapsto (v_\mathrm{p}^D, \dots), D_E \mapsto (v_\mathrm{p}^{D_E}, \dots)\}$, $\widetilde{\mathcal{S}} = \{D \mapsto (\widetilde{v}_\mathrm{p}^D, \dots), D_E \mapsto (\widetilde{v}_\mathrm{p}^{D_E}, \dots)\}$ and the functions assigning sequences of real-valued terms are given as $Z = \{D \mapsto \langle 0, 1.1, 5.1, 6.1\rangle, D_E \mapsto \langle 0, 1, 5.1, 6.1\rangle\}$ and $\widetilde{Z} = \{D \mapsto \langle \widetilde{z}_1^D, \widetilde{z}_2^D, \widetilde{z}_3^D, \widetilde{z}_4^D\rangle, D_E \mapsto \langle \widetilde{z}_1^{D_E}, \widetilde{z}_2^{D_E}, \widetilde{z}_3^{D_E}, \widetilde{z}_4^{D_E}\rangle\}$. .

We show a part of $\mathsf{tr_U}(\varrho, M, z, \widetilde{Z}, \Pi, g)$ to make clear that here the formula uses temporal variables, where the value is subject to perturbation, instead of constants:

$$(\widetilde{z}_1^{D_E} \leq z < \widetilde{z}_2^{D_E} \implies (\mathsf{tr_w}(\langle\rangle, M_E, \langle \widetilde{z}_1^{D_E}\rangle, \langle p_1^E\rangle, g_E) \wedge \\ \mathsf{tr_a}(z - \widetilde{z}_1^{D_E}, \emptyset, p_1^E, p_2^E, g_E) \wedge F(\langle p_2^E, p_3^E, p_4^E\rangle, g_E))) \\ \wedge(\widetilde{z}_2^{D_E} \leq z < \widetilde{z}_3^{D_E} \implies \dots)$$

where $M_E = M_0 \boxbar \{E\}$. We saw in Example 5.2.9 that with constant speed the position in the final data tuple p_f is the initial position plus the speed times the freeze time. Thus, for a freeze time of $z = 1$ the positions are $D_E.v^4_\mathsf{p} = 34$ and $D.v^4_\mathsf{p} = 24$. Further, the value of the sensor function is $D.v^4_\Omega = 9$ and $D_E.v^4_\Omega = 16.5$.

We choose values for the perturbed temporal variables $\widetilde{z}_2^{D_E}$ and $\widetilde{z}_2^{D}$ (i.e. the perturbed time stamps of the first action of both cars) such that at the freeze time the car C_2 has set its reservation and E has not yet withdrawn its reservation. That is, we choose $\widetilde{z}_2^{D_E} = 1.1$ and $\widetilde{z}_2^{D} = 1$. For the spatial perturbations we choose to perturb the sensor function of D and the position of D_E by 1, i.e. $\widetilde{v_\Omega}^{D} = 10$ and $\widetilde{v_\mathsf{p}}^{D_E} = 33$. We do not perturb the other variables. Now, the perturbed safety envelopes of D_E and D have an overlap of length of 1 and their reservations both contain lane 2. Hence, the formula safe is not satisfied by the perturbed model, which means that safe does not hold globally (in the temporal sense) with a temporal robustness of 0.1 and a spatial robustness of 1. That is, $\varrho_0(M_0) \not\models^{0.1,1}_\mathsf{seq} \Box\,\mathsf{safe}$. △

We prove that our construction is correct.

Lemma 6.1.19. *Given proper finite MLSLS model $M = (CS, TS, \Omega, V, \nu)$ and formula ϕ with $\mathsf{freeVar}(\phi) \subseteq \mathsf{dom}\,\nu$, an $\emptyset$-terminated timed word $\varrho \in (\mathcal{P}(\mathsf{Act}_{\mathsf{cars}\,M}) \times \mathbb{T})^+$ and $\epsilon, \delta \in \mathbb{R}_{>0}$ we have*

$$\varrho(M) \models^{\epsilon,\delta}_\mathsf{seq} \Box\,\phi \text{ iff } \mathsf{tr}^{\epsilon,\delta}_\Box(\varrho, M, \phi) \textit{ is valid }.$$

Proof. Let $Z, Z', \Pi, g, p', p_\mathsf{f}$ be as defined in the definition of $\mathsf{tr}^{\epsilon,\delta}_\Box$. That means, Z contains the time stamps in ϱ, Z' is a structure of fresh temporal variables, g is a bijective mapping between some set $\mathbb{D} \subseteq \mathsf{DVar}$ and $\mathsf{cars}\,M$, Π is a structure assigning to each $D \in \mathbb{D}$ a sequence of data tuples such that (Π, g, M, ϱ) is sane and p' is a data tuple such that (p', g, M) is sane.
Case 1 (only if). We start with the "only if"-direction and proceed by contraposition. The general idea is that we extract values from a satisfying assignment that represent a point in time t, a timed word ϱ' and an MLSLS model M' such that ϱ' is ϵ-similar to ϱ, M' is δ-similar to $\varrho'(M)@t$ and M' does not satisfy ϕ.

Assume that the negation of the right hand side is satisfiable. That is, assume that

$$\exists z, Z', \mathcal{S}', x_1', x_2'. (\mathsf{tr}_{\mathsf{U}}(\varrho, M, z, Z', \Pi, g) \wedge \mathsf{sim}_{\mathsf{t}}(w, (Z, g), (Z', g), \epsilon) \\ \wedge \mathsf{sim}_{\mathsf{m}}((p_{\mathsf{f}}, g), (p', g), \delta) \wedge \mathsf{sane}^{\mathsf{r}}(p') \wedge \neg \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p', \phi)) \quad (6.4)$$

is satisfiable, where $p' = (DS, l_1, l_2, x_1', x_2', f, \mathbb{D}, \mathcal{S}', D_E)$.

Let h be a satisfying assignment that assigns to the existentially quantified variables suitable values and let ϱ' be the timed word we get by replacing for each car the time stamp of each action by the value as assigned to the variables in Z' as defined in Lemma 6.1.16. Then $d_{\mathsf{t}}(\varrho, \varrho') \leq \epsilon$.

Note that we can replace ϱ in tr_{U} by ϱ' because the delays are taken from the real terms in Z'. From ϱ we only use the discrete actions, the order of which for each car individually, is equal in ϱ and ϱ' (ensured by $d_{\mathsf{t}}(\varrho, \varrho') \leq \epsilon$). Let Z'' be the structure we get by replacing the variables in Z' by the values assigned by h. Then we can replace z and Z' in $\mathsf{tr}_{\mathsf{U}}(\varrho', M, z, Z', \Pi, g)$ by t and Z'' without affecting satisfaction.

From Lemma 5.3.4 and because p_{f} is the composition of the last data tuples in Π, we know that satisfaction of $\mathsf{tr}_{\mathsf{U}}(\varrho', M, t, Z'', \Pi, g)$ implies satisfaction of $\mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, M_t)$. This means that the values assigned to the variables in p_{f} represent the model M_t. Next, the formulas $\mathsf{sim}_{\mathsf{m}}((p_{\mathsf{f}}, g), (p', g), \delta)$ and $\mathsf{sane}^{\mathsf{r}}(p')$ ensure that p' contains a representation of an MLSLS model M' that is δ-similar to M_t. From Lemma 3.4.7 and $h \models \mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p', g, M') \wedge \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p', \phi)$ we conclude $M' \not\models \phi$. As $d_{\mathsf{t}}(\varrho, \varrho') \leq \epsilon$ and $d_{\mathsf{m}}(M_t, M') \leq \delta$ it follows that $\varrho(M) \not\models_{\mathsf{seq}}^{\epsilon,\delta} \Box \phi$.

Case 2 (if). We continue with the "if"-case and proceed by contraposition. The general idea is to find FORCF formulas describing the different aspects of the transition sequence and the static MLSLS formula and show that they are satisfiable for themselves. Then we argue that their conjunction also is satisfiable.

Assume $\varrho(M) \not\models_{\mathsf{seq}}^{\epsilon,\delta} \Box \phi$, which means that there is a point in time t, a timed word $\varrho' = (w', \tau')$ and an MLSLS model M' such that ϱ' is ϵ-similar to ϱ, M' is δ-similar to $\varrho'(M)@t$ and M' does not satisfy ϕ. For $C \in \mathsf{cars}\, M$ let $\varrho_C' = (w_C', \tau_C')$ be such that ϱ_C' is equal to $\varrho' \restriction \{C\}$ except that ϱ_C' has no interior empty sets of actions and is $\emptyset$-terminated. Let $\varrho_C = (w_C, \tau_C)$ be defined similarly. Further, let $M_C = M \boxplus \{C\}$, $n_C = \#\varrho_C'$, $\varrho_C'(M_C) = M_{C,0} \xrightarrow{\tau_C'(1)} M_{C,1} \xrightarrow{w_C'(1)} M_{C,2} \ldots \xrightarrow{\tau_C'(n) - \tau_C'(n-1)} M_{C,2n-1} \xrightarrow{w_C'(n)} M_{C,2n}$ and let $k_C \in \{0, \ldots, n_C\}$ be the largest index such that $\tau_C'(k_C) \leq t < \tau_C'(k_C + 1)$ or

$k_C = n$ if $\mathsf{last}\,\tau'_C \leq t$. Then we use the abbreviations

$$\begin{aligned}
F_1 &\equiv \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p_{\mathsf{f}}, g, \varrho'(M)@t) \enspace , \\
\psi_1 &\equiv \bigwedge_{D\in\mathsf{dom}\,g} \Big(\bigwedge_{i\in\{0,\dots,k_{g(D)}\}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p^D_{i+1}, M_{g(D),i}) \bigwedge_{j\in\{k_{g(D)+1},\dots,n_{g(D)}\}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p^D_{j+1}, \varrho'_{g(D)}(M_{g(D)})@t)\Big) \enspace , \\
\psi_2 &\equiv \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p_{\mathsf{f}}, g, M_t) \wedge \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p', g, M') \wedge \mathsf{sim}_{\mathsf{m}}((p_{\mathsf{f}}, g), (p', g), \delta) \enspace , \\
\psi_3 &\equiv \mathsf{sim}_{\mathsf{t}}(w, (Z, g), (Z', g), \epsilon) \enspace ,
\end{aligned}$$

where p^D_k is an abbreviation for $\Pi(D)(k)$. Now, ψ_1 essentially represents for every car $C \in \mathsf{cars}\,M$ the MLSLS models encountered in $\varrho'_C(M_C)$ until time t. The formula ψ_1 is satisfiable because it is a conjunction of many formulas of the form $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(p, M'')$ for some MLSLS model M'' and some data tuple p and each single of these formulas is satisfiable and constrains disjoint sets of variables. For ψ_2, a satisfying assignment simply assigns the values in M' and M_t. For ψ_3, a satisfying assignment assigns the time stamps in τ' to the variables in Z' (cf. Lemma 6.1.15).

Next, we argue that the conjunction of the four formulas above also is satisfiable. As $\bigwedge_{D\in\mathbb{D}} \mathsf{tr}^{\mathsf{lra}}_{\mathsf{m,E}}(\mathsf{last}\,\Pi(D), \varrho'_{g(D)}(M_{g(D)})@t)$ is a subformula of ψ_1, it follows from Lemma 5.3.5 that $\psi_1 \iff \psi_1 \wedge F_1$. Even though $\psi_1 \wedge F_1$ and ψ_2 share the variables in p_{f} it is not too difficult to see that their conjunction also is satisfiable. The reason is that the constraints on their common variables are on both sides equivalent to F_1. As ψ_3 shares no variables with ψ_1 or ψ_2 we can also add ψ_3 such that $\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge F_1$ is satisfiable and let h be a satisfying assignment.

Next, from Lemma 5.3.4 we know that ψ_1 is equivalent to $\mathsf{tr}_{\mathsf{U}}(\varrho', M, t, Z'', \Pi, g)$, where $Z'' : \mathbb{D} \to \mathsf{Seq}\,\mathbb{R}$ is a structure containing the time points of ϱ'. Similar to the "only if"-direction, as $d_{\mathsf{t}}(\varrho, \varrho) \leq \epsilon$ holds we can replace ϱ' by ϱ, while preserving equivalence. Additionally, we can replace t and Z'' by z and Z'. To get a satisfying assignment we can extend h to assign z and the variables in Z' the value t and the values in Z''.

We remind that Lemma 3.4.7 states that satisfaction of a given MLSLS formula by a given MLSLS model can be recognised with FOLRA (which is a fragment of FORCF). Further, Lemma 3.4.6 states that the constraints representing an MLSLS model with FOLRA formulas are satisfiable. From these two lemmas and $M' \not\models \phi$ we deduce that $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p', g, M') \implies \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p', \neg\phi)$ is valid and that $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p', g, M')$ is satisfiable. Again we argue that we can add $\mathsf{tr}^{\mathsf{lra}}_{\mathsf{m}}(p', g, M') \wedge \mathsf{tr}^{\mathsf{lra}}_{\mathsf{f}}(p', \neg\phi)$ to the conjunction such that the formula remains satisfiable. That is, we find that

$$\mathsf{tr}_{\mathsf{U}}(\varrho, M, z, Z', \Pi, g) \wedge \mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p_{\mathsf{f}}, g, \varrho'(M)@t) \wedge \mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}(p', g, M') \\ \wedge \mathsf{sim}_{\mathsf{m}}((p_{\mathsf{f}}, g), (p', g), \delta) \wedge \mathsf{sim}_{\mathsf{t}}(w, (Z, g), (Z', g), \epsilon) \wedge \mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}(p', \neg\phi)$$

is satisfiable, which means that $\mathsf{tr}_{\Box}^{\epsilon,\delta}(\varrho, M, \phi)$ is not valid.

□

From the previous lemma it follows that $\varrho(M) \models_{\mathsf{seq}}^{\epsilon,\delta} \Box\,\phi$ holds iff $\mathsf{tr}_{\Box}^{\epsilon,\delta}(\varrho, M, \phi)$ is valid. As satisfiability of the first-order theory of real-closed fields [Tar51] is decidable, we get the following theorem. As we are mainly interested in general decidability, we did not investigate the complexity of our procedure. We refer to Lemma 2.4.2 for the complexity of FORCF.

Theorem 6.1.20. *For two values $\epsilon, \delta \in \mathbb{R}_{>0}$ it is decidable whether an MLSLS formula holds globally in an MLSLS transition sequence with spatial robustness δ and temporal robustness ϵ.*

6.2 Discussion

Spatio-temporal robustness has been studied before for more abstract formalisms [DM10; Que13]. However, here the data for which we want to achieve robustness have a specific meaning because the underlying model of MLSLS is dedicated to modelling motorway traffic. To this end, we study spatio-temporal robustness, taking the meaning of data into account.

In real-time systems we distinguish between time-driven and event-driven real-time systems [Kop91]. In MLSLS we have two kinds of data values: the event-driven values are claims, reservations and the acceleration. The time-driven values are position, speed and the braking distance (given by the sensor function). We study temporal robustness only for claims and reservations. For this we use the methodology from timed languages, where time stamps are perturbed [GHJ97]. Additionally, we study spatial robustness for the time-driven values position and braking distance in a static "timeless" manner at the level of traffic snapshots. In [FP09] such a "timeless" approach to spatial robustness has been done for Metric Temporal Logic.

One of the goals in the definition of original MLSL was to reduce complexity of spatial reasoning by separating the spatial aspects from the car dynamics

[HLO+11]. In this sense, the introduction of temporal robustness by perturbing time stamps seems well suited for MLSL because we separate temporal robustness from spatial robustness, which simplifies reasoning.

A disadvantage of our approach is that at the linking of time-driven and event-driven values (here acceleration, which is event-driven and affects future evolution of time-driven values) we do not achieve temporal robustness, as it affects spatial robustness. In other words, we do not allow perturbation of actions that change the acceleration of a car.

For our approach to temporal robustness we consider similarity of timed words. A definition how to quantify similarity of timed words is defined in [GHJ97]. However, there the requirement is made that timed words have an infinite distance if they do not agree on the order of events. In [AFS04] the authors define a quantitative notion of (bi)similarity. However, they define that the i-th position in one sequence is compared to the i-th position in another sequence, i.e. they do not consider that the order of events may not always be relevant. Since in our thesis the timed words originate from a distributed system, which makes it unreasonable to always consider the order of events as relevant. Hence, we define an independence relation in the sense of [Maz86] and in our definition of similarity allow for independent events to change their order. To the best of our knowledge, a quantitative comparison of timed words under consideration of causality has not been used before.

6.3 Related Work

There exists a lot of work on monitoring temporal properties in dense time formalisms. This was then extended to checking how robustly (in the spatial sense) a signal satisfies a Metric Temporal Logic formula [FP09]. In [DM10] the authors extend this to consider spatio-temporal robustness of Signal Temporal Logic (STL), a temporal logic that works with dense time and dense data defined in [MN04]. In [DM10] the authors compute a spatio-temporal robustness degree for a given real-time signal and an STL formula. The signal is given as a sequence of time-stamped measurements and the value of the signal in between these measurements is linearly interpolated. The authors propose a new real-valued valuation function for STL. Using this new semantics they provide algorithms to compute for a given signal and formula the spatial robustness degree δ, and the temporal robustness degree ϵ for a given desired spatial robustness δ. That means, if a signal s and a formula ϕ have a robustness degree of (ϵ, δ), then we can perturb the signal temporally by ϵ and spatially by δ and the resulting signal still satisfies ϕ. Formally, to define temporal robustness they use a retiming function $\alpha : \mathbb{T} \to \mathbb{T}$ that temporally distorts a signal s to a new signal s' by defining $s'(t) = s(\alpha(t))$.

In [Que13; QFD11] the authors define a notion of spatio-temporal similarity for traces of hybrid systems. They use this to define ϵ-δ-refinement of hybrid systems. A hybrid system α ϵ-δ-refines a hybrid system β iff for every trace σ_α of α, there is a trace σ_β of β such that σ_α and σ_β are ϵ-δ-similar. Their notion of spatio-temporal similarity is close to the notion of spatio-temporal robustness used in [DM10]. The authors define a variation of MTL. Then they show for a formula ϕ, if β satisfies ϕ and α ϵ-δ-refines β, then α satisfies $\phi^{\epsilon,\delta}$, where $\phi^{\epsilon,\delta}$ is created by shaking the constants in ϕ.

In [AD14] the authors perform online monitoring of spatial properties for a driving car. In contrast to our work, they take a very low level view (little abstraction) and they cannot easily check arbitrary spatial properties.

7 Model Checking Temporal Properties with a Multi-Valued Semantics

In Chapters 5 and 6 we considered temporal properties requiring that something holds globally. We used a Boolean semantics for these properties. That is, they either could be satisfied or not. Another typical temporal property is that something happens *eventually*. In this chapter we consider properties of the form that something happens *soon*. To formalise such properties we use discounting in temporal logics using a multi-valued semantics.

In economics, discounting represents that money earned sooner can be reinvested earlier and hence yields more revenue than money earned later. Discounting has been introduced into temporal logics to represent that something good happening earlier is more important than similar events happening later [dHM03]. A typical example is a rail-road crossing. Consider the property “eventually the gates are open”. While a controller leaving the gates closed an hour after the train has passed is safe and alive, it is not useful. We can use discounting to express that the controller should not wait unnecessarily long before opening the gates. The discount here is a scalar defining the slope of an exponential function assigning weights to events based on their (relative) time of occurrence. In [dFH+05; dHM03; ABK14] such weighted evaluation of temporal properties has been described as quantifying the *temporal quality* of a system.

So far discounting in logics only has been studied for discrete-time temporal logics (LTL, CTL*, μ-calculus) [dFH+05; ABK14; MR14; Man12; dHM03]. Here, we study discounting in the dense-time logic Duration Calculus (DC) [ZHR91]. Our interest in DC arises from its expressiveness, being able to express properties of accumulated durations and its closeness to MLSL.

We define *Discounted Duration Calculus* (DDC), where the truth value is real-valued in the interval $[0, 1]$, instead of Boolean. A truth value closer to

1 means higher temporal quality. With DDC we can express properties such as $\Diamond^d \phi$ (meaning "*soon* with *discount* $d \in [0,1]$ the system satisfies ϕ"), where ϕ is a DDC formula and $\Diamond$ is the *right neighbourhood modality* from [ZH97]. To evaluate the truth value of $\Diamond^d \phi$ on the interval $[t_0, t_1]$ we search for a neighbouring interval $[t_1, t_2]$ such that the discounting factor $d^{t_2 - t_1}$ multiplied with the truth value of ϕ on $[t_1, t_2]$ is maximal. We point out that we use exponential discounting because this is the most common from of discounting. However, other discounting mechanisms are possible.

This chapter is based on the work published in [OFH16]. In this thesis we added proofs left out in the publication. Additionally, we slightly generalised the notion of "whenever E happens, then ϕ holds". Finally, we do not connect DDC through timed words with timed automata. Instead, we directly connect runs of timed automata with trajectories, which we use to define the semantics of DDC.

7.1 Discounted Duration Calculus

To introduce discounting into a temporal logic we found it helpful to have a notion of "now" and discount as we go into the future. Hence, we use an adapted version of DC, where the chop operator is replaced by the right neighbourhood modality. As atomic formulas, we allow comparison of linear combinations of durations with constants. For the following definition we recall that $\mathbb{B}(V)$ is the set of propositional logic formulas using the set of state variables V as variables (cf. Page 14). Further, for $S \in \mathbb{B}(V)$ we say that S is a *state expression*.

