

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

Proofs for Traffic Safety

Combining Diagrams and Logic

Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften

vorgelegt von

Sven Linker

Tag der Disputation: 20.02.2015

Gutachter:

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

Abstract

Due to the increasing interest in autonomously driving cars, safety issues of such systems are of utmost importance. Safety in this sense is primarily the absence of collisions, which is inherently a spatial property. Within computer science, typical models of cars include specifications of their behaviour, where the space a car needs for operating safely is a function of time. This complicates proofs of safety properties tremendously.

In this thesis, we present methods to separate reasoning on space from the dynamical behaviour of cars. To that end, we define an abstract model with an emphasis on spatial transformations of the situation on the road. Based on this model, we develop two formalisms: We give the definitions of a modal logic suited to reason about safety properties of arbitrarily many cars. Furthermore, we present a diagrammatic language to ease the specification of such properties. We formally prove that no collisions arise between cars obeying a small set of requirements.

Zusammenfassung

Durch das steigende Interesse an autonomen Fahrzeugen gewinnt deren Sicherheit immer stärker an Bedeutung. Hierbei ist Sicherheit gleichbedeutend mit Kollisionsfreiheit, eine grundsätzlich räumliche Eigenschaft. Fahrzeugmodelle in der Informatik beinhalten Spezifikationen des dynamischen Verhaltens, so dass der zum sicheren Betrieb nötige Raum abhängig von der Zeit ist. Dies erschwert Sicherheitsbeweise enorm.

In dieser Arbeit stellen wir Methoden vor, um Schlussfolgerungen über den Raum vom Fahrzeugverhalten abzutrennen. Hierzu definieren wir ein abstraktes Modell mit dem Schwerpunkt auf den räumlichen Veränderungen der Straßensituation. Darauf aufbauend entwickeln wir zwei Formalismen: Wir definieren eine Modallogik, mit der Aussagen über die Sicherheit von beliebig vielen Fahrzeugen bewiesen werden können. Weiterhin stellen wir Diagramme zur einfacheren Spezifikation solcher Eigenschaften vor. Wir beweisen die Kollisionsfreiheit zwischen Fahrzeugen, die wenige Anforderungen erfüllen.

Acknowledgements

I could not have finished my thesis without the help of many different people. First and foremost, I thank Ernst-Rüdiger Olderog for supporting me throughout my time at the University of Oldenburg, both during my undergraduate studies as well as during my time as a Ph.D. student. Furthermore, I thank Michael Reichhardt Hansen for being my external examiner and for the helpful feedback he gave during my visit in Lyngby.

While I wrote the text of this document, Manuel Giesecking, Martin Hilscher and Maïke Schwammlinger were willing to suffer and proofread different chapters. I am especially indebted to Manuel Giesecking, who checked many of the proofs. All of the remaining errors are due to myself.

The pleasant atmosphere in the working group “Correct System Design” was strongly shaped by all of my former and current colleagues along the different groups working on theoretical computer science in Oldenburg: Marion Bramkamp, Björn Engelmann, Evgeny Erofeev, Johannes Faber, Hans Fleischhack, Nils-Erik Flick, Sibylle Fröschle, Manuel Giesecking, Andrea Göken, Martin Hilscher, Sören Jeserich, Stephanie Kemper, Heinrich Ody, Christoph Peuser, Jan-David Quesel, Hendrik Radke, Uli Schlachter, Maïke Schwammlinger, Tim Strazny, Mani Swaminathan, Patrick Uven, Ira Wempe and Elke Wilkeit. Thank you very much.

The beginning of my Ph.D. studies was influenced immensely by my fellows in the graduate school “TrustSoft”: Amna Asif, Ahmad El Maamoun, Kinga Kiss-Iakab, Henrik Lipskoch, Nils Müllner, Felix Oppermann, Hendrik Radke and Astrid Rakow. Many thanks to all of you for the instructive meetings, the great trips to Dagstuhl and the nice discussions. It was an honour to meet you. Especially, I would like to thank Ira Wempe for being a great coordinator of the graduate school and for listening to all the problems we had.

Of course, there are people who supported me outside of academia. I would like to thank Christian Bruns, Markus Dahlke, Karsten Fritzsche, Manuel Giesecking, Niels Hapke, Markus Lohmeyer, Andreas Schäfer, Alexander Skobel and Tim Strazny for their constant support and time, for relaxed discussions, for great vacation trips, and in general for being friends. Special thanks go to my current and former flatmates Holger Brettschneider, Marcus Dreßler, Sebastian Richter and in particular Cathrin Vogel for helping me during the hard times of the last years, as well as for being part of the fun activities we had together. Thank you, I owe you very much.

Contents

1	Introduction	1
1.1	Contributions	3
1.2	Structure of this Thesis	4
2	Preliminaries	5
2.1	Mathematical Notations	5
2.2	Temporal Logic and Interval Logic	7
2.3	Labelled Natural Deduction	8
2.4	Graph Rewriting	11
3	Spatial Model of Traffic	19
3.1	Abstract Road	20
3.2	Bounded Visibility	25
3.3	Sensor Models	27
3.4	Related Work	28
4	Modal Logic for Freeway Traffic	31
4.1	Syntax and Semantics	31
4.2	Proof System	36
4.3	Undecidability of Spatial MLSL	59
4.4	Related Work	65
5	Visual Logic for Freeway Traffic	69
5.1	Concrete Syntax	70
5.2	Conveniences	76
5.3	Formalising Sanity	77
5.4	Abstract Syntax	80
5.5	Semantics	97
5.6	Decidability of Spatial Traffic Diagrams	104
5.7	Related Work	107
6	Combining Text and Diagrams	111
6.1	Comparison of Expressivity	111
6.2	Verification Framework	121

6.3	Related Work	122
7	Case study	125
7.1	Controller Specification	126
7.2	Safety Proof	127
7.3	Refining the Specification	133
8	Conclusion	137
8.1	Summary	137
8.2	Future Work	138
	Bibliography	141
	Index	150
	Symbol Index	150
	Subject Index	152

List of Figures

1.1	Decomposing Reasoning about Traffic	2
2.1	Semantics of the chop modality	7
2.2	Example of a Graph Rewriting Rule	13
2.3	The graph G	13
2.4	Possible Results of Applying r once to G	14
2.5	The Hyperedge Replacement System \mathcal{R}	16
2.6	Example of a Graph Rewriting Rule with an Application Condition	16
3.1	Situation on a Freeway at a Single Point in Time	20
5.1	Spatial Traffic Diagram	70
5.2	Example of a Topological Sequence	72
5.3	Example of a Lane Sequence	73
5.4	Example of Distance Arrows	73
5.5	Omission of Notation for Conjunction	76
5.6	Ambiguities with Sequences	76
5.7	Sanity Conditions as Diagrams (I)	78
5.7	Sanity Conditions as Diagrams (II)	79
5.7	Sanity Conditions as Diagrams (III)	79
5.8	Connectedness of Cars	80
5.9	Alignment of Cars	80
5.10	Abstract Syntax of Fig. 5.1	83
5.11	Concrete and Abstract Syntax of a Diagram with Duration Arrows	84
5.12	Rule Sets R_{LS} , R_{SEP} , R_{CH} and R_{OCC} : Lane/Topological Sequences	86
5.13	Rule Sets R_{BL} , R_{LA} , R_T and R_I : Structure of Spatial Diagrams	87
5.14	Hyperedge Replacement Grammar for the Definition of Paths	89
5.15	Rule R_{CS}^{sing} : Distance Arrow from Left to Right of a Topological Situation	89
5.16	Different Possibilities for Distance Arrows	90
5.17	Application Condition for Distance Arrows between Different Lanes	90
5.18	Rules R_{CS}^1 and R_{CS}^2 : Distance Arrow between Cars	91
5.19	Rules R_{CS}^3 and R_{CS}^4 : Distance Arrow attached to Left of Target	92
5.20	Rules R_{CS}^5 and R_{CS}^6 : Distance Arrow attached to Right of Source	93

5.21	Rule R_{CS}^7 : Distance Arrow on the Same Lane from Right to Left	94
5.22	Rule R_{CS}^8 : Distance Arrow on Different Lanes from Right to Left	94
5.23	Rule Sets R_S , R_{BD} , R_L and R_{AT} : Structure of Traffic Diagrams	95
5.24	Semantics for Logical Connectives	98
5.25	Semantics of Sequences	99
5.26	Semantics of Layers	100
5.27	Semantics of Lane Sequences	101
5.28	Semantics of Topological Sequences	102
5.29	Semantics of Distance Arrows	103
5.30	Concrete and Abstract Syntax of an Example Diagram	104
5.31	Example of a Spatial Diagram	104
5.32	Problematic Example for Decidability	106
5.33	A Time-Distance Diagram for Railways	107
5.34	A Time-Space Diagram	108
6.1	Schema of a Spatial Diagram	112
6.2	Example for the Metric Transformation	113
6.3	New Rule R_{BD}^5 : Formulas	121
6.4	Example of a Combination of EMLSL and Traffic Diagrams	122
6.5	Proving Safety with EMLSL and Traffic Diagrams	122
7.1	Positive Conditions for the Creation of Reservations	134
7.2	Withdrawal of Reservations	135
7.3	Distance Controller Preserving Free Space in Front of Cars	136

List of Tables

4.1	Instructions for Counter c_1 of a Two-Counter Machine	59
5.1	Diagrammatic Elements of Traffic Diagrams	71
5.2	Terminal Types of the Abstract Syntax	82

1

Introduction

The amount of individual traffic is still on the rise, and will probably continue to do so in the near future. The reduction of accidents in spite of the increase of traffic density is therefore a main goal of much research involving the development of cars. Advanced driver assistance systems support the human driver by supplying different kinds of information. They may, e.g., display warnings whenever the current velocity is higher than allowed. Furthermore, they often provide visual means to inform the driver whether changing lanes is possible in a safe way. However, the human element within traffic is still a source for unsafe situations, e.g., when drivers overestimate the capabilities of their car, or are exhausted due to a long drive.

To further ensure safety, the development of automated cars capable of driving autonomously has been and is still of strong interest. The PATH project [Hsu+94] was one of the pioneering projects constructing and analysing the behaviour of fully autonomous cars. The results of this project led to the identification of several manoeuvres the controllers of such cars have to support. Furthermore, it was possible to derive constraints ensuring safety, i.e., the prevention of collisions. Several safe controllers for cars have been presented, e.g., by Lygeros et al. [LGS98] and Jula et al. [JKI99] within this project.

In all of these works, the dominant role in both the specification of the controllers, as well as in proving their safety, is played by the car dynamics. Safety in these approaches is always defined as the avoidance of collisions, which is an inherently spatial property. By using differential equations, the positions and braking distances of cars are only derived elements of the model, which tremendously increases the complexity of proofs.

Intuitively, safety of traffic is only dependent on the local environment of each car. Consider for example two cars C and D . If the distance between both is very large, e.g., 100 kilometres, the behaviour of one should not concern the safety of the other. Instead of mimicking such a property on a syntactic level, it should be inherent in the model definition itself to avoid clutter in specifications.

We want to specifically address the problem of safety on freeways with these ideas in

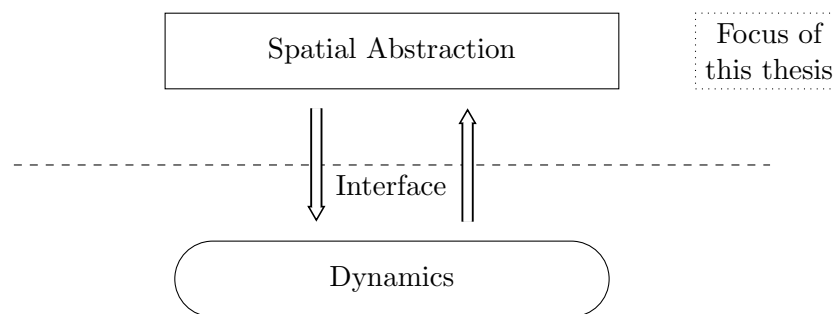


Figure 1.1: Decomposing Reasoning about Traffic

mind. To that end, we want to reason *locally* with an emphasis on the *spatial* properties of traffic. For that, we want to examine how spatial properties and locality restrictions can be incorporated into a model of traffic as first-class citizens, and in what way such a model eases proofs of spatial safety predicates. That is, we decompose the reasoning process into two levels as shown in Fig. 1.1. On the upper level, the topology of space and its evolution over time is the main focus of reasoning. That is, the model should reflect which parts of the freeway are occupied by cars and how these parts change while the cars drive along their current lane, and when they change lanes. On the lower level, the dynamics specify the concrete trajectories, how the cars perform these changes. Hence there has to be an exchange of information between both levels, as indicated by the arrows in Fig. 1.1. On the one hand, the abstraction has to get information about the size of the occupied space from the dynamics. On the other hand, mode changes within the layer of dynamics can be initiated by the upper level. Now assume that we have specified a protocol for the behaviour of cars on the spatial abstraction and shown that it prohibits collisions. Then we only have to prove that the dynamic layer respects the constraints for the behaviour implied by the upper level, and that it provides the spatial abstraction with the correct information for our protocol to work.

For formal reasoning purposes, the use of logics and especially modal logics has a long standing tradition within computer science. Logic provides a good trade-off between being succinct and precise. Furthermore, such a logical approach benefits from meta-theoretical research, e.g., the development of proof systems or methods to decide satisfiability of formulas algorithmically. Proof systems allow the user to reason only on formulas and to disregard the concrete semantic structures, as long as the main properties of the semantics are captured within axioms and proof rules. They may also be implemented within interactive or automated theorem provers, thus providing the user with the guarantee to create a correct proof. Tools checking satisfiability may be used for example to decide whether a formal syntactic proof should be attempted. For this, the desired property is negated and then checked for satisfiability. If the algorithm decides that the negated property can be satisfied, all attempts to formally prove the validity of the property are futile. Due to these advantages, many approaches to prove safety (or other interesting properties) of software and hardware devices make extensive use of logical methods and formulas in general.

However, within engineering practice, formulas and mathematical descriptions lack this acceptance. There, visual languages play a more prominent role. For example, for software engineering, the Unified Modelling Language (UML) [RJB04; UML12] is well-established. It incorporates, e.g., Message Sequence Charts (MSCs) [ITU96], which are used to define the communication between several independent objects, and timing diagrams [UML12], which are able to express timing relations between events occurring in chip-designs. Similarly, depictions of electronic circuits are used widely to communicate about specifications and implementations of computing devices. But most of these diagrammatic languages lack a precise semantics. For example, within MSCs, it is not always clear whether a diagram states a possible or a necessary course of events.

Still, several diagrammatic languages have been provided with a formal semantics. While general mathematical properties may be captured by formal Euler- and Venn-Diagrams [Shi95; AB96], only few diagrams extensively used in computer science were formalised in this way. MSCs have been enhanced by a mathematical semantics in the definition of Life Sequence Charts [DH01], which also extend MSCs with methods to distinguish universal and existential sequences. Schlör examined timing diagrams formally [Sch01]. In the context of real-time systems, Kleuker presented a visual specification language called Constraint Diagrams [Kle00].

1.1 Contributions

With these considerations at hand, the contributions of this work are the following. We define a model of freeway traffic that clearly distinguishes space from the dynamical behaviour of cars. This model also emphasises the local environment of a single car, and by that is a suitable model for the remainder of this thesis. An explicit formalisation of the sensors provided by the cars on the freeway completes the model.

Using this model as the semantics, we present the *extended multi-lane spatial logic* (EMLSL) to reason locally about spatial properties of freeway traffic. While the logic cannot express concrete properties of the dynamics, it is able to distinguish different discrete changes in the space on the road. Within the logic, both qualitative aspects, i.e., the topological situations on the freeway, as well as quantitative aspects, e.g., the length of free space ahead of a car or the number of lanes between two different cars, can be referred to. As a first step towards tool assistance, we present a formal proof system in the style of natural deduction and derive several useful rules within this system.

To further ease the use of our approach, we define a formal diagrammatic language called *Traffic Diagrams*, which uses the abstract model of traffic as a semantics. Similar to the logic, the diagrams incorporate methods to reason about the existence of transitions and to measure distances between cars. For a clearly defined syntax, we do not only define the visual elements used within the diagrams, but give a formal syntactic definition in terms of hypergraphs. This formal syntax may be used to take advantage of already existing methods and tool support for diagram parsing.

We compare the expressiveness of Traffic Diagrams and EMLSL. As it will turn out, the diagrams are not as expressive as the logic. To that end, we present a translation

of diagrams into equivalent formulas. Furthermore, we discuss how the diagrams and formulas may be used in conjunction to exploit their respective advantages.

Finally, we apply all of these formalisms to specify a small set of requirements restricting the possible behaviour of cars. In addition, we define a safety predicate, which is expressing the absence of collisions. We then formally prove the safety predicate to be invariant along all possible transition sequences of the abstract model of traffic with the help of the deductive system for the logic.

1.2 Structure of this Thesis

After this introduction, we present definitions and conventions that we will use throughout this thesis in Chap. 2. Chapter 3 is devoted to the development of the abstract model of traffic that we use as a semantics. In Chap. 4, we define the logic EMLSL and prove its undecidability. Furthermore, this chapter contains the proof system together with several derived rules. Chapter 5 contains the definitions needed for the visual language of Traffic Diagrams. We present both the concrete and abstract syntax as well as a formal semantics. In addition, the chapter includes formalisations of certain properties the abstract model possesses and a sketch for a decidable subset of diagrams. The combination of EMLSL and Traffic Diagrams is explored in Chap. 6, in which we also compare the expressiveness of both approaches. Chapter 7 presents the application of the proof system and the diagrams to a case study, defining the required constraints for safe behaviour within freeway traffic. Finally, Chap. 8 concludes the thesis.

2

Preliminaries

Contents

2.1	Mathematical Notations	5
2.2	Temporal Logic and Interval Logic	7
2.3	Labelled Natural Deduction	8
2.4	Graph Rewriting	11

In this chapter, we present most of the work this thesis is based upon. We first give the basic notations used throughout the following work in Sect. 2.1. In Sect. 2.2 and Sect. 2.3 we recall the preliminaries for the first part of the thesis, i.e., the concepts and definitions of different interval logics and labelled natural deduction, respectively. Following that, we present graph rewriting in Sect. 2.4, which plays a crucial role for the syntax of the diagrams in this thesis.

2.1 Mathematical Notations

In this section, we introduce general mathematical notation. While the notation itself follows mathematical standards, we include it for reference purposes.

Sets For a set S , the powerset of S , i.e., the set of all subsets of S , is denoted by $\mathcal{P}(S)$. We use the usual notations \cup , \cap and \setminus for the union, intersection and difference of sets. To denote the disjoint union, we use the symbol \uplus . The Cartesian product of the sets S and T , i.e., the set of all ordered tuples (s, t) with elements taken from the sets S and T , is denoted by $S \times T$.

We use the standard notations for the different sets of numbers, i.e., \mathbb{N} for the positive

integers and \mathbb{R} for the real numbers. The positive reals are denoted by

$$\mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\} .$$

In this thesis, we will make extensive use of intervals over both the positive integers and the real numbers. We denote closed borders of intervals with brackets and open borders with parenthesis. Furthermore, we also allow for infinity as a right border. For example, a typical interval we will use is the interval of all positive reals $[0, \infty)$ ($= \mathbb{R}_+$). We denote the set of all *real-valued intervals*, i.e., intervals over the reals with \mathcal{I} . Sometimes, we want to use variables as the borders of an interval. We will use the notation \mathcal{I}_{Var} to refer to the set of all *variable intervals*, i.e., where the borders are either variables or real numbers. Observe that $\mathcal{I} \subset \mathcal{I}_{\text{Var}}$.

Let $i \in \mathcal{I}$ with the borders a and b , i.e., $i = [a, b]$, $i = (a, b]$, $i = [a, b)$ or $i = (a, b)$. We call the set $i \setminus \{a, b\}$ the *interior of i* , denoted by $\mathfrak{I}(i)$.

Relations and Functions If S and T are sets, $R \subseteq S \times T$ is a *relation between S and T* . The *domain* of R is the set of all elements of S which are related to an element of T . Similarly, the *range* of R is the set of all elements of T which are related to elements of S . If both the domain and range of R are subsets of the same set S , i.e., $R \subseteq S \times S$, then we call R a relation on S . We denote the reflexive and transitive closure of a relation R on a set S by R^* .

A *function* is a relation $f \subseteq S \times T$, which is functional and total, i.e.,

$$\begin{aligned} (s, t_1) \in f \text{ and } (s, t_2) \in f & \quad \text{implies} \quad t_1 = t_2 & \quad \text{(Functionality),} \\ \text{for all } s \in S \text{ there is a } t \in T & \quad \text{such that} \quad (s, t) \in f & \quad \text{(Totality) .} \end{aligned}$$

We denote the function f itself by $f: S \rightarrow T$ and its elements $(s, t) \in f$ by $f(s) = t$. If a relation f is only functional, but not total, we call f a *partial function*. If $f: S \rightarrow T$ is a (partial or total) function, the *image* of $S' \subseteq S$ is given by

$$f(S') = \{t \in T \mid \exists s \in S' \bullet f(s) = t\} .$$

Injectivity of a function means, that each element of the range is related to exactly one element of the function's domain, i.e.,

$$(s_1, t) \in f \text{ and } (s_2, t) \in f \quad \text{implies} \quad s_1 = s_2 \quad \text{(Injectivity) .}$$

If f is injective, we denote the *preimage* of an element t of T by $f^{-1}(t) = s$, where $f(s) = t$. To denote the *function modification*, we use notation taken from the Z specification language [Smi00]. That is, for the function f , we use $f \oplus \{x \mapsto y\}$ to denote the function which coincides with f except for $f(x) = y$. For two functions $f: S \rightarrow T$ and $g: T \rightarrow U$, the function $g \circ f: S \rightarrow U$ is the *composition* of g and f , given by $(g \circ f)(x) = g(f(x))$ as long as both $f(x)$ and $g(f(x))$ are defined.

A *sequence* is a function f with either the natural numbers or a finite subset of them as its domain. We will often denote a sequence f with the domain $\{0, \dots, n\}$ by $\langle c_0, \dots, c_n \rangle$,

where $f(i) = c_i$ for all $i \in \{0, \dots, n\}$. The elements c_i are the *values* of the sequence. If S is a set, we use the notation S^* for the set of all finite sequences with values in S . It will be clear from the context whether we use this notation for sequences or the reflexive transitive closure of a relation. If we want to apply the function g to all elements of the sequence $\langle c_0, \dots, c_n \rangle$, we also write $g(\langle c_0, \dots, c_n \rangle)$ for the sequence $\langle g(c_0), \dots, g(c_n) \rangle$.

2.2 Temporal Logic and Interval Logic

Reasoning about temporal changes by means of logical formulas has a long history. The first approach to use a *temporal logic*, i.e., a modal logic to formalise properties of time is due to Prior [Pri57]. He introduced the modalities G and F with the intended meaning of “it will always be the case” and “it will be the case”, respectively, as well as H and P , which stand for “it has always been the case” and “it was the case”, respectively. Prior analysed the modalities with respect to a totally ordered linear time scale, i.e., for two time points x and y , either x was later than y , or y was later than x , or both stood for the same time point. Such models are models of *linear time*. For the specification and verification of computational systems, *linear temporal logic* (LTL), as presented by Pnueli [Pnu77], is a typical formalism inspired by Prior’s work. In computer science, logics with semantics of *branching time* are also of strong importance, with the most famous example being *computational tree logic* (CTL), introduced by Emerson and Clarke [EC82]. Instead of interpreting time as a linearly ordered domain, they consider a tree of states, each of which may possess an arbitrary number of children. CTL then provides operators to reason about paths originating at the current node, and about the occurrence of states on these paths.

A different approach to describe temporal properties is due to both Moszkowski [Mos85] and Halpern and Shoham [HS91]. Halpern and Shoham introduced a modal logic, subsequently called HS, where the modalities correspond to Allen’s interval relations [All83]. The main application of HS and its extensions lies in the field of artificial intelligence, where the different relations describe the knowledge about time (or other domains) an agent may possess. Moszkowski introduced *interval temporal logic* (ITL) to specify and verify hardware specifications. The models of both of these approaches are based on (usually finite) intervals. In the following, we will concentrate on ITL.

Within ITL, intervals can be divided into their subintervals, to describe, e.g., the beginning or the end of the intervals. For this purpose, Moszkowski introduced the *chop modality* \frown . An interval $[a, b]$ satisfies a formula $\varphi \frown \psi$, if and only if there is a point c such that $a \leq c \leq b$, where $[a, c]$ satisfies φ and $[c, b]$ satisfies ψ . A depiction of this interpretation is given in Fig. 2.1. The syntax of ITL is basically first-order logic with

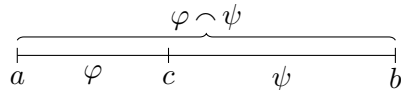


Figure 2.1: Semantics of the chop modality

the addition of the chop-modality:

$$\varphi ::= \perp \mid \theta_1 = \theta_2 \mid \varphi_1 \rightarrow \varphi_2 \mid \forall x \bullet \varphi_1 \mid \varphi_1 \frown \varphi_2 \ ,$$

where θ_1 and θ_2 are terms over a given signature. ITL distinguishes between *rigid* and *flexible* terms. The values of the former are given globally by a valuation of the variables, while the semantics of the latter may change with respect to the interval under consideration. A typical example of a flexible variable would be the length of an interval. We will not go further into the details of the semantics of ITL. Both ITL itself as well as extensions like the Duration Calculus [ZHR91] or Neighbourhood Logic [BRZ99] have been proven valuable for the specification and verification of real-time systems.

2.3 Labelled Natural Deduction

In his PhD thesis, Gentzen [Gen35] developed the calculus of *natural deduction* to present an alternative to axiom-based proof systems. Natural deduction is based on a set of *proof rules* and (temporary) assumptions, to give a tighter connection between the way mathematicians “naturally” prove theorems and the formal proof system. To that end, the system comprises an introduction and an elimination rule for each operator (with the possible exception of the falsum proposition \perp). A *derivation* within natural deduction consists of a tree, where the root is the theorem to prove and the leaves are the assumptions used within the proof. The branches from the root to the assumptions are determined by the structure of the proof rules. Each rule may possess application conditions, which restrict the form of the assumptions in the branches above the application of this rule. These conditions can ensure, e.g., freshness of variables or terms for rules concerning the quantors in first-order logic. A main feature of natural deduction is that rules may *eliminate* assumptions from the derivation. The underlying idea of this method is that the information of the eliminated assumption is already comprised within the conclusion of the rule. For example, if we can derive the truth of a formula ψ from the assumption that the formula φ holds, then $\varphi \rightarrow \psi$ reflects exactly this information. Hence, we can eliminate φ from the set of assumptions and disregard it in the rest of the derivation. This mechanism is a formal counterpart to the informal mathematical strategy to use temporary assumptions within a proof. For example, within case distinctions we temporarily assume the truth of each case and derive our desired conclusion independently. Within natural deduction, we would use the elimination rule for the disjunction operator for such a purpose. A *proof* in natural deduction is a derivation, where all assumptions have been eliminated. In derivations, we will mark eliminated assumptions by enclosing them in square brackets. To identify the application of the rule which was the reason why this particular assumption was eliminated, we add a unique index to both the assumption and the application of the rule within the tree.

While this approach is well-suited for classical logic and intuitionistic logic, other non-classical logics impose problems. For example, to define rules covering general modalities, application conditions for the rules have to constrain the set of all assumptions, not only the set of assumptions involved in the formula to prove [Pra06].