Definition 7.1.1 (Syntax of DDC)**.** For $n \in \mathbb{N}, i \in \{0, \ldots, n\}$ and a set of state variables V let $d, k_i, c \in \mathbb{Q}$, where $d \in [0,1]$, $\gtrsim \in \{\geq, >\}$ and $S_i \in \mathbb{B}(V)$. Then the formulas ϕ of *Discounted Duration Calculus* (abbreviated *DDC*) are defined by the grammar

$$\phi ::= \Sigma_{i=0}^{n} k_i * \textstyle\int S_i \gtrsim c \mid \neg\phi \mid \phi \vee \phi \mid \Diamond^d \phi \ .$$

Let $\mathrm{DDC}_{<1}$ denote the fragment of DDC where all discounts d are strictly less than 1. △

In [OFH16] we defined the semantics of DDC on trajectories, which we derived from timed words. The timed words then were generated or accepted by timed automata. Here, we do not use timed words to connect timed automata to

DDC. This has two reasons. The first reason is that the use of timed words in [OFH16] is inconsistent with our use of timed words in previous chapters of this thesis. In the previous chapters we use timed words as a time-stamped sequences of events, where each event affects the current state. In [OFH16] we use timed words as a time-stamped sequence of states. The other reason is that [OFH16] is not clear in how timed words are generated. Thus, in this thesis we decided to connect runs of timed automata directly to trajectories. Our approach to connect timed automata to DDC closely follows [Hoe06]. There the author connects Duration Calculus to so-called Phase Event Automata, which are similar to timed automata.

The *semantics of a DDC formula* ϕ on the basis of a trajectory τ is a function

$$\tau(\phi) : \mathit{Intv} \to [0,1] \ ,$$

where $\mathit{Intv} = \{[t,t'] \subseteq \mathsf{span}(\tau) \mid t \leq t'\}$ denotes the set of bounded and closed real intervals contained in $\mathsf{span}(\tau)$. The function $\tau(\phi)$ assigns to $[t,t']$ a *satisfaction value* in the real interval $[0,1]$, where closer to 1 means better.

Discounts d occur only in connection with the right neighbourhood modality $\Diamond^{d}\phi$, which expresses that an adjacent interval to the right of the current interval satisfies ϕ. The discount d is used to decrease the satisfaction value as the length of the adjacent interval necessary to satisfy ϕ. The modal formula $\Diamond^{d}\phi$ can be understood as "*soon* ϕ holds".

Definition 7.1.2 (Semantics of DDC)**.** For $n \in \mathbb{N}, i \in \{0,\dots,n\}$ let $d, k_i, c \in \mathbb{Q}$, where $d \in [0,1]$, $\gtrsim \in \{\geq, >\}$ and $S_i \in \mathbb{B}(V)$. The *semantics of a formula* ϕ, given a trajectory τ and an interval $[t_0,t_1]$, yields a value $\tau(\phi)\,[t_0,t_1] \in [0,1]$ defined inductively as follows:

$$\begin{aligned}
\tau(\Sigma_{i=0}^{n} k_i \textstyle\int S_i \gtrsim c)\,[t_0,t_1] &= \begin{cases} 1 & \text{if } \Sigma_{i=0}^{n} k_i \int_{t=t_0}^{t_1} \tau(S_i)(t)\,\mathrm{d}t \gtrsim c \ , \\ 0 & \text{otherwise} \ , \end{cases} \\
\tau(\neg\phi)\,[t_0,t_1] &= 1 - \tau(\phi)\,[t_0,t_1] \ , \\
\tau(\phi_0 \vee \phi_1)\,[t_0,t_1] &= \mathsf{max}(\tau(\phi_0)\,[t_0,t_1], \tau(\phi_1)\,[t_0,t_1]) \ , \\
\tau(\Diamond^{d}\phi)\,[t_0,t_1] &= \sup_{\substack{t_2 \in \mathsf{span}(\tau),\\ t_2 \geq t_1}} (d^{t_2 - t_1} * \tau(\phi)\,[t_1,t_2]) \ .
\end{aligned}$$

△

We use common abbreviations, such as true, false, $\wedge$, $<$ and $\leq$. If we have a discount of 1 we usually do not write it explicitly. That is, for a DDC formula

ϕ we define $\Diamond\,\phi \equiv \Diamond^{\,1}\phi$. We define as abbreviation a modality

$$\Box^{\,d}\phi \equiv \neg\Diamond^{\,d}\neg\phi \ ,$$

which can be understood as "ϕ holds for a *long* time". For some interval $[t_0, t_1]$ the semantics is

$$\tau(\Box^{\,d}\phi)\,[t_0, t_1] = 1 - \sup_{\substack{t_2 \in \mathsf{span}(\tau),\\ t_2 \geq t_1}} \left(d^{t_2 - t_1} * (1 - \tau(\phi)\,[t_1, t_2])\right) \ .$$

We point out that in the formula above the supremum searches for a small $t_2 \geq t_1$ that makes the truth value of $\tau(\phi)$ on $[t_1, t_2]$ small. Further, the greater the interval $[t_1, t_2]$ is chosen, the greater the truth value of $\Box^{\,d}\phi$ becomes. Note that the truth value of $\Box^{\,d}\phi$ increases with the decrease of d, while the truth value of $\Diamond^{\,d}\phi$ decreases with the decrease of d.

To express that a state expression S holds throughout an interval, we use the abbreviation

$$\lceil S \rceil \equiv \textstyle\int \neg S = 0 \wedge \ell > 0 \ ,$$

where ℓ is an abbreviation of $\int \mathsf{true}$ for an arbitrary state expression S'.

From [Hoe06; ZH04] we take the idea to encode events, which we represent as value changes of state variables. That means that at time t the event S occurs if there is t_0, t_1 with $t_0 < t < t_1$ such that the truth value of S on the interval $[t_0, t)$ is different from the truth value of S on $[t, t_1)$. We formalise this as

$$\updownarrow S \equiv \lceil \neg S \rceil \wedge \Diamond \lceil S \rceil \vee \lceil S \rceil \wedge \Diamond \lceil \neg S \rceil \ .$$

Note that here $\updownarrow S$ requires proper intervals, while in [Hoe06] point-intervals are required. The difference stems from the fact that here S "happens" at the right end of the current interval, while in [Hoe06] S "happens" during the current interval.

With $\Diamond\,\Diamond\,\phi$ we express that on some right interval, which may or may not be adjacent to the current interval, ϕ holds. We shall use the abbreviation $\mathrm{F}_{\updownarrow S}\,\phi$ to denote that there is some future point in time t, where S "happens" and ϕ holds. That is, ϕ holds on $[t, t]$ and S changes its value at t. We formalise this as

$$\mathrm{F}_{\updownarrow S}\,\phi \equiv \Diamond\,\Diamond\,(\updownarrow S \wedge \Diamond\,(\ell = 0 \wedge \phi)) \ .$$

Further, we define

$$\mathrm{G}_{\updownarrow S}\,\phi \equiv \neg\mathrm{F}_{\updownarrow S}\,\neg\phi \ .$$

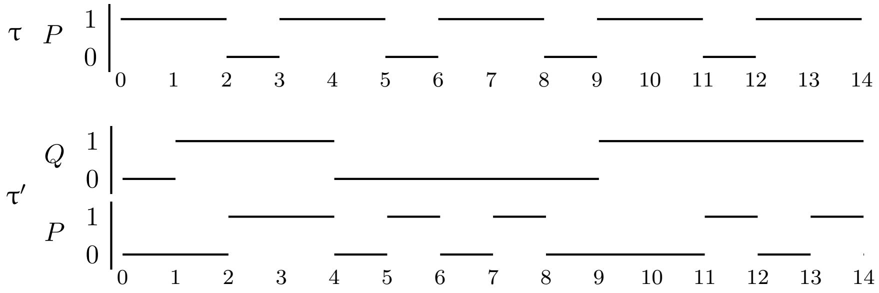


Figure 7.1: Graphical representation of two trajectories. We assume that the variables do not change their values after time point 14.

To understand the intuition of this formula we point out that $\mathrm{G}_{\updownarrow S}\,\phi$ is equivalent to $\Box\Box(\updownarrow S \implies \Box(\ell = 0 \implies \phi))$. Thus, the formula $\mathrm{G}_{\updownarrow S}\,\phi$ means that whenever S happens, then also ϕ holds.

Example 7.1.3. Consider the three formulas ϕ_0, ϕ_1 and ϕ_2 shown below:

- $\phi_0 \equiv \Diamond^{0.8}(\int P \geq 3)$, which means "soon P holds for at least 3 time units",
- $\phi_1 \equiv \Diamond^{0.9}\,\Box^{0.8}(\int P - \int \neg P \leq 3)$, which means "soon P should hold no more than 3 time units more than $\neg P$, for a long time", and
- $\phi_2 \equiv \mathrm{G}_{\updownarrow Q}\,\Diamond^{0.8}(\int P \geq 2)$, which means "every time Q changes its value, then soon P holds for at least 2 time units".

We evaluate these formulas on the trajectories depicted in Figure 7.1. In our examples we use $\approx$ to denote equality after rounding to two positions behind decimal point.

Evaluate ϕ_0 on τ: The earliest point when $\int P \geq 3$ is satisfied is at $t = 4$. We calculate:

$$\begin{aligned} & \tau(\Diamond^{0.8}(\textstyle\int P \geq 3))\ [0,0] \\ = \ & \mathsf{sup}_{t \in \mathsf{span}(\tau)}(0.8^t * \tau(\textstyle\int P \geq 3)\,[0,t]) \\ = \ & 0.8^4 \approx 0.41 \end{aligned}$$

Evaluate ϕ_1 on τ: In ϕ_1 the inner modality is given t_0 and chooses the smallest t_1 such that $\int P - \int \neg P \leq 3$ is violated. The outer modality chooses t_0 such that the product of its discount 0.9^{t_0} multiplied with the truth value archived by the inner modality becomes maximal. We calculate (assuming that $t_0, t_1 \in \mathsf{span}(\tau)$):

$$\begin{aligned}
& \tau(\Diamond^{0.9}\, \Box^{0.8}(\textstyle\int P - \int\neg P \leq 3))\,[0,0] \\
= \;& \mathsf{sup}_{t_0 \geq 0}(0.9^{t_0} * (1 - \mathsf{sup}_{t_1 \geq t_0}(0.8^{t_1 - t_0} * (1 - \tau(\textstyle\int P - \int\neg P \leq 3)\,[t_0, t_1])))) \\
= \;& 0.9^2 * (1 - 0.8^{12-2} * (1 - \tau(\textstyle\int P - \int\neg P \leq 3)\,[t_0, t_1]) \\
= \;& 0.9^2 * (1 - 0.8^{12-2} * (1 - 0)) \approx 0.72
\end{aligned}$$

Evaluate ϕ_2 on τ': We evaluate $\phi' \equiv \Diamond^{0.8}(\int P \geq 2)$ on all point intervals $[t, t]$, where Q changes its value. For τ' these points are $1, 4$ and 9. The truth value is $\mathsf{min}(\tau'(\phi')\,[1,1], \tau'(\phi')\,[4,4], \tau'(\phi')\,[9,9])$, which evaluates to $\mathsf{min}(0.8^{4-1}, 0.8^{8-4}, 0.8^{14-9}) \approx 0.33$. $\triangle$

Comparison of Duration Calculus and Discounted Duration Calculus

Our logic Discounted Duration Calculus (DDC) is strongly inspired by Duration Calculus (DC). We briefly consider the common features and the differences of these two logics. The main *differences of DC and DDC* are that

- DC features the contracting chop modality, while DDC uses the expanding right neighbourhood modality, and
- DC has a Boolean semantics, while DDC has a real-valued semantics.

The main *commonalities of DC and DDC* are that

- both are interval-based real-time temporal logics and
- DDC uses the integral operator first defined and used in temporal logics in DC.

7.2 Approximate Model Checking

In this section we prove that for a relevant fragment of DDC it is approximable how well a model satisfies a formula, where the model is given as a timed automaton [AD94].

As model we use timed automata that have state variables that hold in states. Additionally, our timed automata have a set of allowed initial clock valuations, where the initial value of a clock may be different from 0. Further, we assume that our timed automata are *strongly non-Zeno* [AMP+98]. This is the case iff there is a nonzero constant $c \in \mathbb{R}_{>0}$ such that in every control cycle at least c units of time passes. Formally, for every path $l_0 \xrightarrow{e_0} \dots \xrightarrow{e_{n-1}} l_n$ with $l_0 = l_n$ there is an edge that resets some clock x and an edge or a location with a constraint $x \geq c$. For ease of exposition we assume this constant to be a natural number greater than 0. We assume strong non-Zenoness because we reduce general approximative model checking to approximative time-bounded model checking. Then, if the timed automaton is strongly non-Zeno, we can reduce approximative time-bounded model checking to approximative step bounded model checking. Additionally, we assume that our timed automata contain no deadlocks, which is standard.

Classically, in language based model checking we check if all behaviours of a model (here a timed automaton A) satisfy a property ϕ. That is, we check $\mathcal{L}(A) \subseteq \mathcal{L}(\phi)$. This is equivalent to $\mathcal{L}(A) \cap \overline{\mathcal{L}(\phi)} = \emptyset$. In model checking we search the complete state space and either find a witness in $\mathcal{L}(A) \cap \overline{\mathcal{L}(\phi)}$, which proves that A does not satisfy ϕ. Or there is no such witness. In this case A satisfies the property ϕ. Thus, we arrive at the following definition of a model checking function. As we work with real-valued truth values, here model checking gives a value in the interval $[0, 1]$. Further, here we consider trajectories instead of words.

Definition 7.2.1 (Model Checking Timed Automata)**.** Let A be a timed automaton and ϕ a DDC formula. We define *model checking* as computing

$$\mathsf{mc}(A, \phi) = \begin{cases} \mathsf{inf}_{\tau \in \mathcal{T}(A)}(\tau(\phi)\,[0, 0]) & \text{if } \mathcal{T}(A) \neq \emptyset \ , \\ 1 & \text{otherwise} \ . \end{cases}$$

Note that $\mathsf{mc}(A, \phi) \in [0, 1]$. △

We give our definition of approximate model checking.

Definition 7.2.2 (Approximate Model Checking)**.** Let A be a timed automaton, ϕ be a $\text{DDC}_{<1}$ formula and let $\epsilon \in (0, 1]$ be the desired precision. Then *approximate model checking* is the task to compute a truth value $v \in \mathbb{R}$ with $0 \leq v \leq 1$ such that

$$v \in \mathsf{mc}(A, \phi) \pm \epsilon \ ,$$

where $\mathsf{mc}(A, \phi) \pm \epsilon$ is the set of all real values that are ϵ-close to $\mathsf{mc}(A, \phi)$ (cf. Page 12). △

Next we define the time-bounded prefix for trajectories similar to how we defined it for timed words.

Definition 7.2.3 (Time-Bounded Prefix)**.** Let V be a set of state variables and an interval $T \subseteq \mathbb{T}$, where we allow T to be infinite. For a trajectory $\tau : V \to T \to \{0,1\}$ and $\delta \in \mathbb{T}$ we define the *time-bounded prefix of* τ *until* δ as

$$\mathsf{pre}(\tau, \delta) = \{P \mapsto ([0,\delta] \lhd \tau(P)) \mid P \in V\} \ .$$

We abbreviate $\mathsf{pre}(\tau,\delta)$ with τ_δ. △

The intuition of the definition above is that we cut off everything after δ. If $\delta \notin \mathsf{span}\,\tau$, then the domain restriction ensures that the trajectory remains unchanged.

We want to compute $v \in \mathsf{sup}_{\tau \in \mathcal{T}(A)}(\tau(\phi)\,[0,0]) \pm \epsilon$. However, before that we introduce the following lemma, which states that we can bound the error when computing the satisfaction of a formula by a trajectory on a shortened trajectory. The error done depends on the length of the span of the shorter trajectory and on the maximal discount occurring in the formula.

Example 7.2.4. Consider the formula $\phi_0 \equiv \Diamond^{0.8} \textstyle\int P \geq 3$ and the trajectory τ from Example 7.1.3. Then $\tau_3(\Diamond^{0.8} \textstyle\int P \geq 3)\,[0,0] = 0$ and $\tau_4(\Diamond^{0.8} \textstyle\int P \geq 3)\,[0,0] = 0.8^4$. The difference between $\tau_3(\phi_0)\,[0,0]$ and $\tau_4(\phi_0)\,[0,0]$ is less than the discount to the power of the length of the shorter trajectory, i.e. less than 0.8^3. △

We proceed to the lemma mentioned earlier.

Lemma 7.2.5. *Given a DDC$_{<1}$ formula ϕ, a trajectory τ, two bounds $\delta, \delta' \in \mathbb{R}_{>0} \cup \{\infty\}$ with $\delta \leq \delta'$ we have*

$$|\tau_\delta(\phi)\,[0,0] - \tau_{\delta'}(\phi)\,[0,0]| \leq d^\delta \ ,$$

where d is the maximal discount occurring in ϕ and $d = 1$ if there is none.

Proof. We first prove a slightly different claim and show later that this claim implies the lemma. The claim is that for all $\epsilon \in \mathbb{R}_{>0}$ there is $\epsilon' \in (0,\epsilon]$ such that

$$|\tau_\delta(\phi)\,[0,0] - \tau_{\delta'}(\phi)\,[0,0]| \leq d^\delta + \epsilon' \ .$$

The idea behind ϵ and ϵ' is that we can show that the difference does not exceed d^δ by “much”. In other words, the amount the difference exceeds d^δ is arbitrary small. We use both values near the end of the proof. We generalise the inequality above to an arbitrary current interval to have a usable induction hypothesis. For all intervals $I = [t_0, t_1]$ with $t_0 \leq t_1 \leq \delta$ and $t_0, t_1 \in \mathbb{R}_{\geq 0}$ and all $\epsilon \in \mathbb{R}_{>0}$ there is $\epsilon' \in (0, \epsilon]$ such that

$$|\tau_\delta(\phi)\, I - \tau_{\delta'}(\phi)\, I| \leq d^{\delta - t_1} + \epsilon' \ ,$$

where $I = [t_0, t_1]$, $t_0 \leq t_1 \leq \delta$ and $t_0, t_1 \in \mathbb{R}_{\geq 0}$. We proceed by induction on the structure of ϕ.

Induction base.

Let $\phi \equiv \sum_{i=0}^{n} k_i \textstyle\int S_i \gtrsim c$. Then $\tau_\delta(\phi)\, I$ and $\tau_{\delta'}(\phi)\, I$ ignore everything outside I, which means that both values are equal.

Induction hypothesis.

For all ϕ_1 and ϕ_2, $\delta, \delta' \in \mathbb{R}_{>0} \cup \{\infty\}$ with $\delta \leq \delta'$, all intervals $[t_0, t_1] = I \subseteq [0, \delta]$ and all $\epsilon \in \mathbb{R}_{>0}$ there is $\epsilon' \in (0, \epsilon]$ such that

$$|\tau_\delta(\phi_1)\, I - \tau_{\delta'}(\phi_1)\, I| \leq d_1^{\delta - t_1} + \epsilon' \ , \tag{IH1}$$

$$|\tau_\delta(\phi_2)\, I - \tau_{\delta'}(\phi_2)\, I| \leq d_2^{\delta - t_1} + \epsilon' \ . \tag{IH2}$$

where d_i is the maximal discount occurring in ϕ_i.

Induction step.

Case 1 ($\phi \equiv \neg\phi_1$). Now, $|\tau_\delta(\neg\phi_1)\, I - \tau_{\delta'}(\neg\phi_1)\, I|$ evaluates to $|\tau_{\delta'}(\phi_1)\, I - \tau_\delta(\phi_1)\, I|$. The claim follows from the IH.

Case 2 ($\phi \equiv \phi_1 \vee \phi_2$). The absolute difference $|\tau_\delta(\phi)\, I - \tau_{\delta'}(\phi)\, I|$ evaluates to $|\,\mathsf{max}(\tau_\delta(\phi_1)\, I, \tau_\delta(\phi_2)\, I) - \mathsf{max}(\tau_{\delta'}(\phi_1)\, I, \tau_{\delta'}(\phi_2))|$. We do a case distinction on what the two maxima evaluate to, i.e. whether both sides of the difference evaluate the same subformula.

Subcase 2.1 ($\tau_\delta(\phi_1)\, I \geq \tau_\delta(\phi_2)\, I$ and $\tau_{\delta'}(\phi_1)\, I \geq \tau_{\delta'}(\phi_2)$). In this case $|\tau_\delta(\phi)\, I - \tau_{\delta'}(\phi)\, I|$ is equal to $|\tau_\delta(\phi_1)\, I - \tau_{\delta'}(\phi_1)\, I|$. Using the IH it follows that $|\tau_\delta(\phi_1)\, I - \tau_{\delta'}(\phi_1)\, I| \leq d_1^{\delta - t_1} + \epsilon'$. As $d_1 \leq d$ we conclude $d_1^{\delta - t_1} + \epsilon' \leq d^{\delta - t_1} + \epsilon'$, which finishes this case.

Subcase 2.2 ($\tau_\delta(\phi_1)\, I < \tau_\delta(\phi_2)\, I$ and $\tau_{\delta'}(\phi_1)\, I < \tau_{\delta'}(\phi_2)$). This case is symmetric to the previous one.