A way to preserve the underlying structure of natural deduction is to augment the formulas in the proof with information about the semantics. This leads to systems of *labelled natural deduction* (LND) [Sim94; BMV98; Vig00]. In labelled natural deduction, one of the basic entities are *labelled formulas* $w : \varphi$, where φ is a formula of the logic and w is a name of a world. Intuitively, the labelled formula $w : \varphi$ means that the formula φ holds at world w . The reachability relation between the worlds R is defined by *relational formulas* of the form wRv , where w and v are names for worlds. The formula wRv expresses that v is reachable from w . By adding suitable rules concerning relational formulas, different types of reachability relations may be defined. For example, the addition of the following rules defines, that R is both reflexive and transitive, and hence is the underlying reachability relation of all $S4$ -frames.

$$\frac{}{wRw} \quad \frac{wRv \quad vRu}{wRu}$$

For an operator \odot , we denote the *elimination rule* of \odot by $\odot E$. Similarly, we use the notation $\odot I$ for the *introduction rule* of \odot . The rules concerning classical propositional and first-order logic operators are similar to standard natural deduction rules. In the presentation of the rules, we will denote the *syntactical substitution* of x by a term t in the formula ϕ by $\phi[x \mapsto t]$. Following the notations of Basin et al. [BMV98], we use the name $\perp E$ for reductio ad absurdum. Note that this rule is the only one concerning falsum.

$$\begin{array}{c} \frac{w : \varphi \rightarrow \psi \quad w : \varphi}{w : \psi} \rightarrow E \quad \frac{[w : \varphi] \quad \vdots \quad w : \psi}{w : \varphi \rightarrow \psi} \rightarrow I \quad \frac{[w : \neg\varphi] \quad \vdots \quad v : \perp}{w : \varphi} \perp E \\ \\ \frac{w : \varphi \wedge \psi}{w : \varphi} \wedge E \quad \frac{w : \varphi \wedge \psi}{w : \psi} \wedge E \quad \frac{w : \varphi \quad w : \psi}{w : \varphi \wedge \psi} \wedge I \\ \\ \frac{w : \forall x \bullet \varphi}{w : \varphi[x \mapsto t]} \forall E \quad \frac{w : \varphi[x \mapsto t]}{w : \forall x \bullet \varphi} \forall I \end{array}$$

The application condition for $\forall I$ is that t may not occur in any assumption $w : \varphi[x \mapsto t]$ depends on. The rule $\perp E$ allows for a contradiction to be propagated along the reachable worlds. That is, we take \perp to be a *global* contradiction [Vig00]. A main advantage of the labelling approach is that the behaviour and intention of the modalities can be captured by rules which only use the typical mechanisms of natural deduction, i.e., application conditions and elimination of assumptions. Furthermore, the introduction and elimination rules for a box modality can be defined without any further knowledge about the properties of the reachability relation R :

$$\square E \frac{[wRv] \quad \vdots \quad w : \square\varphi}{v : \varphi} \quad \square I \frac{v : \varphi}{w : \square\varphi}$$

The application condition of the introduction rule $\Box\text{I}$ is that v is different from w and may not occur free in any assumption $v: \varphi$ depends on, except for wRv . The box elimination rule states that whenever $\Box\varphi$ holds on w , and v is reachable from w , then φ holds on v . The introduction rule states that if we can deduce the truth of φ on v , where the only assumption on v is that it is reachable from w , then we know that φ holds on all worlds reachable from w . Hence $\Box\varphi$ holds on w . This is exactly the intended semantics of a box-like modality.

We will often use the rules for the derived operators, i.e., \vee , \neg , \leftrightarrow , \exists and diamond-like modalities. To spare the reader the search of these rules in the literature, we include the rules for these operators.

$$\begin{array}{c}
 \frac{w: \neg\varphi \quad w: \varphi}{w: \perp} \neg\text{E} \qquad \frac{[w: \varphi] \quad \vdots}{\frac{w: \perp}{w: \neg\varphi}} \neg\text{I} \\
 \\
 \frac{[w: \varphi] \quad [w: \psi] \quad \vdots \quad \vdots}{w: \varphi \vee \psi \quad v: \chi \quad v: \chi} \vee\text{E} \qquad \frac{w: \varphi}{w: \varphi \vee \psi} \vee\text{I} \qquad \frac{w: \psi}{w: \varphi \vee \psi} \vee\text{I} \\
 \\
 \frac{[w: \varphi] \quad [w: \psi] \quad \vdots \quad \vdots}{w: \psi \quad w: \varphi} \leftrightarrow\text{I} \qquad \frac{w: \varphi \quad w: \varphi \leftrightarrow \psi}{w: \psi} \leftrightarrow\text{E} \qquad \frac{w: \psi \quad w: \varphi \leftrightarrow \psi}{w: \varphi} \leftrightarrow\text{E} \\
 \\
 \frac{w: \varphi[x \mapsto t]}{w: \exists x \bullet \varphi} \exists\text{I} \qquad \frac{[w: \varphi] \quad \vdots \quad \vdots}{w: \exists x \bullet \varphi \quad v: \chi} \exists\text{E} \\
 \\
 \frac{v: \varphi \quad wRv}{w: \Diamond\varphi} \Diamond\text{I} \qquad \frac{[v: \varphi] [wRv] \quad \vdots \quad \vdots}{w: \Diamond\varphi \quad u: \chi} \Diamond\text{E}
 \end{array}$$

The application condition for $\exists\text{E}$ is that x appears free neither in any assumption $v: \chi$ depends on (except for $w: \varphi$) nor in $v: \chi$ itself. Similarly, in the rule $\Diamond\text{E}$, v has to be different from both u and w and may not appear in any assumption $u: \chi$ depends on, except for wRv . Usually, the rules for the quantifiers have to assert the existence of the terms t substituted for the variable x . In this thesis, we will assume constant and infinite domains of quantification, and therefore the existence of the terms is guaranteed, allowing us to omit these additional assumptions.

LND has been transferred to interval logics by Rasmussen [Ras01; Ras02]. In his work, the reachability relation is ternary and labelled formulas are of the form $[a, b]: \varphi$, where $[a, b]$ is an interval on which φ is true. Rasmussen used a generalised interval logic called *signed interval logic* (SIL), where intervals may also have a “negative length”, i.e.,

he allows for intervals $[a, b]$, where $a > b$. He defined rules capturing, e.g., the *single decomposition property* of intervals [Dut95], and the behaviour of the chop-modalities, and achieved a sound and complete proof system.

For a well-defined proof system, Rasmussen also needed to express whether a term or formula is rigid and whether a formula does not contain a chop-modality, i.e., is *chop-free*. Since these properties are both of syntactic nature, predicates for rigidity and for chop-freeness can be straightforwardly defined. Furthermore, Rasmussen extended the proof system for SIL with suitable axioms and rules to embrace, for example, Duration Calculus or Neighbourhood logic.

2.4 Graph Rewriting

In this section we present the basic ideas of graph rewriting based on hypergraphs. Even though we present the formal definitions, we will mostly focus on the intuitive notions. However, we will give references to complete formalisations of the approaches described in this section.

The first thing needed for the definition of a graph rewriting system is a notion of graphs. In this thesis, we will use *typed hypergraphs*, a generalisation of graphs.

Definition 2.1 (Typed Hypergraph). *Let T be a set of types and \mathbb{O} a set of labels. A typed hypergraph $G = (\mathcal{V}, \mathcal{E}, \tau, \theta, \mathfrak{l})$ over T and \mathbb{O} consists of a set of vertices \mathcal{V} , hyperedges (or edges for short) \mathcal{E} , an attachment function $\tau: \mathcal{E} \rightarrow \mathcal{V}^*$, a type function $\theta: \mathcal{E} \rightarrow T$ and a labelling function $\mathfrak{l}: \mathcal{E} \rightarrow \mathbb{O}$. In contrast to edges in usual graphs, a hyperedge e may connect an arbitrary number of vertices, we say it visits the nodes via its tentacles. We use typed hypergraphs, i.e., the type $\theta(e)$ of an edge e determines the number of vertices e must be visiting. Hence for all edges e of the same type, the sequence $\tau(e)$ is of the same length. We denote the set of all typed hypergraphs by the set \mathbb{G} .*

Even though, the attachment function of a hypergraph returns sequences, we will introduce a more mnemonic notation to refer to the elements of the sequence. Since the length of the sequence for all edges of one type is equal, we will not speak of the index of a node within the sequence, but refer to it by a short string, e.g., i or at . This will be used to have a more intuitive way to describe graphs. If we refer to different graphs G and H , we will sometimes use the notation \mathcal{V}_G and \mathcal{V}_H to denote the set of vertices of the corresponding graph, and similarly for the other elements of the graphs. For the visualisations of typed hypergraphs, we use small black circles to denote the vertices and grey rectangles with rounded corners to denote the hyperedges, where the type of the edge is inscribed in the rectangle. The tentacles of the edges are given by the labelled connections between the edges and the vertices. Labels are presented as rectangles, which are connected with the edge they label by a dashed line. For the definition of graph rewriting systems, we now have to introduce *graph homomorphisms*.

Definition 2.2 (Graph Homomorphisms). *Let G and H be two typed hypergraphs. The two functions $f_{\mathcal{V}}: \mathcal{V}_G \rightarrow \mathcal{V}_H$ and $f_{\mathcal{E}}: \mathcal{E}_G \rightarrow \mathcal{E}_H$ form a graph homomorphism, if they are*

edge preserving and compatible with the attachment function, as well as the typing and labelling functions. That is, for all edges $e \in \mathcal{E}_G$ such that $\tau_G(e) = \langle v_0, \dots, v_n \rangle$ we have

$$\begin{aligned}\tau_H(f_{\mathcal{E}}(e)) &= \langle f_{\mathcal{V}}(v_0), \dots, f_{\mathcal{V}}(v_n) \rangle , \\ \theta_H(f_{\mathcal{E}}(e)) &= \theta_G(e) , \\ \mathfrak{l}_H(f_{\mathcal{E}}(e)) &= \mathfrak{l}_G(e) .\end{aligned}$$

The pair $f_{\mathcal{V}}$ and $f_{\mathcal{E}}$ will simply be abbreviated by $f: G \rightarrow H$ and by abuse of notation, we will use the name f to denote both functions. In general, we will only consider injective graph homomorphisms, i.e., both $f_{\mathcal{V}}$ and $f_{\mathcal{E}}$ have to be injective functions.

Graph rewriting systems [Roz97] are a generalisation of formal grammars. Where formal grammars replace occurrences of strings with other strings, graph rewriting systems allow for the replacement of graphs with other, possibly more complex graphs. Hence such rewriting systems allow for the creation of a language of graphs. Even though graph rewriting systems are often defined in terms of category theory, we give a definition within set theory, following the presentation of Baldan et al.[BKK03]. First, we have to define what a *graph rewriting rule* consists of.

Definition 2.3 (Rewriting Rule). A graph rewriting rule (or production) $p = \langle L, R, \alpha \rangle$ is defined by its left-hand side (LHS) L and its right-hand side (RHS) R , as well as the injective function $\alpha: \mathcal{V}_L \rightarrow \mathcal{V}_R$, which we will usually indicate by labelling corresponding vertices of L and R by natural numbers.

A rule p is applicable to a graph G , if there exists an injective graph homomorphism $m: L \rightarrow G$, a match of L in G . Given a match m of L in G , the application of p to G results in a new graph H , given by

$$\begin{aligned}\mathcal{V}_H &= \mathcal{V}_G \uplus (\mathcal{V}_R \setminus \alpha(\mathcal{V}_L)) \\ \mathcal{E}_H &= (\mathcal{E}_G \setminus m(\mathcal{E}_L)) \uplus \mathcal{E}_R\end{aligned}$$

and with the function $\bar{m}: \mathcal{V}_R \rightarrow \mathcal{V}_H$ given by $\bar{m}(v) = m(\alpha^{-1}(v))$ if $v \in \alpha(\mathcal{V}_L)$ and $\bar{m}(v) = v$ otherwise, the attachment, type and labelling function are defined as

$$\begin{aligned}e \in \mathcal{E}_G \setminus m(\mathcal{E}_L) &\Rightarrow \tau_H(e) = \tau_G(e), & \theta_H(e) &= \theta_G(e), & \mathfrak{l}_H(e) &= \mathfrak{l}_G(e) \\ e \in \mathcal{E}_R &\Rightarrow \tau_H(e) = \bar{m}(\tau_R(e)), & \theta_H(e) &= \theta_R(e), & \mathfrak{l}_H(e) &= \mathfrak{l}_R(e)\end{aligned}$$

We denote the application of a rule p to the graph G resulting in H by $G \Rightarrow_p H$.

Intuitively, the application of a rule p to a graph G consists of replacing an occurrence of a subgraph F of G that matches the LHS with the RHS, where vertices are identified by the function α .

Example 2.1. Consider the rule r shown in Fig. 2.2. The injective function of r is given by the nodes labelled 1 and 2. The labels determine, which nodes are identified in the LHS and RHS during the application of the rule. So let us consider the graph G shown in Fig. 2.3. We can apply r at three different occurrences of the LHS of r to G . This yields the three graphs shown in Fig. 2.4. Of course, r could be applied to these graphs again, until no edge of the type S occurs in the results anymore.

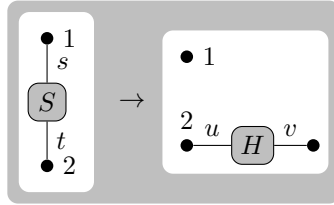
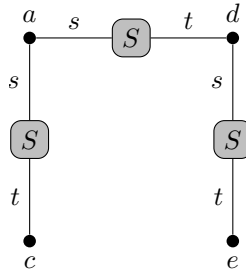


Figure 2.2: Example of a Graph Rewriting Rule

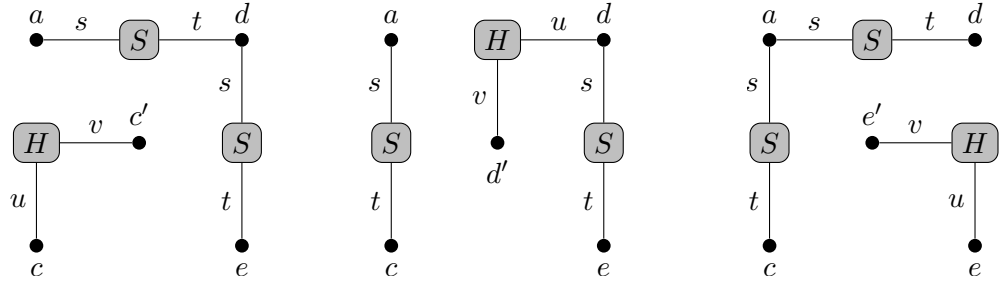
Figure 2.3: The graph G

A *graph rewriting system* (or graph transformation system) consists basically of a set of rewriting rules. In addition, it contains the *axiom*, denoting the graph where all derivations of the system have to start at.

Definition 2.4 (Graph Rewriting System). A graph rewriting system $\mathcal{G} = (T, S, P)$ consists of a set of types T , an axiom S , i.e., a typed hypergraph over T , and a set of rewriting rules P , where all graphs occurring in P are typed hypergraphs over T . If H is the result of an arbitrary sequence of applications of rules within \mathcal{G} to the graph G , we write $G \Rightarrow_{\mathcal{G}}^* H$, or simply $G \Rightarrow^* H$, if \mathcal{G} is clear from the context. We say that there exists a derivation of H from G .

Unfortunately, this approach is not expressive enough for our purposes. We need the possibility to constrain the application of rules more than just through the mere occurrence of a subgraph. For that, we employ *nested conditions* [HP09] or rather their extension *HR* conditions* [HR10; Rad13].

Intuitively, nested conditions allow not only for the statement whether a certain subgraph matching the left-hand side exists, but also for further conditions on the environment of this match. For example, it is possible to state that a certain other subgraph shall *not* exist, for a rule to be applicable. HR* conditions extend this notion by allowing for *variables* within the conditions, which have to be instantiated by graphs that are created by a hyperedge replacement system. A *hyperedge replacement system* is a graph rewriting system, where the LHS of each rule may only refer to one hyperedge at once. For example, the rule r given above is such a rule.



(a) Replacing the Left Edge (b) Replacing the Middle Edge (c) Replacing the Right Edge

 Figure 2.4: Possible Results of Applying r once to G

Definition 2.5 (Graph Substitution). *Let X be a set of edge types and \mathcal{R} a hyperedge replacement system. A graph substitution induced by \mathcal{R} is a function $\sigma: X \rightarrow \mathbb{G}$ where $x \mapsto_{\mathcal{R}}^* \sigma(x)$ for all $x \in X$. We denote the application of a substitution σ to the graph G , i.e., the simultaneous replacement of all occurrences of edges in the domain of σ within G by G^σ .*

The main syntactic element of HR^* conditions is $\exists(h, c)$, where $h: P \rightarrow C$ is a graph homomorphism and c a graph condition. For a given graph G , this notation shall describe the existence of a subgraph C^σ within G , i.e., a subgraph described by the application of a suitable graph substitution σ to C . The subgraph C^σ then has to further satisfy the condition c . This construction allows us, e.g., to express the existence of a path of arbitrary length within G . With the condition c , additional constraints on the path (or even the environment of the path) can be defined. The other crucial element of the conditions is $\exists(P \sqsupseteq C, c)$, where P and C are graphs and c is a graph condition. This notation gives us the ability to “cut” certain elements from graphs used within the conditions and only refer to the elements of C subsequently (respectively, the elements of C^σ for a substitution σ). With this construction, we can circumvent the injectivity of the underlying graph morphisms.

The full syntax of HR^* conditions is given by the following definition.

Definition 2.6 (HR^* Graph Condition). *Let \mathcal{R} be a hyperedge replacement system and G be a hypergraph possibly containing hyperedges that can be replaced by \mathcal{R} . For a hypergraph P , the set of HR^* conditions over P is given inductively as follows:*

1. \top is a HR^* condition over P .
2. Let P be a subgraph of C , where $h: P \rightarrow C$ is the inclusion of P in C and let c be a HR^* condition over C , then $\exists(h, c)$ is a HR^* condition over P .
3. Let C be a graph and c a HR^* condition over C , then $\exists(P \sqsupseteq C, c)$ is a HR^* condition over P .

4. Let c and c' be HR^* conditions over P , then both $\neg c$ and $c \wedge c'$ are HR^* conditions over P .

If the domain of the inclusion in the second case of the definition is obvious from the context, we may omit it. That is, instead of $\exists(P \rightarrow C, c)$, we would write $\exists(C, c)$.

We now define the semantics of a HR^* condition based on graph homomorphisms, to have a clearly defined notion of satisfaction. In the following definition, we use the notation c^σ for a condition c and a substitution σ , do denote the recursive application of σ to all graphs occurring within c .

Definition 2.7 (Semantics of HR^* Conditions). *Let $g: P \rightarrow G$ be a graph homomorphism. Then the satisfaction of c by g , denoted by $g \models c$ is defined as follows.*

1. g satisfies \top .
2. g satisfies $\exists(h, c)$ iff $h: P \rightarrow C$ and there is a substitution σ which replaces all variable edges of C and a graph homomorphism $q: C^\sigma \rightarrow G$ with $q \circ h^\sigma = g$ ¹ such that q satisfies c^σ .
3. g satisfies $\exists(P \supseteq C, c)$ iff there is a substitution σ such that $C^\sigma \subseteq P$ and a graph homomorphism $f: C^\sigma \rightarrow G$ satisfying c^σ such that g restricted to C^σ coincides with f , i.e., $g|_{C^\sigma} = f$.
4. g satisfies $\neg c$ iff g does not satisfy c .
5. g satisfies $c \wedge c'$ iff g satisfies c and g satisfies c' .

If g has the domain \emptyset , i.e., $g: \emptyset \rightarrow G$ and g satisfies the HR^* condition c , we also say that G satisfies c .

We define the abbreviations for the missing Boolean connectives and universal quantification as usual. Furthermore, we will omit any conditions of the form \top , e.g., instead of $\exists(C, \top)$, we only write $\exists(C)$. Observe that the substitution σ in the semantics of the conditions replaces each occurrence of a variable edge by the same graph. This will be unfortunate for our main purpose, when we want to state the existence of several, structurally different paths defined by the same hyperedge replacement system. However, Radke showed that simultaneous replacement of variables by substitutions and replacement of variables with different graphs (which are still derivable by the replacement system) are equally expressive [Rad13]². We chose to present the semantics based on substitutions, since they require less notational overhead, but will use the conditions as if we defined the semantics based on replacement.

¹ h^σ is the graph homomorphism which coincides with h for all elements of P , but has C^σ as its range. Observe that all elements of C have a unique counterpart within C^σ , and hence h^σ is uniquely determined.

²The main idea of the proof that substitution is as expressive as replacement is to create a new variable type for each occurrence of a variable within a condition. The replacement rules for these new types are then defined to be similar to the original variable type. The other direction is more involved.

If we want to formally add a HR^* condition as an application condition to a rule $r = \langle L, R, \alpha \rangle$, we use the following approach. We start with the condition $\exists(L \sqsupseteq I, c_0)$, where I is the graph induced by the restriction of α to only the nodes of L , i.e., it consists of a discrete graph. Then, within the condition c_0 , we can constrain possible connections between these nodes, e.g., claim the existence of a path. This extra step is needed, since we want to “reuse” the elements of the left-hand side within the condition, which is normally prevented by the injectivity of all homomorphisms involved.

Definition 2.8 (Application of a Rule with a HR^* condition). *Let $r = \langle L, R, \alpha \rangle$ be a rewriting rule enhanced with the HR^* condition c over L . Then r is applicable to the graph G , if there exists a match $m: L \rightarrow G$ and $m \models c$. The application of r to G is then defined as in Definition 2.3.*

Visually, we use an abbreviation. If we want to add an application condition $\exists(L \sqsupseteq I, c)$ to a rule $r = \langle L, R, \alpha \rangle$, we depict c to the left of the LHS of r and separate c from r with a white triangle.

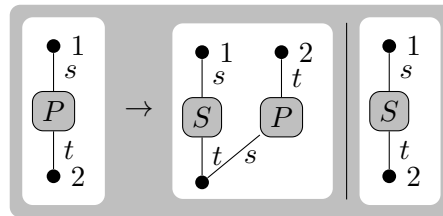


Figure 2.5: The Hyperedge Replacement System \mathcal{R}

Example 2.2. *As an example, consider the hyperedge replacement system \mathcal{R} , consisting of the rules shown in Fig. 2.5. It replaces the hyperedge P by a path of S edges of an arbitrary length greater than zero. Using \mathcal{R} , we can define a HR^* condition to restrict the applications of the rule r of Fig. 2.2. Consider the modification of r as shown in Fig. 2.6.*

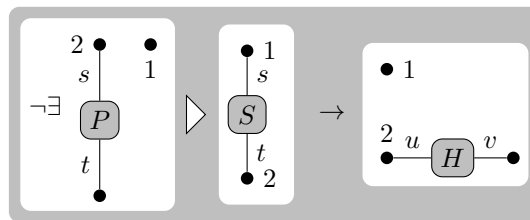


Figure 2.6: Example of a Graph Rewriting Rule with an Application Condition

The application condition states that the rule may only be applied, whenever there is no path of S edges starting at the node labelled 2. That is, the second application of r as shown in Fig. 2.3 would not be possible anymore. Observe that it prohibits the

unlabelled node and the node with the label 1 to be identified due to the injectivity of the homomorphisms involved.

If there are several rules with the same left-hand side, we use a notation similar to the extended Backus-Naur form to achieve a compact depiction of such rules (see, e.g., the rules defining the hyperedge replacement system \mathcal{R} in Fig. 2.5). Then, if a rule has more than one alternative, we use the following convention to easily refer to each of these.

Convention 2.1. *If there is more than one alternative for a left-hand side of rules, we will add superscripts 1, 2, 3, ... to the names of the rules to refer to the first, second, third, ... alternative of the right-hand sides of the rule.*

3

Spatial Model of Traffic

Contents

3.1	Abstract Road	20
3.2	Bounded Visibility	25
3.3	Sensor Models	27
3.4	Related Work	28

In this chapter we define the model we use to reason about traffic situations. Here we concentrate on a model of freeway traffic [Hil+11], i.e., we do not consider cars driving in opposing directions or intersections of streets. A main property we want to maintain in the model is its independence of the dynamics of the cars. Such a model will enable us to decompose reasoning about traffic safety into two parts. On the upper level, spatial arguments allow to show traffic safety properties, e.g., disjointness of space needed for emergency braking manoeuvres. Then, on the lower level, controllers only need to comply with the spatial constraints of the model, and safety for the overall system, i.e., the freeway, will follow. Still, to keep the model descriptive, we will give a simple type of dynamics, but will also give the required (but still very weak) restrictions on possible car dynamics. As long as the actual car dynamics adhere to the spatial constraints implied by our model, the properties proven with the techniques presented in this thesis will hold, even with more concrete and expressive dynamics.

Figure 3.1 shows an exemplary situation on a freeway. For each car, we have both indicated its *physical size* (the small polygon) as well as its *braking distance*, i.e., the distance it needs in case of an emergency braking to come to a complete standstill. We will call the sum of these the *safety envelope* of the car. Note that already this picture contains an important abstraction: car *A* has been depicted as driving on two lanes at the same time. This notion shall indicate that *A* is presently engaged in a lane-change manoeuvre. Hence we already abstracted from the concrete physical position of *A*. The

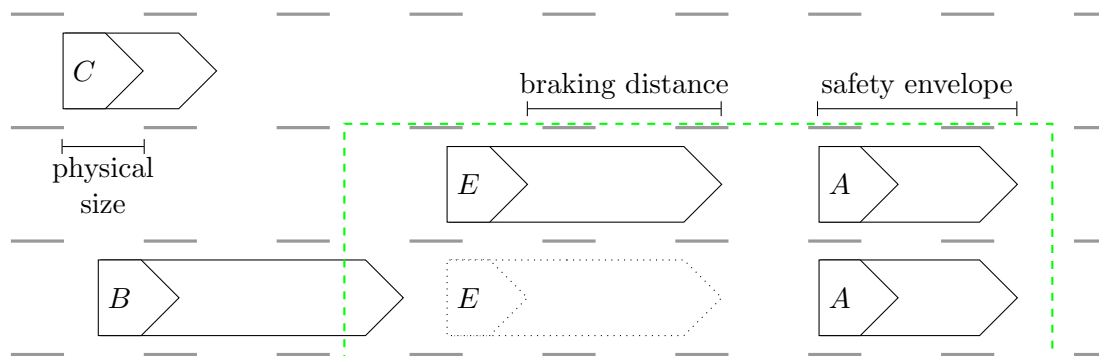


Figure 3.1: Situation on a Freeway at a Single Point in Time

dotted instance of car E denotes that E has currently set its turn signals to indicate its desire to change to the lane to its right. In the next section, we will formalise situations similar to this figure and describe the possible behaviour of cars. The dashed rectangle shall denote the finite part of space that the car E perceives at the moment, as implied by the sensors implemented in E .

The model of traffic is divided into different sections. First, we will define how all cars may behave on the street, i.e., we give an abstraction of the freeway and what actions cars may perform on this abstraction. Afterwards, we will restrict this model to the part a single car may perceive at a point in time. Then, we will make use of an abstract function Ω_E , which defines the behaviour of the sensors of a car E . Finally, we are able to define how each car perceives different cars on the freeway: simply as their physical sizes, or with additional knowledge of their braking distances.

3.1 Abstract Road

We allow for an infinite (but countable) number of cars on the street. Each car is associated with a unique identifier, which may be thought of, e.g., as its license plate. The set of such car identifiers is \mathbb{I} . We will usually denote elements of \mathbb{I} with uppercase letters, e.g., $C, D \in \mathbb{I}$. The road itself comprises an arbitrary but fixed number $N > 1$ of lanes, which are given by the set $\mathbb{L} = \{1, \dots, N\} \subset \mathbb{N}$. We will make use of addition and the total order on natural numbers, subsequently.