Subcase 2.3 ($\tau_\delta(\phi_1)\, I \geq \tau_\delta(\phi_2)\, I$ and $\tau_{\delta'}(\phi_1)\, I < \tau_{\delta'}(\phi_2)$). In this case the difference evaluates to $|\tau_\delta(\phi_1)\, I - \tau_{\delta'}(\phi_2)\, I|$. We collect the assumptions of this case in the following equations:

$$\tau_\delta(\phi_1)\, I \geq \tau_\delta(\phi_2)\, I \ , \tag{7.1}$$

$$\tau_{\delta'}(\phi_1)\, I < \tau_{\delta'}(\phi_2)\, I \ . \tag{7.2}$$

Using the induction hypothesis and the assumptions of this case we get the following two chains of inequalities, which bound $\tau_\delta(\phi_1)\, I$ from below and from above:

$$\tau_\delta(\phi_1)\, I \overset{(7.1)}{\geq} \tau_\delta(\phi_2)\, I \overset{IH2}{\geq} \tau_{\delta'}(\phi_2)\, I - d_2^{\delta - t_1} - \epsilon' \ ,$$
$$\tau_\delta(\phi_1)\, I \overset{IH1}{\leq} \tau_{\delta'}(\phi_1)\, I + d_1^{\delta - t_1} + \epsilon' \overset{(7.2)}{<} \tau_{\delta'}(\phi_2)\, I + d_1^{\delta - t_1} + \epsilon' \ .$$

The two inequalities above mean that the absolute difference of $\tau_\delta(\phi_1)\, I$ and $\tau_{\delta'}(\phi_2)\, I$ is at most $\mathsf{max}(d_1^{\delta-t_1}, d_2^{\delta-t_1}) + \epsilon'$. As $\mathsf{max}(d_1, d_2) = d$, the lemma follows.

Subcase 2.4 ($\tau_\delta(\phi_1)\, I < \tau_\delta(\phi_2)\, I$ and $\tau_{\delta'}(\phi_1)\, I \geq \tau_{\delta'}(\phi_2)$). This case is symmetric to the previous case.

We conclude that the lemma holds for $\phi \equiv \phi_1 \vee \phi_2$.

Case 3 ($\phi \equiv \Diamond^{d_0} \phi_1$). We proceed to the last case $\phi \equiv \Diamond^{d_0} \phi_1$ with $d_0 \leq d$ and $d_0 \in [0,1]$. We define two functions $f : X \to [0,1], f' : X' \to [0,1]$ with $X = \{x \in \mathbb{R} \mid t_1 \leq x \leq \delta\}$ and $X' = \{x \in \mathbb{R} \mid t_1 \leq x \leq \delta'\}$ as

$$f(t) = d_0^{t-t_1} * \tau_\delta(\phi_1)\, [t_1, t] \ ,$$
$$f'(t) = d_0^{t-t_1} * \tau_{\delta'}(\phi_1)\, [t_1, t] \ .$$

Then we have to show

$$|\,\mathsf{sup}\, f - \mathsf{sup}\, f'| \leq d^{\delta - t_1} + \epsilon' \ ,$$

which is equivalent to

$$\mathsf{sup}\, f \leq \mathsf{sup}\, f' + d^{\delta - t_1} + \epsilon' \ \text{ and} \tag{7.3}$$
$$\mathsf{sup}\, f' \leq \mathsf{sup}\, f + d^{\delta - t_1} + \epsilon' \ . \tag{7.4}$$

Subcase Equation (7.3). We start by showing $\mathsf{sup}\, f \leq \mathsf{sup}\, f' + d^{\delta - t_1} + \epsilon'$. Let $s \in \mathsf{Seq}_\infty\, X$ be an infinite sequence with $\mathsf{lim}_{i \to \infty} f(s(i)) = \mathsf{sup}\, f$ and note that $X \subseteq X'$, which means that we can use s together with f'. As $f(s(i))$ with growing i converges towards $\mathsf{sup}\, f$, there is $n \in \mathbb{N}$ such that the sequence from n onwards is $\frac{\epsilon'}{2}$-close to $\mathsf{sup}\, f$. Let $t = s(n)$. In the steps below we use the IH with the precision set to $\epsilon_0 = \frac{\epsilon'}{2}$. Also, note that $d_0 \leq d$.

We get

$$
\begin{aligned}
|f(t) - f'(t)| &= |d_0^{t-t_1} * \tau_\delta(\phi_1)\,[t_1, t] - d_0^{t-t_1} * \tau_{\delta'}(\phi_1)\,[t_1, t]| \\
&= d_0^{t-t_1} * |\tau_\delta(\phi_1)\,[t_1, t] - \tau_{\delta'}(\phi_1)\,[t_1, t]| \\
&\overset{IH}{\leq} d_0^{t-t_1} * d^{\delta-t} + \epsilon_0' \leq d^{t-t_1} * d^{\delta-t} + \epsilon_0' \leq d^{t-t_1+(\delta-t)} + \epsilon_0' \\
&\leq d^{\delta-t_1} + \epsilon_0' \ ,
\end{aligned}
\tag{7.5}
$$

where $\epsilon_0' \in (0, \frac{\epsilon'}{2}]$.

As $f(t)$ and $\mathsf{sup}\, f$ are $\frac{\epsilon'}{2}$-close, we know $|\mathsf{sup}\, f - f'(t)| \leq |f(t) - f'(t)| + \frac{\epsilon'}{2}$. Combining this with Equation (7.5) we get

$$
|\mathsf{sup}\, f - f'(t)| \overset{\frac{\epsilon'}{2}\text{-close}}{\leq} |f(t) - f'(t)| + \frac{\epsilon'}{2} \overset{(7.5)}{\leq} d^{\delta-t_1} + \epsilon_0' + \frac{\epsilon'}{2} \leq d^{\delta-t_1} + \epsilon' \ ,
$$

which implies $\mathsf{sup}\, f \leq f'(t) + d^{\delta-t_1} + \epsilon'$. As $f'(t) \leq \mathsf{sup}\, f'$ we can weaken the upper bound to $\mathsf{sup}\, f \leq \mathsf{sup}\, f' + d^{\delta-t_1} + \epsilon'$.

Subcase Equation (7.4). We continue by showing $\mathsf{sup}\, f' \leq \mathsf{sup}\, f + d^{\delta-t_1} + \epsilon'$. This direction mostly works as the other one. However, we have to watch out as the domain of f' may be larger than the domain of f. Let $s' \in \mathsf{Seq}_\infty X'$ be an infinite sequence converging to the supremum of f'. Then there is n such that $t = s'(n)$ is $\frac{\epsilon'}{2}$-close to $\mathsf{sup}\, f'$. We make a case distinction on whether $t > \delta$.

If $t > \delta$, then $f'(t)$ and $f(\delta)$ both are less or equal $d^{\delta-t_1}$, which implies the desired property. Formally, as all formulas evaluate on all trajectories to some value ≤ 1, we know $f'(t) \leq d_0^{t-t_1}$. Next, we conclude $d_0^{t-t_1} \leq d^{\delta-t_1}$, which implies $\mathsf{sup}\, f' \leq d^{\delta-t_1} + \frac{\epsilon'}{2}$. Similarly, we know $f(\delta) \leq d^{\delta-t_1}$, which means $|f(\delta) - \mathsf{sup}\, f'| \leq d^{\delta-t_1} + \frac{\epsilon'}{2}$. This difference gives us an upper bound for $\mathsf{sup}\, f'$, i.e. $\mathsf{sup}\, f' \leq f(\delta) + d^{\delta-t_1} + \frac{\epsilon'}{2}$. As $f(\delta) \leq \mathsf{sup}\, f$, we can weaken this upper bound to conclude $\mathsf{sup}\, f' \leq \mathsf{sup}\, f + d^{\delta-t_1} + \epsilon'$.

If $t \leq \delta$ we can proceed as in the case of Equation (7.3) from Equation (7.5) onwards.

This finishes the case $\phi \equiv \lozenge^{d_0} \phi_1$.

Now, if for all $\epsilon \in \mathbb{R}_{>0}$ there is $\epsilon' \in (0, \epsilon]$ such that $|\tau_\delta(\phi)\, I - \tau_{\delta'}(\phi)\, I| \leq d^{\delta-t_1} + \epsilon'$, then we also know $|\tau_\delta(\phi)\, I - \tau_{\delta'}(\phi)\, I| \leq d^{\delta-t_1}$. Otherwise, we would have an ϵ

for which the property does not hold. By choosing $I = [0,0]$ and $t_1 = 0$ we see that the lemma holds. $\square$

With the previous lemma we can approximate the satisfaction of a DDC$_{<1}$ formula by an infinite trajectory. For this we compute the point in time $\delta = \mathsf{log}_d\, \epsilon$ such that the value of v is almost not affected by any suffix of the trajectory starting at time δ. This is possible because all modalities in DDC$_{<1}$ are discounted by less than 1 and hence the effect of a trajectory on the truth value becomes less and less as time advances. Note that for other discounting functions, e.g. $\frac{1}{1+d*(t-t')}$, other computations are necessary. However, for any computable strictly monotonic discounting function with limit 0 such a point, after which the effect on the truth value is $\leq \epsilon$, is computable. From the previous lemma we get the following corollary.

Corollary 7.2.6. *Given a DDC$_{<1}$ formula ϕ and an allowed error ϵ, let d be the largest discount constant occurring in ϕ such that for all other discounts d' in ϕ we have $d' \leq d$ and let $\delta = \mathsf{log}_d\, \epsilon$. Then for any trajectory τ we have*

$$|\tau(\phi)\,[0,0] - \tau_\delta(\phi)\,[0,0]| \leq \epsilon \ .$$

To check to what extent a timed automaton satisfies a formula we use a bounded unfolding of the transition relation. The following lemma specifies which variables we use in the unfolding. The construction can be found, e.g., in [TMM02; BL11; KJN12].

Lemma 7.2.7 (Bounded Unfolding, [TMM02; BL11; KJN12])**.** *Given a timed automaton $A = (L, \rightarrow, \mathsf{Init}, I, \mathsf{Lab}, V, \Lambda, \mathcal{X})$, a step bound m and a time-bound δ let $\underline{t} = \langle \underline{t_0}, \ldots, \underline{t_m} \rangle$, $\underline{P} = \langle \underline{P_0}, \ldots, \underline{P_m} \rangle$ with $P \in V$ be sequences of LRA variables and let $\underline{V} = \{\underline{P} \mid P \in V\}$ be a set of such sequences. We can create a LRA formula $F_A(\underline{t}, \underline{V}, m, \delta)$ such that*

- *if A has a run $\pi = \langle ((l_0, \beta_0, \nu_0), t_0), \ldots, (l_m, \beta_m, \nu_m), t_m), \ldots \rangle$ with $t_m = \delta$, then*

$$\begin{array}{rl} & \{\underline{t_i} \mapsto t_i \mid i \in \{0, \ldots, m\}\} \\ \uplus & \{\underline{P_j} \mapsto \beta_j(P) \mid P \in V, j \in \{0, \ldots, m\}\} \end{array} \models F_A(\underline{t}, \underline{P}, m, \delta) \text{ and}$$

- *for any assignment $h \models F_A(\underline{t}, \underline{V}, m, \delta)$ the trajectory τ with $\tau(P)(t) = h(\underline{P_i})$, where $P \in V$, $h(\underline{t_i}) \leq t < h(\underline{t_{i+1}})$ and $t \leq \delta$ is the δ-prefix of some trajectory $\tau' \in \mathcal{T}(A)$, i.e. $\mathsf{pre}(\tau', \delta) = \tau$.*

The intuition of the lemma above is that the prefix until δ of a trajectory is described using variables $\underline{t_i}, \underline{P_i}$, for $0 \leq i \leq m$. If in the interval $[t_i, t_{i+1})$ the propositional variable P holds, then $\underline{P_i}$ is assigned the value true, and otherwise the value false. Note that as we assume that the timed automaton contains no deadlocks, any prefix of a run can be extended to an infinite run.

7.2.1 Encoding of the Semantics for Formulas

We encode the semantics of DDC in FOLRA. As the semantics of DDC uses exponentials, we cannot encode the exact semantics. However, we can approximate the truth value with finite but arbitrary high precision. We use this encoding to prove that approximative model checking of $\mathrm{DDC}_{<1}$ for strongly non-Zeno timed automata is computable.

The general idea is to introduce for each subformula a fresh variable. Then we constrain this variable to take the truth value of the formula on the given interval and the given trajectory. In more detail, for a $\mathrm{DDC}_{<1}$ formula ϕ and a current interval given as FOLRA terms θ_0 and θ_1 we define a FOLRA formula $x \;\mathsf{isSemOf}^m\; \phi\; \theta_0\; \theta_1$ expressing that the free variable x approximates the truth value of ϕ on $[\theta_0, \theta_1]$. This formula is defined by induction over the structure of ϕ. We connect this to the bounded unfolding of the transition relation via conjunction. That is, for $F_A(\underline{t}, \underline{V}, m, \delta) \wedge (x \;\mathsf{isSemOf}^m\; \phi\; 0\; 0)$ and a satisfying assignment h the value $h(x)$ approximates the truth value of ϕ on a trajectory τ with a matching run π of A, where $\underline{t}, \underline{V}$ represents the first δ time units of this trajectory and m is the number of transitions taken in the run π during the first δ time units.

We use the following FOLRA abbreviations to express that the free variable x is equal to the greater (resp. smaller) term of θ_1 and θ_2:

$$\begin{aligned}\mathsf{MAX}(x, \theta_1, \theta_2) &\equiv (\theta_1 < \theta_2 \implies x = \theta_2) \wedge (\theta_1 \geq \theta_2 \implies x = \theta_1)\ ,\\ \mathsf{MIN}(x, \theta_1, \theta_2) &\equiv (\theta_1 < \theta_2 \implies x = \theta_1) \wedge (\theta_1 \geq \theta_2 \implies x = \theta_2)\ .\end{aligned}$$

Encoding of $\tau(\Sigma_{j=0}^{n} k_j \int S_j \gtrsim c)\,[\theta_0, \theta_1]$

We show the encoding of $k \int S \gtrsim c$. The generalisation to linear combinations of durations is easily done in FOLRA. First, we need some abbreviations as preparation. We use

$$x \;\mathsf{isOverlap}_i\; \theta_0\; \theta_1 \equiv \exists y_0, y_1.\, \mathsf{MAX}(y_0, \underline{t_i}, \theta_0) \wedge \\ \mathsf{MIN}(y_1, \underline{t_{i+1}}, \theta_1) \wedge \mathsf{MAX}(x, 0, y_1 - y_2)$$

with $i \in \{0, \dots, m-1\}$ to specify that the variable free x measures the overlap of the intervals $[\theta_0, \theta_1]$ and $[\underline{t_i}, \underline{t_{i+1}}]$. For a state expression S let $\underline{S_i}$ be the FOLRA formula we get by replacing every occurrence of $P \in V$ by $\underline{P_i}$. Then we define

$$x \;\mathsf{isDur}_i\; S\; \theta_0\; \theta_1 \equiv (\underline{S_i} \implies (x \;\mathsf{isOverlap}_i\; \theta_0\; \theta_1)) \wedge (\neg \underline{S_i} \implies x = 0)$$

to specify that the fresh variable x measures how long S holds in the interval $[\theta_0, \theta_1] \cap [\underline{t_i}, \underline{t_{i+1}}]$. We lift the previous formula to measure how long S holds in the interval $[\theta_0, \theta_1]$ (without restricting the interval further). We define

$$x \;\mathsf{isDur}^m\; S\; \theta_0\; \theta_1 \equiv \big((\exists y_0, \dots, y_{m-1}.\, x = \sum_{i=0}^{m-1} y_i \wedge \bigwedge_{i=0}^{m-1} (y_i \;\mathsf{isDur}_i\; S\; \theta_0\; \theta_1)\big) \;.$$

Now we can define that in $x \;\mathsf{isSemOf}^m\; k\!\int\! S \gtrsim c\; \theta_0\; \theta_1$ the free variable x is 1 if the comparison is satisfied and 0 otherwise. We define

$$x \;\mathsf{isSemOf}^m\; k\!\int\! S \gtrsim c\, \theta_0\, \theta_1 \equiv \big((\exists y.\, (y \;\mathsf{isDur}^m\; S\, \theta_0\, \theta_1) \wedge k{*}y \gtrsim c) \implies x = 1\big) \\ \wedge \big(\neg(\exists y.\, (y \;\mathsf{isDur}^m\; S\; \theta_0\; \theta_1) \wedge k * y \gtrsim c) \implies x = 0\big) \;.$$

It is not difficult to generalise this to cover linear combinations of accumulated durations.

Encoding of $\tau(\neg\phi)\,[\theta_0, \theta_1]$

For negation we compute the truth value of the subformula and subtract the result from one. We define

$$x \;\mathsf{isSemOf}^m\; (\neg\phi)\; \theta_0\; \theta_1 \equiv \exists y.\, (y \;\mathsf{isSemOf}^m\; \phi\; \theta_0\; \theta_1) \wedge x = 1 - y \;.$$

Encoding of $\tau(\phi_0 \vee \phi_1)\,[\theta_0, \theta_1]$

For disjunction we use two quantified formulas to compute the truth value of the subformulas and take the larger value as result. That is,

$$x \;\mathsf{isSemOf}^m\; (\phi_0 \vee \phi_1)\; \theta_0\; \theta_1 \equiv \\ \exists y_0, y_1.\, (y_0 \;\mathsf{isSemOf}^m\; \phi_0\; \theta_0\; \theta_1) \wedge (y_1 \;\mathsf{isSemOf}^m\; \phi_1\; \theta_0\; \theta_1) \wedge \mathsf{MAX}(x, y_0, y_1) \;.$$

Encoding of $\tau(\Diamond^{d}\phi)\,[\theta_0,\theta_1]$

To compute the truth value of $\Diamond^{d}\phi$ we first introduce an abbreviation to compute the least upper bound.

For the following two abbreviations let $F(\bar{y})$ be a FOLRA formula having $\bar{y}$ (and possibly others) as free variables and let $\theta(\bar{y})$ be a FOLRA term. Then, for some fresh variable x we define the abbreviations

$$\begin{aligned}\mathsf{UB}(x,\theta(\bar{y}),F(\bar{y})) &\equiv \forall\bar{y}.F(\bar{y}) \implies x \geq \theta(\bar{y}) \;, \\ \mathsf{LUB}(x,\theta(\bar{y}),F(\bar{y})) &\equiv \mathsf{UB}(x,\theta(\bar{y}),F(\bar{y})) \wedge \forall z.\,(\mathsf{UB}(z,\theta(\bar{y}),F(\bar{y})) \implies z \geq x) \;.\end{aligned}$$

This means that $\mathsf{UB}(x,\theta(\bar{y}),F(\bar{y}))$ restricts x to be an upper bound of $\theta(\bar{y})$ for all values for $\bar{y}$ that make $F(\bar{y})$ true. Similarly, $\mathsf{LUB}(x,\theta(\bar{y}),F(\bar{y}))$ means that x is the least upper bound of $\theta(\bar{y})$ for any values for $\bar{y}$ that satisfy $F(\bar{y})$.

Example 7.2.8. For an example consider the functions $f_i : \mathbb{R}_{>0} \to \mathbb{R}$ with $i \in \{1,2\}$ and $f_1(x) = -x$ and $f_2(x) = x$. It is easy to see that the least upper bound of f_1 is 0, and that f_2 has no upper bound. We compute the least upper bound of f_1 with the formula

$$\mathsf{LUB}(x,-y,y>0) \;,$$

where $y > 0$ is used to restrict the allowed inputs, $-y$ computes the result which is assigned to x. Note that y is quantified inside the formula. The upper formula unfolds to

$$\mathsf{UB}(x,-y,y>0) \wedge \forall z.\,(\mathsf{UB}(z,-y,y>0) \implies z \geq x) \;,$$

which unfolds to

$$\forall y_1.\,(y_1 > 0 \implies x \geq -y_1) \wedge \forall z.\,((\forall y_2.\,y_2 > 0 \implies z \geq -y_2) \implies z \geq x) \;.$$

Now, a satisfying assignment assigns x the value of the least upper bound of f_1. To satisfy the first conjunct we can assign any positive number. The second conjunct restricts the result to $x = 0$.

For f_2 we create the formula

$$\mathsf{LUB}(x,y,y>0) \;,$$

which unfolds to

$$(\forall y_1.\,(y_1 > 0 \implies x \geq y_1)) \wedge \forall z.\,((\forall y_2.\,y_2 > 0 \implies z \geq y_2) \implies z \geq x) \;.$$

That is, we remove the minus sign. We see that the first conjunct is not satisfiable. Thus, there is no upper bound for f_2. △

When r, t, d range over a bounded domain we can approximate an exponential function $r * d^t$ with an arbitrary precision using linear approximations. Below we will use the abbreviation y isApproxOf $r\ d\ t$ to denote that y is an approximation of $r * d^t$.

We compute the truth value of $\tau(\Diamond^d \phi)\,[\theta_0, \theta_1]$ with the following formula:

$$x \ \mathsf{isSemOf}^m \ (\Diamond^d \phi)\ \theta_0\ \theta_1 \equiv \exists t, r.\, \mathsf{LUB}(x, y, F(t, y, r)) \text{ with}$$
$$F(t, y, r) \equiv (r \ \mathsf{isSemOf}^m\ \phi\ \theta_1\ t) \wedge (y \ \mathsf{isApproxOf}\ \ r\ d\ (t - \theta_1)) \wedge \theta_1 \leq t \leq \underline{t_l}\ .$$

With $F(t, y, r)$ we ensure that r takes the truth value of ϕ on the interval $[\theta_1, t]$ and y approximates $r * d^{t-\theta_1}$ such that $t \in [\theta_1, \underline{t_l}]$. Note that r, t and d are all bounded. That is, r represents the truth value of a subformula and thus is from $[0, 1]$, t is from $[0, \delta]$ and d is a constant.

We prove correctness of our encoding. In the next lemma we show that our encoding can be used to approximate the satisfaction of a DDC$_{<1}$ formula by *some* trajectory of a timed automaton. In the lemma after the next lemma, we show how to approximate the greatest lower bound of the satisfaction of a DDC$_{<1}$ formula by all trajectories of a timed automaton.

For the first lemma consider some automaton A with state variables V and a run $\pi = \langle (l_0, \beta_0, \nu_0), t_0), \ldots, (l_m, \beta_m, \nu_m), t_m), \ldots \rangle$ with $t_m = \delta$. Note that if a run does not contain a configuration with time stamp δ, we can simply add such a configuration by splitting the appropriate delay step into two delay steps. Thus, the trajectory τ_δ matching the prefix of π until δ can be represented by a satisfying assignment to $F_A(\underline{t}, \underline{V}, m, \delta)$, where $\underline{t}$ and the sequences in $\underline{V}$ have length $m + 1$. The following lemma states that our FOLRA encoding of the semantics of DDC serves to approximate the satisfaction of a DDC$_{<1}$ formula ϕ by the δ-bounded prefix τ_δ of some trajectory τ of A with a matching run π with m steps before δ time units. Note that we relate the satisfaction values of ϕ by τ_δ and τ with Corollary 7.2.6.

Lemma 7.2.9. *Let $m \in \mathbb{N}, \delta, \epsilon \in \mathbb{R}_{>0}$ and let $\underline{t}, \underline{V}$ be sequences of LRA variables of length $m+1$. Further, let A be a timed automaton with a run that has exactly m steps before δ time and let ϕ be a DDC$_{<1}$ formula. Then,*

$$F_A(\underline{t}, \underline{V}, m, \delta) \wedge (x \ \mathsf{isSemOf}^m \ \ \phi\ 0\ 0)$$

is satisfiable and for any satisfying assignment h we have

$$h(x) \in [\tau(\phi)\,[0,0] - \epsilon, \tau(\phi)\,[0,0] + \epsilon] \cap [0,1] \ ,$$

where τ, is defined as $\tau(P)(t) = h(\underline{P_i})$ with i such that $h(\underline{t_i}) \le t < h(\underline{t_{i+1}})$, $t \le \delta$ and $P \in V$, is the δ-bounded prefix of some trajectory in $\mathcal{T}(A)$.

Proof. First of all, we point out that by constructing τ as described in the lemma the formula $F_A(\underline{t}, \underline{V}, m, \delta)$ ensures that for some trajectory $\tau' \in \mathcal{T}(A)$ we have $\tau = \mathsf{pre}(\tau', \delta)$.

We proceed by induction on the structure of ϕ. To have a usable induction hypothesis we generalise the lemma to an arbitrary current interval given by terms instead of values. Additionally, we concretise how the error ϵ is used. We prove for all LRA terms θ_0, θ_1 that

$$0 \le \theta_0 \le \theta_1 \le \delta \implies F_A(\underline{t}, \underline{V}, m, \delta) \wedge (x \ \ \mathsf{isSemOf}^m \ \ \phi\ \theta_0\ \theta_1)$$

is satisfiable and for all satisfying assignments h with $0 \le t_0 \le t_1 \le \delta$ we have

$$h(x) \in [\tau(\phi)\,[t_0, t_1] - \frac{(\delta - t_1) * \epsilon}{\delta}, \tau(\phi)\,[t_0, t_1] + \frac{(\delta - t_1) * \epsilon}{\delta}] \cap [0,1] \ ,$$

where $t_0 = h(\theta_0), t_1 = h(\theta_1)$. The intuition here is that the allowed error ϵ is a resource that we spend with every linear approximation. It has to last until the current interval reaches δ. Thus, until time t_1 we are allowed to use $\frac{t_1 \epsilon}{\delta}$ of the error, and from t_1 onwards we may use $\frac{(\delta - t_1)\epsilon}{\delta}$.

Induction base.

Let $\phi \equiv \sum_{j=0}^{n} k_j \int S_j \gtrsim c$. For the start, let us assume that θ_0 and θ_1 are given as the values t_0 and t_1. The formula $(x \ \ \mathsf{isSemOf}^m \ \ \phi\ \theta_0\ \theta_1)$ unfolds to

$$\begin{aligned} &\big((\exists y^0, \ldots, y^n. \bigwedge_{j=0}^{n} (y^j \ \ \mathsf{isDur}^m \ \ S_j\ t_0\ t_1) \wedge \sum_{j=0}^{n} k_j * y^j \gtrsim c) \implies x = 1\big) \\ \wedge \quad &\big(\neg(\exists y^0, \ldots, y^n. \bigwedge_{j=0}^{n} (y^j \ \ \mathsf{isDur}^m \ \ S_j\ t_0\ t_1) \wedge \sum_{j=0}^{n} k_j * y^j \gtrsim c) \implies x = 0\big) \ . \end{aligned}$$

We make a case distinction on whether the inequality holds in τ. Assume it holds in τ with the current interval $[t_0, t_1]$. Then we can choose values for y^j, y^j_i with $j \in \{0, \ldots, n\}$ and $i \in \{0, \ldots, m-1\}$ such that $h(y^j) =$

$\sum_{i=0}^{m-1} h(y_i^j)$, $h(y_i^j) = \int_{\mathsf{max}(\underline{t_i},t_0)}^{\mathsf{min}(\underline{t_{i+1}},t_1)} S \,\mathrm{d}t$ and $\sum_{i=0}^{n} k_j * h(y^j) \gtrsim c$. The intuition is that y_i^j measures how much S_j holds in the interval $[t_0, t_1] \cap [\underline{t_i}, \underline{t_{i+1}}]$. Then $(y^j \;\; \mathsf{isDur}^m \;\; S_j \; t_0 \; t_1)$ and $(y_i^j \;\; \mathsf{isDur}_i \;\; S_j \; t_0 \; t_1)$ are satisfied. By assigning x the value 1 the formula $(x \;\; \mathsf{isSemOf}^m \;\; \sum_{j=0}^{n} k_j \int S_j \gtrsim c \; t_0 \; t_1)$ is satisfied. Then, for any satisfying assignment h we have $h(x) = \tau(\phi)\,[t_0, t_1]$.

Now for the other direction. Assume that $\sum_{j=0}^{n} k_j \int S_j \gtrsim c$ is not satisfied by τ and $[t_0, t_1]$. Then we cannot assign values to the variables y^j, y_i^j with $j \in \{0, \ldots, n\}$ and $i \in \{0, \ldots, m-1\}$ such that $(y^j \;\; \mathsf{isDur}^m \;\; S_j \; t_0 \; t_1)$ is satisfied. This means that in $(x \;\; \mathsf{isSemOf}^m \;\; \sum_{j=0}^{n} k_j \int S_j \gtrsim c \; t_0 \; t_1)$ the positive case is not satisfiable. Further, the negative case is satisfied by assigning x the value 0, which implies $h(x) = \tau(\phi)\,[t_0, t_1]$.

If θ_0 and θ_1 contain variables, we can assign arbitrary values. If $0 \leq h(\theta_0) \leq h(\theta_1) \leq \delta$, then we can reuse the previous argumentation. Otherwise, if the inequalities are not satisfied, the lemma is vacuously true.

Induction hypothesis.

We get the induction hypothesis that for all LRA terms θ_0, θ_1, all $m \in \mathbb{N}, \epsilon, \delta \in \mathbb{R}_{>0}$, all DDC$_{<1}$ formulas ϕ and all timed automata A, if A has a run that has at least m steps before δ time, then

$$0 \leq \theta_0 \leq \theta_1 \leq \delta \implies F_A(\underline{t}, \underline{V}, m, \delta) \wedge (x \;\; \mathsf{isSemOf}^m \;\; \phi \; \theta_0 \; \theta_1)$$

is satisfiable and for all satisfying assignments h with $0 \leq t_0 \leq t_1 \leq \delta$ we have

$$h(x) \in [\tau(\phi)\,[t_0, t_1] - \frac{(\delta - t_1) * \epsilon}{\delta}, \tau(\phi)\,[t_0, t_1] + \frac{(\delta - t_1) * \epsilon}{\delta}] \cap [0, 1] \;\; ,$$

where $h(\theta_0) = t_0, h(\theta_1) = t_1$.

Induction step.

Case 1 ($\phi \equiv \neg\phi_1$). The formula $(x \;\; \mathsf{isSemOf}^m \;\; \neg\phi_1 \; \theta_0 \; \theta_1)$ unfolds to

$$\exists y. ((y \;\; \mathsf{isSemOf}^n \;\; \phi_1 \; \theta_0 \; \theta_1) \wedge x = 1 - y) \;\; .$$

From the IH it follows that the inner formula is satisfiable and that for any satisfying assignment h we have $h(y) \in \tau(\phi_1)\,[t_0, t_1] \pm \frac{(\delta - t_1)*\epsilon}{\delta}$ restricted to $[0, 1]$, where $h(\theta_0) = t_0, h(\theta_1) = t_1$. This implies

$$h(x) = 1 - h(y) \in 1 - \tau(\phi_1)\,[t_0, t_1] \pm \frac{(\delta - t_1) * \epsilon}{\delta} \;\; .$$

Case 2 ($\phi \equiv \phi_0 \vee \phi_1$). Now, ($x$ $\mathsf{isSemOf}^m$ $\phi_0 \vee \phi_1$ θ_0 θ_1) unfolds to

$$\exists y_0, y_1 .\, (y_0 \ \mathsf{isSemOf}^n \ \phi_0 \ \theta_0 \ \theta_1) \wedge (y_1 \ \mathsf{isSemOf}^n \ \phi_1 \ \theta_0 \ \theta_1) \wedge \mathsf{MAX}(x, y_0, y_1) \ .$$

By IH the first two subformulas are satisfiable for all terms θ_0, θ_1. Let h_0, h_1 be two satisfying assignments. As these subformulas do not share any variables, except those in θ_0, θ_1, their conjunction, and also the complete formula, is satisfiable. A satisfying assignment assigns x the value $\mathsf{max}(h_0(y_0), h_1(y_1))$. As $\tau(\phi_i)\,[t_0, t_1] \in h_i(y_i) \pm \frac{(\delta - t_1)*\epsilon}{\delta}$ with $i \in \{0, 1\}$ we conclude $h(x) \in \tau(\phi_0 \vee \phi_1)\,[t_0, t_1] \pm \frac{(\delta - t_1)*\epsilon}{\delta}$, where h is any satisfying assignment and $h(\theta_0) = t_0, h(\theta_1) = t_1$.

Case 3 ($\phi \equiv \Diamond^d \phi_1$). The formula ($x$ $\mathsf{isSemOf}^n$ $\Diamond^d \phi_1$ θ_0 θ_1) is equivalent to

$$\exists z, r .\, \mathsf{LUB}\big(x, y, (r \ \mathsf{isSemOf}^m \ \phi_1 \ \theta_1 \ z) \wedge (y \ \mathsf{isApproxOf} \ r \ d \ (z - \theta_1)) \wedge \theta_1 \leq z \leq \delta\big) \ .$$

From the IH we know that for all values for z in between θ_1 and δ the formula (r $\mathsf{isSemOf}^m$ ϕ_1 θ_1 z) is satisfiable and that all satisfying assignments h ensure $h(r) \in \tau(\phi_1)\,[t_1, t_2] \pm \frac{(\delta - t_2)*\epsilon}{\delta}$, where $t_1 = h(\theta_1), t_2 = h(z)$.

Next, whatever value is chosen for r, we know that (y $\mathsf{isApproxOf}$ r d $(\theta_1 - z)$) is satisfiable and that all satisfying assignments h ensure $h(y) \in h(r) * d^{t_2 - t_1} \pm \frac{(t_2 - t_1)*\epsilon}{\delta}$. We show that the nested linear approximations do not cause problems. Formally, we show for any satisfying assignment h that $h(y) \in \tau(\phi_1)\,[t_1, t_2] * d^{t_2 - t_1} \pm \frac{(\delta - t_2)*\epsilon}{\delta}$. In the first step we replace r by the value it computes, while adding the uncertainty of this computation to the error interval (right of $\pm$). In the second step we drop $d^{t_2 - t_1}$, which is less or equal 1. This might increase the error interval. We have

$$\begin{aligned}
& h(y) \in h(r) * d^{t_2 - t_1} \pm \frac{(t_2 - t_1) * \epsilon}{\delta} \\
\text{iff } & h(y) \in \tau(\phi_1)\,[t_1, t_2] * d^{t_2 - t_1} \pm \left(\frac{(t_2 - t_1) * \epsilon}{\delta} + d^{t_2 - t_1} * \frac{(\delta - t_2) * \epsilon}{\delta} \right) \\
\text{implies } & h(y) \in \tau(\phi_1)\,[t_1, t_2] * d^{t_2 - t_1} \pm \left(\frac{(t_2 - t_1) * \epsilon}{\delta} + \frac{(\delta - t_2) * \epsilon}{\delta} \right) \\
\text{iff } & h(y) \in \tau(\phi_1)\,[t_1, t_2] * d^{t_2 - t_1} \pm (t_2 - t_1 + \delta - t_2) * \frac{\epsilon}{\delta} \\
\text{iff } & h(y) \in \tau(\phi_1)\,[t_1, t_2] * d^{t_2 - t_1} \pm \frac{(\delta - t_1) * \epsilon}{\delta} \ .
\end{aligned}$$

Now, (x $\mathsf{isSemOf}^m$ $\Diamond^d\phi_1\ \theta_0\ \theta_1$) searches for the least upper bound for y using r and t_2 and assigns it to x. That is,

$$\begin{aligned} h(x) &\in \sup_{t_2\in[t_1,\delta]} (d^{t_2-t_1} * \tau(\phi_1)\,[t_1,t_2]) \pm \frac{(\delta - t_1) * \epsilon}{\delta} \\ \text{iff } h(x) &\in \tau(\Diamond^d\phi_1)\,[t_0,t_1] \pm \frac{(\delta - t_1) * \epsilon}{\delta} \ . \end{aligned} \qquad \square$$

We use our approximation of the semantics in FOLRA and the bounded unfolding to prove that approximative model checking is computable. For this, we first need another abbreviation. Similar as for the least upper bound we define a formula to compute the largest lower bound. Let $F(\bar{y})$ be a FOLRA formula containing $\bar{y}$ as free variables and let $\theta(\bar{y})$ be a FOLRA term. Then, for some fresh variable x we define

$$\begin{aligned} \mathsf{LB}(x,\theta(\bar{y}),F(\bar{y})) &\equiv \forall \bar{y}.F(\bar{y}) \implies x \leq \theta(\bar{y}) \ , \\ \mathsf{GLB}(x,\theta(\bar{y}),F(\bar{y})) &\equiv \mathsf{LB}(x,\theta(\bar{y}),F(\bar{y})) \wedge \forall z.\,(\mathsf{LB}(z,\theta(\bar{y}),F(\bar{y})) \implies z \leq x) \ . \end{aligned}$$

Note that we did not investigate the complexity of our reduction. For the complexity of FOLRA we refer to Lemma 2.4.6.

Theorem 7.2.10 (Approximate Model Checking is Computable)**.** *Given a strongly non-Zeno timed automaton A and a DDC$_{<1}$ formula ϕ and a desired precision $\epsilon \in \mathbb{R}_{>0}$, the approximate model checking problem is effectively computable: there is a procedure computing $v \in [0,1]$ such that*

$$v \in \mathsf{mc}(A,\phi) \pm \epsilon \ .$$

Proof. We split the allowed error ϵ into two parts. Let $\epsilon_1, \epsilon_2 > 0$ be such that $\epsilon_1 + \epsilon_2 = \epsilon$. We use ϵ_1 to only consider trajectories of time-bounded length, and ϵ_2 to approximate the semantics of DDC.

We know from Corollary 7.2.6 that we can limit the time horizon of interest to $\delta = \log_d \epsilon_1$, where d is the largest discount occurring in ϕ. As A is strongly non-Zeno we know that in any cycle at least one time unit passes. Let l be the length of the longest cycle in the transition graph of A. Then, in any run of A within δ time units at most $m = \lceil l\delta \rceil$ (where $\lceil\cdot\rceil$ denotes rounding to the next larger integer) can occur.

Consider the formula

$$\psi \equiv \mathsf{GLB}(x, y, \bigvee_{i=0}^{m} F_A(\underline{t}^i, \underline{V}^i, i, \delta) \wedge (y \;\; \mathsf{isSemOf}^m \;\; \phi\ 0\ 0))$$

where $\underline{t}^i = \langle \underline{t}_0, \ldots, \underline{t}_i \rangle, \underline{V}^i = \{\langle \underline{P}_0, \ldots, \underline{P}_i \rangle \mid P \in V\}$ and we allow an error of ϵ_2 in $y \;\; \mathsf{isSemOf}^m \;\; \phi\ 0\ 0$.

If ψ is not satisfiable we know that for all $i \in \{0, \ldots, m\}$ the automaton A does not have a run that has i steps until δ time passes. As we have chosen m large enough to include all cycles, this means that A does not have any runs. Hence, $\mathcal{T}(A) = \emptyset$, which implies $\mathsf{mc}(A, \phi) = 1$. We choose $v = 1$ and the lemma holds.

Now, assume that ψ is satisfiable by some assignment h. Then, from Lemma 7.2.7, Lemma 7.2.9 and because GLB computes the greatest lower bound it follows that

$$h(x) \in \inf_{\tau \in \mathcal{T}(A)} (\tau_\delta(\phi)\,[0, 0]) \pm \epsilon_2 \ . \tag{7.6}$$

Next, from Corollary 7.2.6 we know that by bounding ourselves to the δ prefix we perform at most an error of ϵ_1. Hence,

$$|\inf_{\tau \in \mathcal{T}(A)} (\tau_\delta(\phi)\,[0, 0]) - \inf_{\tau \in \mathcal{T}(A)} (\tau(\phi)\,[0, 0])| \le \epsilon_1 \ . \tag{7.7}$$

Combining Equations 7.6 and 7.7 we conclude $h(x) \in \inf_{\tau \in \mathcal{T}(A)} (\tau(\phi)\,[0, 0]) \pm \epsilon$, which by definition is equivalent to $h(x) \in \mathsf{mc}(A, \phi) \pm \epsilon$. We conclude that by choosing $v = h(x)$ the lemma holds. □

7.2.2 Extending the Model Checkable Fragment

So far we can only approximate the satisfaction value for $\mathrm{DDC}_{<1}$. However, with this fragment we can only reason about time-bounded prefixes of infinite behaviours. This severely restricts the usefulness of this fragment. In this section we extend the fragment for which we can perform model checking to include formulas of the form $\mathrm{G}_{\updownarrow S}\, \phi$, where S is a state expression and ϕ is a $\mathrm{DDC}_{<1}$ formula. With these kind of formulas we can reason about infinite behaviours. However, this requires additional mild restrictions on the model.

When the timed automaton has upper bounds for the values of all clocks in all locations the set of reachable states is computable with a finite representation. The goal of this constraint is to avoid over-approximation introduced

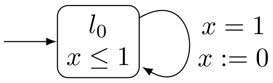


Figure 7.2: A timed automaton with two clocks x and y and no state variables. The clock x is reset every second, while y is never used.

by the *normalisation* step of reachability algorithms [BY03]. For an example consider the timed automaton in Figure 7.2 with two clocks x, y. The reachable configurations are $\{(l_0, f_\emptyset, (u,v)) \mid u \in [0,1], v = u + k, k \in \mathbb{N}\}$, where $f_\emptyset$ is the valuation for the (empty) set of state variables and (u,v) are the values of the clocks x and y. We see that the fractional part of x and y are always equal. With normalisation we abstract from the concrete value of the clock y and merely remember $y > 1$. However, this is an over-approximation. To avoid this over-approximation we require that all locations have an upper bound for every clock. We call a timed automaton that has in every location for every clock an upper bound *strongly bounded*.

We concretise the earlier ideas. For a state expression S and a timed automaton $A = (L, \rightarrow, \mathsf{Init}, I, \mathsf{Lab}, V, \Lambda, \mathcal{X})$ we denote the set of configurations where the state expression S just changed its value with $f(S,A)$. We define this formally as

$$\begin{aligned} f(S,A) = \{(l,\beta,\nu) \mid \\ &\exists (l_0,\beta_0,\nu_0) \in \mathsf{Init}, (l,\beta,\nu), (l',\beta',\nu') \in (L \times \mathsf{Val}(V) \times \mathsf{Val}(\mathcal{X})). \\ &\quad (l_0,\beta_0,\nu_0) \rightarrow^* (l',\beta',\nu') \rightarrow (l,\beta,\nu) \text{ and } \beta \models S \not\Leftrightarrow \beta' \models S\} \ . \end{aligned}$$

For a strongly bounded timed automaton A the set $f(S,A)$ is computable.

Lemma 7.2.11 ([BY03; BLP+99]). *For a strongly bounded timed automaton A and a state expression S the set $f(S,A)$ is computable and has a finite symbolic representation in linear real arithmetic.*

Proof. Since, the automaton is strongly bounded, we can use common approaches to symbolically explore the reachable states of the timed automaton without using normalisation [BY03; BLP+99]. These approaches use difference bound matrices, which are formulas of linear real arithmetic represented as matrices with efficient operations on them. □

We reduce computing the satisfaction of $\mathrm{G}_S\,\phi$ by A to computing the satisfaction of ϕ by a transformed automaton A'.