For simplicity, we assume each lane to be of infinite extension, so that we do not have to consider start- or endpoints of the road. Hence we take the extension of the freeway to be the set of real numbers \mathbb{R} . Throughout this thesis, we chose to use continuous time, i.e., the time domain \mathbb{T} is fixed to be $\mathbb{T} = \mathbb{R}_+$.

As shown in Fig. 3.1, we distinguish between two spatial properties for each car. First, each car *reserves* a certain amount of space on the freeway. This space is determined by the actual position on the freeway and the physical size of the car. Furthermore, depending on the model of the cars sensors, it may include its braking distance. A typical safety property would include that the reservations of all cars are disjoint during the

whole observation.

In contrast to reservations, the space *claimed* by a car may be thought of as a test, or a virtual image of the car, to check whether a lane change manoeuvre is possible. Hence a claim models that a car sets its turn signals to indicate an upcoming lane change.

In Fig. 3.1, the reservations are drawn solid, while the claim of E is given by a dotted polygon. We will use this convention throughout this thesis.

Definition 3.1 (Traffic Snapshot). *Let \mathbb{L} be a fixed, finite set of lanes and \mathbb{I} a countably infinite set of car identifiers. A structure $\mathcal{TS} = (res, clm, pos, spd, acc)$, is a traffic snapshot, where res, clm, pos, spd and acc are functions*

- $res : \mathbb{I} \rightarrow \mathcal{P}(\mathbb{L})$ such that $res(C)$ is the set of lanes the car C reserves,
- $clm : \mathbb{I} \rightarrow \mathcal{P}(\mathbb{L})$ such that $clm(C)$ is the set of lanes the car C claims,
- $pos : \mathbb{I} \rightarrow \mathbb{R}$ such that $pos(C)$ is the position of the car C along the lanes,
- $spd : \mathbb{I} \rightarrow \mathbb{R}$ such that $spd(C)$ is the current speed of the car C ,
- $acc : \mathbb{I} \rightarrow \mathbb{R}$ such that $acc(C)$ is the current acceleration of the car C .

This model of the freeway is still very broad. To make a tighter connection between real freeways and this abstract notion, we restrict the model in several ways. First, we require all cars to reserve at least one, and at most two lanes. A car reserving two lanes at once is assumed to be in the process of changing from one lane to the other. Furthermore, a car may only set its turn signals, if it is not already engaging a lane-change. Finally, a car may only try to change to a lane adjacent to its current lane. These requirements are captured in the following *sanity conditions* of traffic snapshots.

Definition 3.2 (Sanity Conditions). *A traffic snapshot \mathcal{TS} is sane, if the following conditions hold for all $C \in \mathbb{I}$.*

1. $res(C) \cap clm(C) = \emptyset$
2. $1 \leq |res(C)| \leq 2$
3. $0 \leq |clm(C)| \leq 1$
4. $1 \leq |res(C)| + |clm(C)| \leq 2$
5. $|res(C)| = 2$ implies $\exists n \in \mathbb{L} \bullet res(C) = \{n, n + 1\}$
6. $clm(C) \neq \emptyset$ implies $\exists n \in \mathbb{L} \bullet res(C) \cup clm(C) = \{n, n + 1\}$

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

Example 3.1. *We formalise Fig. 3.1 as a traffic snapshot $\mathcal{TS} = (res, clm, pos, spd, acc)$. We will only present the subsets of the functions for the cars visible in the figure. Assuming*

that the set of lanes is $\mathbb{L} = \{1, 2, 3\}$, where 1 denotes the lower lane and 3 the upper one, the functions defining the reservations and claims of \mathcal{TS} are given by

$$\begin{array}{llll} \text{res}(A) = \{1, 2\} & \text{res}(B) = \{1\} & \text{res}(C) = \{3\} & \text{res}(E) = \{2\} \\ \text{clm}(A) = \emptyset & \text{clm}(B) = \emptyset & \text{clm}(C) = \emptyset & \text{clm}(E) = \{1\} \end{array}$$

For the function pos , we chose arbitrary real values which still satisfy the relative positions of the cars in the figure. Similarly, we instantiate the function spd such that the safety envelopes of the cars could match the figure. For example, since the safety envelope of B is larger than the safety envelope of C , B has to drive with a higher velocity. For simplicity, we assume that all cars are driving with constant velocity at the moment, i.e., for all cars, the function acc returns zero.

$$\begin{array}{llll} \text{pos}(A) = 28 & \text{pos}(B) = 3.5 & \text{pos}(C) = 2 & \text{pos}(E) = 14 \\ \text{spd}(A) = 8 & \text{spd}(B) = 14 & \text{spd}(C) = 4 & \text{spd}(E) = 11 \end{array}$$

This traffic snapshot satisfies the sanity conditions.

To allow for changes of spatial situations, we have to define transitions between traffic snapshots. The possible transitions may be categorised in two different ways. First, we may distinguish *local* transitions from *global* transitions, the former describing, e.g., how a single car creates a claim, or mutates its existing claim into a reservation. The only global transition is the passing of time, in which all cars change their positions and velocities according to their dynamics.

However, the passing of time does not capture the whole of the dynamics in our setting, since we allow for instantaneous changes of accelerations, out of simplicity. The *dynamic* transitions consist of these discrete changes of accelerations and the time-passing transitions. The other types of transitions are essentially changes in the spatial configuration on the freeway, which we call *spatial* transitions.

Definition 3.3 (Transitions). *The following transitions describe the changes that may occur at a traffic snapshot $\mathcal{TS} = (\text{res}, \text{clm}, \text{pos}, \text{spd}, \text{acc})$.*

$$\begin{array}{l} \mathcal{TS} \xrightarrow{\text{c}(C,n)} \mathcal{TS}' \Leftrightarrow \begin{array}{l} \mathcal{TS}' = (\text{res}, \text{clm}', \text{pos}, \text{spd}, \text{acc}) \\ \wedge |\text{clm}(C)| = 0 \wedge |\text{res}(C)| = 1 \\ \wedge \text{res}(C) \cap \{n+1, n-1\} \neq \emptyset \\ \wedge \text{clm}' = \text{clm} \oplus \{C \mapsto \{n\}\} \end{array} \\ \mathcal{TS} \xrightarrow{\text{wd } \text{c}(C)} \mathcal{TS}' \Leftrightarrow \begin{array}{l} \mathcal{TS}' = (\text{res}, \text{clm}', \text{pos}, \text{spd}, \text{acc}) \\ \wedge \text{clm}' = \text{clm} \oplus \{C \mapsto \emptyset\} \end{array} \\ \mathcal{TS} \xrightarrow{\text{r}(C)} \mathcal{TS}' \Leftrightarrow \begin{array}{l} \mathcal{TS}' = (\text{res}', \text{clm}', \text{pos}, \text{spd}, \text{acc}) \\ \wedge \text{clm}' = \text{clm} \oplus \{C \mapsto \emptyset\} \\ \wedge \text{res}' = \text{res} \oplus \{C \mapsto \text{res}(C) \cup \text{clm}(C)\} \end{array} \end{array}$$

$$\begin{aligned}
 \mathcal{TS} \xrightarrow{\text{wd } r(C,n)} \mathcal{TS}' &\Leftrightarrow \mathcal{TS}' = (\text{res}', \text{clm}, \text{pos}, \text{spd}, \text{acc}) \\
 &\quad \wedge \text{res}' = \text{res} \oplus \{C \mapsto \{n\}\} \\
 &\quad \wedge n \in \text{res}(C) \wedge |\text{res}(C)| = 2 \\
 \mathcal{TS} \xrightarrow{t} \mathcal{TS}' &\Leftrightarrow \mathcal{TS}' = (\text{res}, \text{clm}, \text{pos}', \text{spd}', \text{acc}) \\
 &\quad \wedge \forall C \in \mathbb{I}: \text{pos}'(C) = \text{pos}(C) + \text{spd}(C) \cdot t + \frac{1}{2} \text{acc}(C) \cdot t^2 \\
 &\quad \wedge \forall C \in \mathbb{I}: \text{spd}'(C) = \text{spd}(C) + \text{acc}(C) \cdot t \\
 \mathcal{TS} \xrightarrow{\text{acc}(C,a)} \mathcal{TS}' &\Leftrightarrow \mathcal{TS}' = (\text{res}, \text{clm}, \text{pos}, \text{spd}, \text{acc}') \\
 &\quad \wedge \text{acc}' = \text{acc} \oplus \{C \mapsto a\}
 \end{aligned}$$

The spatial transitions are the following. The car C may *create a claim* on the lane n via the transition $\text{c}(C, n)$, if it does not hold a claim at the moment, and n is adjacent to its current reservation. It may furthermore *withdraw its claim* by the transition $\text{wd } \text{c}(C)$. The *creation of a reservation* $r(C)$ merges the current claim of C with its reservation and removes the claim. If the car C reserves two lanes at once, it may *withdraw its reservation* to the lane n via $\text{wd } r(C, n)$, provided n is an element of its current reservation. Observe that neither the creation of a reservation nor the withdrawal of a claim have any preconditions. Hence these transitions may occur at any time.

The dynamic transitions given above are very specific, which seems to contradict our aim to abstract from the dynamics of cars. However, these transitions are only given exemplarily, to have a defined behaviour of cars within this thesis. The results in the following chapters are independent of the concrete instantiation of the dynamics, as long as the changes of positions and velocities of cars are continuous. Interesting models of car dynamics, e.g. given by results of control theory in fact *are* continuous in this sense [Bya+09]. Usually the discrete changes allowed by the dynamics define the level of abstraction, i.e., the point, at which the dynamics no longer accommodate the physical reality.

Due to these reasons, we also combine passing of time and changes of accelerations to *evolutions*.

Definition 3.4 (Evolution). *An evolution of duration t starting in \mathcal{TS} and ending in \mathcal{TS}' is a transition sequence*

$$\mathcal{TS} = \mathcal{TS}_0 \xrightarrow{t_0} \mathcal{TS}_1 \xrightarrow{\text{acc}(C_0, a_0)} \dots \xrightarrow{t_n} \mathcal{TS}_{2n-1} \xrightarrow{\text{acc}(C_n, a_n)} \mathcal{TS}_{2n} = \mathcal{TS}',$$

where $t = \sum_{i=0}^n t_i$, $a_i \in \mathbb{R}$ and $C_i \in \mathbb{I}$ for all $0 \leq i \leq n$. We denote this evolution by

$$\mathcal{TS} \xrightarrow{t} \mathcal{TS}'.$$

We furthermore need a notion for the occurrence of arbitrary many transitions. For that, we just collect all behaviour between two different snapshots with the concept of *abstract transitions*.

Definition 3.5 (Abstract Transitions). *Let $T = \mathcal{TS}_0 \xrightarrow{\lambda_0} \dots \xrightarrow{\lambda_{n-1}} \mathcal{TS}_n$ be a transition sequence, where λ_i is an arbitrary transition label (for $0 \leq i < n$). Then $\mathcal{TS}_0 \Rightarrow \mathcal{TS}_n$ is an abstract transition.*

Example 3.2. *The following trace shows an exemplary transition sequence starting at the traffic snapshot defined in Example 3.1. At first, all cars move along their dynamics for t_1 seconds. Then the car C claims lane 2. Afterwards, t_2 seconds pass. Subsequently, C changes its claim to a reservation on lane 2 and after driving for t_{lc} seconds on both lanes (moving over), it then withdraws all reservations but the one for lane 2.*

$$\mathcal{TS} \xrightarrow{t_1} \mathcal{TS}_2 \xrightarrow{c(C,2)} \mathcal{TS}_3 \xrightarrow{t_2} \mathcal{TS}_4 \xrightarrow{r(C)} \mathcal{TS}_5 \xrightarrow{t_{lc}} \mathcal{TS}_6 \xrightarrow{\text{wd } r(C,2)} \mathcal{TS}_7$$

Furthermore, $\mathcal{TS} \Rightarrow \mathcal{TS}_7$.

However, there is no traffic snapshot \mathcal{TS}' such that $\mathcal{TS} \xrightarrow{c(A,3)} \mathcal{TS}'$, since the reservation of car A already comprises two lanes.

The transitions are well-defined in the sense, that a transition starting in a sane traffic snapshot, will again result in a sane snapshot. That is, the transitions preserve the sanity conditions of Def. 3.2.

Lemma 3.1 (Preservation of Sanity). *Let \mathcal{TS} be a sane traffic snapshot. Then, each structure \mathcal{TS}' reachable by a transition is a sane traffic snapshot.*

Proof. We proceed by a case distinction. If the transition leading from \mathcal{TS} to \mathcal{TS}' is the passing of time, or the change of an acceleration, the sanity conditions are still satisfied in \mathcal{TS}' , since they only concern the amount and place of claims and reservations.

The removal of a claim $\mathcal{TS} \xrightarrow{\text{wd } c(C)} \mathcal{TS}'$ sets $clm'(C) = \emptyset$. We distinguish two cases. If $clm(C) = \emptyset$, then $\mathcal{TS} = \mathcal{TS}'$ and hence satisfies the conditions trivially. Let $clm(C) \neq \emptyset$. After the transition, condition 1 holds trivially, condition 2 is not affected, condition 3 holds, as does condition 4. While condition 5 is not affected, condition 6 holds trivially.

Now let $\mathcal{TS} \xrightarrow{c(C,n)} \mathcal{TS}'$. Then by definition of the transition, $res(C)$ on \mathcal{TS} contains exactly one element, and $clm(C)$ is empty. On \mathcal{TS}' , $clm'(C)$ contains exactly n . Since $\{n+1, n-1\} \cap res(C) \neq \emptyset$, n cannot be an element of $res'(C)$. Hence the conditions 1 to 6 are satisfied.

Consider $\mathcal{TS} \xrightarrow{\text{wd } r(C,n)} \mathcal{TS}'$. Since $|res(C)| = 2$, condition 4 ensures that $clm(C) = \emptyset$, by which condition 1, 3, 4, 5 and 6 hold in \mathcal{TS}' . Condition 2 holds, since we overwrite $res(C)$ with $\{n\}$.

Finally, let $\mathcal{TS} \xrightarrow{r(C)} \mathcal{TS}'$. Again we have to consider two cases. First, if $clm(C) = \emptyset$, then $\mathcal{TS} = \mathcal{TS}'$, and hence the sanity conditions hold. If $clm(C) \neq \emptyset$, we get by condition 3 that $clm(C) = \{n\}$ for some $n \in \mathbb{L}$. By condition 4, $|res(C)| = 1$, and by condition 1, we get that after the transition $|res(C)| = 2$, i.e., condition 2 holds. Condition 1 and 6 hold now trivially. Condition 3 holds since we reset $clm'(C) = \emptyset$ and similarly for condition 4. Condition 5 holds, since condition 6 holds on \mathcal{TS} . \square

In the rest of this thesis, we will only consider sane traffic snapshots.

3.2 Bounded Visibility

An important assumption for our approach is that safety of a manoeuvre on the freeway should not be dependent on the behaviour of cars in too far distance. Even though such cars may indirectly influence the spatial situation near a car C , e.g., by performing an emergency braking, C should be in the position to ensure a safe execution of manoeuvres by observation of the cars in its proximity. Hence we assume that the information available to each actor of a manoeuvre is limited to a finite part of the freeway around it. This idea is formalised in the definition of *views*.

Definition 3.6 (View). *For a given traffic snapshot \mathcal{TS} with a set of lanes \mathbb{L} , a view V is defined as a structure $V = (L, X, E)$, where*

- $L = [l, n] \subseteq \mathbb{L}$ is an interval of lanes that are visible in the view,
- $X = [r, t] \subseteq \mathbb{R}$ is the extension that is visible in the view,
- $E \in \mathbb{I}$ is the identifier of the car under consideration, called owner of the view.

A subview of V is obtained by restricting the lanes and extension we observe. For this we use sub- and superscript notation: $V^{L'} = (L', X, E)$ and $V_{X'} = (L, X', E)$, where L' and X' are subintervals of L and X , respectively.

The maximal view of E should at least comprise the space needed for braking in the worst possible case, i.e., the distance needed to come to a standstill for the car with the worst brakes at maximal velocity. This distance is subsequently called *horizon* of the system. Furthermore, we assume that in such a maximal view, every car may perceive all lanes of the freeway within the horizon.

Definition 3.7 (Standard View). *For a traffic snapshot $\mathcal{TS} = (res, clm, pos, spd, acc)$ with \mathbb{L} as its set of lanes and a car $E \in \mathbb{I}$ we define the standard view of E to be*

$$V_s(E, \mathcal{TS}) = (\mathbb{L}, [pos(E) - h, pos(E) + h], E) ,$$

where the horizon h is chosen such that a car driving at maximum speed can, with lowest maximal deceleration, come to a standstill within the horizon h .

In our approach we want to emphasise local reasoning with respect to the owner of a view. If this car moves along the road, i.e., in terms of transitions, whenever time passes, we have to ensure that the view owned by this car also moves with the same speed. Formally, if a traffic snapshot \mathcal{TS}_0 evolves to \mathcal{TS}_1 in the time t , i.e. $\mathcal{TS}_0 \xrightarrow{t} \mathcal{TS}_1$, the extension X of a view $V = (L, X, E)$ has to be shifted by the difference of the positions of E in \mathcal{TS}_0 and \mathcal{TS}_1 . For this purpose, we introduce the function mv , which, given two snapshots \mathcal{TS} , \mathcal{TS}' and a view V , computes the view V' corresponding to V after moving from \mathcal{TS} to \mathcal{TS}' .

Definition 3.8 (Moving a View). *Let \mathcal{TS} and \mathcal{TS}' be two traffic snapshots with $\mathcal{TS} = (res, clm, pos, spd, acc)$ and $\mathcal{TS}' = (res', clm', pos', spd', acc')$. Furthermore, let $V = (L, [r, s], E)$ be a view. Then the result of moving V from \mathcal{TS} to \mathcal{TS}' is given by $mv_{\mathcal{TS}'}^{\mathcal{TS}}(V) = (L, [r + x, s + x], E)$, where $x = pos'(E) - pos(E)$.*

With the possibility to move a view from one snapshot to another, we abuse the notions of transitions to connect tuples of traffic snapshots and views. That is, if a transition exists between traffic snapshots \mathcal{TS} and \mathcal{TS}' , it also exists between \mathcal{TS}, V and \mathcal{TS}', V' , where V' is either equal to V if the transition is spatial, or the result of moving V to \mathcal{TS}' if the transition is an evolution. For abstract transitions, we also have to move the view to the new snapshot, since time may have passed between the traffic snapshots.

Convention 3.1 (Extended Transitions). *Let \mathcal{TS} be a traffic snapshot, V any suitable view for \mathcal{TS} and $*$ $\in \{r(C), wd\ r(C, n), c(C, n), wd\ c(C) \mid C \in \mathbb{I}, n \in \mathbb{L}\}$. Then we may also write*

$$\begin{aligned} \mathcal{TS}, V &\xrightarrow{*} \mathcal{TS}', V && \text{for } \mathcal{TS} \xrightarrow{*} \mathcal{TS}' , \\ \mathcal{TS}, V &\overset{t}{\rightsquigarrow} \mathcal{TS}', mv_{\mathcal{TS}'}^{\mathcal{TS}}(V) && \text{for } \mathcal{TS} \overset{t}{\rightsquigarrow} \mathcal{TS}' , \\ \mathcal{TS}, V &\Rightarrow \mathcal{TS}', mv_{\mathcal{TS}'}^{\mathcal{TS}}(V) && \text{for } \mathcal{TS} \Rightarrow \mathcal{TS}' . \end{aligned}$$

The model defined so far is clearly capable of capturing qualitative properties of space on the freeway. Since we are also interested in quantitative properties, e.g., the distance between two cars, we introduce two measures on the set of lanes and the extension of the views. For the extension, this notion coincides with the length measurement of Duration Calculus (DC) [ZHR91], while the measure on the lanes is simply its cardinality.

Definition 3.9 (Measures). *Let $I_R = [r, t]$ be a real-valued interval, i.e. $r, t \in \mathbb{R}$. The measure of I_R is the norm $\|I_R\| = t - r$. Similarly, the measure of a discrete interval I_D is its cardinality $|I_D|$.*

The definition of subviews induces a relation between views, which we will formalise subsequently. Since the set of lanes and the extension of a view are always discrete and real-valued intervals, respectively, we chose to employ relations as induced by other interval based logics, like Interval Temporal Logic (ITL) [Mos85] or DC [ZHR91]. That is, the view $V_1 = (L, [r, s], E)$ and $V_2 = (L, [s, t], E)$ are in horizontal relation with $V = (L, [r, t], E)$, where $s \in [r, t]$.

For vertical relations, we have to be more careful, since the set of lanes is discrete, in contrast to the extension of the views. If we would use a similar notion of vertical relations, i.e., $V_1 = ([l, m], X, E)$, $V_2 = ([m, n], X, E)$ and $V = ([l, n], X, E)$, two problems would arise. First, the lane m would be part of both subviews, which clearly contradicts the idea of separating them.¹ Furthermore, we could never achieve a view without lanes. While this does not seem to be a problem at first, it would complicate the definition of a modal logic with traffic snapshots and views as semantics (cf. Chap.4). An indication of this problem we can already describe here is that the measures on the horizontal and

¹In the case of continuous intervals, we do not mind that s is part of both subviews, since we are generally only interested in cars with a size greater than zero.

vertical dimensions would behave differently. For example, the smallest possible interval in horizontal direction is $[s, s]$ for an arbitrary $s \in \mathbb{R}$. This interval has the measure $||[s, s]|| = 0$. However, the smallest interval in vertical direction would be $[m, m]$ with $m \in \mathbb{N}$, which has the measure $||[m, m]|| = 1$. To avoid this asymmetry, we give a slightly more intricate notion of chopping discrete intervals.

Definition 3.10 (Chopping Discrete Intervals). *Let I be a discrete interval, i.e., $I = [l, n]$ for some $l, n \in \mathbb{L}$ or $I = \emptyset$. Then $I = I_1 \ominus I_2$ if and only if $I_1 \cup I_2 = I$, $I_1 \cap I_2 = \emptyset$, and either $\max(I_1) + 1 = \min(I_2)$ or $I_1 = \emptyset$ or $I_2 = \emptyset$.*

However, for a symmetric definition, we will also use a corresponding notation for continuous intervals.

Definition 3.11 (Chopping Continuous Intervals). *Let $X = [r, t]$ and $s \in [r, t]$. Furthermore, we denote $[r, s]$ by X_1 and $[s, t]$ by X_2 . Then $X = X_1 \oplus X_2$.*

We define the following relations on views to have a consistent description of reachable views in vertical and horizontal direction.

Definition 3.12 (Relations of Views). *Let V_1, V_2 and $V = (L, X, E)$ be views of a snapshot \mathcal{TS} . Then $V = V_1 \ominus V_2$ if and only if $L = L_1 \ominus L_2$, $V_1 = V^{L_1}$ and $V_2 = V^{L_2}$. Furthermore, $V = V_1 \oplus V_2$ if and only if $X = X_1 \oplus X_2$ and $V_1 = V_{X_1}$ and $V_2 = V_{X_2}$.*

Note that views with different owners may never be related to each other.

Example 3.3. *Consider again Fig. 3.1. The dashed rectangle indicates the view $V = (L, X, E)$ we want to define. Inspection of Example 3.1 shows that $L = \{1, 2\}$. For the extension, we only have to choose values such that the relations of the figure are preserved, i.e., both E and A fit fully into the extension, the safety envelope of B is partially contained in X , while no part of C overlaps with it. For example, we can choose $X = [12, 42]$, if we assume that the safety envelope of C is smaller than eight spatial units and the safety envelope of B is larger than 9.5 spatial units.*

3.3 Sensor Models

Subsequently we will use a car dependent sensor function $\Omega_E : \mathbb{I} \times \mathcal{TS} \rightarrow \mathbb{R}_+$ which, given a car identifier and a traffic snapshot, provides the length of the corresponding car, as perceived by E . The idea behind this function is to parameterise the capabilities of different sensors within the model. For example, in previous work we presented two types of obtainable knowledge of cars [Hil+11]. The simplest type is called *perfect knowledge* and assumes that the sensors return not only the physical size of a car, but in addition the length it needs for safe braking in case of an emergency. Both lengths together comprise the *safety envelope* of the car. For a more realistic scenario, it is sensible to assume that a car E knows its own safety envelope, but only the physical sizes of the other cars. Such different assumptions may be defined by instantiating the sensor function appropriately.

Furthermore, the sensor function may be thought of as a link between the abstract setting presented here and concrete controllers for different car manoeuvres. Controllers

must on the one hand return correct values according to the sensor function, e.g., while computing safety envelopes, and on the other hand may only use the type of knowledge determined by the function.

In this thesis, we will only consider the case of perfect knowledge of every car, to gain a very symmetrical setting. As shown in previous work, using an asymmetrical model of knowledge leads to the need of explicit communication between cars [Hil+11]. In the setting of perfect knowledge however, we can use the implicit communication via reservations and claims in particular, and still prove collision freedom.

The sensor function together with a view gives rise to the way the space on the freeway is perceived by the different cars. For a given view $V = (L, X, E)$ and a traffic snapshot $\mathcal{TS} = (res, clm, pos, spd, acc)$ we use the following abbreviations:

$$res_V : \mathbb{I} \rightarrow \mathcal{P}(L) \text{ with } C \mapsto res(C) \cap L \quad (3.1)$$

$$clm_V : \mathbb{I} \rightarrow \mathcal{P}(L) \text{ with } C \mapsto clm(C) \cap L \quad (3.2)$$

$$len_V : \mathbb{I} \rightarrow \mathcal{P}(X) \text{ with } C \mapsto [pos(C), pos(C) + \Omega_E(C, \mathcal{TS})] \cap X \quad (3.3)$$

The functions (3.1) and (3.2) are restrictions of their counterparts in \mathcal{TS} to the sets of lanes considered in this view. The function (3.3) gives us the part of the view occupied by a car C . Observe that using this definition without further restrictions, cars may have a point-like extension. To have a more realistic model, we require the values of the sensor function to be greater than zero for all cars and traffic snapshots.

Convention 3.2. *For all cars $C \in \mathbb{I}$, all views $V = (L, X, E)$ and all traffic snapshots \mathcal{TS} , we require $\Omega_E(C, \mathcal{TS}) > 0$.*

Example 3.4. *In this example we finish the formalisation of Fig. 3.1 by specifying a possible instantiation of the sensor function. Compare the values in this example to both Example 3.1 and 3.3.*

$$\begin{array}{ll} \Omega_E(A, \mathcal{TS}) = 11 & \Omega_E(B, \mathcal{TS}) = 11.5 \\ \Omega_E(C, \mathcal{TS}) = 7 & \Omega_E(E, \mathcal{TS}) = 13 \end{array}$$

With this sensor definition, the derived functions of \mathcal{TS} and V are as follows.

$$\begin{array}{llll} res_V(A) = \{1, 2\} & res_V(B) = \{1\} & res_V(C) = \emptyset & res_V(E) = \{2\} \\ clm_V(A) = \emptyset & clm_V(B) = \emptyset & clm_V(C) = \emptyset & clm_V(E) = \{1\} \\ len_V(A) = [28, 39] & len_V(B) = [12, 15] & len_V(C) = \emptyset & len_V(E) = [14, 27] \end{array}$$

Observe how the space occupied by B is reduced to fit into the view, and how the reservation of C is invisible for E , since the view only comprises both lower lanes.