Lemma 7.2.12. *Let ϕ be a DDC$_{<1}$ formula, $A = (L, \to, \mathsf{Init}, I, \mathsf{Lab}, V, \Lambda, \mathcal{X})$ a strongly bounded timed automaton and S a state expression. Then, for the timed automaton $A' = (L, \to, f(S, A), I, \mathsf{Lab}, V, \Lambda, \mathcal{X})$ we have*

$$\mathsf{mc}(A, \mathrm{G}_{\updownarrow S}\,\phi) = \mathsf{mc}(A', \phi) \ .$$

Proof. First, let us see what $\tau(\mathrm{G}_{\updownarrow S}\,\phi)\,[0,0]$ evaluates to, for some $\tau \in \mathcal{T}(A)$. We have

$$\begin{aligned}
&\tau(\mathrm{G}_{\updownarrow S}\,\phi)\,[0,0] \\
&= \tau(\neg \mathrm{F}_{\updownarrow S}\,\neg\phi)\,[0,0] \\
&= 1 - \sup_{t_1 \geq 0}(\sup_{t_2 \geq t_1}(\min(\tau(\updownarrow S)\,[t_0,t_1], 1 - \tau(\phi)\,[t_1,t_1])) \ .
\end{aligned}$$

Now, if $\mathcal{T}(A) = \emptyset$, then also $\mathcal{T}(A') = \emptyset$ and the lemma holds.

If $\mathcal{T}(A') = \emptyset$, then we know that on all trajectories of A the formula $\updownarrow S$ never is satisfied. Thus, $\mathsf{mc}(A, \mathrm{G}_{\updownarrow S}\,\phi)$ and $\mathsf{mc}(A', \phi)$ both evaluate to 1. This finishes the special cases.

Next we assume that both automata accept some trajectories. Then we can reformulate the lemma as

$$\inf_{\tau \in \mathcal{T}(A)}(\tau(\mathrm{G}_{\updownarrow S}\,\phi)\,[0,0]) \geq \inf_{\tau' \in \mathcal{T}(A')}(\tau'(\phi)\,[0,0]) \ , \tag{7.8}$$

$$\inf_{\tau' \in \mathcal{T}(A')}(\tau'(\phi)\,[0,0]) \geq \inf_{\tau \in \mathcal{T}(A)}(\tau(\mathrm{G}_{\updownarrow S}\,\phi)\,[0,0]) \ . \tag{7.9}$$

We show that for any trajectory $\tau \in \mathcal{T}(A)$ there is $\tau' \in \mathcal{T}(A')$ such that $\tau(\mathrm{G}_{\updownarrow S}\,\phi)\,[0,0] \geq \tau'(\phi)\,[0,0]$. This implies that Equation (7.8) holds. Further, we show a similar property for the other direction.

For any trajectory $\tau \in \mathcal{T}(A)$ and a corresponding run π of A and for arbitrary t_0, t_1, where $\tau(\updownarrow S)\,[t_0,t_1] = 1$, there is a run π' of A' such that from time t_1 onwards π and π' are equal. Thus, there is a trajectory $\tau' \in \mathcal{T}(A')$ corresponding to π' such that $\tau(\mathrm{G}_{\updownarrow S}\,\phi)\,[0,0] = \tau'(\phi)\,[0,0]$. Hence, Equation (7.8) holds.

For the other direction consider some trajectory $\tau' \in \mathcal{T}(A')$ and a corresponding run $\pi' = \langle((l'_1, \beta'_1, \nu'_1), 0), \dots\rangle$. There is a run $\pi = \langle((l_1, \beta_1, \nu_1), 0), \dots, ((l_i, \beta_i, \nu_i), t_i), ((l'_1, \beta'_1, \nu'_1), t_{i+1}), \dots\rangle$ such that $\beta_i \models S \iff\!\!\!\!\!\!/\ \ \beta'_1 \models S$, i.e. from index $i+1$ onwards π is equal to π' (except for the time stamps). It follows that $\tau(\updownarrow S)\,[t_i, t_{i+1}] = 1$ and $\tau(\phi)\,[t_{i+1}, t_{i+1}] = \tau'(\phi)\,[0,0]$. Next, as $\tau(\mathrm{G}_{\updownarrow S}\,\phi)\,[0,0]$ tries to find the lowest possible truth value of ϕ we know that $\tau(\mathrm{G}_{\updownarrow S}\,\phi)\,[0,0] \leq \tau'(\phi)\,[0,0]$. Thus, Equation (7.9) holds. Note that the prefix of π leading to $((l'_1, \beta'_1, \nu'_1), t_{i+1})$ may even yield a truth value smaller than $\tau'(\phi)\,[0,0]$. □

We conclude that for our globally properties approximate model checking is computable. As for Theorem 7.2.10 we did not investigate the complexity of our procedure. For the complexity of FOLRA we refer to Lemma 2.4.6.

Theorem 7.2.13. *The approximate model checking problem is computable for* $\mathrm{G}_{\updownarrow S}\,\phi$ *and* A*, where* S *is a state expression,* ϕ *is a DDC*$_{<1}$ *formula and* A *is a strongly non-Zeno strongly bounded timed automaton.*

Proof. Follows from Theorem 7.2.10 and Lemma 7.2.12. □

7.3 Examples

To support our claims that we can reason about interesting problems with DDC we provide two examples in this section.

7.3.1 Production Cell

We consider two drilling machines that generate heat while drilling. These machines independently of each other process work pieces of different sizes, and the drilling time needed to finish a work piece depends on the size of the piece. If a machine drills for a long time without interruption the machine becomes too hot. If the machine is too hot, it will gradually take damage. It is undesirable to always avoid that the machine becomes too hot, because then production will be too low. The desired property is that the machine soon cools down, after it became too hot.

Let $i \in \{0, 1\}$. We represent that machine i became too hot by changing the value of the propositional variable H_i, that the machine is drilling by D_i and the durability of the machine by the discount (here 0.9, where closer to 1 means more durable). Further, there are coefficients (here $1, 2$) representing how quickly the temperature changes over time in the respective locations and here 5 is the desired cooldown to achieve after the machine has become too hot.

We formalise the desired property as

$$\phi \equiv \mathrm{G}_{\updownarrow H_0}(\Diamond^{\,0.9}(\textstyle\int\neg D_0 - 2\int D_0 \geq 5)) \wedge \mathrm{G}_{\updownarrow H_1}(\Diamond^{\,0.9}(\int\neg D_1 - 2\int D_1 \geq 5))\,.$$

Let the conjuncts of ϕ be ϕ_0 and ϕ_1. We show the controllers of the drilling machines and some helper automata in Figure 7.3 on Page 224. Note that the resulting network of timed automata is delaying, strongly non-Zeno and strongly bounded (cf. Pages 17, 205 and 220).

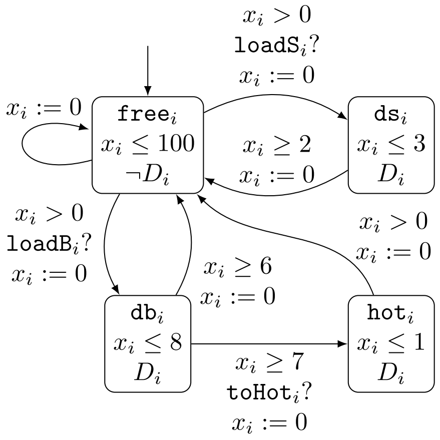
Computing the Satisfaction Value

Here we focus on the satisfaction value of the subformula $\mathrm{G}_{\updownarrow H_0}(\Diamond^{\,0.9}(\int\neg D_0 - 2\int D_0 \geq 5))$. However, the satisfaction value for the other subformula is the same.

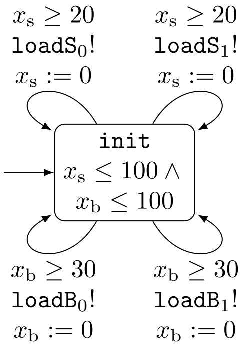
Let $A = (A_0 \parallel A_1 \parallel B_0 \parallel B_1 \parallel C) = (L, \rightarrow, \mathsf{Init}, I, \mathsf{Lab}, V, \Lambda, \mathcal{X})$ be the parallel composition of our automata. To approximate the satisfaction value of $\mathrm{G}_{\updownarrow H_0}(\Diamond^{\,0.9}(\int\neg D_0 - 2\int D_0 \geq 5))$ by A we apply Lemma 7.2.12 for the first subformula and create $A' = (L, \rightarrow, f(H_0, A), I, \mathsf{Lab}, V, \Lambda, \mathcal{X})$, where $f(H_0, A)$ is the set of initial configurations in which the state variable H_0 has just changed its value, i.e. where the edge from $\mathtt{db}_0$ to $\mathtt{hot}_0$ just has been taken (cf. Page 220 for the definition of $f(H_0, A)$). Below we show the set $f(H_0, A)$. Note that we do not include the location of B_0 and B_1 and also the valuation of the state variables as both hold little useful information here. The set $f(H_0, A)$ is given as

$$\begin{gathered} f(H_0, A) = \{((\mathtt{hot}_0, \mathtt{free}_1, \mathtt{init}), \nu) \mid \nu \in \big(7 \leq x_{\mathsf{b}} \leq 8 \wedge x_0 = 0 \wedge x_{\mathsf{s}} \leq 100 \wedge \\ ((x_1 + 2 \leq x_{\mathsf{s}}) \vee (x_{\mathsf{b}} + 2 < x_{\mathsf{s}} \wedge x_1 \leq 100 \wedge x_{\mathsf{b}} \leq x_1))\big)\} \cup \\ \{((\mathtt{hot}_0, \mathtt{ds}_1, \mathtt{init}), \nu) \mid \nu \in (7 \leq x_{\mathsf{b}} \leq 8 \wedge x_1 = x_{\mathsf{s}} \wedge x_{\mathsf{s}} \leq 3 \wedge x_0 = 0)\}\ . \end{gathered}$$

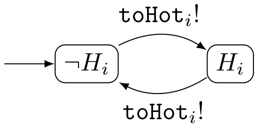
Note that we computed the initial configurations with Uppaal Tiga [CDF+05] by computing a winning strategy for the property *control* : $A[\,]$ *true* with the options `-c -w 2 -n 2`. Here, $((\mathtt{hot}_0, \mathtt{ds}_1, \mathtt{init}), \nu)$ are the possible configurations where H_0 has just changed its value and A_1 currently is drilling a small working piece. For the configurations $((\mathtt{hot}_0, \mathtt{free}_1, \mathtt{init}), \nu)$, where A_1 is not drilling, there are two kinds of configurations possible. The first kind $(x_1 + 2 \leq x_{\mathsf{s}})$, represents that so far no small piece has been drilled, or that the last small piece was drilled by A_1. The second kind $(x_{\mathsf{b}} + 2 < x_{\mathsf{s}} \wedge x_1 \leq 100 \wedge x_{\mathsf{b}} \leq x_1)$, represents that the last small piece was drilled by A_0 before it started on the current big piece.



(a) Drilling machine controller A_i.



(b) Work piece spawner C.



(c) Helper automaton B_i.

Figure 7.3: In the top left-hand side we see the controller A_i for $i \in \{0, 1\}$ of a drilling machine. It takes 6 to 8 time units to drill a big piece and 2 to 3 time units to drill a small piece. The upper bounds of 100, serve to make Lemma 7.2.12 applicable, the other upper bounds restrict the maximal drilling time needed for small and big working pieces. The self loop in $\texttt{free}_i$ serves to make the parallel composition $A_0 \parallel A_1 \parallel B_0 \parallel B_1 \parallel C$ deadlock free. In the top right-hand side we see C, which controls how quickly work pieces may appear and that assigns the working pieces nondeterministically to the machines. In the bottom we show the helper automaton B_i controlling the state variable H_i.

Let the desired precision be $\epsilon = 0.1$. As $\delta = \log_{0.9} \epsilon$ is less than 21.86 we can choose $\delta = 22$ according to Corollary 7.2.6. The approximation of the satisfaction value is

$$\inf_{\tau \in \mathcal{T}(A')} (\sup(0.9^t * \tau_{22}(\textstyle\int \neg D_0 - 2 \int D_0 \geq 5)\,[0,t] \mid 0 \leq t \leq 22)) \pm \epsilon \ .$$

As the method of the previous section is not implemented we calculate the satisfaction of ϕ_0 by hand. Hence, we do not add further imprecision caused by linear approximation. We are looking for a run π of A', such that in the trajectory induced by π the smallest t for which $\tau_{22}(\int \neg D_0 - 2 \int D_0 \geq 5)\,[0,t]$ holds, is large.

A run that maximises the time t needed to satisfy $\int \neg D_0 - 2 \int D_0 \geq 5$ is depicted below. The intuition of the run is that after the machine finished a big working piece, it is assigned a small working piece. The location of C always is `init`. Hence, the configurations in the run have the form $((l, \nu(x_0), \nu(x_{\mathsf{s}}), \nu(x_{\mathsf{b}})), t)$ where l is a location from A_0, $\nu(y)$ is the value of the clock y under the clock valuation ν and t is a time stamp. Note that again, we do not include the locations of A_1, B_0, B_1, C, the clocks of A_1 and the state variable valuation. The run is

$$\begin{aligned}\pi = \langle &((\mathtt{hot}_0, 0, 19, 7), 0), ((\mathtt{hot}_0, 1, 20, 8), 1), ((\mathtt{free}_0, 0, 20, 8), 1),\\ &((\mathtt{free}_0, 1, 21, 9), 2), ((\mathtt{ds}_0, 0, 0, 9), 2), (\mathtt{ds}_0, 3, 3, 12), 5),\\ &(\mathtt{free}_0, 0, 3, 12), 5), (\mathtt{free}_0, 17, 20, 29), 22)\rangle \ .\end{aligned}$$

It is clear that $(\mathtt{hot}_0, 0, 19, 7)$ is in $f(H_0, A)$. The run spends 4 time units in locations where D_0 holds ($\mathtt{hot}_0$, $\mathtt{ds}_0$). Thus, it takes 13 time units in locations where $\neg D_0$ holds ($\mathtt{free}_0$) until $\int \neg D_0 - 2 \int D_0 \geq 5$ is satisfied, which means that altogether we need 17 time units. We have

$$\inf_{\tau \in \mathcal{T}(A')} (\tau(\Diamond^{0.9}(\textstyle\int \neg D_0 - 2 \int D_0 \geq 5))) \in \ 0.9^{17} \pm 0.1 \ .$$

After rounding to two positions behind decimal point this is close to 0.17 ± 0.1. In general, by considering only bounded prefixes of all runs we introduce an error. However, in our example the result 0.9^{17} is exact, because A' does not have a run that results in a lower truth value.

We see that the controllers of the drilling machines satisfy our cooldown property poorly, i.e. that the machines often overheat. To fix this we could introduce a scheduler in between the controllers A_0, A_1 and the spawner of the working pieces C. This scheduler would then assign the working pieces to machines in a way that avoids assigning two successive working pieces to the same machine. As the model then would be quite big, we would need automation to compute the satisfaction value for the larger example.

7.3.2 Hazard Warning

As another example we consider a variation of the hazard warning protocol from [OS17]. There, a car on a motorway detects a hazard and intends to warn other cars farther away of the hazard via limited range broadcast communication. For this the car first computes a *communication chain* containing the IDs of the cars that should in turn forward the warning. Afterwards, the car sends a warning together with the chain it computed to the next car in the chain (cf. Figure 7.4). The authors verify that their protocol ensures delivery of the warning within a certain time-bound.

We extend their approach by partitioning the road into three sectors near, middle and far according to their distance to the hazard. Then we define DDC formulas expressing that

"for all cars C after a hazard has been detected
soon with urgency d_s car C is informed of the hazard" ,

where d_s is given by the sector the car C is in.

Here we assume that the cars forwarding the warning are precomputed. However, we can use MLSL to determine the next car to forward the warning. For this we could use the view to serve as communication range and create an MLSL formula that describes the location of a suitable car.

The Property

We formalise the property described above. We introduce the three sectors: far, middle and close. A car is

- in the *near sector*, if it is less than 300 m away from the hazard,
- in the *middle sector*, if it is between 300 m and 1000 m away from the hazard and
- in the *far sector*, if it is more than 1000 m away from the hazard.

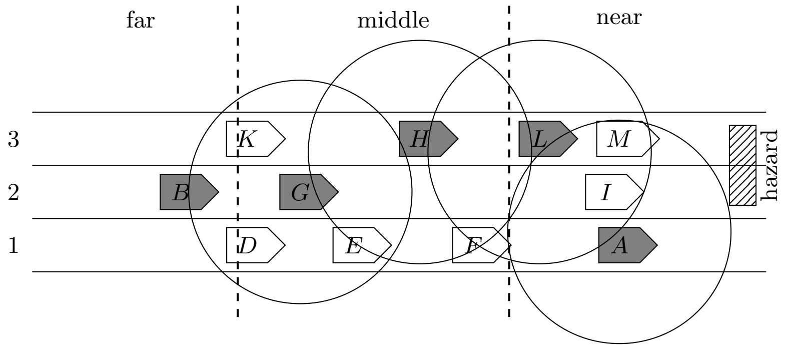


Figure 7.4: Overview of the hazard warning protocol. Car A learns of the hazard on the lanes 2 and 3, computes a communication chain $\langle A, L, H, G, B\rangle$. We partition the road by the distance to the hazard into the three sectors far, middle and near.

We assume a maximum speed of $130\,\mathrm{km\,h^{-1}}$, which is almost equivalent to $36.2\,\mathrm{m\,s^{-1}}$ and a maximum deceleration of $12\,\mathrm{m\,s^{-2}}$. Following classical mechanics a car travels $\frac{-v^2}{2d}$ spatial units after braking until it stops, where d is the deceleration and v is the current speed. Thus, a car with a speed of $130\,\mathrm{km\,h^{-1}}$ travels about 54.6 metres after initiating emergency braking.

Within the definition of DDC we always assumed an exponential discounting function. However, any strictly decreasing function with a range within $[0, 1]$ can be used. For cars in the near sector we consider a delay of $0.5\,\mathrm{s}$ until a car is informed as very good (as the car can clearly avoid entering the hazard), which we represent by a truth value of 0.9. To delays of $1\,\mathrm{s}$ and $3\,\mathrm{s}$ we assign satisfaction values of 0.6 and 0.1. By linear interpolation, we form a function

$$f(t) = \begin{cases} 1 + (t-0) * \frac{0.9-1}{0.5-0} & \text{if } 0 \leq t < 0.5 \\ 0.9 + (t-0.5) * \frac{0.6-0.9}{1-0.5} & \text{if } 0.5 \leq t < 1 \\ 0.6 + (t-1) * \frac{0.1-0.6}{3-1} & \text{if } 1 \leq t < 3 \end{cases}$$

We extend f to a discounting function by letting it asymptotically approach 0. Thus, after simplification we get the discounting function η_{n} for the sector near as defined below (visualised in Figure 7.5 on Page 229, top right):

$$\eta_{\mathsf{n}}(t) = \begin{cases} 1 - 0.2t & \text{if } 0 \leq t < 0.5 \ , \\ 1.2 - 0.6t & \text{if } 0.5 \leq t < 1 \ , \\ 0.85 - 0.25t & \text{if } 1 \leq t < 3 \ , \\ 0.1 * 0.9^{t-3} & \text{otherwise} \ . \end{cases}$$

For the sectors middle and far we provide the pairs of delay until a car is informed and truth value in Figure 7.5 on Page 229.

Now, to formalise the property we use two signals representing events: H represents that a hazard has been detected and I_C represents that car C has been informed of the hazard. Thus, the formulas $\updownarrow H$ and $\updownarrow I_C$ are satisfied at the point in time when the respective event has happened. Let s be one of the sectors near, middle or far and let CS_s be the set of cars in the sector s. We define

$$\phi_s \equiv \bigwedge_{C \in CS_s} \mathrm{G}_{\updownarrow H} \left(\Diamond^{\eta_s} \updownarrow I_C\right) \ ,$$

$$\phi \equiv \bigwedge_{s \in \{\mathsf{n},\mathsf{m},\mathsf{f}\}} \phi_s \ .$$

The DDC formula ϕ_s formalises that for each car in sector s, after a hazard is detected, soon this car is informed.

The Protocol

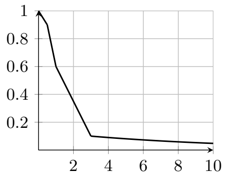
At first, some car C learns of a hazard, which causes $\updownarrow H$ to be satisfied. It then sends a broadcast signal to inform other cars within its communication range of the hazard, and picks a specific car to forward the warning. If the other car does not forward the warning within a given time, C again sends the broadcast signal. The other cars behave similarly, until all cars are informed.