3.4 Related Work

Different models for traffic on freeways have been proposed throughout the past. A main difference between these approaches, is whether they explicitly define space or only model

the connection structure between cars. The former approach was taken within the PATH project by Hsu et al. [Hsu+94; LGS98], where different minimal manoeuvres for cars partaking in automated traffic have been defined. The behaviour of the cars themselves is modelled by differential equations. Combining the manoeuvres in this approach allows for more sophisticated manoeuvres on the freeway, for example the construction of so-called platoons of several cars to lower the overall fuel-consumption by exploiting the slipstream of the cars in front.

Platzer defined an extension of dynamic logic allowing for the use of differential equations within programs [Pla10a]. It has been used by Platzer and Quesel to model railways [PQ09] and extended to cope with an infinite number of cars changing lanes on a multi-lane freeway [Pla10b; LPN11]. In contrast to our work, the explicit dynamics of the cars and the spatial configuration on the road are inherently interwoven. This stems from the generality of their approach, where the semantics itself is a Kripke model, in which the states are given by first-order structures. Hence all assumptions and definitions of freeway traffic have to be given by formulas of their logic. Since we consider a more restricted and specific semantics, we can omit these definitions. However, there are striking similarities to our model of freeway traffic. Cars changing lanes are modelled as using two lanes at once, for example. Furthermore, they use constant domains of the semantic first-order states, which resembles our assumption of a constant set of cars.

Banach and Butler presented a formalisation of a cruise controller, which keeps the velocity constant as long as possible [BB13] and a controller which tries to keep the car at the center of a lane [BB14]. They started with a specification given in pure Event-B [Abr10] without any hybrid elements and gradually refined events and non-deterministic behaviour, to finally achieve a verified hybrid formalisation of these controllers. In contrast to our approach, they explicitly solve the differential equations within their specifications.

Damm, Hungar and Olderog presented a method to verify safe behaviour within traffic by using a proof rule [DHO06]. The conditions for safe behaviour are divided into the premisses of the proof rule and can be verified independently. In that sense, they split the goals to be analysed, by providing a protocol the cars have to implement. This protocol requires the cars to enter certain correction modes, whenever the cars are approaching an unsafe situation. However, they base their model on hybrid automata with explicit differential equations in the modes.

The interaction between controllers for different purposes, namely keeping a given target velocity and staying within one lane, has been studied by Damm, Möhlmann and Rakow [DMR14]. They use hybrid automata for the definition of the controllers and hence rely strongly on concrete differential equations for the behaviour. Furthermore, their approach is not directly defined to cope with lane changes, even though the protocol can be extended to gradually change the lane instead of stabilising to the mid of the current lane.

None of these models of car traffic distinguish between what we call reservations and the communication of the intention to change the lane, i.e., a claim. That is, they treat a lane-change as a manoeuvre of a single car, who does not need to announce its actions. In contrast, in our model all cars able to perceive a claim can take this information into

account.

Toben [Tob08] has used the “spotlight principle” [WW07] to verify distributed systems with a dynamically changing link structure. His running example is based on car platoons. The spotlight abstraction restricts the set of agents under consideration to a finite number. Only these distinguished cars are kept explicitly, while the cars outside this “spotlight” are abstracted to a process with arbitrary behaviour. Our restriction of the road to a finite view similarly restricts the set of agents visible to a single distinguished car. However, we do not directly get that the number of cars in the view is finite. Furthermore, cars outside of a view are not considered at all, since we assume that they are currently not able to communicate with the owner of this view.

The presented model can be extended to take oncoming traffic into account [HLO13]. The main difference is that the set of cars has to be divided into disjoint subsets \mathbb{I}_{\rightarrow} and \mathbb{I}_{\leftarrow} to refer to the direction a car is driving into. Then, most of the definitions in this chapter can be used without changes, except for the calculation of the perceived length len_V of a car. There the different directions cause for a slightly altered definition.

A different extension is to get rid of the concrete dynamics introduced in Definition 3.3. We could introduce a new function called, e.g., $dyn: \mathbb{I} \rightarrow (\mathbb{T} \rightarrow \mathbb{R})$ which returns for a given car its continuous dynamical behaviour. While time passes, all cars would evolve according to the functions returned by dyn . Instead of changing the acceleration of a car C , dyn would be updated to reflect the mode change within C . For a more concise presentation, we refrained from giving this even more abstract definition. However, the results in this thesis are not affected by this change.

4

Modal Logic for Freeway Traffic

Contents

4.1	Syntax and Semantics	31
4.2	Proof System	36
4.2.1	Relational Properties	40
4.2.2	Rigidity	42
4.2.3	First-Order Logic	43
4.2.4	Chopping, Decomposition and Transitions	44
4.2.5	Specifying the Domains	45
4.2.6	Spatial Properties of Spatial Atoms	46
4.2.7	Spatial Atoms and Transitions	46
4.2.8	Invariance	52
4.2.9	Derived Rules	54
4.3	Undecidability of Spatial MLSL	59
4.4	Related Work	65

In this chapter, we present the *extended multi-lane spatial logic* (EMLSL) which is explicitly defined to deal with traffic situations described by the model in the previous chapter. That is, the main focus of reasoning lies on the spatial aspects of traffic, while most of the dynamics is hidden within the semantics.

This chapter is based on previous work [LH13], but contains a more structured presentation of the proof system, as well as a more thorough and complete discussion.

4.1 Syntax and Semantics

We employ three sorts of variables. The set of variables ranging over car identifiers is denoted by $CVar$, with typical elements c and d . For referring to lengths and quantities

of lanes, we use the sorts RVar and LVar ranging positive over real numbers and natural numbers, respectively. The set of all variables is denoted by Var. To refer to the car owning the current view, we use the special constant ego. Furthermore, we use the syntax ℓ for the length of a view, i.e., the length of the extension of the view and ω for the width, i.e., the number of lanes. For simplicity, we only allow for addition between correctly sorted terms. However, it is straightforward to augment the definition with further arithmetic operations.

Definition 4.1 (Terms of EMLSL). *We use the following definition of terms.*

$$\theta ::= n \mid r \mid \text{ego} \mid u \mid \ell \mid \omega \mid \theta_1 + \theta_2,$$

where $n \in \mathbb{N}$, $r \in \mathbb{R}_+$, $u \in \text{Var}$ and θ_i are both of the same sort, and not elements of $\text{CVar} \cup \{\text{ego}\}$. We denote the set of terms with Θ .

In addition to the atoms \perp denoting a contradiction, $=$ for equality and \leq for the order between elements of the dimensions, we use two spatial atoms $re(c)$ and $cl(c)$, which shall be true, iff the current view consists of one single lane which is completely filled with the reservation of the car denoted by c (or its claim, respectively). To reason about views with more lanes and different topological relations between cars, we can *chop* views either horizontally with the binary modality \frown , or vertically which is denoted by stacking formulas on top of each other. Furthermore, we use unary universal modalities for all of the possible spatial transitions between traffic snapshots and for evolutions. The modality for evolutions is metric, i.e., it is possible to constrain the length of the evolutions by the interval the modality is annotated with. Observe that the spatial modalities use a car variable in their subscript. This variable will be evaluated like other variables in the formulas. I.e., the modalities are parameterised by these variables. The modality \mathbf{G} is a universal modality with respect to abstract transitions, i.e., it can be used to define invariance properties. Finally, EMLSL is closed under all first-order operators.

Definition 4.2 (Syntax of EMLSL). *The syntax of formulas of the extended multi-lane spatial logic EMLSL is given as follows.*

$$\phi ::= \perp \mid \theta_1 = \theta_2 \mid \theta_1 \leq \theta_2 \mid re(c) \mid cl(c) \mid \phi_1 \rightarrow \phi_2 \mid \forall z \bullet \phi_1 \mid \phi_1 \frown \phi_2 \mid \begin{array}{c} \phi_2 \\ \phi_1 \end{array} \mid M\phi$$

where $M \in \{\Box_{r(c)}, \Box_{c(c)}, \Box_{wd\ c(c)}, \Box_{wd\ r(c)}, \Box_I, \mathbf{G}\}$, $I \in \mathcal{I}$, $c \in \text{CVar} \cup \{\text{ego}\}$, $z \in \text{Var}$, and $\theta_1, \theta_2 \in \Theta$ are of the same sort. For the atom $\theta_1 \leq \theta_2$, we also require that θ_i are not elements of $\text{CVar} \cup \{\text{ego}\}$. We denote the set of all EMLSL formulas by Φ .

Definition 4.3 (Valuation and Modification). *A valuation is a function $\nu: \text{Var} \cup \{\text{ego}\} \rightarrow \mathbb{I} \cup \mathbb{R}_+ \cup \mathbb{N}$. The function $\nu \oplus \{x \mapsto \alpha\}$ is a modification of ν , which coincides with ν except possibly for x . We silently assume valuations and their modifications to respect the sorts of variables. For a view $V = (L, X, E)$, we lift ν to a function ν_V evaluating terms, where variables and ego are interpreted as in ν , and $\nu_V(\ell) = \|X\|$ and $\nu_V(\omega) = |L|$. The function $+$ is interpreted as addition.*

Definition 4.4 (Semantics). *In the following, let θ_i be terms of the same sort, $I \in \mathcal{I}$, $c \in \text{CVar} \cup \{\text{ego}\}$ and $z \in \text{Var}$. The satisfaction of formulas with respect to a traffic snapshot \mathcal{TS} , a view $V = (L, X, E)$ and a valuation ν with $\nu(\text{ego}) = E$ is defined inductively as follows:*

$$\begin{aligned}
 \mathcal{TS}, V, \nu \not\models \perp & \quad \text{for all } \mathcal{TS}, V, \nu \\
 \mathcal{TS}, V, \nu \models \theta_1 = \theta_2 & \Leftrightarrow \nu_V(\theta_1) = \nu_V(\theta_2) \\
 \mathcal{TS}, V, \nu \models \theta_1 \leq \theta_2 & \Leftrightarrow \nu_V(\theta_1) \leq \nu_V(\theta_2) \\
 \mathcal{TS}, V, \nu \models \text{re}(c) & \Leftrightarrow |L| = 1 \text{ and } \|X\| > 0 \text{ and} \\
 & \quad \text{res}_V(\nu(c)) = L \text{ and } X = \text{len}_V(\nu(c)) \\
 \mathcal{TS}, V, \nu \models \text{cl}(c) & \Leftrightarrow |L| = 1 \text{ and } \|X\| > 0 \text{ and} \\
 & \quad \text{clm}_V(\nu(c)) = L \text{ and } X = \text{len}_V(\nu(c)) \\
 \mathcal{TS}, V, \nu \models \phi_1 \rightarrow \phi_2 & \Leftrightarrow \mathcal{TS}, V, \nu \models \phi_1 \text{ implies } \mathcal{TS}, V, \nu \models \phi_2 \\
 \mathcal{TS}, V, \nu \models \forall z \bullet \phi & \Leftrightarrow \forall \alpha \in \mathbb{I} \cup \mathbb{R}_+ \cup \mathbb{N} \bullet \mathcal{TS}, V, \nu \oplus \{z \mapsto \alpha\} \models \phi \\
 \mathcal{TS}, V, \nu \models \phi_1 \wedge \phi_2 & \Leftrightarrow \exists V_1, V_2 \bullet V = V_1 \oplus V_2 \text{ and} \\
 & \quad \mathcal{TS}, V_1, \nu \models \phi_1 \text{ and } \mathcal{TS}, V_2, \nu \models \phi_2 \\
 \mathcal{TS}, V, \nu \models \begin{matrix} \phi_2 \\ \phi_1 \end{matrix} & \Leftrightarrow \exists V_1, V_2 \bullet V = V_1 \ominus V_2 \text{ and} \\
 & \quad \mathcal{TS}, V_1, \nu \models \phi_1 \text{ and } \mathcal{TS}, V_2, \nu \models \phi_2 \\
 \mathcal{TS}, V, \nu \models \Box_{r(c)} \phi & \Leftrightarrow \forall \mathcal{TS}' \bullet \mathcal{TS} \xrightarrow{r(\nu(c))} \mathcal{TS}' \text{ implies } \mathcal{TS}', V, \nu \models \phi \\
 \mathcal{TS}, V, \nu \models \Box_{c(c)} \phi & \Leftrightarrow \forall \mathcal{TS}', n \bullet \mathcal{TS} \xrightarrow{c(\nu(c), n)} \mathcal{TS}' \text{ implies } \mathcal{TS}', V, \nu \models \phi \\
 \mathcal{TS}, V, \nu \models \Box_{\text{wd } c(c)} \phi & \Leftrightarrow \forall \mathcal{TS}' \bullet \mathcal{TS} \xrightarrow{\text{wd } c(\nu(c))} \mathcal{TS}' \text{ implies } \mathcal{TS}', V, \nu \models \phi \\
 \mathcal{TS}, V, \nu \models \Box_{\text{wd } r(c)} \phi & \Leftrightarrow \forall \mathcal{TS}', n \bullet \mathcal{TS} \xrightarrow{\text{wd } r(\nu(c), n)} \mathcal{TS}' \text{ implies } \mathcal{TS}', V, \nu \models \phi \\
 \mathcal{TS}, V, \nu \models \Box_I \phi & \Leftrightarrow \forall \mathcal{TS}', t \bullet t \in I \wedge \mathcal{TS} \xrightarrow{t} \mathcal{TS}' \text{ implies } \mathcal{TS}', \text{mv}_{\mathcal{TS}'}^{\mathcal{TS}'}(V), \nu \models \phi \\
 \mathcal{TS}, V, \nu \models \mathbf{G} \phi & \Leftrightarrow \forall \mathcal{TS}' \bullet \mathcal{TS} \Rightarrow \mathcal{TS}' \text{ implies } \mathcal{TS}', \text{mv}_{\mathcal{TS}'}^{\mathcal{TS}'}(V), \nu \models \phi
 \end{aligned}$$

In addition to the standard abbreviations of the remaining Boolean operators and the existential quantifier, we use $\top \equiv \neg \perp$. Furthermore, we introduce a set of derived modalities and abbreviations in the following convention.

Convention 4.1 (Abbreviations). *An important derived modality of our previous work [Hil+11] is the somewhere modality*

$$\langle \phi \rangle \equiv \top \frown \left(\begin{matrix} \top \\ \phi \\ \top \end{matrix} \right) \frown \top.$$

Further, we use its dual operator everywhere. We abbreviate the modality somewhere along the extension of the view with the operator \Diamond_ℓ , similar to the on some subinterval

modality of DC. For the metric modality, we allow for two simplifications, if the interval I is a singleton set and if the bounds are not relevant. We also introduce the dual operator to the invariance modality.

$$\begin{aligned} [\phi] &\equiv \neg \langle \neg \phi \rangle & \diamond_{\ell} \phi &\equiv \top \wedge \phi \wedge \top & \square_{\ell} \phi &\equiv \neg \diamond_{\ell} \neg \phi \\ \square_{\tau} \phi &\equiv \square_{[0, \infty)} \phi & \square_x \phi &\equiv \square_{[x, x]} \phi & \mathbf{F} \phi &\equiv \neg \mathbf{G} \neg \phi \end{aligned}$$

Likewise, abbreviations can be defined to express the modality on some lane. Furthermore, we define the diamond modalities for the transitions as usual, i.e., $\diamond_* \phi \equiv \neg \square_* \neg \phi$, where $*$ $\in \{r(c), c(c), \text{wd } r(c), \text{wd } c(c), I\}$.

Example 4.1. We first present some examples φ_i for EMLSL formulas.

$$\begin{aligned} \varphi_1 &\equiv \ell = x \\ \varphi_2 &\equiv \langle cl(\text{ego}) \rangle \\ \varphi_3 &\equiv \langle re(c) \rangle \\ \varphi_4 &\equiv \square_{r(\text{ego})} \left\langle \begin{array}{l} re(\text{ego}) \\ re(\text{ego}) \end{array} \right\rangle \\ \varphi_5 &\equiv \diamond_{c(a)} \top \\ \varphi_6 &\equiv \langle \square_{c(b)} cl(b) \rangle \end{aligned}$$

For evaluating their semantics, we recall the traffic snapshot \mathcal{TS} , the view V and the sensor function defined in Example 3.1 to 3.4, i.e., the formalisation of Fig. 3.1. We only repeat the values of the view and of the derived functions res_V , clm_V and len_V . The view is given by $V = (\{1, 2\}, [12, 42], E)$ and the corresponding restrictions of \mathcal{TS} are as follows:

$$\begin{aligned} res_V(A) &= \{1, 2\} , & res_V(B) &= \{1\} , & res_V(C) &= \emptyset , & res_V(E) &= \{2\} , \\ clm_V(A) &= \emptyset , & clm_V(B) &= \emptyset , & clm_V(C) &= \emptyset , & clm_V(E) &= \{1\} , \\ len_V(A) &= [28, 39] , & len_V(B) &= [12, 15] , & len_V(C) &= \emptyset , & len_V(E) &= [14, 27] . \end{aligned}$$

Let ν be defined by $\nu(x) = 30$, $\nu(a) = A$, $\nu(b) = B$, $\nu(c) = C$ and $\nu(\text{ego}) = E$. Then the following relations hold:

$$\begin{aligned} \mathcal{TS}, V, \nu &\models \varphi_1 \\ \mathcal{TS}, V, \nu &\models \varphi_2 \\ \mathcal{TS}, V, \nu &\not\models \varphi_3 \\ \mathcal{TS}, V, \nu &\models \varphi_4 \\ \mathcal{TS}, V, \nu &\not\models \varphi_5 \\ \mathcal{TS}, V, \nu &\models \varphi_6 \end{aligned}$$

The first formula is true, since the length of V is exactly 30 spatial units. The formula φ_2 is true, since we can find the subview $V_2 = (L_2, X_2, E)$ with $L_2 = \{1\}$ and $X_2 = [14, 27]$,

for which $cl_{V_2}(E) = L_2$ and $len_{V_2}(E) = X_2$. The reasoning why \mathcal{TS} , V and ν do not satisfy φ_3 is similar. For φ_4 , observe that there is only one transition for E mutating its claim to a reservation. After this transition, $res_V(E) = \{1, 2\}$ holds. Hence, there is a subview of V , such that φ_4 is satisfied after all transitions with the label $r(E)$. The formula φ_5 is not satisfied by the given model, since there is no transition, where A may create a new claim (as explained in Example 3.2). Finally, φ_6 is satisfied, because when B creates a claim, the only lane the claim can be created on is lane 2. Hence, we can find a subview V_6 , which consists of $L_6 = \{2\}$ and $X_6 = [12, 15]$. This view and the snapshot emerging from the transition satisfy φ_6 . Observe that the presented extensions for the satisfaction of φ_2 and φ_6 are the maximal extensions possible. We could also have chosen a subinterval of these extensions with a length greater than zero, and still have a satisfying model.

In the first definition of MLSL, we included the atom *free* to denote free space on the road, i.e., space which is neither occupied by a reservation nor by a claim. It was not possible to derive this atom from the others, since we were unable to express the existence of exactly one lane and a non-zero extension in the view. However, in the current presentation, *free* can be defined within EMLSL.

$$free \equiv \ell > 0 \wedge \omega = 1 \wedge \forall c \bullet \Box_\ell \neg (cl(c) \vee re(c))$$

Furthermore, we can define $\ell < r \equiv \neg(\ell = r \wedge \top)$ and use the superscript ϕ^r to abbreviate the schema $\phi \wedge \ell = r$. For reasons of clarity, we will not always use this abbreviation and write out the formula instead, to emphasise the restriction.

As an example, the following formula defines the behaviour of a distance controller, i.e., as long as the car starts in a situation with free space in front of it, the formula demands that after an arbitrary time, there is still free space left.

$$\forall x, y \bullet \Diamond_\ell \left(\begin{array}{c} \omega = x \\ re(\text{ego}) \wedge free \\ \omega = y \end{array} \right) \rightarrow \Box_\tau \left(\Diamond_\ell \left(\begin{array}{c} \omega = x \\ re(\text{ego}) \wedge free \\ \omega = y \end{array} \right) \right)$$

We have to relate the lane in both the antecedent and the conclusion by the atoms $\omega = x$ and $\omega = y$ respectively. If we simply used $\langle re(\text{ego}) \wedge free \rangle$, it would be possible for the reservations to be on different lanes, and hence, we would not ensure that free space is in front of each of ego's reservations at every point in time. However, the formula does not constrain how the situations may change, whenever reservations or claims are created or withdrawn.

Observe that it is crucial to combine acceleration and time transitions into a single modality \Box_I . Let ego drive on lane m with a velocity of v . If we only allowed for the passing of time without any changes of accelerations, this formula would require all cars on m in front of ego to have a velocity $v_f \geq v$, while all cars behind ego had to drive with $v_b \leq v$. Hence the evolutions allow for more complex behaviour in the underlying model.

Like for ITL [Mos85] or DC [ZHR91], we call a term or formula *flexible* whenever its satisfaction is dependent on the current traffic snapshot and view. Otherwise the formula

is *rigid*. However, since the spatial dimensions of EMLSL are not directly interrelated, we also distinguish *horizontally rigid* and *vertically rigid* formulas. The satisfaction of the former is independent of the extension of views, while for the latter, the amount of lanes in a view is of no influence. If a formula is only independent of the current traffic snapshot, we call it *dynamically rigid*.

Definition 4.5 (Types of Rigidity). *Let ϕ be a formula of EMLSL. We call ϕ dynamically rigid, if it does not contain any spatial atom, i.e., $re(c)$ or $cl(c)$ as a subformula. Furthermore, we call ϕ horizontally rigid, if it is dynamically rigid and in addition does not contain ℓ as a term. Similarly, ϕ is vertically rigid, if it is dynamically rigid and does not contain ω as a term. If ϕ is both vertically and horizontally rigid, it is simply rigid.*

Example 4.2. *Each equality constraint between variables $c = d$ is a rigid formula. In contrast, $\ell = x$ is only vertically rigid, and $\omega = y$ is only horizontally rigid. Since both of these formulas are dynamically rigid, so is $\ell = x \wedge \omega = y$. The formula $\langle re(c) \rangle$ is not rigid in any way.*

Lemma 4.1. Let ϕ , ϕ_H and ϕ_V be formulas of EMLSL, such that ϕ is dynamically rigid, ϕ_H is horizontally rigid and ϕ_V is vertically rigid. Then for all traffic snapshots \mathcal{TS} , \mathcal{TS}' , views V , V_1 , V_2 and valuations ν ,

1. $\mathcal{TS}, V, \nu \models \phi$ iff $\mathcal{TS}', V, \nu \models \phi$
2. Let $V = V_1 \oplus V_2$. Then $\mathcal{TS}, V, \nu \models \phi_H$ iff $\mathcal{TS}, V_i, \nu \models \phi_H$ (for $i \in \{1, 2\}$).
3. Let $V = V_1 \ominus V_2$. Then $\mathcal{TS}, V, \nu \models \phi_V$ iff $\mathcal{TS}, V_i, \nu \models \phi_V$ (for $i \in \{1, 2\}$).

Proof. By induction on the structure of EMLSL formulas. □

4.2 Proof System

In this section, we define a system of labelled natural deduction [Gab96; BMV98; Vig00] for the full logic EMLSL. That is, the rules of the deduction system do not operate on formulas ϕ , but on *labelled formulas* $w : \phi$, where w is a term of a *labelling algebra* and ϕ is a formula of EMLSL. They may connect the derivations of formulas and relations between the terms w to allow for a tighter relationship between both. The labelling algebra is more involved than for standard modal logics, since EMLSL is in essence a multi-dimensional logic, where the modalities are not interdefinable. Obviously, the spatial modalities cannot be defined by the dynamic modalities and vice versa. Furthermore, neither can the dynamic modalities be defined by each other in general. Consider, e.g., the modalities $\Box_{r(c)}$ and $\Box_{c(c)}$. Both of these modalities rely on different transitions between the models, which are only indirectly related.

The labels of the algebra consist of tuples ts, v . Similarly to the semantics, ts is the name of a traffic snapshot and v the name of a view. The algebra contains three different kinds of relations. The relations of the form $v = v_1 \oplus v_2$ and $v = v_1 \ominus v_2$ define ternary reachability relations between views for the spatial modalities. Relations between

whole labels, e.g., $ts, v \xrightarrow{r(c)} ts', v'$ describe the behaviour of transitions. The relations within the labelling algebra for traffic snapshots directly correspond to the dynamic modalities. For example, we have $ts, v \xrightarrow{c(c)} ts', v'$, whenever there exists an $n \in \mathbb{N}$ such that $\mathcal{TS} \xrightarrow{c(\nu(C), n)} \mathcal{TS}'$, where v and v' denote the same view and ts (ts') denotes the snapshot \mathcal{TS} (\mathcal{TS}' , respectively). Finally, since we have transitions describing the passing of time, we need a new type of variables called *timing variables* t . They may only be used in formulas related to the passing of time, i.e., $ts, v \xrightarrow{t} ts', v'$, and in inequalities of the form $t_1 \trianglelefteq t_2$.

We do not give a deduction system for the transitions between snapshots, since the conditions needed to hold between them are of a very complex nature, i.e., they are definable only with the power of full first-order logic with functions, identity and arithmetic. Hence we would not achieve a system with a nice distinction between the relational deductions and the deductions of labelled formulas [BMV98; Vig00]. Instead, we simply assume the existence of the relations between snapshots whenever needed. That is, we will often have, e.g., the existence of a transition in our set of assumptions. However, we will give rules to describe the behaviour of views and snapshots under spatial transitions. That is, we can ensure that the views themselves do not change under such transitions. This is due to the fact that the only possibility for a view to change is during evolutions and abstract transitions. Only for them the view is moved according to the differences between the source and target snapshot of the transition. For views, we have a more sophisticated set of rules. We introduce a new quantifier \mathbb{E} , which ranges over views. We use it to relate views and define the possibilities to chop views both horizontally and vertically.

Subsequently, we assume that we have countably infinite sets of *names for traffic snapshots* \mathfrak{TS} and *names for views* \mathfrak{V} .

Definition 4.6 (Labelled Formulas and Relational Formulas). *Let ts be a name for a traffic snapshot, v a name for a view and ϕ a formula according to Definition 4.2. Then $ts, v : \phi$ is a labelled formula of EMLSL. Locality formulas are of the form*

$$\rho ::= v = v_1 \oplus v_2 \mid v = v_1 \ominus v_2 \mid \mathbb{E}v \bullet \rho$$

where $v, v_1, v_2 \in \mathfrak{V}$. Furthermore, we call $ts, v \xrightarrow{*} ts', v'$ dynamic formulas, where $ts, ts' \in \mathfrak{TS}$, $v, v' \in \mathfrak{V}$ and $\xrightarrow{*}$ is a relation of the labelling algebra. Finally, formulas of the form $t_1 \trianglelefteq t_2$, are called timing formulas, where t_1 and t_2 are timing terms, i.e., either elements of \mathbb{T} , timing variables or a sum of timing terms. If we do not need the distinction between dynamic and locality formulas, we simply write relational formulas.

Observe that we use the same notation for the syntactic constructs of locality formulas, as well as the chopping operation as their semantic counterpart. In general, the distinction should be clear from the context.

To have a meaningful soundness result of the calculus, we give the relation of the semantics of labelled formulas and normal formulas. For that, we have to relate the names of views and traffic snapshots with their semantic counterparts. That is, we need valuations for both types of names and extend the valuations of variables to timing terms.

Definition 4.7 (Snapshot and Locality Valuation). *A function $\sigma: \mathfrak{TS} \rightarrow \mathbb{T}\mathfrak{S}$ relating names of traffic snapshots with traffic snapshots themselves is called a snapshot valuation. Similarly, a function $\lambda: \mathfrak{V} \rightarrow \mathbb{V}$ is called a locality valuation. Modifications of both snapshot and locality valuations are defined similarly to modifications of normal valuations. Furthermore, we extend valuations ν to also relate timing variables t to elements of \mathbb{T} . We silently assume that $\nu(t) = t$ if t is a constant element of \mathbb{T} and lift the valuations to evaluate timing terms.*

The satisfaction of labelled formulas and relational formulas is defined with respect to a snapshot valuation, a locality valuation and a valuation for variables. The first two types of valuations map the labels of formulas to a model of EMLSL, such that the labelled formula can be evaluated by the satisfaction relation of EMLSL. For the satisfaction of dynamic formulas, recall that we lifted the transitions within the model to relate tuples of traffic snapshots and views in Convention 3.1.

Definition 4.8 (Satisfaction of Labelled Formulas and Relational Formulas). *A valuation ν , a snapshot valuation σ and a locality valuation λ satisfy a labelled formula $ts, v: \phi$, written $\sigma, \lambda, \nu \models ts, v: \phi$ if and only if $\sigma(ts), \lambda(v), \nu \models \phi$. Furthermore,*

$$\begin{aligned}
 \sigma, \lambda, \nu \models ts_1, v_1 \xrightarrow{r(c)} ts_2, v_2 &\Leftrightarrow \sigma(ts_1), \lambda(v_1) \xrightarrow{r(\nu(c))} \sigma(ts_2), \lambda(v_2), \\
 \sigma, \lambda, \nu \models ts_1, v_1 \xrightarrow{\text{wd } r(c)} ts_2, v_2 &\Leftrightarrow \exists n \bullet \sigma(ts_1), \lambda(v_1) \xrightarrow{\text{wd } r(\nu(c), n)} \sigma(ts_2), \lambda(v_2), \\
 \sigma, \lambda, \nu \models ts_1, v_1 \xrightarrow{c(c)} ts_2, v_2 &\Leftrightarrow \exists n \bullet \sigma(ts_1), \lambda(v_1) \xrightarrow{c(\nu(c), n)} \sigma(ts_2), \lambda(v_2) \\
 \sigma, \lambda, \nu \models ts_1, v_1 \xrightarrow{\text{wd } c(c)} ts_2, v_2 &\Leftrightarrow \sigma(ts_1), \lambda(v_1) \xrightarrow{\text{wd } c(\nu(c))} \sigma(ts_2), \lambda(v_2) \\
 \sigma, \lambda, \nu \models ts_1, v_1 \xrightarrow{t} ts_2, v_2 &\Leftrightarrow \sigma(ts_1), \lambda(v_1) \xrightarrow{\nu(t)} \sigma(ts_2), \lambda(v_2) \\
 \sigma, \lambda, \nu \models ts_1, v_1 \Rightarrow ts_2, v_2 &\Leftrightarrow \sigma(ts_1), \lambda(v_1) \Rightarrow \sigma(ts_2), \lambda(v_2) \\
 \sigma, \lambda, \nu \models t_1 \leq t_2 &\Leftrightarrow \nu(t_1) \leq \nu(t_2) \\
 \sigma, \lambda, \nu \models v = v_1 \oplus v_2 &\Leftrightarrow \lambda(v) = \lambda(v_1) \oplus \lambda(v_2) \\
 \sigma, \lambda, \nu \models v = v_1 \ominus v_2 &\Leftrightarrow \lambda(v) = \lambda(v_1) \ominus \lambda(v_2) \\
 \sigma, \lambda, \nu \models \mathbb{E}v \bullet \rho &\Leftrightarrow \exists V \bullet \sigma, \lambda \oplus \{v \mapsto V\}, \nu \models \rho
 \end{aligned}$$

where ρ is a locality formula. We lift the satisfaction relation to sets of formulas. Let Γ and Δ be a set of labelled formulas and of relational formulas, respectively. Then

$$\begin{aligned}
 \sigma, \lambda, \nu \models \Delta &\Leftrightarrow \forall \eta \in \Delta \bullet \sigma, \lambda, \nu \models \eta \\
 \sigma, \lambda, \nu \models \Gamma &\Leftrightarrow \forall (ts, v: \phi) \in \Gamma \bullet \sigma, \lambda, \nu \models ts, v: \phi \\
 \sigma, \lambda, \nu \models (\Gamma, \Delta) &\Leftrightarrow \sigma, \lambda, \nu \models \Gamma \text{ and } \sigma, \lambda, \nu \models \Delta \\
 (\Gamma, \Delta) \models ts, v: \phi &\Leftrightarrow \sigma, \lambda, \nu \models (\Gamma, \Delta) \text{ implies } \sigma, \lambda, \nu \models ts, v: \phi \\
 &\text{for all } \sigma, \lambda, \nu \\
 (\Gamma, \Delta) \models \eta &\Leftrightarrow \sigma, \lambda, \nu \models (\Gamma, \Delta) \text{ implies } \sigma, \lambda, \nu \models \eta \\
 &\text{for all } \sigma, \lambda, \nu,
 \end{aligned}$$

where η is a relational formula.

Definition 4.9 (Derivation). A derivation of a labelled formula $ts, v: \phi$ from a set of labelled formulas Γ and a set of relational formulas Δ is a tree, where the root is $ts, v: \phi$, each leaf is an element of Γ or Δ , and each node within the tree is a result of an application of one of the rules defined subsequently. We denote the existence of such a derivation by $(\Gamma, \Delta) \vdash ts, v: \phi$.

Following Rasmussen [Ras01], we define predicates for chop-freeness of formulas and rigidity of terms and formulas. To increase the number of deducible theorems, we differentiate between *vertical* and *horizontal* chop-freeness and rigidity, as well as *dynamic* rigidity. These properties are especially important for the correct instantiation of terms, i.e., for the elimination of universal quantifiers.

Example 4.3. Consider the formula

$$\forall x \bullet \left(\begin{array}{l} \ell = x \\ \ell = x \end{array} \rightarrow \ell = x \right),$$

which is a theorem of EMLSL, since the length of a view is not changed by chopping vertically. If we use classical universal quantifier instantiation and substitute the vertically flexible term ω for x , then we would get

$$\begin{array}{l} \ell = \omega \\ \ell = \omega \end{array} \rightarrow \ell = \omega. \quad (4.1)$$

Now let \mathcal{TS} be a traffic snapshot, V a view and ν a valuation satisfying the antecedent of (4.1). Then V can be vertically chopped into V_1 and V_2 such that its length equals its width on both V_1 and V_2 . Now let $\nu_V(\ell) = c$. Then also $\nu_{V_1}(\omega) = c$ and $\nu_{V_2}(\omega) = c$. Since V consists of both these subviews, $\nu_V(\omega) = 2c$. But the conclusion of (4.1) states that V should satisfy $\omega = \ell = c$. This is clearly a contradiction. If we require that the term t to be substituted for x is vertically rigid, we ensure that the value of t is the same for V_1 , V_2 and V . For example, we could substitute x by the vertically rigid term ℓ .

The example shows that have to we examine two things for a substitution of t for x in a formula φ to be allowed: On the one hand, it is important whether φ contains any chops, both vertical and horizontal. On the other hand, whether the term t is vertically or horizontally flexible determines its suitability for the substitution. Then, if φ does not contain a horizontal chop (i.e., it is *horizontally chop-free*) and t is vertically rigid, then we may substitute t for x in φ . Similarly, if φ is *vertically chop-free* (i.e., it does not contain a vertical chop), and t is horizontally rigid, we can substitute t for x in φ . If φ does not contain any chop, we can substitute *any* term t for x . Likewise, if t is completely rigid, then φ may contain chops of any kind.

We denote vertical (horizontal) chop-freeness by the predicate vcf (hcf) and vertical (horizontal, dynamical) rigidity by vri (hri , dri). If a term or formula θ is completely rigid, we use the notation $\text{ri}(\theta)$. The rules for the definition of all these predicates are straightforward, since both rigidity and chop-freeness are syntactic properties. All atomic formulas are vertically and horizontally chop-free. For \odot being a Boolean operator or the horizontal chop \frown , the following rules give vertical chop-freeness.

$$\frac{\text{vcf}(\phi) \quad \text{vcf}(\psi)}{\text{vcf}(\phi \otimes \psi)} \text{vcf} \otimes \text{I} \quad \frac{\text{vcf}(\phi \otimes \psi)}{\text{vcf}(\phi)} \text{vcf} \otimes \text{E} \quad \frac{\text{vcf}(\phi \otimes \psi)}{\text{vcf}(\psi)} \text{vcf} \otimes \text{E}$$

The rules for quantifiers and the horizontal rules are defined similarly.

For terms, ℓ is vertically rigid and ω is horizontally rigid. The spatial atoms $re(c)$ and $cl(c)$ are neither horizontally nor vertically rigid, since they require the view to possess an extension greater than zero and exactly one lane. Equality is both vertically and horizontally rigid, as long as both compared terms are rigid. Below, we show some exemplary rules, where \otimes is an arbitrary binary operator.

$$\frac{\text{hri}(\phi) \quad \text{hri}(\psi)}{\text{hri}(\phi \otimes \psi)} \text{hri} \otimes \text{I} \quad \frac{\text{hri}(\phi \otimes \psi)}{\text{hri}(\phi)} \text{hri} \otimes \text{E} \quad \frac{\text{hri}(\phi \otimes \psi)}{\text{hri}(\psi)} \text{hri} \otimes \text{E}$$

The final rules of this section relate rigidity with horizontal and vertical rigidity. For the introduction of general rigidity, observe that we do not need to take dynamic rigidity into concern, since both horizontal as well as vertical rigidity imply dynamic rigidity.

$$\frac{\text{hri}(\phi) \quad \text{vri}(\phi)}{\text{ri}(\phi)} \text{riI} \quad \frac{\text{ri}(\phi)}{\text{hri}(\phi)} \text{riE} \quad \frac{\text{ri}(\phi)}{\text{vri}(\phi)} \text{riE}$$

4.2.1 Relational Properties

We have different rules for the relations between labels. First, we state that each view V is decomposable into two subviews. This is true, since we allow for the empty view, i.e., the view without lanes or with a point-like extension. We use \mathbb{E} to denote existential quantification over views. To use the relations between views, we have to be able to instantiate views, i.e., we have to introduce a rule for *elimination of existential quantifiers over views*. As a side condition for this elimination rule, we require that $ts, v'' : \phi$ is not dependent on any assumption including v' as a label, except for the locality formula ρ . The rule itself is a straightforward adaptation of the classical rule. Again, we only show the case for the vertical relations.

$$\frac{}{\mathbb{E}v' \bullet v = v' \ominus v} V0 \quad \frac{v = v_1 \ominus v_2 \quad v_2 = v'_1 \ominus v'_2}{\mathbb{E}v' \bullet v = v' \ominus v'_2} V\text{Com} \quad \frac{[\rho] \quad \vdots \quad \mathbb{E}v' \bullet \rho \quad ts, v'' : \phi}{ts, v'' : \phi} \mathbb{E}E$$

$$\frac{}{\mathbb{E}v' \bullet v = v \ominus v'} V0 \quad \frac{v = v_1 \ominus v_2 \quad v_1 = v'_1 \ominus v'_2}{\mathbb{E}v' \bullet v = v'_1 \ominus v'_2} V\text{Com} \quad \frac{}{\mathbb{E}v_1, v_2 \bullet v = v_1 \ominus v_2} V\text{Dec}$$

Now we define rules for the interaction between views and traffic snapshots along discrete transitions. The first set of rules (EV) describe the equality of views during discrete transitions. The other set allows for the substitution of subviews for views in the labels. This is a direct consequence of convention 3.1 and the semantics of dynamic and locality formulas.

$$\begin{array}{c}
 \frac{ts, v \xrightarrow{*} ts', v'}{ts, v \xrightarrow{*} ts', v} \text{EV} \qquad \frac{ts, v \xrightarrow{*} ts', v'}{ts, v' \xrightarrow{*} ts', v'} \text{EV} \\
 \\
 \frac{ts, v \xrightarrow{*} ts', v \quad v = v_1 \oplus v_2}{ts, v_1 \xrightarrow{*} ts', v_1} \text{SV} \qquad \frac{ts, v \xrightarrow{*} ts', v \quad v = v_1 \oplus v_2}{ts, v_2 \xrightarrow{*} ts', v_2} \text{SV} \\
 \\
 \frac{ts, v_1 \xrightarrow{*} ts', v_1 \quad v = v_1 \oplus v_2}{ts, v \xrightarrow{*} ts', v} \text{SV} \qquad \frac{ts, v_2 \xrightarrow{*} ts', v_2 \quad v = v_1 \oplus v_2}{ts, v \xrightarrow{*} ts', v} \text{SV}
 \end{array}$$

A similar set of rules exists for the vertical relations. In all of these rules, we require $*$ $\in \{r(c), c(c), \text{wd } r(c), \text{wd } c(c) \mid c \in \text{CVar} \cup \{\text{ego}\}\}$. We also need a set of rules to use the invariance of views between discrete transitions to “move” formulas from one view to another, if these views are connected by a transition. Note that we do not require the formula to be rigid of any kind, since semantically, the views are the same.

$$\begin{array}{c}
 \frac{ts, v \xrightarrow{*} ts', v' \quad ts, v' : \phi}{ts, v : \phi} \text{IV} \qquad \frac{ts, v \xrightarrow{*} ts', v' \quad ts, v : \phi}{ts, v' : \phi} \text{IV} \\
 \frac{ts, v \xrightarrow{*} ts', v' \quad ts', v' : \phi}{ts', v : \phi} \text{IV} \qquad \frac{ts, v \xrightarrow{*} ts', v' \quad ts', v : \phi}{ts', v' : \phi} \text{IV}
 \end{array}$$

A very important difference in the rules IV to the rules in the following section is that the views v and v' are related via discrete transitions. That means essentially that the extension, number of lanes and the owners of these views are the same. Observe that the snapshot in the labelled formulas is fixed.

The rules relating different types of transitions mostly refer to the evolutions and the abstract transitions. The labelling for abstract transitions as well as for evolutions is both reflexive and transitive. Furthermore, we may abstract any given transition to an abstract transition.

$$\begin{array}{c}
 \frac{}{ts, v \xrightarrow{0} ts, v} \xrightarrow{t} r \qquad \frac{ts_1, v_1 \xrightarrow{t_1} ts_2, v_2 \quad ts_2, v_2 \xrightarrow{t_2} ts_3, v_3}{ts_1, v_1 \xrightarrow{t_1+t_2} ts_3, v_3} \xrightarrow{t} tr \\
 \\
 \frac{}{ts, v \Rightarrow ts, v} \Rightarrow r \qquad \frac{ts_1, v_1 \Rightarrow ts_2, v_2 \quad ts_2, v_2 \Rightarrow ts_3, v_3}{ts_1, v_1 \Rightarrow ts_3, v_3} \Rightarrow tr \\
 \\
 \frac{ts, v \xrightarrow{*} ts', v'}{ts, v \Rightarrow ts', v'} \text{abs}
 \end{array}$$

In the rule *abs*, the asterisk $*$ may be instantiated by any transition type.

Finally, let t, t_1, t_2 and t_3 be either timing variables or constants of \mathbb{T} . We then define rules to let \trianglelefteq be a partial order compatible with addition. We also require all timing variables and constants of \mathbb{T} to be at least zero.

$$\begin{array}{c}
 \frac{}{0 \trianglelefteq t} \trianglelefteq 0 \qquad \frac{}{t \trianglelefteq t} \trianglelefteq r \\
 \\
 \frac{t_1 \trianglelefteq t_2 \quad t_2 \trianglelefteq t_3}{t_1 \trianglelefteq t_3} \trianglelefteq tr \qquad \frac{t_1 \trianglelefteq t_2 \quad t_2 \trianglelefteq t_1}{t_1 = t_2} \trianglelefteq sy \qquad \frac{t_1 \trianglelefteq t_2}{t_1 + t_3 \trianglelefteq t_2 + t_3} \trianglelefteq +
 \end{array}$$

Lemma 4.2. The rules concerning the different types of relational formulas are sound.

Proof. The rules $V0$ are sound, since for each locality valuation λ and name of a view v such that $\lambda(v) = (L, X, E)$, we can take the view $V_0 = (\emptyset, X, E)$ to get $\lambda(v) = V_0 \oplus \lambda(v)$ and $\lambda(v) = \lambda(v) \oplus V_0$. Similar reasoning yields soundness of $VDec$. For $VCom$, assume we have a locality valuation λ , such that $\lambda(v) = \lambda(v_1) \oplus \lambda(v_2)$ and $\lambda(v_2) = \lambda(v'_1) \oplus \lambda(v'_2)$. Let $\lambda(v) = (L, X, E)$, $\lambda(v_1) = (L_1, X, E)$, $\lambda(v_2) = (L_2, X, E)$, $\lambda(v'_1) = (L'_1, X, E)$ and $\lambda(v'_2) = (L'_2, X, E)$. Since both $L_1 \cap L_2 = \emptyset$ and $L'_1 \cap L'_2 = \emptyset$, we also know $(L_1 \cup L'_1) \cap L'_2 = \emptyset$. Furthermore, if the sets of lanes are not empty, $\max(L_1) + 1 = \min(L_2)$ and $\max(L'_1) + 1 = \min(L'_2)$ as well as $\max(L_1) < \max(L'_1)$. Hence $\max(L_1 \cup L'_1) + 1 = \min(L'_2)$. That is, for $V' = (L_1 \cup L'_1, X, E)$, we have $\lambda(v) = V' \oplus \lambda(v'_2)$. If some of the sets of lanes are empty, soundness follows similarly. Soundness of $\mathbb{E}E$ is proved like for the standard rule for the elimination of the existential quantifier. For the rules EV , observe that the during discrete transitions, the view is not changed. Hence if for a snapshot valuation σ , a locality valuation λ and a valuation ν , we have, e.g., $\sigma(ts), \lambda(v) \xrightarrow{r(\nu(c))} \sigma(ts'), \lambda(v')$, then we can deduce $\lambda(v) = \lambda(v')$, i.e., both types of the rule EV are sound. For the rules SV , we just have to recognise that the existence of a discrete transition is fully independent of the views at hand, i.e., the semantics of the dynamic formula is not affected, when the view changes. The proofs for the soundness of IV are similar. Soundness of the rules and axioms concerning the evolutions and abstract transitions follows straightforwardly from the semantic definitions of these transitions. Finally, for the rules concerning the partial order on time, the soundness follows since \mathbb{R}_+ is an additive ordered group. \square

4.2.2 Rigidity

The intuition of rigidity is formalised in the following rules. Whenever a formula is horizontally rigid, the formula holds on all views horizontally reachable from the current view. The rules for vertically rigidity are similar.

$$\begin{array}{c}
 \frac{ts, v: \phi \quad \text{hri}(\phi) \quad v = v_1 \oplus v_2}{ts, v_1: \phi} R_H \quad \frac{ts, v: \phi \quad \text{hri}(\phi) \quad v = v_1 \oplus v_2}{ts, v_2: \phi} R_H \\
 \frac{ts, v_1: \phi \quad \text{hri}(\phi) \quad v = v_1 \oplus v_2}{ts, v: \phi} R_H \quad \frac{ts, v_2: \phi \quad \text{hri}(\phi) \quad v = v_1 \oplus v_2}{ts, v: \phi} R_H \\
 \frac{ts, v: \phi \quad \text{dri}(\phi) \quad ts, v \xrightarrow{*} ts', v'}{ts', v': \phi} R_D \quad \frac{ts', v': \phi \quad \text{dri}(\phi) \quad ts, v \xrightarrow{*} ts', v'}{ts, v: \phi} R_D
 \end{array}$$

Again, the difference of the rules R_D to the rules IV is, that in the former the whole label is transformed, where in the latter, only the view may change.

Lemma 4.3. The rules concerning the different types of rigidity are sound.

Proof. Immediate by Lemma 4.1. \square

4.2.3 First-Order Logic

For the first-order operators, we use the typical definitions of labelled natural deduction rules [BMV98]. The only difference lies in the rules for quantification. We may instantiate an universally quantified variable with a horizontally (vertically) rigid, if the formula is vertically (horizontally) chop-free. If the formula is completely chop-free, we may instantiate the variable with an arbitrary term. Similarly, rigid terms may instantiate x in arbitrary formulas. In all cases, a side condition for the instantiation is that s respects the sort of x .

$$\frac{ts, v: \forall x \bullet \phi \quad \text{hcf}(\phi) \quad \text{vri}(s)}{ts, v: \phi[x \mapsto s]} \forall\text{E} \quad \frac{ts, v: \forall x \bullet \phi \quad \text{vcf}(\phi) \quad \text{hri}(s)}{ts, v: \phi[x \mapsto s]} \forall\text{E}$$

$$\frac{ts, v: \forall x \bullet \phi \quad \text{hcf}(\phi) \quad \text{vcf}(\phi)}{ts, v: \phi[x \mapsto s]} \forall\text{E} \quad \frac{ts, v: \forall x \bullet \phi \quad \text{hri}(s) \quad \text{vri}(s)}{ts, v: \phi[x \mapsto s]} \forall\text{E}$$

The introduction rule for the universal quantifier is adapted, such that the variable to be quantified has to be completely rigid.

$$\frac{[\text{hri}(x)] \quad [\text{vri}(x)] \quad \vdots \quad ts, v: \phi}{ts, v: \forall x \bullet \phi} \forall\text{I}$$

The application condition is that x may not occur free in any assumption on which $ts, v: \phi$ depends, except for $\text{hri}(x)$ and $\text{vri}(x)$.

The rules concerning equality are a direct translation from Rasmussen's work [Ras01]

$$\frac{}{ts, v: s = s} \text{Refl} \quad \frac{ts, v: s = t}{ts, v: t = s} \text{Sym} \quad \frac{ts, v: s = t \quad ts, v: t = u}{ts, v: s = u} \text{Trans}$$

$$\frac{ts, v: \phi[x \mapsto s] \quad ts, v: s = t \quad \text{ri}(s) \quad \text{ri}(t)}{ts, v: \phi[x \mapsto t]} \text{Subst}_{\text{ri}}$$

$$\frac{ts, v: \phi[x \mapsto s] \quad ts, v: s = t \quad \text{vcf}(\phi) \quad \text{hcf}(\phi)}{ts, v: \phi[x \mapsto t]} \text{Subst}_{\text{cf}}$$

Lemma 4.4. The rules concerning first-order quantifiers and equality are sound.

Proof. The proof for equality, substitution and the introduction rule of the universal quantifier is straightforward.

Now consider the case of the elimination of the universal quantifier, where ϕ is horizontally chop-free and s is vertically rigid, and let $\sigma, \lambda, \nu \models ts, v: \forall x \bullet \phi$. Then, the semantic of the universal quantifier yields that $\sigma, \lambda, \nu \oplus \{x \mapsto \alpha\} \models ts, v: \phi$ for all α respecting the sort of x . In particular, since x and s are of the same sort, there is an α' such that $\nu(s) = \alpha'$. This valuation is the same for all subviews of $\lambda(v)$ used for the evaluation of ϕ , since ϕ may at most contain vertical chops and s is vertically rigid. Hence $\sigma, \lambda, \nu \oplus \{x \mapsto \alpha'\} \models ts, v: \phi$ is true and equivalent to $\sigma, \lambda, \nu \models ts, v: \phi[x \mapsto s]$. The other cases of the elimination rule are proved similarly. \square

4.2.4 Chopping, Decomposition and Transitions

The elimination and introduction rules for the chop modalities are adopted from Rasmussen [Ras01], and resemble the rules for existential quantification. We only show the case for the horizontal chop, the rules for vertical chopping are obtained straightforwardly, by replacing horizontal modalities and relations by the vertical ones.

$$\frac{\frac{ts, v_1: \phi \quad ts, v_2: \psi \quad v = v_1 \oplus v_2}{ts, v: \phi \wedge \psi} \wedge I \quad \frac{\begin{array}{c} [ts, v_1: \phi] \\ [ts, v_2: \psi] \\ [v = v_1 \oplus v_2] \\ \vdots \\ [ts, v: \phi \wedge \psi] \end{array} \quad \frac{ts, v: \phi \wedge \psi \quad ts', v': \chi}{ts', v': \chi} \wedge E}{ts', v': \chi} \wedge E$$

The application conditions for the chop elimination rules are, that the names v_1 and v_2 differ from v and v' and may not occur free in any assumption on which $ts', v': \chi$ depends, except for $ts, v_1: \phi$, $ts, v_2: \psi$ and $v = v_1 \oplus v_2$ ($v = v_1 \ominus v_2$ respectively).

The chopping of intervals is not ambiguous, i.e., there is a unique view of a certain length at the beginning of a view. This is the *single decomposition property* [Dut95] of interval logics and captured in the following rules. When there are two vertical chops of a view, and the upper parts are of equal width, we can derive that the same formulas hold on the lower parts and vice versa. Even though we only show the vertical set of rules, similar rules hold for the horizontal chopping of views. The structure of the rules follows the presentation of Rasmussen [Ras01].

$$\frac{ts, v_1: \phi \quad ts, v_2: \omega = s \quad ts, v'_2: \omega = s \quad \text{vri}(s) \quad v = v_1 \ominus v_2 \quad v = v'_1 \ominus v'_2}{ts, v'_1: \phi} VD$$

$$\frac{ts, v_2: \phi \quad ts, v_1: \omega = s \quad ts, v'_1: \omega = s \quad \text{vri}(s) \quad v = v_1 \ominus v_2 \quad v = v'_1 \ominus v'_2}{ts, v'_2: \phi} VD$$

The additivity of length and width can be formalised by the following rules.

$$\frac{ts, v_1: \omega = s \quad ts, v_2: \omega = t \quad \text{vri}(s) \quad \text{vri}(t) \quad v = v_1 \ominus v_2}{ts, v: \omega = s + t} V + I$$

$$\frac{\begin{array}{c} [ts, v_1: \omega = s] \\ [ts, v_2: \omega = t] \\ [v = v_1 \ominus v_2] \\ \vdots \\ [ts, v: \omega = s + t] \end{array} \quad \frac{\text{vri}(s) \quad \text{vri}(t) \quad ts', v': \phi}{ts', v': \phi} V + E}{ts', v': \phi} V + E$$

Similar to the elimination of chops, the application condition for $V + E$ ($H + E$, respectively) is that the names v_1 and v_2 differ from v and v' and may not occur free in any assumption, on which $ts', v': \phi$ depends, except for $ts, v_1: \omega = s$, $ts, v_2: \omega = t$ and $v = v_1 \ominus v_2$ ($ts, v_1: \ell = s$, $ts, v_2: \ell = t$ and $v = v_1 \oplus v_2$, respectively).

The modalities describing the discrete transitions are defined along the lines of Basin et al. [BMV98]. If a transition from the current snapshot is known to be possible, the box modalities may be eliminated. If we can prove that under the assumption of a transition $* \in \{r(d), c(d), \text{wd } r(d), \text{wd } c(d)\}$, ϕ holds on the now reachable snapshot, then also $\Box_*\phi$ is true. The application condition for $\Box_*\text{I}$ is that neither ts' nor v' may occur in any assumption on which $ts', v': \phi$ depends, except for $ts, v \xrightarrow{*} ts', v'$ and that they both differ from ts and v .

$$\frac{[ts, v \xrightarrow{*} ts', v'] \quad \frac{ts, v \xrightarrow{*} ts', v' \quad ts, v: \Box_*\phi}{ts', v': \phi} \Box_*\text{E} \quad \frac{ts', v': \phi}{ts, v: \Box_*\phi} \Box_*\text{I}}{\vdots}$$

For the dynamic transitions, we need to include assertions about the duration t for which time evolves. That is, we add two more premisses, which state that t stays within the bounds of the modality.

$$\frac{[ts, v \xrightarrow{t} ts', v'] \quad [a \leq t] \quad [t \leq b] \quad \frac{ts, v \xrightarrow{t} ts', v' \quad ts, v: \Box_{[a,b]}\phi \quad a \leq t \quad t \leq b}{ts', v': \phi} \Box_t\text{E} \quad \frac{ts', v': \phi}{ts, v: \Box_{[a,b]}\phi} \Box_t\text{I}}{\vdots}$$

Similar to the rules $\Box_*\text{I}$, the application condition for $\Box_t\text{I}$ is that ts' and v' are different from ts and v and neither ts' nor v' nor t appear in any assumption, on which $ts', v': \phi$ depends, except for $ts, v \xrightarrow{t} ts', v'$, $a \leq t$, and $t \leq b$. Furthermore ts and v have to be different from ts' and v' .