We model the protocol as the parallel composition of the automata A_{Haz}, A^i_{Fwd}, A^i_{Det} and A_{Ch}, where $i \in \{0, \dots n-1\}$ and n is the number of cars in the communication chain. Here, A_{Haz} controls the hazard, A^i_{Fwd} (resp. A^i_{Det}) is the hazard forwarding (resp. hazard detection) controller of the i-th car in the communication chain and A_{Ch} is the channel controller. In the following we explain these controllers. Note that we use Uppaal data variables to model state variables of DDC. An example is the data variable `H`, which is the Uppaal representation of the state variable H.

sector near:

delay	truth value
0 s	1
0.5 s	0.9
1 s	0.6
3 s	0.1

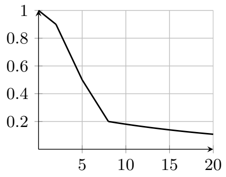
$$\eta_{\mathsf{n}}(t) = \begin{cases} 1 - 0.2t & \text{if } 0 \leq t < 0.5 \\ 1.2 - 0.6t & \text{if } 0.5 \leq t < 1 \\ 0.85 - 0.25t & \text{if } 1 \leq t < 3 \\ 0.1 * 0.9^{t-3} & \text{otherwise} \end{cases}$$



sector middle:

delay	truth value
0 s	1
2 s	0.9
5 s	0.5
8 s	0.2

$$\eta_{\mathsf{m}}(t) = \begin{cases} 1 - 0.05t & \text{if } 0 \leq t < 2 \\ \frac{7}{6} - \frac{-2}{15}t & \text{if } 2 \leq t < 5 \\ 1 - 0.1t & \text{if } 5 \leq t < 8 \\ 0.2 * 0.95^{t-8} & \text{otherwise} \end{cases}$$



sector far:

delay	truth value
0 s	1
10 s	0.9
20 s	0.7
40 s	0.2

$$\eta_{\mathsf{f}}(t) = \begin{cases} 1 - 0.01t & \text{if } 0 \leq t < 10 \\ 1.1 - 0.02t & \text{if } 10 \leq t < 20 \\ 1.2 - 0.25t & \text{if } 20 \leq t < 40 \\ 0.2 * 0.95^{t-40} & \text{otherwise} \end{cases}$$

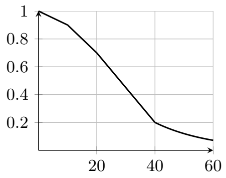


Figure 7.5: The rows represent the discounting functions for the sectors near, middle and far (top to bottom). In the left column we show for each sector a table relating the delay until a car is informed of a hazard and our respective satisfaction with the delay. In the middle column we show the discounting function resulting from linear interpolation, with an asymptotic finish towards 0. In the right column we show the discounting function as a graph, where the x-axis is the value of t, i.e. the delay, and the y-axis is the truth value.

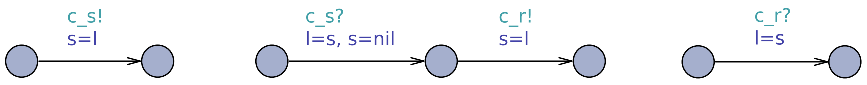


Figure 7.6: Visualisation of the value passing pattern in Uppaal. The value of a local variable `l` from an automaton A_1 (left) is asynchronously sent through a shared variable `s` and an intermediate automaton (middle) serving as channel to a local variable `l` of an automaton A_2 (right). The pattern uses a design decision of Uppaal that in synchronous transitions updates of sending transitions (marked with !) are performed before updates of receiving transitions (marked with ?).

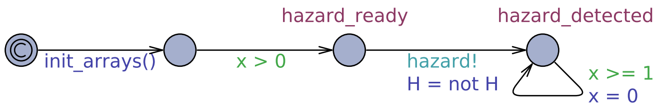


Figure 7.7: The hazard controller A_{Haz} informs the cars of the hazard with **`hazard`**!. The update `H` = **`not`** `H` ensures that when this transition is taken, the formula $\updownarrow H$ is satisfied.

To model the exchange of data between different cars, our controllers use the *value passing pattern* from Uppaal [BDL, Section 7]. As an example consider the transitions depicted in Figure 7.6.

We explain A_{Haz}. The hazard is controlled by a timed automaton, where the change of the value of the data variable `H` causes the DDC formula $\updownarrow H$ to be satisfied. We require that nonzero time passes before the hazard is detected. See also Figure 7.7.

We proceed to explain A^i_{Fwd} and A^i_{Det}. We model the communication chain in Uppaal by assigning to the i-th car in the chain the ID i and let i_C be the ID of car C. Then the first car in the chain is the closest to the hazard. We assume that with its communication range a car can reach the cars that are adjacent in the chain. We model the sending of the chain by sending its own position in it.

For each car we have two controllers. A hazard detection controller that initiates the protocol when the controller is informed of a hazard. The other controller is a hazard forwarding controller that forwards a warning it received from another car. When the detection controller of a car C is made aware of the hazard it changes the value of the data variable `I` for this car (with `set_is_informed`()). When this happens the formula $\updownarrow I_C$ is satisfied. If there are no other cars to inform the controller returns to its initial location and is done. Otherwise the controller starts to forward the warning to nearby cars. If the controller does not receive a confirmation that the warning has been received it resends the warning after some delay. For car C receiving the value $i_C + 1$ serves as confirmation. The forwarding controller behaves almost the same as the detector, with the difference that it is informed by another car and not by the environment. Further, the forwarding controller also sends a signal when it is the last car in the chain, to let the previous car know that it has been informed successfully.

The forwarding and detection controllers have two phases. A waiting phase, where the controller either waits to be warned of a hazard, or has been warned and performed its obligation to inform the next car. The other phase is a communication phase, where the controller tries to inform the next car of the hazard. This happens in transitions with `h_e`! and the variable update `s` = `id` as explained for the value sending pattern. When a controller receives a value (transitions with `h_r`?) it always checks if it was supposed to receive this value with the guards `not near`(`l`) and `near`(`l`), i.e. it checks whether the car sending the value is nearby. If it was not supposed to receive the value, it deletes the value it received (`l` = `nil`).

Finally, we explain $A_{\mathtt{Ch}}$. We model limited range broadcast communication as a separate automaton; the channel controller. First, we explain the variables of the controller and then the behaviour. The controller has an array `clocks` to keep track of how long ago a communication started. We assume that it takes `t_c` time units to finish a communication. Additionally, the controller has a list of cars currently communicating, sorted descending by the values in the array `clocks`. We query and update this list with the functions `is_empty`, `push`, `head` and `tail`. The first car in the list is the next car whose communication finishes. We store this car in the variable `n`.

We describe the behaviour. The automaton has two main control states, indicating whether the list of ongoing communications is empty or not. If the list is empty and a car starts emitting a signal (indicated by `h_e`?), the controller stores the value to be send (which we assume is the ID of the sending car) in a local variable `l`, resets the corresponding clock, and adds the ID to the internal list. Afterwards, the controller waits in `not_empty` for the next communication to finish. If, while waiting, another car starts to communicate

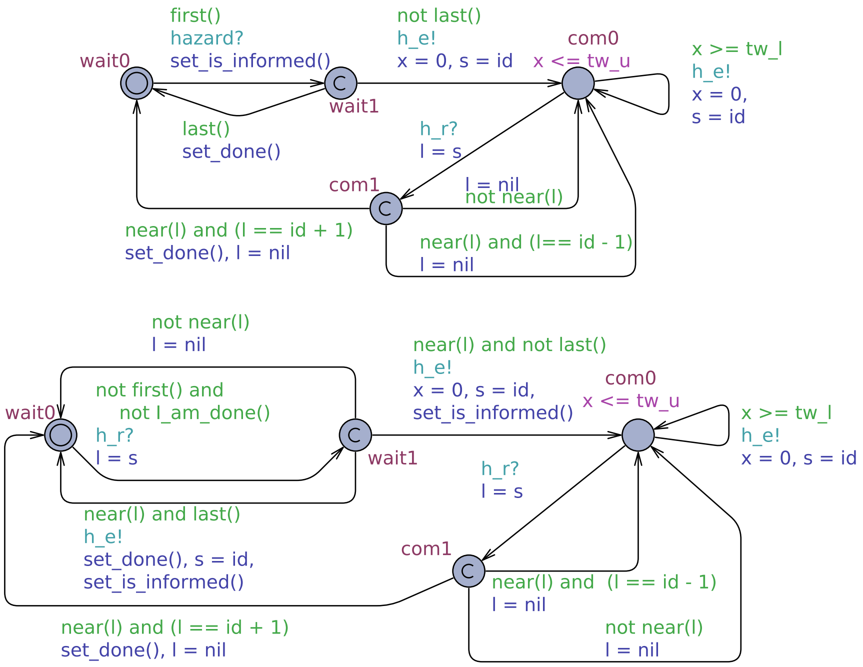


Figure 7.8: The hazard detection controller A^i_{Det} (top) and hazard forwarding controller A^i_{Fwd} (bottom). The only difference between the detection and forwarding controller lies in the transitions leaving the locations `wait0` and `wait1`.

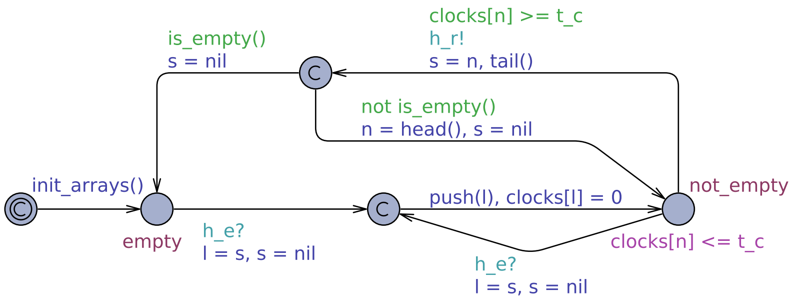


Figure 7.9: The channel controller A_{Ch}.

we perform the same actions for this car as for the earlier one. When the next communication has finished we set the global variable `s` to the ID of the sender, inform all cars that a communication has finished and update the internal list by removing the in first entry. The cars themselves will sort out if they are in the communication range of the sender. Afterwards, if the list is not empty, we set the variable `n` to the ID of the car whose communication is due next.

We point out that `h_e` is a normal channel (between two participants) and that `h_r` is a broadcast channel to inform multiple cars simultaneously. In general, as we have multiple automata with transitions labelled with `h_e`! which, at a given time, they may be forced to take, and only a single automaton with transitions labelled with `h_e`?, such a setting may lead to deadlocks. We avoid this by ensuring that all transitions with `h_e`! lead to noncommitted locations, while all transitions with `h_e`? lead (through committed locations) to locations from where another transition labelled with `h_e`? starts. We can easily verify deadlock freedom with Uppaal.

Computing the Satisfaction Value

In Chapter 7 we introduced an algorithm to perform approximate model checking for a fragment of DDC. However, the algorithm is not implemented. Thus, we use an alternative approach to approximate how well our protocol satisfies ϕ for a particular distribution of cars.

Before we compute the satisfaction value we make a small detour. We argued that this example showcases that the fragment of DDC for which approximative model checking automatically is possible, is useful. For automation our constraints on timed automata are that they are delaying, strongly non-Zeno and strongly bounded (cf. Pages 17, 205 and 220). An automaton is delaying if between any value change of its state variables nonzero time passes. It is strongly non-Zeno if there is a strictly positive constant c such that in any control cycle at least c time units pass. And it is strongly bounded if every clock has in every location an upper bound for the maximal value the clock can reach.

For the ease of exposition we presented the hazard warning protocol in a way where these constraints are not satisfied. However, we briefly outline that we can slightly adapt the automata to satisfy these constraints. These adaptations do not alter the behaviour in any way.

Our protocol is delaying. To see this we point out that we only have the state variables H and I_C for all cars C under consideration. The state variable H is represented by the data variable $\mathtt{H}$ and I_C is represented by $A^{i_C}_{\mathsf{Det}}.\mathtt{I}$ for the first car in the communication chain and by $A^{i_C}_{\mathsf{Fwd}}.\mathtt{I}$ for the other cars. All of these data variables are changed at most once some nonzero time after initialisation, which means that the protocol indeed is delaying.

Next, to see that our protocol is strongly non-Zeno we have to look at the complete network, and not at the individual automata.

At last, we can make our protocol strongly bounded without affecting the behaviour. Note that except A_{Ch} all automata have exactly one local clock, and that we do not have global clocks. To make our protocol strongly bounded we can for all automata except A_{Ch} and all noncommitted locations without an upper bound add an nonzero upper bound and let the clock be reset whenever the bound is reached. In A_{Ch} we have inactive clocks and active clocks that currently track an ongoing communication. Inactive clocks are currently not used and we can arbitrarily reset them and place upper bounds on them. For active clocks, as all communications take $\mathtt{t_c}$ time units, we can add this as an upper bound. We can reset an active clock when the communication is finished and the clock becomes inactive.

We come back from the detour and compute how well our hazard warning protocol satisfies ϕ for the distribution of cars depicted in Figure 7.4. In this distribution we have two cars in sector near, two cars in sector middle and one car in sector far. Additionally, we choose a desired precision of $\epsilon = 0.05$. The approach we use here is to discretize the interval of truth values $[0, 1]$ by a value that is smaller than the desired precision ϵ to have some tolerance in the values we actually chose. This results in finer discretization than necessary. Let this

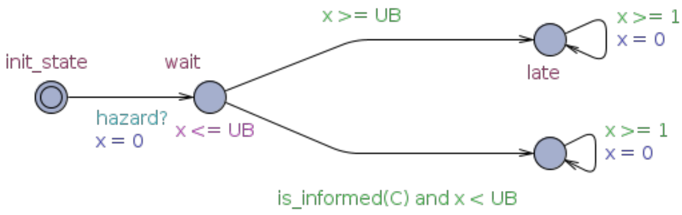


Figure 7.10: Observer A_{Obs} of when car C is informed. If C is either later than some upper bound, stored in the constant `UB`, or never informed of the hazard, then the automaton enters the bad location `late`.

set of discretised truth values be V. Then we check for each sector s, each truth value $v \in V$ and for each car C in sector s, if C is always either never informed or $\eta_s^{-1}(v)$ or more time units informed after the hazard has been detected. Note that the inverse of η_s is defined on the range $(0, 1]$ as discounting functions are strongly decreasing and asymptotically converge to 0. Finally, the result is the smallest truth value for which we found a run. Let the set of relevant points in time for our set of truth values be $T_s = \{\eta_s^{-1}(v) \mid v \in V\}$. For $t \in T_s$ we check whether on all paths the car C is informed t or more time units after the hazard has been detected. If this holds, the overall truth value is less or equal $\eta_s(t)$. To formalise this we add the automaton in Figure 7.10 to the protocol to create the network A_t^C, where the Uppaal constants `UB` and `C` are set to t and i_C. Then we check with Uppaal if A_t^C satisfies the property

$$\psi \equiv A_{\mathsf{Haz}}.\texttt{hazard\_detected} \leadsto A_{\mathsf{Obs}}.\texttt{late} \ ,$$

where $\leadsto$ is the *leads-to* operator of Uppaal. Our approximation then is

$$\min_{\substack{s \in \{\mathsf{n},\mathsf{m},\mathsf{f}\},\\ C \in CS_s}} (\eta_s(t) \mid t \in T_s, A_t^C \models \psi) \ .$$

As each communication attempt succeeds and takes 1 time unit it is not surprising that we find that the i-th car in the communication chain is *not* informed later than i time units. Using our set of points in time T_s to discretize the possible truth values for each discounting function independently we get the results summarised in Table 7.1. Note that we mentioned earlier that we discretize with a smaller value than $\epsilon = 0.05$. In fact, we used a varying step

Table 7.1: For each sector we show for the car informed last in this sector two choices of delays from T_s. In a column we use ✓ to indicate $A_t^C \models \psi$ and ✗ to indicate $A_t^C \not\models \psi$. This means for example that it always takes at least 2.83 and never 3.06 or more time units to inform car 3. Similarly, it takes at least 1.01 and never 4.01 or more time units to inform car 4. Note that the indices refer to the communication chain $\langle L, H, G, B\rangle$ shown in Figure 7.4, from which we removed car A as it detected the hazard. Thus, car 1 is L.

	car 1 in sector near		car 3 in sector middle		car 4 in sector far	
reachable	✓	✗	✓	✗	✓	✗
time-bound UB	1	1.06	2.83	3.06	1.01	4.01
discounting value $\eta_s(\mathtt{UB})$	0.6	0.585	0.789	0.759	0.990	0.960

size of 0.025 ± 0.01 to only have values in T_s with at most two digits behind the decimal point. This seems helpful as Uppaal only allows integer values in transition guards. In Table 7.1 it may seem odd that we have an earlier time for car B than for car G, even though in the communication chain car G occurs before car B. But this is due to our choice of time points T_s. The overall satisfaction of ϕ by our protocol is 0.6, which is ok, but not good.

7.3.3 Hazard Warning with Communication Failures

We introduce communication failures (e.g. by hardware glitches or simultaneous communication attempts of nearby cars). We express the quality of communication as a DDC property and check our earlier property "after a hazard every car soon is informed of the hazard" under the assumption of this quality. Note that we only sketch this example, and that we did not actually model it.

With the formula

$$\mathrm{G}_{\updownarrow F}\,(\Box^{\eta_{\mathsf{f}}} \lceil F \rceil \vee \lceil \neg F \rceil \vee \ell = 0)$$

we express that after a communication failure, for a long time (with discount η_{f}) communication is free of failures. Let s be one of the sectors near, middle and far and let CS_s be the set of cars in the sector s. Then the final formula is

$$\phi_s \equiv \bigwedge_{C \in CS_s} \left(\mathrm{G}_{\updownarrow F} \left(\Box^{\eta_{\mathsf{f}}} \lceil F \rceil \vee \lceil \neg F \rceil \vee \ell = 0 \right) \implies \mathrm{G}_{\updownarrow H} \left(\Diamond^{\eta_s} \updownarrow I_C \right) \right) .$$

For the discounting function measuring the communication quality we introduce another set of relevant time points T_{f} that discretises the interval of truth values $[0, 1]$ according to η_{f}. To compute how strongly the protocol with communication failures A satisfies the formula ϕ_s we create for each car $C \in CS_s$, each $t_0 \in T_s$ and $t_1 \in T_{\mathsf{f}}$ a model $A^C_{t_0,t_1}$. Now, $A^C_{t_0,t_1}$ requires at least t_1 time units to pass between two failures and the location $A_{\mathsf{Obs}}.\mathtt{late}$ only is reached, when either C is never informed, or C is informed t_0 or more time units after the hazard has occurred. If this manipulated model satisfies the TCTL formula

$$\psi \equiv A_{\mathsf{Haz}}.\mathtt{hazard\_detected} \leadsto A_{\mathsf{Obs}}.\mathtt{late} .$$

the truth value of ϕ_s for car C on the original model is less or equal

$$\mathsf{max}(1 - \eta_{\mathsf{f}}(t_1), \eta_s(t_0)) .$$

Our approximated overall satisfaction is

$$\mathop{\mathsf{min}}_{\substack{s \in \{\mathsf{n},\mathsf{m},\mathsf{f}\}, \\ C \in CS_s}} (\mathsf{max}(1 - \eta_{\mathsf{f}}(t_1), \eta_s(t_0)) \mid t_0 \in T_s, t_1 \in T_{\mathsf{f}}, A^C_{t_0,t_1} \models \psi) .$$

7.4 Related Work

Robustness of Temporal Logics

In most approaches to formalise a robust temporal logic we also have a real-valued interpretation [FH05; FP09]. However, there we want to know how much we can perturb a behaviour such that the property still holds. In other words, given that there usually is a difference between how a behaviour is perceived with sensors and how it actually occurs in the physical world, we want to quantify the maximal difference between these two behaviours, such that the result on the first behaviour is transferable to the second behaviour.

With discounting we consider different properties. We consider properties where the maxim “the sooner the better” holds, and we check “how good is it”. This rating favouring earlier satisfaction is not considered with temporal robustness.

Model Checking DC

In [ZH04] the authors define model checking of linear duration invariants, which is a fragment of DC of the form

$$c_0 \leq \ell \leq c_1 \implies \sum_{i=1}^{n} k_i * \textstyle\int S_i \leq c_2 \ ,$$

where $c_0, c_1, c_2, k_i \in \mathbb{R}$, S_i is a state expression and ℓ is the length of the current interval. They do this via a reduction to linear programming, which essentially is the quantifier and disjunction free fragment of FOLRA. The linear duration invariants they consider subsume for example the property

$$\mathrm{G}\,(\ell \geq 60 \implies 20 * \textstyle\int \mathit{Leak} \leq \ell)$$

of the gas burner case study [RRH93], where G means globally. The property requires that in any interval of length ≥ 60, gas should leak for at most 5% of the time. As model the authors consider a restricted form of timed automata featuring a single clock that is reset on every transition.

In [FH07] the authors devise an algorithm for model checking formulas without negation and where all comparisons only feature upper bounds, i.e. the fragment given by the following EBNF:

$$\phi ::= \sum_{i=1}^{n} k_i * \textstyle\int S_i \lesssim c \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \frown \phi_2 \ ,$$

where $\lesssim \in \{<, \leq\}$, $c \in \mathbb{R}$ and $\frown$ is the chop operator as in MLSL. As model the authors consider timed automata. They reduce the model checking problem to optimal reachability for multi priced timed automata.

In [MFH+08] the authors define model checking for a fragment of DC allowing infinite data types. As model the authors use phase event automata, which are similar to timed automata. The fragment of DC the authors consider features the atoms $\ell \sim k$, $\lceil S \rceil$ and disallows the integral operator. Note that their state expressions are quite powerful, as they allow arbitrary quantifier free first-order logic expressions over timed variables. An example is $\lceil \mathit{pos}_1 < \mathit{pos}_2 \rceil$, where pos_1 and pos_2 are timed variables with range $\mathbb{R}$. For model checking they follow the automata theoretic approach [VW86] and use the symbolic constraint solver ARMC [PR07] as backend. Because of the infinite domains their procedure may not terminate.

Discounting

In [dFH+05] the authors define a version of computation tree logic (CTL) with discounting, for which they define algorithms to perform model checking on Markov chains and transition systems. They propose two different semantics, which they call path semantics and fixpoint semantics. The path semantics is a natural introduction of discounting into CTL. Let $c \in [0, 1]$ be a discount. Then, to get the truth value of $\exists\Diamond_c\phi$, in a state s they first take the supremum of all paths starting in s, and then they take the supremum of when we check ϕ on the path, where each step into the future is discounted by c. This is similar to our approach. The fixpoint semantics is a restriction of a semantics for μ-calculus with discounting, defined in [dHM03], to CTL. The authors compare and investigate the two semantics and show that on transition systems the semantics are equivalent, but not on Markov chains.