Lemma 4.5. The rules concerning chops, the single decomposition property and the additivity of lengths as well as width of views are sound.

Proof. The proof is a straightforward adaptation of the proofs in the work of Rasmussen [Ras01], Basin et al. [BMV98] and Viganò [Vig00], respectively. \square

4.2.5 Specifying the Domains

We need further rules to specify certain properties of our different domains. For example, $\ell \frown$ states that the domain of the extensions is dense, while $\ell 0$ prohibits negative extensions.

$$\frac{ts, v: \ell > 0}{ts, v: \ell > 0 \frown \ell > 0} \ell \frown \quad \frac{}{ts, v: \ell \geq 0} \ell 0 \quad \frac{}{ts, v: \omega \geq 0} \omega 0$$

$$\frac{}{ts, v: x \leq x} \quad \frac{ts, v: x \leq y \quad ts, v: y \leq z}{ts, v: x \leq z} \quad \frac{ts, v: x \leq y}{ts, v: x + z \leq y + z}$$

In these rules, we silently assume that the terms x , y and z are of the same sort, and especially are not car variables, i.e., $x, y, z \notin \text{CVar} \cup \{\text{ego}\}$.

4.2.6 Spatial Properties of Spatial Atoms

In this subsection, we present rules which reflect the connection of reservations and claims to the spatial domain. On the one hand, we have rules which are straightforward implications of the intended semantics of the atoms, e.g., that the length of a view which satisfies $re(c)$ has to be greater than zero and has to consist of exactly one lane. On the other hand, we have rules which state that reservations and claims are dense in the extension of views. In comparison with Rasmussen's presentation of Duration Calculus [Ras01], these rules seem to be very specific, and one might be tempted to assume that they are already derivable in the given system. However, since we do not have any type of term resembling integration, we are, e.g., not yet able to state directly that on a view V satisfying $re(c)$, each subview also has to satisfy $re(c)$. So these rules are needed due to the very specific and narrow type of atoms we permit within EMLSL.

$$\begin{array}{c}
\frac{ts, v: re(c)}{ts, v: \ell > 0} \text{ExtRe} \quad \frac{ts, v: cl(c)}{ts, v: \ell > 0} \text{ExtCl} \quad \frac{ts, v: re(c)}{ts, v: \omega = 1} \text{LaneRe} \quad \frac{ts, v: cl(c)}{ts, v: \omega = 1} \text{LaneCl} \\
\\
\frac{ts, v: re(c)}{ts, v: re(c) \wedge re(c)} \text{dRe} \quad \frac{ts, v: cl(c)}{ts, v: cl(c) \wedge cl(c)} \text{dCl} \\
\\
\frac{ts, v: re(c) \quad ts, v_1: \ell > 0 \quad v = v_1 \oplus v_2}{ts, v_1: re(c)} \text{cRe} \quad \frac{ts, v: cl(c) \quad ts, v_1: \ell > 0 \quad v = v_1 \oplus v_2}{ts, v_1: cl(c)} \text{cCl} \\
\\
\frac{ts, v: re(c) \quad ts, v_2: \ell > 0 \quad v = v_1 \oplus v_2}{ts, v_2: re(c)} \text{cRe} \quad \frac{ts, v: cl(c) \quad ts, v_2: \ell > 0 \quad v = v_1 \oplus v_2}{ts, v_2: cl(c)} \text{cCl}
\end{array}$$

Lemma 4.6. The rules concerning the spatial properties of atoms are sound.

Proof. Immediate by the semantics of $re(c)$ and $cl(c)$ and the density of the extension domain. \square

4.2.7 Spatial Atoms and Transitions

Finally, we have to define how the spatial atoms behave with respect to occurring transitions. There are two types of rules in general, *stability rules* and *activity rules*. Stability rules define which atoms stay true after a snapshot changes according to a certain transition. The activity rules state how the reservations and claims of cars will change according to the transitions. In all of the rules within this section, the assumed transition will be shown as the middle premiss, the spatial property to be reasoned about will be the left premiss, and the relation between the car d responsible for the transition and the car c , whose property is visible in the view, will be shown on the right.

If a transition occurs, the truth of all reservations and claims of cars not involved in the transition are unchanged. For the creation of a claim, the following stability rules show that the reservations and claims of other cars are unchanged. We have similar stability rules for the other types of transitions.

$$\frac{ts, v: cl(c) \quad ts, v \xrightarrow{c(d)} ts', v' \quad ts, v: c \neq d}{ts', v': cl(c)} \xrightarrow{c} S$$

$$\frac{ts, v: re(c) \quad ts, v \xrightarrow{c(d)} ts', v' \quad ts, v: c \neq d}{ts', v': re(c)} \xrightarrow{c} S$$

There is an additional stability rule for the creation of reservations. It states that the already existing reservation of a car creating a new reservation is not altered.

$$\frac{ts, v: re(c) \quad ts, v \xrightarrow{r(d)} ts', v' \quad ts, v: c = d}{ts', v': re(c)} \xrightarrow{r} S$$

The activity rule for $c(c)$ implies two properties. First, a claim may only be created when only one reservation exists. Second, the newly created claim resides on one side of the existing reservation. Observe that we require the view under consideration to comprise both adjacent lanes of the reservation. If we dropped this assumption (i.e., removed the subformulas $\omega = 1$), it would be possible for the newly created claim to reside outside of the view V , and hence the conclusion would not be satisfied.

$$\frac{\begin{array}{c} \neg(re(c) \vee cl(c)) \wedge \omega = 1 \\ ts, v: \quad re(c) \quad ts, v \xrightarrow{c(d)} ts', v' \quad ts, v: c = d \\ \neg(re(c) \vee cl(c)) \wedge \omega = 1 \end{array}}{\begin{array}{c} \neg(re(c) \vee cl(c)) \wedge \omega = 1 \quad cl(c) \\ ts', v': \quad re(c) \quad \vee \quad re(c) \\ \quad \quad \quad cl(c) \quad \quad \quad \neg(re(c) \vee cl(c)) \wedge \omega = 1 \end{array}} \xrightarrow{c} A$$

The rule for the creation of reservations in between traffic snapshots is the following. It only ensures that if we observed the claim of the car before the transition, we can perceive its reservation afterwards.

$$\frac{ts, v: cl(c) \quad ts, v \xrightarrow{r(d)} ts', v' \quad ts, v: c = d}{ts', v': re(c)} \xrightarrow{r} A$$

The following activity rules define the withdrawal of reservations and claims. Observe that for the withdrawal of a claim, we do not specify the spatial properties of the traffic snapshot and view the transition originates from. If the claim of a car d was withdrawn, we can be sure that no claim of d exists at any part of the freeway.

$$\frac{ts, v: \quad re(c) \quad ts, v \xrightarrow{wd \ r(d)} ts', v' \quad ts, v: c = d}{ts', v': \quad re(c) \quad \vee \quad \neg re(c) \quad re(c)} \xrightarrow{wd \ r} A$$

$$\frac{ts, v \xrightarrow{wd \ c(d)} ts', v' \quad ts, v: c = d}{ts', v': \neg cl(c)} \xrightarrow{wd \ c} A$$

We also have rules for “backwards” reasoning, i.e., if our current snapshot is reachable from another, we may draw conclusions about the originating snapshot. Again, we differentiate between activity and stability rules. The stability rules all follow the same

schema, where $*$ \in $\{c, \text{wd } c, \text{wd } r\}$. Observe that these rules all have the existence of a reservation at the destination traffic snapshot and view as a premiss. For checking soundness of these rules intuitively, consider $\xleftarrow{\text{wd } r} S$. If after the withdrawal of a reservation, we can still perceive a reservation at the given part of the freeway, it had to be there before the withdrawal. Similar reasoning holds for the other backwards stability rules.

$$\frac{ts', v': re(c) \quad ts, v \xrightarrow{* (d)} ts', v' \quad ts, v: c = d}{ts, v: re(c)} \xleftarrow{*} S$$

$$\frac{ts', v': re(c) \quad ts, v \xrightarrow{* (d)} ts', v' \quad ts, v: c \neq d}{ts, v: re(c)} \xleftarrow{*} S$$

$$\frac{ts', v': re(c) \quad ts, v \xrightarrow{r (d)} ts', v' \quad ts, v: c \neq d}{ts, v: re(c)} \xleftarrow{r} S$$

For the backwards activity rule for the creation of a reservation, we can only ensure that either a claim or a reservation was perceived before the transition, since we can not distinguish the newly created reservation.

$$\frac{ts', v': re(c) \quad ts, v \xrightarrow{r (d)} ts', v' \quad ts, v: c = d}{ts, v: re(c) \vee cl(c)} \xleftarrow{r} A$$

For the backwards activity rule for creation of claims however, we can be sure that no claim existed before the transition occurred.

$$\frac{ts', v': cl(c) \quad ts, v \xrightarrow{c (d)} ts', v' \quad ts, v: c = d}{ts, v: \neg cl(c)} \xleftarrow{c} A$$

If we want to reason backwards along the withdrawal of a reservation, we have to ensure that the lanes next to the perceived reservation after the transition are contained in the view. Otherwise we could not state that both reservations were present within the view before the transition occurred.

$$\frac{\begin{array}{c} \omega = 1 \\ ts', v': re(c) \\ \omega = 1 \end{array} \quad ts, v \xrightarrow{\text{wd } r (d)} ts', v' \quad ts, v: c = d}{\begin{array}{c} re(c) \quad \omega = 1 \\ ts, v: re(c) \vee re(c) \\ \omega = 1 \quad re(c) \end{array}} \xleftarrow{\text{wd } r} A$$

Observe that we cannot define a backwards activity rule for the withdrawal of a claim. If we know that after the withdrawal of the claim of c , we cannot perceive a claim of c , we can still not be sure whether the claim was visible in our view before the transition was taken. It could be the case that no part of c was ever visible within the view.

Lemma 4.7. The rules concerning the changes of spatial atoms along transitions are sound.

Proof. We only show the proofs for the activity rules. The proofs for the stability rules are straightforward, since only the reservations and claims of the car taking a transition may change.

Case 1: Consider an application of $\xrightarrow{\text{wd } r} A$. We assume

$$(\Gamma_1, \Delta_1) \models ts, v: \begin{array}{l} re(c) \\ re(c) \end{array} \quad \text{and} \quad (\Gamma_2, \Delta_2) \models ts, v \xrightarrow{\text{wd } r(d)} ts', v' \quad \text{and} \quad (\Gamma_3, \Delta_3) \models ts, v: c = d .$$

Now we have to show, that with $\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 = \Gamma$ and $\Delta_1 \cup \Delta_2 \cup \Delta_3 = \Delta$, we get

$$(\Gamma, \Delta) \models ts', v': \begin{array}{l} re(c) \\ \neg re(c) \end{array} \vee \begin{array}{l} \neg re(c) \\ re(c) \end{array} .$$

Let σ, λ and ν be suitable valuations such that $\sigma, \lambda, \nu \models (\Gamma, \Delta)$, i.e., $\sigma, \lambda, \nu \models (\Gamma_1, \Delta_1)$, $\sigma, \lambda, \nu \models (\Gamma_2, \Delta_2)$ and $\sigma, \lambda, \nu \models (\Gamma_3, \Delta_3)$. Hence

$$\begin{aligned} \sigma, \lambda, \nu \models ts, v: \begin{array}{l} re(c) \\ re(c) \end{array} , \\ \sigma, \lambda, \nu \models ts, v \xrightarrow{\text{wd } r(d)} ts', v' , \\ \sigma, \lambda, \nu \models ts, v: c = d . \end{aligned}$$

Let $\lambda(v) = V_1 \ominus V_2$, such that $\sigma(ts), V_1, \nu \models re(c)$ and $\sigma(ts), V_2, \nu \models re(c)$ with $V_i = (L_i, X, E)$. Furthermore, by the semantics of $re(c)$ we get that $|L_i| = 1$. We know that there is a lane n_0 , such that $\sigma(ts), \lambda(v) \xrightarrow{\text{wd } r(\nu(d), n_0)} \sigma(ts'), \lambda(v)$. Let $n_0 \in L_1$. Then by Definition 3.3, and since $\nu_{\lambda(v)}(c) = \nu_{\lambda(v)}(d)$, we have that $res'_{V_2}(\nu(c)) = res'_{V_2}(\nu(d)) = \emptyset$, which means $\sigma(ts'), V_2, \nu \not\models re(c)$, that is, $\sigma(ts'), V_2, \nu \models \neg re(c)$. Furthermore, we have $n_0 \in res'_{V_1}(\nu(d))$, i.e., $\sigma(ts'), V_1, \nu \models re(c)$. Observe that $\lambda(v) = \lambda(v')$, i.e., we have $\lambda(v') = V_1 \ominus V_2$. By definition of the vertical chop, we get

$$\sigma(ts'), \lambda(v'), \nu \models \begin{array}{l} \neg re(c) \\ re(c) \end{array}$$

and hence

$$\sigma(ts'), \lambda(v'), \nu \models \begin{array}{l} re(c) \\ \neg re(c) \end{array} \vee \begin{array}{l} \neg re(c) \\ re(c) \end{array} .$$

If $n_0 \in L_2$, the reasoning is analogous. All in all, we get that

$$\sigma, \lambda, \nu \models ts', v': \begin{array}{l} re(c) \\ \neg re(c) \end{array} \vee \begin{array}{l} \neg re(c) \\ re(c) \end{array} .$$

Case 2: Consider an application of $\xrightarrow{r} A$. Then let

$$(\Gamma_1, \Delta_1) \models ts, v: cl(c) \quad \text{and} \quad (\Gamma_2, \Delta_2) \models ts, v \xrightarrow{r(d)} ts', v' \quad \text{and} \quad (\Gamma_3, \Delta_3) \models ts, v: c = d .$$

Now let $\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 = \Gamma$ and $\Delta_1 \cup \Delta_2 \cup \Delta_3 = \Delta$. Let σ, λ and ν be valuations such that $\sigma, \lambda, \nu \models (\Gamma, \Delta)$. That is,

$$\begin{aligned} \sigma, \lambda, \nu &\models ts, v: cl(c) , \\ \sigma, \lambda, \nu &\models ts, v \xrightarrow{r(d)} ts', v' , \\ \sigma, \lambda, \nu &\models ts, v: c = d . \end{aligned}$$

We assume $\sigma(ts) = (res, clm, pos, spd, acc)$ and $\sigma(ts') = (res', clm', pos', spd', acc')$. Thus, $clm_{\lambda(v)}(\nu(c)) = L$, where L are the lanes of $\lambda(v) = \lambda(v')$. By Definition 3.3 we get that $res'(\nu(d)) = res(\nu(d)) \cup clm(\nu(d))$ and, since $\nu_{\lambda(v)}(c) = \nu_{\lambda(v)}(d)$, $res'_{\lambda(v)}(\nu(c)) = clm_{\lambda(v)}(\nu(c)) = L$. So $\sigma(ts'), \lambda(v'), \nu \models re(c)$ and by that $\sigma, \lambda, \nu \models ts', v': re(c)$.

Case 3: Consider an application of $\xrightarrow{wd\ c} A$. Let $(\Gamma_1, \Delta_1) \models ts, v \xrightarrow{wd\ c(d)} ts', v'$ and $(\Gamma_2, \Delta_2) \models ts, v: c = d$. Furthermore, we let $\Gamma = \Gamma_1 \cup \Gamma_2$ and $\Delta = \Delta_1 \cup \Delta_2$. Hence if $\sigma, \lambda, \nu \models (\Gamma, \Delta)$, then $\sigma(ts), \lambda(v) \xrightarrow{wd\ c(\nu(d))} \sigma(ts'), \lambda(v')$ is true as well as $\nu_{\lambda(v)}(c) = \nu_{\lambda(v)}(d)$. That is, if $\sigma(ts') = (res', clm', pos', spd', acc')$, then $clm'(\nu(c)) = \emptyset$. Hence, since $\lambda(v) = \lambda(v')$, we have $\sigma(ts'), \lambda(v'), \nu \models \neg cl(c)$. Thus $\sigma, \lambda, \nu \models ts', v': \neg cl(c)$.

Case 4: Consider an application of $\xrightarrow{c} A$. We assume

$$(\Gamma_1, \Delta_1) \models ts, v: \begin{array}{l} \neg(re(c) \vee cl(c)) \wedge \omega = 1 \\ re(c) \\ \neg(re(c) \vee cl(c)) \wedge \omega = 1 \end{array} ,$$

$(\Gamma_2, \Delta_2) \models ts, v \xrightarrow{c(d)} ts', v'$ and $(\Gamma_3, \Delta_3) \models ts, v: c = d$. Furthermore, let $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$, $\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3$ and $\sigma, \lambda, \nu \models (\Gamma, \Delta)$. That is, for $\lambda(v) = (L, X, E)$ and $\sigma(ts) = (res, clm, pos, spd, acc)$, we know that L contains exactly three elements, say $L = \{n_1, n_2, n_3\}$, where $n_2 = n_1 + 1$ and $n_3 = n_2 + 1$. Then we get $res(\nu(c)) = \{n_2\}$ and $clm(\nu(c)) = \emptyset$. Now consider $\sigma(ts') = (res', clm', pos', spd', acc')$. Since $\sigma, \lambda, \nu \models ts, v \xrightarrow{c(d)} ts', v'$, we have $\sigma(ts) \xrightarrow{c(\nu(d), n')} \sigma(ts')$ for either $n' = n_1$ or $n' = n_3$ (by Def. 3.3). Furthermore, due to $\nu_{\lambda(v)}(c) = \nu_{\lambda(v)}(d)$ we know that $clm'(\nu(c)) = \{n'\}$. Say $n' = n_1$. Then $\sigma(ts'), \lambda(v')^{\{n_1\}}, \nu \models cl(c)$. Note that the extension of $\lambda(v')^{\{n_1\}}$ has to be greater than zero, since the subview $\lambda(v)^{\{n_2\}}$ already satisfies $re(c)$ and $\lambda(v) = \lambda(v')$. Due to $res = res'$, we get

$$\begin{aligned} &\sigma(ts'), \lambda(v'), \nu \models \begin{array}{l} \neg(re(c) \vee cl(c)) \wedge \omega = 1 \\ re(c) \\ cl(c) \end{array} \\ \Rightarrow &\sigma, \lambda, \nu \models ts', v': \begin{array}{l} \neg(re(c) \vee cl(c)) \wedge \omega = 1 \\ re(c) \\ cl(c) \end{array} \\ \Rightarrow &\sigma, \lambda, \nu \models ts', v': \begin{array}{l} \neg(re(c) \vee cl(c)) \wedge \omega = 1 \\ re(c) \\ cl(c) \end{array} \vee \begin{array}{l} cl(c) \\ re(c) \\ \neg(re(c) \vee cl(c)) \wedge \omega = 1 \end{array} . \end{aligned}$$

The case where $n' = n_3$ is similar.

Case 5: Consider an application of $\overset{c}{\leftarrow}A$. Then $(\Gamma_1, \Delta_1) \models ts', v': cl(c)$, $(\Gamma_2, \Delta_2) \models ts, v \xrightarrow{c(d)} ts', v'$ and $(\Gamma_3, \Delta_3) \models ts, v: c = d$. We assume $\sigma, \lambda, \nu \models (\Gamma, \Delta)$, i.e., $\sigma, \lambda, \nu \models (\Gamma_1, \Delta_1)$, $\sigma, \lambda, \nu \models (\Gamma_2, \Delta_2)$ and $\sigma, \lambda, \nu \models (\Gamma_3, \Delta_3)$. Since $c = d$ is dynamically rigid, we also have $\sigma, \lambda, \nu \models ts', v': c = d$ by Lemma 4.1. So $\nu_{\lambda(v')}(c) = \nu_{\lambda(v)}(d)$. There can only be a transition creating a new claim for $\nu_{\lambda(v')}(c)$ from $\sigma(ts)$ to $\sigma(ts')$, if $clm(\nu_{\lambda(v')}(c)) = \emptyset$ on $\sigma(ts)$. Hence, for each view V' , we have $\sigma(ts), V', \nu \models \neg cl(c)$. That is in particular $\sigma(ts), \lambda(v), \nu \models \neg cl(c)$, which yields $\sigma, \lambda, \nu \models ts, v: \neg cl(c)$.

Case 6: Consider an application of $\overset{wd\ r}{\leftarrow}A$ and let

$$(\Gamma_1, \Delta_1) \models ts', ts': \begin{array}{l} \omega = 1 \\ re(c) \\ \omega = 1 \end{array},$$

$(\Gamma_2, \Delta_2) \models ts, v \xrightarrow{wd\ r(d)} ts', v'$ and $(\Gamma_3, \Delta_3) \models ts, v: c = d$. Furthermore, let $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$ and $\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3$. Now assume $\sigma, \lambda, \nu \models (\Gamma, \Delta)$, and let $\sigma(ts) = (res, clm, pos, spd, acc)$ as well as $\sigma(ts') = (res', clm', pos', spd', acc')$. We know that the set of lanes L of $\lambda(v') = \lambda(v) = (L, X, E)$ contains exactly three elements, say $L = \{n_1, n_2, n_3\}$, where $n_2 = n_1 + 1$ and $n_3 = n_2 + 1$. By the semantics of the transitions (see Def. 3.3) and EMLSL (see Def. 4.4), we get $res'(\nu(c)) = \{n_2\}$. The transition exists only, when $|res(\nu(c))| = 2$ and $n_2 \in res(\nu(c))$, so there are only two possibilities (due to the sanity conditions of Def. 3.1): $n_1 \in res(\nu(c))$ or $n_3 \in res(\nu(c))$. Say $n_1 \in res(\nu(c))$. Then

$$\sigma(ts), \lambda(v), \nu \models \begin{array}{l} \omega = 1 \\ re(c) \\ re(c) \end{array}$$

and hence

$$\sigma, \lambda, \nu \models ts, v: \begin{array}{l} re(c) \\ \omega = 1 \end{array} \vee \begin{array}{l} \omega = 1 \\ re(c) \\ re(c) \end{array}.$$

The case for $n_3 \in res(\nu(c))$ is similar.

Case 7: Consider an application of $\overset{r}{\leftarrow}A$ and let $(\Gamma_1, \Delta_1) \models ts', v': re(c)$, $(\Gamma_2, \Delta_2) \models ts, v \xrightarrow{r(d)} ts', v'$ and $(\Gamma_3, \Delta_3) \models ts, v: c = d$. Now assume $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$, $\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3$ and $\sigma, \lambda, \nu \models (\Gamma, \Delta)$. Furthermore, let $\sigma(ts) = (res, clm, pos, spd, acc)$ and $\sigma(ts') = (res', clm', pos', spd', acc')$. We then know that $res'_{\lambda(v')}(c) = \{n\}$ where $\lambda(v) = \lambda(v') = (L, X, E)$ with $L = \{n\}$ and $\|X\| > 0$. By Def. 3.3 and $\nu_{\lambda(v)}(c) = \nu_{\lambda(v)}(d)$ we get that $res'(\nu(c)) = res(\nu(c)) \cup clm(\nu(c))$. If $n \in res(\nu(c))$, we have $\sigma(ts), \lambda(v), \nu \models re(c)$, which implies $\sigma(ts), \lambda(v), \nu \models re(c) \vee cl(c)$. Similarly, if $n \in clm(\nu(c))$, we get $\sigma(ts), \lambda(v), \nu \models cl(c)$, which implies $\sigma(ts), \lambda(v), \nu \models re(c) \vee cl(c)$. That is, $\sigma, \lambda, \nu \models ts, v: re(c) \vee cl(c)$. \square

4.2.8 Invariance

For the introduction of the globally modality \mathbf{G} , we employ the rule $\mathbf{G I}$ resembling inductive reasoning. If the current snapshot \mathcal{TS} and view V are a model of the formula ϕ and we can show that ϕ is preserved under all possible transitions, then we know that \mathcal{TS} and V satisfy $\mathbf{G} \phi$. Hence we get the most complicated rule of our calculus, $\mathbf{G I}$.

$$\frac{
 \begin{array}{ccccc}
 [ts, v \Rightarrow ts_1, v_1] & [ts, v \Rightarrow ts_2, v_2] & [ts, v \Rightarrow ts_3, v_3] & [ts, v \Rightarrow ts_4, v_4] & [ts, v \Rightarrow ts_5, v_5] \\
 [ts_1, v_1 \xrightarrow{r(d)} ts'_1, v'_1] & [ts_2, v_2 \xrightarrow{c(d)} ts'_2, v'_2] & [ts_3, v_3 \xrightarrow{wd \ r(d)} ts'_3, v'_3] & [ts_4, v_4 \xrightarrow{wd \ c(d)} ts'_4, v'_4] & [ts_5, v_5 \xrightarrow{t} ts'_5, v'_5] \\
 [ts_1, v_1 : \phi] & [ts_2, v_2 : \phi] & [ts_3, v_3 : \phi] & [ts_4, v_4 : \phi] & [ts_5, v_5 : \phi] \\
 \vdots & \vdots & \vdots & \vdots & \vdots \\
 ts, v : \phi & ts'_1, v'_1 : \phi & ts'_2, v'_2 : \phi & ts'_3, v'_3 : \phi & ts'_4, v'_4 : \phi & ts'_5, v'_5 : \phi
 \end{array}
 }{
 ts, v : \mathbf{G} \phi
 }$$

In the induction rule, the worlds ts_i, v_i may not occur in any assumption on which $ts'_i, v'_i : \phi$ depends except for the assumptions eliminated in this application. Furthermore, the worlds ts'_i, v'_i are all different from each other and from the worlds ts_i, v_i as well as ts, v . Finally, d and t may not occur free in any assumption the premisses depend on, except for the assumptions eliminated by the application of this rule.

Observe that we do not impose any additional assumptions on t . We want t to possibly denote any time in the interval $[0, \infty)$. That is, the only constraint on t is $0 \leq t$. But this timing formula is an axiom of our system (cf. Sect. 4.2.1), which is why we can omit it from the definition of this rule. The rule is inspired by the work of Manna and Pnueli on program verification [MP95]. They defined a proof rule for the invariance of a formula φ , where the premisses state that the current state satisfies φ , and all transition preserve φ .

The elimination rule is much simpler and resembles the elimination rules for box modalities.

$$\frac{ts, v : \mathbf{G} \phi \quad ts, v \Rightarrow ts', v'}{ts', v' : \phi} \mathbf{G E}$$

Lemma 4.8. The introduction and elimination rules for \mathbf{G} are sound.

Proof. The soundness of $\mathbf{G E}$ is straightforward, hence we concentrate on the introduction rule of the invariance modality.