In [ABK16] the authors define LTL[$\mathcal{D}$], which extends linear temporal logic (LTL) with discounting. They approach the model checking problem by solving the threshold problem. That is, for a given real-valued truth value $v \in [0, 1]$, Kripke structure K and LTL[$\mathcal{D}$] formula ϕ they check whether evaluating ϕ on K results in a truth value greater equal v. Their approach is similar to the classical automata based model checking approach for LTL. They combine the recursive unfolding of the until operator with an unfolding of the discounting function. Eventually, when the truth value becomes smaller than v they stop the unfolding. Further, the authors show that with exponential discounting the complexity of their algorithm is in PSPACE, just like the classical approach.

8 Conclusion

8.1 Summary

We proved the satisfiability problem of Multi-Lane Spatial Logic (MLSL) with an unbounded number of lanes to be undecidable. To show this, we reduced the language emptiness problem of the intersection of context-free languages to the lane-unbounded satisfiability problem of MLSL. This reduction holds even if we assume that spatial information is imperfect.

Furthermore, we defined an extension of MLSL called Multi-Lane Spatial Logic with Scopes (MLSLS). We used this extension to show that with a bounded number of cars, the lane-unbounded satisfiability problem is decidable. To prove the correctness of our algorithm we introduced operations on MLSLS models.

We extended this approach to also solve the model problem of MLSLS, i.e. deciding whether a given model satisfies a given formula. For the model problem we again consider precise and imprecise spatial information separately.

We then extended on this to show that also for MLSLS transition sequences the monitoring problem (as the model problem for a given behaviour of a dynamic system and a temporal formula is called) is decidable. For this we related MLSLS transition sequences to timed words. Similar to the static case we defined operations on these transition sequences to allow for a compositional analysis. For these dynamic systems we also consider the case of imprecise information; here robustness has a spatial and a temporal dimension.

Finally, we formalised properties favouring early satisfaction by introducing discounting into a variant of Duration Calculus. With this extension we express properties such as “always after a hazard as occurred, all relevant cars are soon informed of it”. We showed that for a relevant fragment of our extension the model checking problem is approximable, where the model is given as a timed automaton.

8.2 Future Work

8.2.1 Multi-Lane Spatial Logic

Decidability It is desirable to develop a better understanding of the border of decidability of the satisfiability problem of MLSL. To this end, other fragments of MLSL should be investigated. Interesting fragments may be defined, for example by restricting the nesting depth of chop operators or the number of different car variables (similar to two-variable first-order logic [GKV97]). Further, it may be interesting to see if the satisfiability problem with a bounded number of lanes is decidable. In this thesis, we called this the lane-bounded satisfiability problem.

Stochastic MLSL In this thesis we considered only small spatio-temporal perturbations. Additionally, we assumed that we know for certain whether there is a car or not. However, usually this assumption will not be satisfied, i.e. there are objects where there is uncertainty about whether the sensors perceive an obstacle, such as a car, or an object that is not an obstacle, such as a plastic bag. Or, we know that there is a car, but we are uncertain about the lane the car occupies. It is desirable to incorporate this uncertainty into MLSL and define a stochastic spatio-temporal logic.

Monitoring So far we can only use MLSLS to monitor properties only using the temporal globally operator. It is desirable to extend the kind of properties for which we can use MLSL to perform monitoring. To this end, a fully fledged linear time version of MLSL should be defined. This version could be created by adapting our temporal globally operator to a timed version of until, similar to Metric Temporal Logic [Koy90]. Furthermore, an online monitoring algorithm is desirable that can check the property as the behaviour is observed.

Bounded Model Checking of Controllers There are approaches to extend timed automata [AD94] with MLSL to define controllers of traffic manoeuvres [HLO13; Sch18a]. While there exist first steps towards model checking such controllers [Sch18b; BS19], these approaches impose strong restrictions on the controllers. We have shown how to check MLSLS formulas with arithmetic logics and how to extend this to transition sequences. Further, as we have seen in Chapter 7, there are approaches to unfold the transition relation of timed automata with these arithmetic logics [TMM02; BL11; KJN12]. It is worthwhile to connect both approaches, i.e. to perform bounded model checking for timed automata extended with MLSLS, as this would impose less restrictions than the existing approaches in [Sch18b; BS19]. For a logic suitable to define desirable properties of these extended timed automata see the previous paragraph.

Traffic Sequence Charts As we have briefly pointed out in the related work section of Chapter 5, traffic sequence charts were defined to formally specify properties for sets of traffic manoeuvres [DMP+18]. This is very similar to how we used MLSLS in Chapters 5 and 6. Hence, a thorough comparison of the two formalisms is desirable.

8.2.2 Discounted Duration Calculus

Complexity In this thesis we show for a fragment of Discounted Duration Calculus that approximate model checking is *possible*. We did not consider efficiency at all. Thus, our reduction to first-order logic of linear real arithmetic is not suited for implementation because the resulting formula becomes huge. An algorithm that is suitable for implementation, together with a complexity analysis, is desirable.

Weaken Restrictions To show that model checking for a fragment of Discounted Duration Calculus is approximable we had to impose restrictions on the timed automata. That is, our algorithm only works when the timed automata are strongly non-Zeno, and if the globally operator is used, the timed automata also need to be strongly bounded. Weakening these restrictions certainly is desirable.

Bibliography

[ABK14] S. Almagor, U. Boker, and O. Kupferman. “Discounting in LTL”. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Ed. by E. Ábrahám and K. Havelund. Vol. 8413. LNCS. Springer, 2014, pp. 424–439. DOI: `10.1007/978-3-642-54862-8_37`.

[ABK16] S. Almagor, U. Boker, and O. Kupferman. “Formally Reasoning About Quality”. In: *J. ACM* 63.3 (2016), 24:1–24:56. DOI: `10.1145/2875421`.

[AD14] M. Althoff and J. M. Dolan. “Online Verification of Automated Road Vehicles Using Reachability Analysis”. In: *IEEE Transactions on Robotics* 30.4 (2014), pp. 903–918. DOI: `10.1109/TRO.2014.2312453`.

[AD94] R. Alur and D. L. Dill. “A Theory of Timed Automata”. In: *Theoretical Computer Science* 126.2 (1994), pp. 183–235. DOI: `10.1016/0304-3975(94)90010-8`.

[AFH96] R. Alur, T. Feder, and T. A. Henzinger. “The benefits of relaxing punctuality”. In: *J. ACM* 43.1 (1996), pp. 116–146. DOI: `10.1145/227595.227602`.

[AFS04] L. de Alfaro, M. Faella, and M. Stoelinga. “Linear and Branching Metrics for Quantitative Transition Systems”. In: *International Colloquium on Automata, Languages and Programming (ICALP)*. Ed. by J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella. Vol. 3142. LNCS. Springer, 2004, pp. 97–109. DOI: `10.1007/978-3-540-27836-8_11`.

[All83] J. F. Allen. “Maintaining knowledge about temporal intervals”. In: *Communications of the ACM* 26.11 (1983), pp. 832–843. DOI: `10.1145/182.358434`.

[AMP+98] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. “Controller Synthesis for Timed Automata”. In: *IFAC Proceedings Volumes* 31.18 (1998). IFAC Conference on System Structure and Control, pp. 447–452. DOI: 10.1016/S1474-6670(17)42032-5.

[ANB98] H. Andréka, I. Németi, and J. van Benthem. “Modal Languages and Bounded Fragments of Predicate Logic”. In: *J. Philosophical Logic* 27.3 (1998), pp. 217–274. DOI: 10.1023/A:1004275029985.

[BDL] G. Behrmann, A. David, and K. G. Larsen. *A Tutorial on Uppaal.* Updated version of [BDL04]. URL: http://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf.

[BDL04] G. Behrmann, A. David, and K. G. Larsen. “A Tutorial on Uppaal”. In: *Formal Methods for the Design of Real-Time Systems.* Ed. by M. Bernardo and F. Corradini. LNCS 3185. Springer–Verlag, 2004, pp. 200–236. DOI: 10.1007/978-3-540-30080-9_7.

[BHL+17] G. v. Bochmann, M. Hilscher, S. Linker, and E.-R. Olderog. “Synthesizing and verifying controllers for multi-lane traffic maneuvers”. In: *Formal Aspects of Computing* 29.4 (2017), pp. 583–600. DOI: 10.1007/s00165-017-0424-4.

[BL11] B. Badban and M. Lange. “Exact Incremental Analysis of Timed Automata with an SMT-Solver”. In: *Formal Modeling and Analysis of Timed Systems.* Ed. by U. Fahrenberg and S. Tripakis. Vol. 6919. LNCS. Springer, 2011, pp. 177–192. DOI: 10.1007/978-3-642-24310-3_13.

[BLP+99] G. Behrmann, K. G. Larsen, J. Pearson, C. Weise, and W. Yi. “Efficient Timed Reachability Analysis Using Clock Difference Diagrams”. In: *Computer Aided Verification (CAV).* Ed. by N. Halbwachs and D. A. Peled. Vol. 1633. LNCS. Springer, 1999, pp. 341–353. DOI: 10.1007/3-540-48683-6_30.

[BLS11] A. Bauer, M. Leucker, and C. Schallhart. “Runtime Verification for LTL and TLTL”. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 20.4 (2011), 14:1–14:64. DOI: 10.1145/2000799.2000800.

[BM07] A. R. Bradley and Z. Manna. *The calculus of computation - decision procedures with applications to verification.* Springer, 2007. DOI: 10.1007/978-3-540-74113-8.

[BMG+08] D. Bresolin, D. D. Monica, V. Goranko, A. Montanari, and G. Sciavicco. "Decidable and Undecidable Fragments of Halpern and Shoham's Interval Temporal Logic: Towards a Complete Classification". In: *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*. Ed. by I. Cervesato, H. Veith, and A. Voronkov. Vol. 5330. LNCS. Springer, 2008, pp. 590–604. DOI: `10.1007/978-3-540-89439-1_41`.

[BS19] C. Bischopink and M. Schwammberger. "Verification of Fair Controllers for Urban Traffic Manoeuvres at Intersections". publication pending. 2019.

[BY03] J. Bengtsson and W. Yi. "On Clock Difference Constraints and Termination in Reachability Analysis of Timed Automata". In: *International Conference on Formal Engineering Methods (ICFEM)*. Ed. by J. S. Dong and J. Woodcock. Vol. 2885. LNCS. Springer, 2003, pp. 491–503. DOI: `10.1007/978-3-540-39893-6_28`.

[CDF+05] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. "Efficient On-the-Fly Algorithms for the Analysis of Timed Games". In: *Concurrency Theory*. Ed. by M. Abadi and L. de Alfaro. Vol. 3653. LNCS. Springer, 2005, pp. 66–80. DOI: `10.1007/11539452_9`.

[dFH+05] L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. "Model checking discounted temporal properties". In: *Theoretical Computer Science* 345.1 (2005), pp. 139–170. DOI: `10.1016/j.tcs.2005.07.033`.

[dHM03] L. de Alfaro, T. A. Henzinger, and R. Majumdar. "Discounting the Future in Systems Theory". In: *Automata, Languages and Programming*. Ed. by J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger. Vol. 2719. LNCS. Springer, 2003, pp. 1022–1037. DOI: `10.1007/3-540-45061-0_79`.

[DM10] A. Donzé and O. Maler. "Robust Satisfaction of Temporal Logic over Real-Valued Signals". In: *Formal Modeling and Analysis of Timed Systems (FORMATS)*. Ed. by K. Chatterjee and T. A. Henzinger. Vol. 6246. LNCS. Springer, 2010, pp. 92–106. DOI: `10.1007/978-3-642-15297-9_9`.

[DMP+18] W. Damm, E. Möhlmann, T. Peikenkamp, and A. Rakow. "A Formal Semantics for Traffic Sequence Charts". In: *Principles of Modeling*. Ed. by M. Lohstroh, P. Derler, and M. Sirjani. Vol. 10760. LNCS. Springer, 2018, pp. 182–205. DOI: `10.1007/978-3-319-95246-8_11`.

[FH05] M. Fränzle and M. R. Hansen. "A robust interpretation of duration calculus". In: *International Colloquium on Theoretical Aspects of Computing (ICTAC)*. Ed. by Dang Van Hung and Martin Wirsing. Vol. 3722. LNCS. Springer, 2005, pp. 257–271. DOI: `10.1007/11560647_17`.

[FH07] M. Fränzle and M. R. Hansen. "Deciding an Interval Logic with Accumulated Durations". In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Ed. by O. Grumberg and M. Huth. Vol. 4424. LNCS. Springer, 2007, pp. 201–215. DOI: `10.1007/978-3-540-71209-1_17`.

[FHO15] M. Fränzle, M. R. Hansen, and H. Ody. "No Need Knowing Numerous Neighbours - Towards a Realizable Interpretation of MLSL". In: *Correct System Design*. Ed. by R. Meyer, A. Platzer, and H. Wehrheim. Vol. 9360. LNCS. Springer, 2015, pp. 152–171. DOI: `10.1007/978-3-319-23506-6_11`.

[FP09] G. E. Fainekos and G. J. Pappas. "Robustness of Temporal Logic Specifications for Continuous-Time Signals". In: *Theoretical Computer Science* 410.42 (2009), pp. 4262–4291. DOI: `10.1016/j.tcs.2009.06.021`.

[FR74] M. J. Fischer and M. O. Rabin. "Super-Exponential Complexity of Presburger Arithmetic". In: *Complexity of Computation*. Ed. by R. M. Karp. SIAM-AMS Proceedings 7. See [FR98] for reprint. AMS, 1974, pp. 27–41.

[FR75] J. Ferrante and C. Rackoff. "A Decision Procedure for the First Order Theory of Real Addition with Order". In: *SIAM Journal on Computing* 4.1 (1975), pp. 69–76. DOI: `10.1137/0204006`.

[FR98] M. J. Fischer and M. O. Rabin. "Super-Exponential Complexity of Presburger Arithmetic". In: *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Ed. by B. F. Caviness and J. R. Johnson. Reprint of [FR74]. Springer, 1998, pp. 122–135. DOI: `10.1007/978-3-7091-9459-1_5`.

[Frä99] M. Fränzle. “Analysis of Hybrid Systems: An Ounce of Realism Can Save an Infinity of States”. In: *Computer Science Logic (CSL)*. Ed. by J. Flum and M. Rodríguez-Artalejo. Vol. 1683. LNCS. Springer, 1999, pp. 126–140. DOI: `10.1007/3-540-48168-0_10`.

[GAA+12] T. M. Gasser, C. Arzt, M. Ayoubi, A. Bartels, J. Eier, F. Flemisch, D. Häcker, T. Hesse, W. Huber, C. Lotz, M. Maurer, S. Ruth-Schumacher, J. Schwarz, and W. Vogt. *Rechtsfolgen zunehmender Fahrzeugautomatisierung*. Tech. rep. Heft F 83. Bundesanstalt für Straßenwesen, 2012. URL: `https://www.bast.de/BASt_2017/DE/Publikationen/Foko/2013-2012/2012-11.html`.

[GHJ97] V. Gupta, T. A. Henzinger, and R. Jagadeesan. “Robust Timed Automata”. In: *Hybrid and Real-Time Systems*. Ed. by O. Maler. Vol. 1201. LNCS. Springer, 1997, pp. 331–345. DOI: `10.1007/BFb0014736`.

[GKV97] E. Grädel, P. G. Kolaitis, and M. Y. Vardi. “On the decision problem for two-variable first-order logic”. In: *Bulletin of Symbolic Logic* 3.1 (1997), pp. 53–69. DOI: `10.2307/421196`.

[HKT01] D. Harel, D. Kozen, and J. Tiuryn. “Dynamic logic”. In: *SIGACT News* 32.1 (2001), pp. 66–69. DOI: `10.1145/568438.568456`.

[HLO+11] M. Hilscher, S. Linker, E.-R. Olderog, and A. P. Ravn. “An Abstract Model for Proving Safety of Multi-Lane Traffic Manoeuvres”. In: *International Conference on Formal Engineering Methods (ICFEM)*. Ed. by S. Qin and Z. Qiu. Vol. 6991. LNCS. Springer, 2011, pp. 404–419. DOI: `10.1007/978-3-642-24559-6_28`.

[HLO13] M. Hilscher, S. Linker, and E.-R. Olderog. “Proving Safety of Traffic Manoeuvres on Country Roads”. In: *Theories of Programming and Formal Methods*. Ed. by Z. Liu, J. Woodcock, and H. Zhu. Vol. 8051. LNCS. Springer, 2013, pp. 196–212. DOI: `10.1007/978-3-642-39698-4_12`.

[Hoe06] J. Hoenicke. “Combination of processes, data, and time”. PhD thesis. Carl von Ossietzky University of Oldenburg, 2006.

[HR00] T. A. Henzinger and J. Raskin. “Robust Undecidability of Timed and Hybrid Systems”. In: *Hybrid Systems: Computation and Control (HSCC)*. Ed. by N. A. Lynch and B. H. Krogh. Vol. 1790. LNCS. Springer, 2000, pp. 145–159. DOI: `10.1007/3-540-46430-1_15`.

[HS91] J. Y. Halpern and Y. Shoham. “A propositional modal logic of time intervals”. In: *J. ACM* 38.4 (1991), pp. 935–962. DOI: 10.1145/115234.115351.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, 1979. ISBN: 0-201-02988-X.

[Jan94] W. Janssen. *Layered design of parallel systems.* CIP-Gegevens Koninklijke Bibliotheek, 1994. ISBN: 978-90-9007399-6.

[Kha79] L. G. Khachiyan. “A polynomial algorithm in linear programming”. In: *Doklady Academii Nauk SSSR.* Vol. 244. 1979, pp. 1093–1096.

[KJN12] R. Kindermann, T. A. Junttila, and I. Niemelä. “Beyond Lassos: Complete SMT-Based Bounded Model Checking for Timed Automata”. In: *Joint Proceedings of the IFIP WG 6.1 conferences Formal Methods for Open Object-Based Distributed Systems (FMOODS) and Formal Techniques for Networked and Distributed Systems (FORTE).* Ed. by H. Giese and G. Rosu. Vol. 7273. LNCS. Springer, 2012, pp. 84–100. DOI: 10.1007/978-3-642-30793-5_6.

[Kop91] H. Kopetz. “Event-Triggered Versus Time-Triggered Real-Time Systems”. In: *Operating Systems of the 90s and Beyond.* Ed. by A. I. Karshmer and J. Nehmer. Vol. 563. LNCS. Springer, 1991, pp. 87–101. DOI: 10.1007/BFb0024530.

[Koy90] R. Koymans. “Specifying Real-Time Properties with Metric Temporal Logic”. In: *Real-Time Systems* 2.4 (1990), pp. 255–299. DOI: 10.1007/BF01995674.

[LH15] S. Linker and M. Hilscher. “Proof Theory of a Multi-Lane Spatial Logic”. In: *Logical Methods in Computer Science* 11.3 (2015). DOI: 10.2168/LMCS-11(3:4)2015.

[Lin15] S. Linker. “Proofs for traffic safety : combining diagrams and logic”. PhD thesis. Carl von Ossietzky University of Oldenburg, 2015. URL: http://oops.uni-oldenburg.de/2337/.

[LPN11] S. M. Loos, A. Platzer, and L. Nistor. “Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified”. In: *Formal Methods (FM).* Ed. by M. J. Butler and W. Schulte. Vol. 6664. LNCS. Springer, 2011, pp. 42–56. DOI: 10.1007/978-3-642-21437-0_6.

[LPR+95] K. Lodaya, R. Parikh, R. Ramanujam, and P. S. Thiagarajan. “A Logical Study of Distributed Transition Systems”. In: *Information and Computation* 119.1 (1995), pp. 91–118. DOI: 10.1006/inco.1995.1078.

[Man12] E. Mandrali. “Weighted LTL with Discounting”. In: *Implementation and Application of Automata*. Ed. by N. Moreira and R. Reis. Vol. 7381. LNCS. Springer, 2012, pp. 353–360. DOI: 10.1007/978-3-642-31606-7_32.

[Maz86] A. W. Mazurkiewicz. “Trace Theory”. In: *Advances in Petri Nets*. Ed. by W. Brauer, W. Reisig, and G. Rozenberg. Vol. 255. LNCS. Springer, 1986, pp. 279–324. DOI: 10.1007/3-540-17906-2_30.

[MFH+08] R. Meyer, J. Faber, J. Hoenicke, and A. Rybalchenko. “Model checking Duration Calculus: a practical approach”. In: *Formal Aspects of Computing* 20.4-5 (2008), pp. 481–505. DOI: 10.1007/s00165-008-0082-7.

[MGL+15] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner. *Autonomes Fahren - Technische, rechtliche und gesellschaftliche Aspekte*. Springer, 2015. DOI: 10.1007/978-3-662-45854-9.

[MN04] O. Maler and D. Nickovic. “Monitoring Temporal Properties of Continuous Signals”. In: *Proceedings of the joint conference on Formal Modelling and Analysis of Timed Systems (FORMATS) and Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT)*. Ed. by Y. Lakhnech and S. Yovine. Vol. 3253. LNCS. Springer, 2004, pp. 152–166. DOI: 10.1007/978-3-540-30206-3_12.