We assume that $(\Gamma, \Delta) \models ts, v : \phi$ and $(\Gamma_i, \Delta_i) \models ts'_i, v'_i : \phi$ for $1 \leq i \leq 5$. Subsequently, we let

$$\Gamma' = \Gamma \cup \bigcup_{1 \leq i \leq 5} (\Gamma_i \setminus \{ts_i, v_i : \phi\})$$

and

$$\begin{aligned}
 \Delta' = \Delta & \cup (\Delta_1 \setminus \{ts, v \Rightarrow ts_1, v_1, ts_1, v_1 \xrightarrow{r(d)} ts'_1, v'_1\}) \\
 & \cup (\Delta_2 \setminus \{ts, v \Rightarrow ts_2, v_2, ts_2, v_2 \xrightarrow{c(d)} ts'_2, v'_2\}) \\
 & \cup (\Delta_3 \setminus \{ts, v \Rightarrow ts_3, v_3, ts_3, v_3 \xrightarrow{wd \ r(d)} ts'_3, v'_3\}) \\
 & \cup (\Delta_4 \setminus \{ts, v \Rightarrow ts_4, v_4, ts_4, v_4 \xrightarrow{wd \ c(d)} ts'_4, v'_4\}) \\
 & \cup (\Delta_5 \setminus \{ts, v \Rightarrow ts_5, v_5, ts_5, v_5 \xrightarrow{t} ts'_5, v'_5\})
 \end{aligned}$$

We now show that $(\Gamma', \Delta') \models ts, v: \mathbf{G} \phi$ by proving that within the proof context (Γ', Δ') , all snapshots and views reachable from ts, v satisfy ϕ . For that, let σ be a snapshot valuation, λ a locality valuation and ν a valuation such that $\sigma, \lambda, \nu \models (\Gamma', \Delta')$. We proceed by induction on the length of transition sequences of the form $\sigma(ts), \lambda(v) \Rightarrow \mathcal{TS}, V$. The induction base is immediate, since $\sigma(ts), \lambda(v) \Rightarrow \sigma(ts), \lambda(v)$ has the length zero, and $(\Gamma, \Delta) \models ts, v: \phi$ by our assumptions. As the induction hypothesis, we assume that for all transition sequences $\sigma(ts), \lambda(v) \Rightarrow \sigma(ts'), \lambda(v')$ of length n , we have $(\Gamma', \Delta') \models ts', v': \phi$.

So suppose we have a transition sequence $\sigma(ts), \lambda(v) \Rightarrow \mathcal{TS}'', V''$ of length $n + 1$. That is, there are ts', v' and a sequence $\sigma(ts), \lambda(v) \Rightarrow \sigma(ts'), \lambda(v')$ of length n , such that one of the following holds:

1. $\sigma(ts'), \lambda(v') \xrightarrow{r(C)} \mathcal{TS}'', V''$
2. $\sigma(ts'), \lambda(v') \xrightarrow{c(C, m)} \mathcal{TS}'', V''$
3. $\sigma(ts'), \lambda(v') \xrightarrow{\text{wd } r(C, m)} \mathcal{TS}'', V''$
4. $\sigma(ts'), \lambda(v') \xrightarrow{\text{wd } c(C)} \mathcal{TS}'', V''$
5. $\sigma(ts'), \lambda(v') \xrightarrow{t} \mathcal{TS}'', V''$

for some C, m and t respectively.

We prove the first case, the other ones are similar. Assume $\sigma(ts'), \lambda(v') \xrightarrow{r(C)} \mathcal{TS}'', V''$ as well as $\sigma(ts'') = \mathcal{TS}'', \lambda(v'') = V''$ and $\nu(c) = C$. Then let

$$\begin{aligned} \Delta'_1 = & (\Delta_1 \setminus \{ts, v \Rightarrow ts_1, v_1, ts_1, v_1 \xrightarrow{r(d)} ts'_1, v'_1\}) \\ & \cup \{ts, v \Rightarrow ts', v', ts', v' \xrightarrow{r(c)} ts'', v''\} . \end{aligned}$$

By the induction hypothesis, we know that $(\Gamma', \Delta') \models ts', v': \phi$. So let

$$\Gamma'_1 = (\Gamma_1 \setminus \{ts_1, v_1: \phi\}) \cup \{ts', v': \phi\}.$$

Since the application conditions ensure that neither ts_1 nor v_1 is dependent on any other assumptions and since d does not occur free in any other assumption, we get by our assumptions that $(\Gamma'_1, V'_1) \models ts'', v'': \phi$, i.e. $\sigma, \lambda, \nu \models ts'', v'': \phi$. Furthermore, because σ, λ and ν were chosen arbitrarily, we get $(\Gamma', V') \models ts'', v'': \phi$. \square

Theorem 4.1. The calculus of labelled natural deduction for EMLSL is sound.

Proof. In the preceding sections, we have proven all rules of the calculus to be sound. \square

Since models of EMLSL are based on the real numbers, we cannot hope for a complete deduction system.

4.2.9 Derived Rules

In this section we collect useful derived rules. These rules serve different purposes. On the one hand, they show that several expected properties are derivable with the presented proof system. On the other hand, we will employ most of these rules in our case study in Chap. 7.

We first introduce some generally useful shorthands. The rule following is an adaptation of previous work [Lin07].

$$\frac{[ts, v: \ell = t] \text{ [hri}(t)] \text{ [vri}(t)] \quad \vdots \quad \frac{ts', v': \phi}{ts', v': \phi} \ell\text{E}}{ts', v': \phi} \ell\text{E}$$

The rule ℓE has the application condition that t may neither occur free in any assumption on which $ts', v': \phi$ depends except for $ts, v: \ell = t$, $\text{hri}(t)$ and $\text{vri}(t)$ nor may it occur free in $ts', v': \phi$ itself.

Lemma 4.9. The rule ℓE is a derived rule of the proof system for EMLSL.

Proof. We can replace each occurrence of ℓE with the following proof tree:

$$\frac{\frac{\frac{ts, v: \ell = \ell}{ts, v: \exists t \bullet \ell = t} \text{Ref}}{ts, v: \exists t \bullet \ell = t} \exists\text{I} \quad \frac{[ts, v: \ell = t] \text{ [hri}(t)] \text{ [vri}(t)] \quad \vdots \quad ts', v': \phi}{ts', v': \phi} \exists\text{E}}{ts', v': \phi} \exists\text{E}$$

The application condition is implied by the application of $\exists\text{E}$. \square

From now on, we will often omit the assertions of rigidity and chop-freeness due to brevity, since they are easily derivable. The next type of rules concern the removal of empty views.

$$\frac{ts, v_2: \phi \quad ts, v_1: \ell = 0 \quad v = v_1 \oplus v_2}{ts, v: \phi} \text{0E} \quad \frac{ts, v_1: \phi \quad ts, v_2: \ell = 0 \quad v = v_1 \oplus v_2}{ts, v: \phi} \text{0E}$$

$$\frac{ts, v_2: \phi \quad ts, v_1: \omega = 0 \quad v = v_1 \ominus v_2}{ts, v: \phi} \text{0E} \quad \frac{ts, v_1: \phi \quad ts, v_2: \omega = 0 \quad v = v_1 \ominus v_2}{ts, v: \phi} \text{0E}$$

Lemma 4.10. The rules 0E are derived rules of the proof system for EMLSL.

Proof. All proofs are structurally similar. Hence we only show the case where v_1 represents the empty view, and the view is chopped in horizontal direction.

We can replace each of these occurrences of 0E with the following proof tree:

$$\frac{\frac{\frac{\Pi}{\mathbb{E}v' \bullet v = v' \oplus v} V0 \quad \frac{ts, v': \ell = 0 \quad ts, v_1: \ell = 0 \quad ts, v_2: \phi \quad v = v_1 \oplus v_2 \quad [v = v' \oplus v]_3}{ts, v: \phi} HD}{ts, v: \phi} \mathbb{E}\text{E}_3}{ts, v: \phi} \ell\text{E}_4$$

where Π is the following proof tree.

$$\frac{\frac{\frac{\Pi_{=}}{ts, v: x + y = x} \quad \frac{\Pi_{>}}{ts, v: x + y > x}}{ts, v: \perp} \neg E \quad \frac{ts, v: \perp}{ts, v': \ell = 0} \perp E_1}{ts, v': \ell = 0} \ell E_2$$

The proof tree $\Pi_{=}$ is given as follows.

$$\frac{\frac{[ts, v: \ell = x]_4 \quad [ts, v': \ell = y]_2 \quad [v = v' \oplus v]_3}{ts, v: \ell = x + y} H + I \quad [ts, v: \ell = x]_4}{ts, v: x + y = x} \text{Subst}_{cf}$$

And finally $\Pi_{>}$ is given by:

$$\text{Subst}_{cf} \frac{\frac{[ts, v': \ell = y]_2 \quad [ts, v': \ell > 0]_1}{ts, v': y > 0}}{ts, v': x + y > x} RH \quad \frac{[v = v' \oplus v]_3}{ts, v: x + y > x}$$

In this proof we identify $\neg(\ell = 0)$ with $\ell > 0$ and also implicitly use the validity of $x + y > x \rightarrow \neg(x + y = x)$ as well as $x = x + 0$. \square

To give a flavor of how the spatial atoms and the discrete transitions interact, we derive a variant of the *reservation lemma*, which we proved informally in our previous work.

Lemma 4.11 (Reservation [Hil+11]). A reservation of a car c observed directly after c created it, was either already present or is due to a previously existing claim. I.e., assuming $ts, v \xrightarrow{r(d)} ts', v'$ and $c = d$, the formula $\Box_{r(c)} re(c) \leftrightarrow (re(c) \vee cl(c))$ holds. Formally, this is the following rule.

$$\frac{ts, v \xrightarrow{r(d)} ts', v' \quad ts, v: c = d}{ts, v: \Box_{r(c)} re(c) \leftrightarrow (re(c) \vee cl(c))}$$

Proof. The existence of the transition is of major importance for the elimination of the box modality in the proof using the backwards reasoning rule. We use two auxiliary derivations, which allow us to infer the existence of a reservation on the snapshot after taking a transition.

$$\Pi_S: \frac{[ts, v: re(c)]_1 \quad [ts, v \xrightarrow{r(d)} ts', v']_2 \quad ts, v: c = d}{ts', v': re(c)} \rightarrow_S$$

$$\Pi_A: \frac{[ts, v: cl(c)]_1 \quad [ts, v \xrightarrow{r(d)} ts', v']_2 \quad ts, v: c = d}{ts', v': re(c)} \rightarrow_A$$

Now we present the derivations to get the right hand side of the equivalence from the left hand side, and vice versa.

Π_{\rightarrow} :

$$\frac{\frac{\frac{\Pi_S}{ts', v': re(c)} \quad \frac{\Pi_A}{ts', v': re(c)} \quad [ts, v: re(c) \vee cl(c)]_3}{\vee E_1} \quad \frac{\frac{ts', v': re(c)}{ts, v: \Box_{r(d)} re(c)} \Box_{r(d)} I_2}{ts, v: \Box_{r(c)} re(c)} \text{Subst}_{ri}}{ts, v: \Box_{r(c)} re(c)} \text{Subst}_{ri}$$

Π_{\leftarrow} :

$$\frac{\text{Subst}_{ri} \frac{[ts, v: \Box_{r(c)} re(c)]_3 \quad ts, v: c = d}{\Box_{r(d)} E} \quad \frac{ts, v: \Box_{r(d)} re(c) \quad ts, v \xrightarrow{r(d)} ts', v'}{ts', v': re(c)} \quad \frac{ts, v \xrightarrow{r(d)} ts', v' \quad ts, v: c = d}{ts, v: re(c) \vee cl(c)} \leftarrow A}{ts, v: re(c) \vee cl(c)} \leftrightarrow I_3$$

We can use the rule $\leftrightarrow I$ to get the desired derivation.

$$\frac{\frac{\Pi_{\rightarrow}}{ts, v: \Box_{r(c)} re(c)} \quad \frac{\Pi_{\leftarrow}}{ts, v: re(c) \vee cl(c)}}{ts, v: \Box_{r(c)} re(c) \leftrightarrow (re(c) \vee cl(c))} \leftrightarrow I_3$$

□

From now on, we will often omit the names of the rules used in the proofs. However, every application of a rule which eliminates some assumptions will be explicitly denoted, both by its name and with an index to refer to the eliminated assumptions.

As stated in Section 4.1, we use *free* to denote free space on a lane. To show that the abbreviation given is well-behaved, we show the soundness of the following rules.

$$\frac{ts, v: free}{ts, v: \ell > 0} \text{Extfree} \quad \frac{ts, v: free}{ts, v: \omega = 1} \text{Lanefree} \quad \frac{ts, v: free}{ts, v: free \wedge free} \text{densefree}$$

Lemma 4.12. The rules concerning the spatial properties of *free* are sound.

Proof. The rules *Extfree* and *Lanefree* are a simple application of $\wedge E$. The derivation of *densefree* is more involved.

The root of the proof tree is a derivation of the following shape.

$$\frac{\frac{ts, v: free}{ts, v: \ell > 0} \quad \frac{\Pi_1}{ts, v_1: free} \quad \frac{\Pi_2}{ts, v_2: free} \quad [v = v_1 \oplus v_2]_1}{\frac{ts, v: \ell > 0 \wedge \ell > 0}{ts, v: free \wedge free} \wedge E_1}$$

Now we have to show how to derive $ts, v_i: free$ from $ts, v_1: \ell > 0$, $ts, v_2: \ell > 0$ and $v = v_1 \oplus v_2$ for $i = 1$ and $i = 2$. Since both proof trees are structurally similar, we only show the tree for $i = 1$. The main idea is to assume $ts, v_1: \neg free$, which we identify with $ts, v_1: \ell = 0 \vee \neg(\omega = 1) \vee \exists c: ((\top \wedge (re(c) \vee cl(c))) \wedge \top)$ and derive a contradiction. Furthermore, we use the expanded form of $\Box_{\ell} \phi$, i.e. $\neg((\top \wedge \neg \phi) \wedge \top)$ and, to enhance the readability of the proof, we merge two applications of $\vee E$ into one ternary application. So, Π_1 will essentially be

$$\frac{[ts, v_1: \neg free]_2 \quad \frac{[ts, v_1: \ell > 0]_1 \quad [ts, v_1: \ell = 0]_3}{ts, v_1: \perp} \quad \frac{\Pi_1^\omega \quad \Pi_1^\perp}{ts, v_1: \perp} \vee E_3}{\frac{ts, v_1: \perp}{ts, v_1: free} \perp E_2} \vee E_3$$

where Π_1^ω is given as follows.

$$\frac{\frac{ts, v: free}{ts, v: \omega = 1} \quad [v = v_1 \oplus v_2]_1}{ts, v_1: \omega = 1} \quad [ts, v_1: \neg(\omega = 1)]_3}{ts, v_1: \perp}$$

The main part lies in the proof tree Π_1^\perp , which we still have to split.

$$\frac{[ts, v_1: \exists c \bullet (\top \wedge (re(c) \vee cl(c))) \wedge \top]_3 \quad \frac{[ts, v_1: (\top \wedge (re(c) \vee cl(c))) \wedge \top]_4 \quad ts, v_1: \perp}{ts, v_1: \perp} \Pi_1^\wedge}{ts, v_1: \perp} \exists E_4}{ts, v_1: \perp} \wedge E_5$$

The proof tree Π_1^\wedge is an intermediate step to relate the assumption that a reservation or a claim exists in the situation on the left view v_1 with the major premiss that no reservation or claim exists on the whole of v .

$$\frac{\frac{[v = v_1 \oplus v_2]_1 \quad [v_1 = v_L \oplus v_R]_5}{\mathbb{E}v' \bullet v = v_L \oplus v'}{ts, v: \perp} \quad \frac{\Pi_1^v}{ts, v: \perp} \mathbb{E}E_6}{ts, v_1: \perp}$$

The comparison hinted at in the description of Π_1^\wedge is made explicit in Π_1^v . On the left hand side of the proof tree, we join the parts of the views we got in Π_1^\wedge to derive, that a reservation or claim of c exists on the whole view v , which contradicts the assumption that v satisfies *free*.

$$\frac{[ts, v_L: \top \wedge (re(c) \vee cl(c))]_5 \quad \overline{ts, v': \top} \quad [v = v_L \oplus v']_6}{ts, v: (\top \wedge (re(c) \vee cl(c))) \wedge \top} \quad \frac{ts, v: free}{ts, v: \forall c \bullet \neg((\top \wedge (re(c) \vee cl(c))) \wedge \top)} \quad \frac{ts, v: \forall c \bullet \neg((\top \wedge (re(c) \vee cl(c))) \wedge \top)}{ts, v: \neg((\top \wedge (re(c) \vee cl(c))) \wedge \top)}}{ts, v: \perp}$$

As stated above, the proof tree for the situation at v_2 is structurally similar. Finally, we have created a well-structured proof tree, where all application conditions are satisfied, and the only open assumption is $ts, v: free$. \square

We can also derive rules stating the distributivity of chop over disjunctions.

$$\frac{ts, v: (\varphi \vee \psi) \wedge \chi}{ts, v: (\varphi \wedge \chi) \vee (\psi \wedge \chi)} \text{Distr} \wedge \quad \frac{ts, v: (\varphi \wedge \chi) \vee (\psi \wedge \chi)}{ts, v: (\varphi \vee \psi) \wedge \chi} \text{Distr} \wedge$$

Lemma 4.13. The rules $\text{Distr} \wedge$ are derived rules.

Proof. The proof for the left-hand rule is given by the following derivation:

$$\frac{
 \frac{
 \frac{
 \frac{
 \frac{
 [ts, v_1 : \varphi]_1 \quad [ts, v_1 : \psi]_1
 }{[ts, v_2 : \chi]_2}
 }{[v = v_1 \oplus v_2]_2}
 }{ts, v : \varphi \wedge \chi}
 }{ts, v : \psi \wedge \chi}
 }{ts, v : \varphi \wedge \chi \vee \psi \wedge \chi}
 }{[ts, v_1 : (\varphi \vee \psi)]_2}
 }{ts, v : (\varphi \vee \psi) \wedge \chi}
 }{ts, v : \varphi \wedge \chi \vee \psi \wedge \chi}
 }{\wedge E_2}$$

The right-hand rule is derived similarly. The end of the proof tree is given as:

$$\frac{
 \frac{
 \frac{
 \frac{
 \Pi_\varphi \quad \Pi_\psi
 }{ts, v : (\varphi \wedge \chi) \vee (\psi \wedge \chi)}
 }{ts, v : (\varphi \vee \psi) \wedge \chi}
 }{ts, v : (\varphi \vee \psi) \wedge \chi}
 }{ts, v : (\varphi \vee \psi) \wedge \chi}
 }{\vee E_1}$$

The proof tree Π_φ is a simple application of $\forall I$ and $\wedge E$.

$$\frac{
 \frac{
 \frac{
 [ts, v_1 : \varphi]_2
 }{ts, v_1 : \varphi \vee \psi}
 }{[ts, v_2 : \chi]_2}
 }{[v = v_1 \oplus v_2]_2}
 }{ts, v : (\varphi \vee \psi) \wedge \chi}
 }{ts, v : (\varphi \vee \psi) \wedge \chi}
 }{\wedge E_2}$$

The proof tree Π_ψ is structurally similar, where $ts, v_1 : \psi$ is substituted for $ts, v_1 : \varphi$. \square

Even though the somewhere modality is only defined as an abbreviation, we present explicit elimination and introduction rules. These are useful shorthands, since in most practical cases this modality plays an important role (see for example the case study in Chap. 7).

$$\frac{
 \frac{
 \frac{
 [v = v_L \oplus v_m] \quad [v_m = v'_m \oplus v_R] \quad [v'_m = v_D \ominus v''_m] \quad [v''_m = v^{sub} \ominus v_U] \quad [ts, v^{sub} : \varphi]
 }{\vdots}
 }{ts, v : \langle \varphi \rangle}
 }{ts', v' : \chi}
 }{\langle \rangle E}
 }{
 \frac{
 v = v_L \oplus v_m \quad v_m = v'_m \oplus v_R \quad v'_m = v_D \ominus v''_m \quad v''_m = v^{sub} \ominus v_U \quad ts, v^{sub} : \varphi
 }{ts, v : \langle \varphi \rangle}
 }{\langle \rangle I}$$

In the rule $\langle \rangle E$, the intermediate views $v_L, v_R, v_D, v_U, v_m, v'_m, v''_m$, and v^{sub} may not occur in any assumption $ts', v' : \chi$ depends on, except for the assumptions eliminated by the application of this rule.

Lemma 4.14. The somewhere introduction and elimination rules are derived rules.

Proof. The derivations are straightforward applications of the chop introduction and elimination rules, respectively. \square

4.3 Undecidability of Spatial MLSL

In this section we give an undecidability result for the spatial fragment of EMLSL, i.e., we do not need the modalities for the discrete state changes of the model or the evolutions. We will call this fragment *spatial MLSL*, subsequently. We reduce the halting problem of *two-counter machines*, which is known to be undecidable [Min67], to satisfaction of spatial MLSL formulas.

Intuitively, a two-counter machine executes a branching program which manipulates a (control) state and increments and decrements two different counters c_1 and c_2 . Formally, two counter machines consist of a set of states $Q = \{q_0, \dots, q_m\}$, distinguished initial and final states $q_0, q_{fin} \in Q$ and a set of instructions I of the form shown in Tab. 4.1 (the instructions for the counter c_2 are analogous). The instructions mutate configurations of the form $s = (q_i, c_1, c_2)$, where $q_i \in Q$ and $c_1, c_2 \in \mathbb{N}$ into new configurations:

Table 4.1: Instructions for Counter c_1 of a Two-Counter Machine

s	Instruction	s'
(q, c_1, c_2)	$q \xrightarrow{c_1^+} q_j$	$(q_j, c_1 + 1, c_2)$
$(q, 0, c_2)$	$q \xrightarrow{c_1^-} q_j, q_n$	$(q_j, 0, c_2)$
$(q, c + 1, c_2)$	$q \xrightarrow{c_1^-} q_j, q_n$	(q_n, c, c_2)

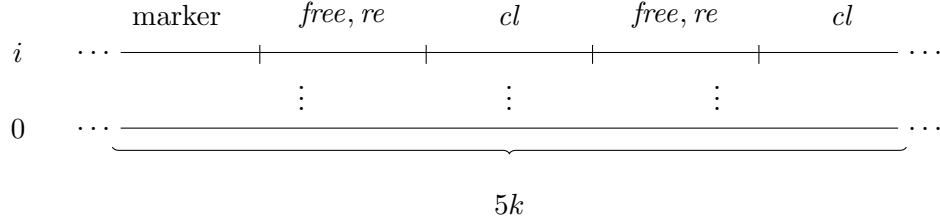
A run from the initial configuration of a two-counter machine $C = (Q, q_0, q_{fin}, I)$ is a sequence of configurations $(q_0, 0, 0) \xrightarrow{i_0} \dots \xrightarrow{i_p} (q_{p+1}, c_{p+1}, c'_{p+1})$, where each i_j is an instance of an instruction within I . If $q_{p+1} = q_{fin}$, the run is *halting*.

We follow the approach of Zhou et al. [ZHS93] for DC. They encode the configurations in recurring patterns of length $4k$ (for $k \in \mathbb{R}_+$), where the first part constitutes the current state, followed by the contents of the first counter. The third part is filled with a marker to distinguish the counters, and is finally followed by the contents of the second counter. Each of these parts is exactly of length k .

Zhou et al. could use distinct observables for the state of the machine, counters and separating delimiters, since DC allows for the definition of arbitrary many observable variables. We have to modify this encoding since within spatial MLSL we are restricted to two predicates for reservations and claims, and the derived predicate for free space, respectively. Furthermore, due to the constraints on EMLSL models in Def. 3.1, we cannot use multiple occurrences of reservations of a unique car to stand, e.g., for the values of one counter. Hence we have to existentially quantify all mentions of reservations and claims. We will never reach an upper limit of existing cars, since we assume \mathbb{I} to be countably infinite.

The current state of the machine q_i is encoded by the number of lanes below the current configuration, the state of each counter is described by a sequence of reservations, separated by free space in between. A single claim identifies the border between the counters. To safely refer to the start of a configuration, we use an additional marker

consisting of a claim, an adjacent reservation and again a claim. Each part of the configurations is assumed to have length k . Free space separates the reservations within one counter from each other and from the delimiters. Intuitively, a configuration is encoded as follows:



To enhance the readability of our encoding, we use the abbreviation

$$\text{marker} \equiv \exists c \bullet cl(c) \wedge \exists c \bullet re(c) \wedge \exists c \bullet cl(c)$$

to denote the start of a configuration.

Like Zhou et al., we ensure that reservations and claims are mutually exclusive.

$$\text{mutex} \equiv \forall c, d \bullet [cl(c) \rightarrow \neg re(d)] \wedge [re(c) \rightarrow \neg cl(d)].$$

We do not have to consider *free*, since it is already defined as the absence of both reservations and claims. Observe that we use the square brackets to denote the *everywhere* modality (cf. Sect. 4.1).

The initial marking $(q_0, 0, 0)$ is then defined by the following formula.

$$\text{init} \equiv \left(\begin{array}{c} [\neg \exists c \bullet cl(c)] \\ \text{marker}^k \wedge \text{free}^k \wedge (\exists c \bullet cl(c))^k \wedge \text{free}^k \wedge (\exists c \bullet cl(c))^k \\ \omega = 0 \end{array} \right) \wedge \top$$

We have to ensure that the configurations occur periodically after every $5k$ spatial units. Therefore, we use the following schema $Per(\mathcal{D})$. Observe that we only require that the lanes surrounding the formula \mathcal{D} do not contain claims. This ensures on the one hand that no configuration lies in parallel with the formula \mathcal{D} , since well-defined configurations have to include claims. On the other hand, it allows for the satisfiability of the formula, since we do not forbid the occurrence of reservations. These are needed for the claims within the configurations, due to the fact that each claim has to be adjacent to a reservation.

$$Per(\mathcal{D}) \equiv \left[\left(\begin{array}{c} [\neg \exists c \bullet cl(c)] \\ \mathcal{D} \\ [\neg \exists c \bullet cl(c)] \end{array} \wedge \ell = 5k \right) \rightarrow \left(\ell = 5k \wedge \begin{array}{c} [\neg \exists c \bullet cl(c)] \\ \mathcal{D} \\ [\neg \exists c \bullet cl(c)] \end{array} \right) \right]$$

Note that we did not constrain on which lane the periodic behaviour occurs. This will be defined by the encoding of the operations.

Now we may define the periodicity of the delimiters and the counters. Here we also have to slightly deviate from Zhou et al.: we are not able to express the statement “almost everywhere *free* or *re(c)* holds” directly. We have to encode it by ensuring that on every subinterval with a length greater than zero, we can find another subinterval which satisfies *free* or *re(c)*. This expresses in particular, that no claim may occur, due to the mutual exclusion property.

$$\begin{aligned} \text{periodic} \equiv & Per((\Box_\ell(\ell > 0 \rightarrow \top \wedge (\text{free} \vee \exists c \bullet \text{re}(c)) \wedge \top) \wedge \omega = 1)^k) \\ & \wedge Per((\exists c \bullet \text{cl}(c))^k) \wedge Per(\text{marker}^k) \end{aligned}$$

We turn to the encoding of the operation $q_i \xrightarrow{c_1^+} q_j$, i.e., the machine goes from q_i to q_j and increments the first counter by one (the other operations can be defined in an analogous manner). Similar to Zhou et al., we use encodings of the form $\neg(\mathcal{D}_1 \wedge \neg\mathcal{D}_2)$, meaning “whenever the beginning of the view satisfies \mathcal{D}_1 , the next part satisfies \mathcal{D}_2 .”