[Mon08] D. Monniaux. “A Quantifier Elimination Algorithm for Linear Real Arithmetic”. In: *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*. Ed. by I. Cervesato, H. Veith, and A. Voronkov. Vol. 5330. LNCS. Springer, 2008, pp. 243–257. DOI: 10.1007/978-3-540-89439-1_18.

[Mos85] B. C. Moszkowski. “A Temporal Logic for Multilevel Reasoning about Hardware”. In: *IEEE Computer* 18.2 (1985), pp. 10–19. DOI: 10.1109/MC.1985.1662795.

[MR14] E. Mandrali and G. Rahonis. “On weighted first-order logics with discounting”. In: *Acta Informatica* 51.2 (2014), pp. 61–106. DOI: 10.1007/s00236-013-0193-3.

[OD08] E. Olderog and H. Dierks. *Real-time systems - formal specification and automatic verification.* Cambridge University Press, 2008. DOI: 10.1017/CBO9780511619953.

[Ody15a] H. Ody. *Undecidability Results for Multi-Lane Spatial Logic.* Reports of SFB/TR 14 AVACS 112. 2015. URL: http://www.avacs.org/fileadmin/Publikationen/Open/avacs_technical_report_112.pdf.

[Ody15b] H. Ody. "Undecidability Results for Multi-Lane Spatial Logic". In: *International Colloquium on Aspects of Computing (ICTAC).* Ed. by M. Leucker, C. Rueda, and F. D. Valencia. Vol. 9399. LNCS. Springer, 2015, pp. 404–421. DOI: 10.1007/978-3-319-25150-9_24.

[Ody17] H. Ody. "Monitoring of Traffic Manoeuvres with Imprecise Information". In: *Workshop on Formal Verification of Autonomous Vehicles.* Ed. by L. Bulwahn, M. Kamali, and S. Linker. Vol. 257. Electronic Proceedings in Theoretical Computer Science (EPTCS). 2017, pp. 43–58. DOI: 10.4204/EPTCS.257.6.

[OFH16] H. Ody, M. Fränzle, and M. R. Hansen. "Discounted Duration Calculus". In: *Formal Methods (FM).* Ed. by J. S. Fitzgerald, C. L. Heitmeyer, S. Gnesi, and A. Philippou. Vol. 9995. LNCS. Springer, 2016, pp. 577–592. DOI: 10.1007/978-3-319-48989-6_35.

[OS17] E.-R. Olderog and M. Schwammberger. "Formalising a Hazard Warning Communication Protocol with Timed Automata". In: *Models, Algorithms, Logics and Tools.* Ed. by L. Aceto, G. Bacci, G. Bacci, A. Ingólfsdóttir, A. Legay, and R. Mardare. Vol. 10460. LNCS. Springer, 2017, pp. 640–660. DOI: 10.1007/978-3-319-63121-9_32.

[Pla10a] A. Platzer. *Logical Analysis of Hybrid Systems - Proving Theorems for Complex Dynamics.* Springer, 2010. DOI: 10.1007/978-3-642-14509-4.

[Pla10b] A. Platzer. "Quantified Differential Dynamic Logic for Distributed Hybrid Systems". In: *Computer Science Logic (CSL).* Ed. by A. Dawar and H. Veith. Vol. 6247. LNCS. Springer, 2010, pp. 469–483. DOI: 10.1007/978-3-642-15205-4_36.

[Pla18] A. Platzer. *Logical Foundations of Cyber-Physical Systems.* Springer, 2018. DOI: 10.1007/978-3-319-63588-0.

[PR07] A. Podelski and A. Rybalchenko. "ARMC: The Logical Choice for Software Model Checking with Abstraction Refinement". In: *Practical Aspects of Declarative Languages (PADL)*. Ed. by M. Hanus. Vol. 4354. LNCS. Springer, 2007, pp. 245–259. DOI: `10.1007/978-3-540-69611-7_16`.

[Pur00] A. Puri. "Dynamical Properties of Timed Automata". In: *Discrete Event Dynamic Systems* 10.1-2 (2000), pp. 87–113. DOI: `10.1023/A:1008387132377`.

[QFD11] J.-D. Quesel, M. Fränzle, and W. Damm. "Crossing the Bridge between Similar Games". In: *Formal Modeling and Analysis of Timed Systems (FORMATS)*. Ed. by U. Fahrenberg and S. Tripakis. Vol. 6919. LNCS. Springer, 2011, pp. 160–176. DOI: `10.1007/978-3-642-24310-3_12`.

[Que13] J.-D. Quesel. "Similarity, Logic, and Games - Bridging Modeling Layers of Hybrid Systems". PhD thesis. Carl von Ossietzky University of Oldenburg, 2013.

[RCC92] D. A. Randell, Z. Cui, and A. G. Cohn. "A Spatial Logic based on Regions and Connection". In: *Principles of Knowledge Representation and Reasoning (KR)*. Ed. by B. Nebel, C. Rich, and W. R. Swartout. Morgan Kaufmann, 1992, pp. 165–176.

[RRH93] A. P. Ravn, H. Rischel, and K. M. Hansen. "Specifying and Verifying Requirements of Real-Time Systems". In: *IEEE Transactions on Software Engineering* 19.1 (1993), pp. 41–55. DOI: `10.1109/32.210306`.

[Sch04] A. Schäfer. "A Calculus for Shapes in Time and Space". In: *International Colloquium on Theoretical Aspects of Computing (ICTAC)*. Ed. by Z. Liu and K. Araki. Vol. 3407. LNCS. Springer, 2004, pp. 463–477. DOI: `10.1007/978-3-540-31862-0_33`.

[Sch06] A. Schäfer. "Specification and verification of mobile real-time systems". PhD thesis. Carl von Ossietzky University of Oldenburg, 2006.

[Sch11] G. Schmidt. *Relational Mathematics*. Vol. 132. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011. DOI: `10.1017/CBO9780511778810`.

[Sch18a] M. Schwammberger. “An abstract model for proving safety of autonomous urban traffic”. In: *Theoretical Computer Science* 744 (2018), pp. 143–169. DOI: `10.1016/j.tcs.2018.05.028`.

[Sch18b] M. Schwammberger. “Introducing Liveness into Multi-lane Spatial Logic lane change controllers using UPPAAL”. In: *International Workshop on Safe Control of Autonomous Vehicles*. Ed. by M. Gleirscher, S. Kugele, and S. Linker. Vol. 269. EPTCS. 2018, pp. 17–31. DOI: `10.4204/EPTCS.269.3`.

[SFK08] M. Swaminathan, M. Fränzle, and J. Katoen. “The Surprising Robustness of (Closed) Timed Automata against Clock-Drift”. In: *IFIP International Conference On Theoretical Computer Science*. Ed. by G. Ausiello, J. Karhumäki, G. Mauri, and C. L. Ong. Vol. 273. IFIP Advances in Information and Communication Technology. Springer, 2008, pp. 537–553. DOI: `10.1007/978-0-387-09680-3_36`.

[Spi92] J. M. Spivey. *Z Notation - a reference manual (2. ed.)* Prentice Hall International Series in Computer Science. Prentice Hall, 1992. ISBN: 978-0-13-978529-0.

[SRR90] E. V. Sørensen, A. P. Ravn, and H. Rischel. *Control Program for a Gas Burner: Part 1: Informal Requirements, PrCoS Case Study 1*. Tech. rep. Report No. ID/DTH EVS2. Department of Computer Science, Technical University of Denmark, 1990.

[SS78] L. A. Steen and J. A. Seebach Jr. *Counterexamples in Topology*. eng. Second Edition. Topological spaces. Springer, 1978. DOI: `10.1007/978-1-4612-6290-9`.

[Tar51] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Tech. rep. Prepared for Publication with the Assistance of J.C.C. McKinsey. 1951. URL: `https://www.rand.org/pubs/reports/R109.html`.

[TMM02] S. L. Torre, S. Mukhopadhyay, and A. Murano. “Optimal-Reachability and Control for Acyclic Weighted Timed Automata”. In: *IFIP International Conference on Theoretical Computer Science*. Ed. by R. A. Baeza-Yates, U. Montanari, and N. Santoro. Vol. 223. IFIP Conference Proceedings. Kluwer, 2002, pp. 485–497. DOI: `10.1007/978-0-387-35608-2_40`.

[VW86] M. Y. Vardi and P. Wolper. "An Automata-Theoretic Approach to Automatic Program Verification (Preliminary Report)". In: *Logic in Computer Science (LICS)*. IEEE Computer Society, 1986, pp. 332–344.

[Wei99] V. Weispfenning. "Mixed Real-Integer Linear Quantifier Elimination". In: *International Symposium on Symbolic and Algebraic Computation (ISSAC)*. Ed. by K. O. Geddes, B. Salvy, and S. S. Dooley. ACM, 1999, pp. 129–136. DOI: 10.1145/309831.309888.

[WHL+15] H. Winner, S. Hakuli, F. Lotz, and C. Singer, eds. *Handbuch Fahrerassistenzsysteme, Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. Springer, 2015. DOI: 10.1007/978-3-658-05734-3.

[WHO18] World Health Organization. *Global status report on road safety 2018: summary*. 2018. URL: https://www.who.int/violence_injury_prevention/road_safety_status/2018.

[Win86] G. Winskel. "Event Structures". In: *Petri Nets: Central Models and Their Properties*. Ed. by W. Brauer, W. Reisig, and G. Rozenberg. Vol. 255. LNCS. Springer, 1986, pp. 325–392. DOI: 10.1007/3-540-17906-2_31.

[ZH04] C. Zhou and M. R. Hansen. *Duration Calculus - A Formal Approach to Real-Time Systems*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2004. DOI: 10.1007/978-3-662-06784-0.

[ZH97] C. Zhou and M. R. Hansen. "An Adequate First Order Interval Logic". In: *Compositionality: The Significant Difference*. Ed. by W. P. de Roever, H. Langmaack, and A. Pnueli. Vol. 1536. LNCS. Springer, 1997, pp. 584–608. DOI: 10.1007/3-540-49213-5_23.

[ZHR91] C. Zhou, C. A. R. Hoare, and A. P. Ravn. "A Calculus of Durations". In: *Information processing letters* 40.5 (1991), pp. 269–276. DOI: 10.1016/0020-0190(91)90122-X.

[ZHS93] C. Zhou, M. R. Hansen, and P. Sestoft. "Decidability and undecidability results for duration calculus". English. In: *Symposium on Theoretical Aspects of Computer Science*. Ed. by P. Enjalbert, A. Finkel, and K. W. Wagner. Vol. 665. LNCS. Springer, 1993, pp. 58–68. ISBN: 978-3-540-56503-1. DOI: 10.1007/3-540-56503-5_8.

Index of Symbols

MLSL(S)
Chapters 2, 3, 4, 5 and 6

$\mathsf{acc}(C)$	acceleration of car C	22
$\boxminus$	anti-restriction of model	55
$\mathsf{cars}\cdot$	car domain of $\cdot$	53
C	car identifier	22
$\mathbb{I}$	set of all car identifiers	22, 53
CVar	set of all car variables	25
$\mathsf{cl}(\cdot)$	claim formula	28, 63
$\mathsf{clm}(C)$	set of claims of car C	22
$\boxplus$	disjoint union of models	55
D	annotation for downwards	35
ego	ego constant	25
E	view owner/ego car	24
$M_\emptyset$	empty MLSLS model	53
X	extension of a view	24
free	formula for free space	27, 63
$\mathsf{freeVar}(\phi)$	set of free car variables in ϕ	62
$\frown$	horizontal chop operator	28, 63
L	lanes of a view	24
$\mathbb{L}$	set of all lanes	22
letter	MLSL encoding of a letter	37
$\mathsf{letter}_{\mathsf{free}}$	as letter but surrounded by free space	37
$\mathsf{letter}^{\mathsf{r}}_{\mathsf{free}}$	like $\mathsf{letter}^{\mathsf{r}}$ but surrounded by free space	124
$\mathsf{letter}^{\mathsf{r}}$	robust MLSL encoding of a letter	123
d_{m}	metric on MLSL(S) models	121, 141
Act	set of all actions	24
$\mathrm{a}(\cdot,\cdot)$	set acceleration action	24
$\mathrm{wd\ c}(\cdot)$	remove claim action	23
$\mathrm{wd\ r}(\cdot,\cdot)$	remove reservation action	23
$\mathrm{c}(\cdot,\cdot)$	set claim action	23
$\mathrm{r}(\cdot)$	set reservation action	23
M	MLSL(S) model	25, 53
$\mathbb{M}$	set of all MLSL(S) models	25, 53
Φ	set of all MLSLS formulas	62
$\mathsf{pos}(C)$	position of car C	22
$\mathsf{re}(\cdot)$	reservation formula	28, 63
$\mathsf{res}(C)$	set of reservations of car C	22
$\boxed{\upharpoonright}$	restriction of model	55
se	safety envelope	26
$\models$	MLSL(S) satisfaction relation	27, 63
$\models^\delta$	robust satisfaction relation	121
$\mathsf{sat}(\phi)$	set of models satisfying ϕ	137
CS	scope of MLSLS model	53
$:$	scope operator	62
cs	set of car variables	62
Ω	sensor function	25
$\lvert M\rvert$	size of car domain of M	53
$\langle\cdot\rangle$	somewhere abbreviation	28
$\mathsf{spd}(C)$	speed of car C	22
$\mathbb{TS}$	set of all traffic snapshots	22
TS	traffic snapshot	22
true	formula that is always satisfied	28
$F(G_{\mathsf{D}}, G_{\mathsf{U}})$	undecidable MLSL formula	41
$F_{\mathrm{robust}}(G_{\mathsf{D}}, G_{\mathsf{U}})$	robust version of $F(G_{\mathsf{D}}, G_{\mathsf{U}})$	125
U	annotation for upwards	35
ν	car variable valuation	25
I_V^M	set of cars in view V in model M	26
V	view of MLSL(S) model	24
$\mathbb{V}$	set of views	24
$\obar$	horizontal view composition	24
$\ominus$	vertical view composition	24
$\eqsim_{\mathsf{m}}$	weak equality of models	69

Timed Words & Transition Sequences
Chapters 2, 5 and 6

$\setminus$	timed anti-restriction	153, 158
$\varpi@t$	in ϖ model at time t	157
$\leftrightharpoons$	causal congruence	180
$\cdot$	concatenation of timed words	8
$\approx$	equality up to empty sets of actions	154, 157
last	last model in transition sequence	157
d_{t}	metric on timed words	183
$\models_{\mathsf{seq}}$	spatio-temporal MLSLS satisfaction relation	161
$\boxed{\Vert\vert}$	disjoint parallel composition of transition sequences	160

$\parallel$	parallel composition of timed words	152
$\mathsf{pre}(\cdot,\cdot)$	timed word prefix	155
$\upharpoonright$	timed restriction	153, 158
span	time span	8, 157
τ	sequence of time stamps	8
ϱ	timed word	8
$(\mathcal{P}(\Sigma) \times \mathbb{T})^{+}$	set of all timed words over Σ	8
ϖ	MLSLS transition sequence	157
w	word	8

Arithmetic Logics – All chapters

$\ominus$	anti-restriction of FOMLA tuples	87
NTerm	set of all terms ranging over $\mathbb{N}$	31
RTerm	set of all real-valued terms	31
κ	FOMLA variable assignment	30
cars	car domain of FOMLA tuple	86
g	car identifier variable assignment	114
DS	car identifier variable scope	75
D	arithmetic car identifier variable	75
$v_{\mathsf{p}}, v_{\Omega}, v_{\mathsf{r1}}, v_{\mathsf{r2}}, v_{\mathsf{c}}, v_{\mathsf{s}}, v_{\mathsf{a}}$	data variables	75, 113
$\mathcal{S}(D)$	data variables of D	75
$\oplus$	disjoint union of FOMLA tuples	86
$(\kappa_{\emptyset}, \Upsilon_{\emptyset})$	empty FOMLA tuple	82
$=_{\mathsf{r}}$	equality up to renaming	87
$\mathsf{mla2m}$	transformation of FOMLA tuples to MLSLS models	89
Υ	FOMLA data tuple	75
$Z(D)$	sequence of real-valued terms of D	165
$\mathrm{T}^{\mathsf{rcf}}$	set of possibly simple FORCF data tuples	162
$\Pi(D)$	sequence of data tuples of D	165
π	sequence of data tuples	164
$\mathsf{m2mla}$	transformation of MLSLS models to FOMLA tuples	91
$\models$	satisfaction relation	29
$*$	multiplication	29
NVar	natural number-valued variables	30
$\bullet$	value for unassigned variables	75
RVar	set of all real-valued variables	29
$\obar$	restriction of FOMLA tuples	87
$D.\mathsf{se}$	abbreviation for safety envelope	77
$\mathsf{sane}(\Upsilon)$	sanity formula	77
$\underline{\ \cdot\ }$	sequence of variables in bounded unfolding	210, 212
DVar	set of car identifier variables	75
$\mathsf{sim}_{\mathsf{m}}$	arithmetic formula for similarity of models	142, 143
$\mathsf{sim}_{\mathsf{t}}$	arithmetic formula for similarity of timed words	186, 187
simple	transformation of proper to simple FOMLA tuple	82
$\lvert(\kappa, \Upsilon)\rvert$	size of car domain of (κ, Υ)	86
α	natural number-valued term representing a view	75
β	real-valued term representing a view	76
θ	real-valued term representing a point in time	31
tr_{a}	transforms a single action-delay pair to FORCF constraints	163
$\mathsf{tr}_{\mathsf{f}}^{\mathsf{lra}}$	transforms MLSLS formula to FOLRA constraints	113
$\mathsf{tr}_{\mathsf{f}}^{\mathsf{mla}}$	transforms MLSLS formulas to FOMLA constraints	77
$\mathsf{tr}_{\mathsf{m}}^{\mathsf{lra}}$	transforms MLSLS models to FOLRA constraints	115
tr_{U}	encodes “transition sequence until time t”	165, 167
tr_{w}	encodes transition sequence for one car	164
h	FORCF variable assignment	29
$\eqsim_{\mathsf{mla}}$	weak equality of FOMLA tuples	83

Timed Automata and DDC Chapters 2 and 7

$\mathbb{B}$	set of Boolean constraints	14
I	clock invariants per location	15
$\mathcal{X}$	set of clocks	15
$\mathrm{DDC}_{<1}$	set of all DDC formulas with discount less than 1	200
$\rightarrow$	set of edges	15
$\updownarrow$	DDC encoding of event	202
$\lceil S \rceil$	DDC formula for “S holds in current interval”	202
$\mathrm{F}_{\updownarrow S}\, \phi$	DDC formula for “eventually S happens and ϕ holds”	202
$\mathrm{G}_{\updownarrow S}\, \phi$	DDC formula for “whenever S happens ϕ holds”	202

Init	set of initial configurations	15
τ	internal action	15
Lab	set of labels for synchronisation	15
l	location	15
L	set of locations	15
$\mathsf{mc}(\cdot,\cdot)$	model checking function	205
$\mathsf{pre}(\cdot,\cdot)$	timed prefix for trajectories	206
τ_δ	prefix of τ until δ	206
$\Diamond$	some right neighbourhood modality	200, 201
$\Box$	every right neighbourhood modality	202
Λ	invariant for state variables	15
V	set of state variables	15
$\mathcal{T}(A)$	set of trajectories of A	18
τ	trajectory	14
β	valuation of state variables	15
$\mathsf{Val}(\cdot)$	set of valuations	15
ν	valuation of clocks	15

Other – All chapters

$\lvert\cdot\rvert$	set cardinality and also absolute value	5, 12
$\mathrm{CNF}^{-\epsilon}$	grammars in CNF without ϵ-rules	10
$\overline{\cdot}$	complementation of sets and right end of interval	5, 13
$\circ$	relational composition	6
$s_1 \cdot s_2$	sequence concatenation	7
G	context-free grammar	9
$\setminus$	difference of sets	5
$\uplus$	disjoint union of sets	6
dom	domain of relation	6
$⩤$	domain anti-restriction	6
$\lhd$	domain restriction	6
$\langle\rangle$	empty sequence	6
front	front of a sequence	7
head	first element of sequence	7
inf	infimum	12
$\cap$	intersection of sets	5
$\underline{\cdot}$	left end of interval	13
$\lVert\cdot\rVert$	length of interval	13
last	last element of sequence	7
$\#\cdot$	length of sequences/words	6, 7
$\prec_{\mathsf{l}}$	lexicographic order relation	7
$*$	multiplication	29
$\mathbb{N}$	set of all natural numbers	5
$\mathcal{N}$	set of nonterminals	9
$\pm$	plus/minus	12
$\mathcal{P}(\cdot)$	powerset	5
$\sqsubseteq$	prefix relation	7
$s[..i]$	prefix of s until $s(i)$	7
$::$	sequence prepending and also appending	7
ran	range of relation	6
$⩥$	range anti-restriction	6
$\rhd$	range restriction	6
$\mathbb{Q}$	set of all rational numbers	5
$\mathbb{R}$	set of all real numbers	5
$\mathbb{R}_{>0}$	set of strictly positive reals	5
$(\!\lvert\cdot\rvert\!)$	relational image	6
$\mathcal{R}$	context-free rewrite rules	9
$\mathsf{Seq}\,\cdot$	set of sequences over $\cdot$	7
S	starting nonterminal	9
$s[i..]$	suffix of s starting at $s(i)$	7
sup	supremum	12
$\equiv$	syntactic equivalence	5
$\mathsf{tail}(\cdot)$	tail of a sequence	7
$\mathcal{T}$	set of terminals	9
Υ	tree	8
$\mathsf{T}(\Sigma)$	set of Σ-labelled trees	9
$\cup$	union of sets	5
w	word	8

Index of Subjects