The formula following F_1 copies the reservations of counter one of state q_i to the corresponding places in counter one in state q_j .

$$\begin{aligned} F_1 \equiv & \neg \left(\left(\begin{array}{c} \top \\ \text{marker}^k \wedge \ell < k \wedge \exists c \bullet \text{re}(c) \wedge ((\exists c \bullet \text{re}(c)) \wedge \top) \wedge \ell = 5k \\ \omega = i \end{array} \right) \wedge \right. \\ & \left. \neg \left(\begin{array}{c} \top \\ \ell = 0 \vee (\exists c \bullet \text{re}(c)) \wedge \top \\ \omega = j \end{array} \right) \right) \end{aligned}$$

We use a similar formula F_{free} to copy the free space before the reservations. Observe that we do not copy the last free part of the counter, since we intend to add another reservation. Due to space limitations, we use the following abbreviation to identify the occurrences of free space in front of reservations:

$$\text{free}_{re} = ((\text{free} \wedge \top) \wedge \neg(\text{free} \wedge \exists c \bullet \text{cl}(c) \wedge \top) \wedge \ell = 5k) .$$

On the one hand, the formula ensures that we find an occurrence of free space at the beginning of the current interval. On the other hand, it prohibits this occurrence to be the last free space at the end of the counter.

$$F_{free} = \neg \left(\left(\begin{array}{c} \top \\ \text{marker}^k \wedge \ell < k \wedge \text{free} \wedge \text{free}_{re} \\ \omega = i \end{array} \right) \wedge \neg \left(\begin{array}{c} \top \\ \ell = 0 \vee (\text{free} \wedge \top) \\ \omega = j \end{array} \right) \right)$$

The formulas F_2 and F_3 handle the addition of another reservation to the counter. We have to distinguish between an empty counter and one already containing reservations.

$$F_2 \equiv \left(\begin{array}{c} \top \\ \text{marker}^k \wedge \text{free}^k \wedge \ell = 5k \\ \omega = i \end{array} \right) \rightarrow \left(\begin{array}{c} \top \\ \top \wedge (\text{free} \wedge \exists c \bullet \text{re}(c) \wedge \text{free})^k \\ \omega = j \end{array} \right)$$

$$F_3 \equiv \left(\begin{array}{c} \text{marker}^k \wedge \ell < k \wedge \exists c \bullet \text{re}(c) \wedge \left(\begin{array}{c} \top \\ (free \wedge \exists c \bullet cl(c) \wedge \top) \wedge \ell = 6k \end{array} \right) \wedge \omega = i \end{array} \right) \rightarrow$$

$$\left(\begin{array}{c} \top \\ \top \wedge (free \wedge \exists c \bullet \text{re}(c) \wedge free \wedge \exists c \bullet cl(c))^k \wedge \omega = j \end{array} \right)$$

In addition, we need formulas which copy of contents of the second counter to the new configuration, similar to F_1 .

Let I_C be the set of the machine's instructions and $F(i)$ be the conjunction of the formulas encoding operation i and q_{fin} its final state. Then

$$\text{halt}(C) \equiv \text{init} \wedge \text{periodic} \wedge \text{mutex} \wedge \bigwedge_{i \in I_C} \square_{\ell} F(i) \wedge \diamond_{\ell} \left(\begin{array}{c} \top \\ \exists c \bullet cl(c) \wedge \omega = \text{fin} \end{array} \right).$$

If and only if $\text{halt}(C)$ is satisfiable, the machine contains a halting run. This holds since only configurations may contain claims (as defined in the formalisation of periodicity), and whenever the machine reaches its final state, it halts. Hence the halting problem of two counter machines with empty initial configuration reduces to satisfiability of spatial MLSL formulas.

Proposition 4.1. Let C be a two-counter machine. Then C has a halting run if and only if $\text{halt}(C)$ is satisfiable.

Proof. “if”.

Let $\mathcal{TS}, V, \nu \models \text{halt}(C)$, where $V = (L, X, E)$. Observe that all variables occurring in $\text{halt}(C)$ are existentially quantified, and hence we may ignore the values of ν . We divide X into parts of length $5k$, i.e., we have $\|X\| = s \cdot 5k + r$, where $0 \leq r < 5k$, which means

$$X = [a, b] = \bigcup_{d=1}^s [a + (d-1) \cdot 5k, a + d \cdot 5k] \cup [a + s \cdot 5k, b].$$

We denote $\bigcup_{d=1}^c [a + (d-1) \cdot 5k, a + d \cdot 5k]$ by X_e . Let $X' = [a + (d'-1) \cdot 5k, a + d' \cdot 5k]$ and $X'' = [a + d' \cdot 5k, a + (d'+1) \cdot 5k]$ for some $0 < d' < s$. Now assume that at X' , lane m contains a configuration, i.e.,

$$\begin{aligned} \mathcal{TS}, V_{X'}^{\{m\}} \models & \text{marker}^k \wedge (\square_{\ell}(\ell > 0 \rightarrow \top \wedge (free \vee \exists c \bullet \text{re}(c)) \wedge \top) \wedge \omega = 1)^k \\ & \wedge \exists c \bullet cl(c)^k \wedge (\square_{\ell}(\ell > 0 \rightarrow \top \wedge (free \vee \exists c \bullet \text{re}(c)) \wedge \top) \wedge \omega = 1)^k \\ & \wedge \exists c \bullet cl(c)^k \end{aligned}$$

By interpreting periodic on $\mathcal{TS}, V_{X' \cup X''}$ we get that there is a lane m' such that

$$\begin{aligned} \mathcal{TS}, V_{X''}^{\{m'\}} \models & \text{marker}^k \wedge (\Box_{\ell}(\ell > 0 \rightarrow \top \wedge (\text{free} \vee \exists c \bullet \text{re}(c)) \wedge \top) \wedge \omega = 1)^k \\ & \wedge \exists c \bullet \text{cl}(c)^k \wedge (\Box_{\ell}(\ell > 0 \rightarrow \top \wedge (\text{free} \vee \exists c \bullet \text{re}(c)) \wedge \top) \wedge \omega = 1)^k \\ & \wedge \exists c \bullet \text{cl}(c)^k \end{aligned}$$

Furthermore, the formula periodic prevents that there exists a lane different from m' containing such a situation, since for it to hold, all other lanes are forbidden to contain claims at X'' . Hence we have exactly one configuration on all parts $[a + (d-1) \cdot 5k, a + d \cdot 5k]$.

We can extract a run for C from \mathcal{TS}, V from $\text{halt}(C)$ by induction on d as follows.

Let $d = 1$. Then init ensures that on lane 0, there is a configuration with no reservations between the marker and the first claim and between the first and the second claim. Hence, we have a run starting and ending with $(q_0, 0, 0)$.

As the induction hypothesis, we assume that for $1 \leq d < s$, we can extract a run $R = (q_0, 0, 0) \rightarrow^* (q_i, c_1, c_2)$ from \mathcal{TS}, V_{X_d} . For $d + 1$, we know by the arguments above, that there exists exactly one configuration on $[a + d \cdot 5k, a + (d + 1) \cdot 5k]$. Since C is deterministic, for the configuration on lane i , there is at most one set of formulas applicable. We only show the case for instruction incrementing counter one.

Let $F_1, F_2, F_3, F_{\text{free}}$ be the applicable formulas, which we will interpret on $X_{d+1} \setminus X_{d-1}$, i.e. the interval $X_+ = [a + (d-1) \cdot 5k, a + (d+1) \cdot 5k]$. This interval is exactly $10k$ long and starts with marker^k on lane i . Then F_1 states that for each reservation in the representation of the first counter, i.e., where $\ell < k \wedge \exists c \bullet \text{re}(c)$ holds, we find a reservation on lane j exactly $5k$ space units onwards. The outermost negation ensures that each possible chop point is considered, in particular the chop points arbitrarily close to the end points of the reservations. F_{free} ensures in a similar way, that for each free space in front of a reservation in this representation, we have free space exactly $5k$ space units onwards on lane j . Hence, all reservations and the free space in between is present on lane j .

Now we consider two cases. When there is no reservation between the marker and the first single claim, then F_2 replaces this free space by a reservation enclosed by free space, i.e., the end configuration of the run was $(q_i, 0, c_2)$ and the resulting configuration is $(q_j, 1, c_2)$. The second counter was copied like the first.

If there was a reservation before the last free space, then F_3 replaces this last free space similarly by a reservation enclosed by free space on lane j , i.e., the configuration (q_i, c_1, c_2) is changed to $(q_j, c_1 + 1, c_2)$. Hence, in both cases we defined the increment of counter 1 together with a state change from q_i to q_j , which is by construction an instruction of C , hence $R \rightarrow (q_j, c_1, c_2)$ is a valid run of C . The other cases are analogous.

Now if we did extract a run from the satisfying model of $\text{halt}(C)$, we have two possibilities. First, if $r = 0$, then the configuration at step s is the last of R . Then the last conjunct of $\text{halt}(C)$ ensures, that a final state was reached, hence R is a halting run.

Otherwise, if $r > 0$, then similarly it is ensured that on this last part of V , the lane corresponding to the final state has been reached. Since also the last change has to be initiated by a formula as before, there is an instruction to complete R to a halting run.

“only if”.

Let $R = (q_0, 0, 0) \rightarrow^* (q_{fin}, c_1, c_2)$ be a halting run of C with $d + 1$ configurations, i.e. $q_d = q_{fin}$. We create a model \mathcal{TS}, V with $V = (L, X, E)$ with $|X| = (d + 1) \cdot 5k$ and $|L| = |Q| + 1$ as follows. For a configuration (q_i, c_1, c_2) at step d' , we define three cars $C_{d',0}, C_{d',1}, C_{d',2}$ with

$$\begin{aligned} \text{pos}(C_{d',e}) &= d' \cdot 5k + e \cdot k/3 && \text{for } e \in \{0, 1, 2\} \\ \text{res}(C_{d',0}) &= \text{res}(C_{d',2}) = \{i + 1\} \\ \text{res}(C_{d',1}) &= \{i\} \\ \text{clm}(C_{d',0}) &= \text{clm}(C_{d',2}) = \{i\} \\ \text{clm}(C_{d',1}) &= \emptyset \\ \Omega_E(C_{d',e}, \mathcal{TS}) &= k/3 && \text{for } e \in \{0, 1, 2\} \end{aligned}$$

These cars satisfy marker^k. For the claims marking the end of counter 1 and 2 respectively, we define $C_{d',4}$ and $C_{d',6}$ as follows.

$$\begin{aligned} \text{pos}(C_{d',4}) &= d' \cdot 5k + 2k \\ \text{pos}(C_{d',6}) &= d' \cdot 5k + 4k \\ \text{res}(C_{d',4}) &= \text{res}(C_{d',6}) = \{i + 1\} \\ \text{clm}(C_{d',4}) &= \text{clm}(C_{d',6}) = \{i\} \\ \Omega_E(C_{d',4}, \mathcal{TS}) &= \Omega_E(C_{d',6}, \mathcal{TS}) = k \end{aligned}$$

For the definition of the first counter, we need the maximum value max of both counters on the whole run. Then we define a sequence of cars $C_{d',3,x}$, where $1 \leq x \leq c_1$ if $c_1 > 0$. For each such car we set

$$\begin{aligned} \text{pos}(C_{d',3,x}) &= d' \cdot 5k + 3k + \left((2x + 1) \cdot \frac{k}{1 + 2 \cdot max} \right) \\ \text{res}(C_{d',3,x}) &= \{i\} \\ \text{clm}(C_{d',3,x}) &= \emptyset \\ \Omega_E(C_{d',3,x}) &= \frac{k}{1 + 2 \cdot max} \end{aligned}$$

Otherwise, no such sequence is added.

For the second counter, we define a similar sequence $C_{d',5,x}$ with $1 \leq x \leq c_2$ if $c_2 > 0$.

If we create such sets of cars for each configuration, the formula $\text{halt}(C)$ is satisfied, if the run is halting. \square

The main theorem of this section is a corollary of Prop. 4.1.

Theorem 4.2. The satisfiability problem of spatial MLSL is undecidable.

Even though we used the full power of spatial MLSL in the proof, i.e., we used both ℓ and ω , the proof would be possible without using the latter. For that, we would not be

able to encode the state of the configuration in the lanes, but by a similar way to the markers in the formulas. For example, the formula $(\exists c \bullet cl(c) \wedge \exists c \bullet re(c) \wedge \exists c \bullet cl(c))^k$ would denote the state q_0 , and with another iteration of $re(c)$, it would denote q_1 and so on. If we remove the references to more than one lane in each of the formulas above, the reservations and claims would already imply that only one lane exists, and hence, the use of ω within the abbreviation *free* could be omitted. This shows that spatial MLSL is already undecidable even if we only use ℓ .

4.4 Related Work

Within this chapter, we concentrated on spatial reasoning with a labelled transition system of metric spaces as semantics. Another variant of logics which are called “spatial” are logics for reasoning about structural properties of discrete systems, e.g. process algebras like the π -calculus [MPW92] or the ambient calculus [CG98], or as a more general approach graphs.

The spatial logic for concurrency [CC01] is a very expressive language to describe processes with dynamically changing structures. It contains several spatial modalities to refer to the structure of processes as well as behavioural modalities to reason about the possible messages a process may send to the environment and first- and second-order quantifiers. With this set of operators, fixpoint operators and a validity predicate can be defined within the logic. The sequent calculus for this logic [CC04] uses a set of constraints to store assumptions about possible fresh names, which is in a sense similar to the labels and relational formulas in our proof system.

Baldan et al. presented a logic for verification purposes of graph transformation systems [BKK03]. Their approach relies on monadic second-order quantification and explicit predicates referring to the labels of edges and their sources and targets. Even though the logic is defined to reason about arbitrary graph structures, the set of predicates is similarly to EMLSL very restricted.

Most related work on spatial logics in the sense of this chapter is focused on purely qualitative spatial reasoning [APB07], e.g., the expressible properties concern topological relations [RCC92]. Shao et al. presented a temporal extension of such a qualitative spatial logic to specify properties of hybrid systems [Sha+13]. They use an approach of Finger and Gabbay [FG92] to add topological spaces to a linear model of time. However, how the spaces evolve over time has to be explicitly defined within the systems they design. This contrasts our definition, where these evolutions are hidden in the model, and hence not directly accessible by the operators of EMLSL.

Logics expressing quantitative spatial properties are rare, an example is Schäfer’s Shape Calculus (SC) [Sch05], which is a very general extension of Duration Calculus. On the one hand the focus of EMLSL lies on a restricted field of application, i.e., freeway traffic, and hence EMLSL is more restricted than SC. On the other hand, a direct translation into SC or one of its decidable subset fails, since in SC quantification is only possible over variables used in length measurements. In EMLSL however, quantification over cars is allowed, which resembles quantification over spatial properties (e.g., the existence

of a reservation at a certain point within a view). This would have to be modelled by quantification on observables in SC. Hence, for identifying a subset of EMLSL where satisfaction is decidable, restricting the number of cars to be finite seems inevitable.

From a more general point of view, EMLSL is a multi-dimensional multi-modal first-order logic using three sorts of variables. Marx and Venema have further examined different properties of multi-dimensional modal logics [MV97]. For them, multi-dimensionality stems from the models themselves, i.e., a logic is multi-dimensional, if the models consist of tuples of elements taken from more basic sets. Our models incorporate many different types of elements, and hence are multi-dimensional in their sense. But they only consider homogeneous models, i.e., where all dimensions are similar, while our dimensions behave very differently. For example, the vertical dimension is discrete, while the horizontal dimension is dense. The temporal dimensions, i.e., the discrete and continuous transitions, again behave in very different ways. Furthermore, EMLSL consists of various different modal operators, which are not interdefinable. However, the modalities are strongly interconnected, e.g. the creation of a reservation only has an effect, if there was a preceding creation of a claim for the same car. Due to this interdependence of the accessibility relations, EMLSL is not simply a fusion [Gab+03] of the corresponding uni-modal languages.

In the definition of EMLSL, we chose to keep the domains of quantification constant on all reachable worlds. In a previous presentation [Hil+11], the quantifiers were evaluated on the set of visible cars within a view. However, for the contents within this chapter, constant domains ease the definition of the proof system and prohibit certain surprising special cases. For example, equality could not be guaranteed to be a rigid predicate in the case of varying domains, since the cars under consideration may be visible in one view, but not in its subview. Such subtle intricacies are typical for first-order modal logics with varying domains [FM98].

For cars, the existence predicate of free logic [Ben86] can be emulated, since existence in our setting corresponds to visibility to the owner of the current view. I.e., we can use the formula $\langle re(c) \vee cl(c) \rangle$ to relativise our quantors. For example instead of $\forall c \bullet \varphi$ we would use $\forall c \bullet \langle re(c) \vee cl(c) \rangle \rightarrow \varphi$ to mimic varying domains. More specifically, we can emulate *decreasing* domains with respect to the spatial accessibility relations.

Labelled natural deduction for (multi-)modal logics has been studied intensely recently. E.g., when the rules for relational formulas can be defined with horn clauses as antecedents, nice meta-theoretical properties like normalisation of proofs can be established [BMV98; Vig00]. In intuitionistic modal logic, similar results are obtained, when the relational theory is defined using only geometric sequents [Sim94]. Unfortunately, even with our restricted set of rules for view relations, these results do not carry over to our setting, since we made use of existential quantification on views. In particular, the rule $\mathbb{E}\mathbb{E}$ merges relational and logical deductions. The set of rules for dynamic transitions is small and only concerns the length of passing time and abstract transitions. Rules stating the possibility of transitions for, e.g., the creation of reservations, need to use spatial atoms as premisses (in this case the existence of a claim) and relational formulas as conclusions (the existence of the transition). Hence, proofs would integrate relational and logical deductions even more.

Rasga et al. investigated the fibring [CSS05] of labelled deductive systems [Ras+02]. They distinguish between two different types of fibrations. The first type of systems is generated by treating all operators as unique, i.e., the operators from the different systems can not interact directly. In particular, if the systems use Boolean operators, these operators are not immediately equivalent. In the second type of fibrations, operator types common to all systems may be merged to a single set of operators. E.g., such a combination is needed, if the fibration should only use one set of Boolean operators and quantors. We assume that the deduction system of Sec. 4.2 is a fibring of the second type, where the Boolean operators are shared between all deduction systems involved. A further classification of EMLSL (or a suitable subset) and its proof system within the framework of fibring and multi-dimensional logics would be of interest. If we can find subsets of the deductive systems which possess suited characteristics, e.g., the finite-model property, we may be able to use preservation results [CSS11]. However, already the definition of EMLSL within this framework is a laborious challenge.

5

Visual Logic for Freeway Traffic

Contents

5.1	Concrete Syntax	70
5.2	Conveniencs	76
5.3	Formalising Sanity	77
5.4	Abstract Syntax	80
5.4.1	Topological Sequences and Lane Sequences	86
5.4.2	Spatial Diagrams	87
5.4.3	Temporal Sequences	94
5.5	Semantics	97
5.6	Decidability of Spatial Traffic Diagrams	104
5.7	Related Work	107

During the work on safe lane change controllers, we often used visual depictions of traffic situations for analysis purposes. Naturally this prevalence of diagrams led to the question whether we could give these depictions a formal semantics and use them during proofs. In this chapter we present a formalisation of our intuitive visualisations. This diagrammatic language of *Traffic Diagrams* is loosely based on previous work on a visual language regarding space and time [Lin10].

The intent of Traffic Diagrams is to enable users to visually reason about spatial and temporal aspects of multi-lane traffic. Therefore, these diagrams are interpreted on the abstract road model of EMLSL. The diagrams shall denote constraints on the current model. That is, the elements shown in the diagram are ensured to be present, while properties not mentioned within a diagram may have arbitrary values. Hence the diagrams restrict the models in a similar fashion as EMLSL.

An example of a purely spatial traffic diagram is given in Fig. 5.1. The depicted spatial situation is contained in a dashed rectangle, called a *layer*. Two adjacent lanes are shown,

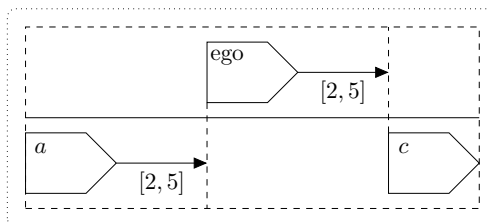


Figure 5.1: Spatial Traffic Diagram

where two cars a and c are driving on the lower lane, and ego is driving on the other. The parts of a lane not occupied by any car shown in the diagrams are denoting free space on the road. Constraints on the distances between the cars are shown as *distance arrows*, labelled with real-valued intervals $i \in \mathcal{I}$. This diagram may, e.g., be part of a very fine grained specification of a car platooning system. It describes a precondition for ego to join the platoon built by a and c by changing lanes. In particular, ego has to check whether the distance to the other two cars is at least two and at most five meters. The idea of the small distances between the cars is that the system may use the slipstream to lower the fuel consumption of the platoon on a global level.

Such spatial diagrams can be connected by *duration arrows* also labelled by intervals to create spatio-temporal *sequences*. A duration arrow constrains the time allowed to pass between the spatial layers, i.e., the freeway situations, it connects. We allow for negation and conjunction as well as existential quantification on the level of layers and of connected sequences of diagrams.

Within traffic diagrams, variables may be used in two ways. Depictions of cars may be labelled with variables, while the bounds of intervals are either variables or real numbers. The variables allow the user to connect different distances and cars along several parts of a diagram.

5.1 Concrete Syntax

A major difference between diagrammatic and standard sentential logics is the stronger distinction of diagrammatic syntax into two levels [How+02; Dau04b; Dau09], the *abstract* (or *type*) and *concrete* (or *token*) syntax. The former is typically a formal mathematical construct, describing the possible relationships between its elements, while the latter describes the representation of the abstract diagrammatic elements. Howse et al. use a mathematical definition of both syntactic levels and relate the syntactic levels by suited homomorphisms [How+02]. In contrast, Dau argues that the concrete syntax does not benefit from such a rigorous treatment [Dau04b; Dau09]. Instead, he emphasises that such a definition would lead to unnecessary technicalities. Furthermore, it would still not solve the problem of relating the mathematical concrete syntax with the drawings on, e.g., paper. For the presentation of traffic diagrams, we follow Dau by defining conventions for drawing the syntactic elements.


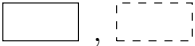

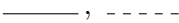
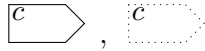
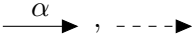

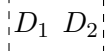


We employ a two-sorted set of variables Var , consisting of the sets of *car variables* CVar

and *length variables* RVar with typical elements $c, d \in \text{CVar}$ and $x, y \in \text{RVar}$, respectively. Similar to EMLSL, we use a constant ego , to denote the current local car. For temporal purposes, we use *real-valued* intervals, i.e., intervals, where both bounds are concrete real values. The other type of intervals are *variable intervals*, where we allow the bounds to be either real numbers or real-valued variables. In both types, ∞ is allowed as a right bound of an interval. Recall that we denote the set of real-valued intervals by \mathcal{I} and the set of variable intervals by \mathcal{I}_{Var} (cf. Sect 2.1). Furthermore, we denote the set of discrete *actions* of cars by

$$\mathcal{A} = \{r(c), c(c), \text{wd } r(c), \text{wd } c(c) \mid c \in \text{CVar}\}.$$

The basic elements of the concrete syntax of traffic diagrams are given in Tab. 5.1.

Table 5.1: Diagrammatic Elements of Traffic Diagrams ($\alpha \in \mathcal{I} \cup \mathcal{A}$ and $i \in \mathcal{I}_{\text{Var}}$)

Diagrammatic Element	Name
	Sequence
	Full/Partial Layer
	Unspecified Space
	Exact/Wide Lane Separation
	Reservation, Claim
	Temporal Arrow
	Distance Arrow
	Conjunction
	Negation
	Existential Quantification

These elements are arranged according to the following definitions. The spatial situation on single lanes is captured in topological sequences as given below. These sequences only describe the qualitative situation, i.e., which cars drive where in relation to other cars on the lane. A car is represented by a *reservation* or a *claim*, denoted by solid and dotted irregular pentagons, respectively.

Representations of cars may intersect with each other, and their horizontal positions may coincide. In the latter case, we shift the depiction of the cars against each other in vertical direction to emphasise the presence of more than one car. Cars have to be

labelled with variables taken from CVar. Unspecified space is depicted as a shaded part of a lane, while free space is just white space between cars and/or unspecified space.

Definition 5.1 (Topological Sequence). *A topological sequence consists of a horizontally arranged sequence of reservations, claims, and unspecified space, in the following manner. All elements have to be of roughly the same height. Reservations and claims may overlap. If they do, the user is free to shift single reservations and claims vertically, to enhance the visibility of the overlaps. All reservations and claims have to be labelled by exactly one variable $c \in \text{CVar}$ or ego. Adjacent reservations or claims have to be labelled differently. Unspecified space may not overlap with any reservation or claim. There may be blanks between the elements of the topological sequence. These blanks are free space.*



Figure 5.2: Example of a Topological Sequence

This definition implies that a topological sequence can be uniquely divided into *topological situations*, i.e., a possible overlap of reservations and claims, unspecified space or free space. This observation gives us the possibility to treat topological situations one at a time.

Example 5.1. *Consider the topological sequence in Fig. 5.2. It consists of seven different topological situations (from left to right):*

1. *the reservation of the car named c,*
2. *the unspecified space,*
3. *the claim of the car named d,*
4. *the free space between the claim of d and the reservation of ego,*
5. *the reservation of ego,*
6. *the overlap of the reservation of ego and the claim of b and*
7. *the claim of the car named b.*

Each of these situations as well as their order within the sequence can be uniquely derived from the depiction in Fig. 5.2.

Since topological sequences describe single lanes, and we want to describe multi-lane traffic, we have to be able to describe the spatial situations on adjacent lanes at once. We also want to have the possibility to omit irrelevant lanes from the specification. We therefore introduce *lane separations*, which have to be drawn between topological sequences. Solid separations denote that the lanes are adjacent, while dashed separations mean that there may be some omitted lanes between the depicted topological sequences.

Definition 5.2 (Lane Sequence). *A lane sequence consists of one or more topological sequences arranged vertically, which are separated by wide or exact lane separations.*

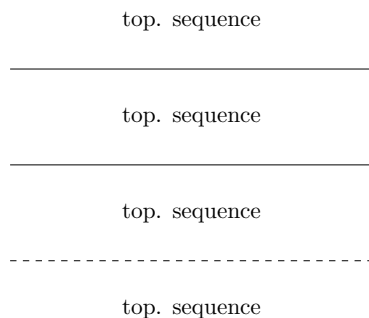


Figure 5.3: Example of a Lane Sequence

Example 5.2. *Consider the abstraction of a lane sequence shown in Fig. 5.3. It consists of four topological sequences, which we do not define any further, and lane separations in between. The lowest lane separation is wide, while both of the upper separations are exact. Note that the sequence does not have lane separations as its upper or lower boundary.*

Up to now, only qualitative aspects of space are describable with the given definitions. Since in the setting of multi-lane traffic quantitative constraints as “within two and three meters” may be of interest, we introduce methods to measure space between cars (both reservations and claims). This is done via *distance arrows*. The limitations given in the definition may seem arbitrary at the moment, but they allow for a non-ambiguous definition of abstract types of diagrams in Sect. 5.4.

Definition 5.3 (Distance Arrows). *An arrow between borders of reservations and/or claims is called a distance arrow. Such an arrow has to be drawn horizontally. Distance arrows may connect reservations and/or claims in different topological sequences. In this case, we require that the arrow is drawn within one of these sequences and the border of the reservation/claim in the other sequence is extended by an auxiliary vertical dashed line to meet the arrow. If the source and target of a distance arrow lie within the same topological sequence, the source must be to the left of the target. Distance arrows are always drawn solid and have to be labelled with an interval $i \in \mathcal{I}_{\text{Var}}$.*

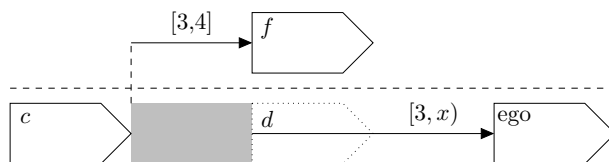


Figure 5.4: Example of Distance Arrows

Example 5.3. Figure 5.4 shows an exemplary lane sequence with two distance arrows. The upper distance arrow connects the right border of the reservation of the car c with the left border of f 's reservation. It is labelled with the closed interval $[3, 4]$. Note how the border of the reservation of c is extended by the dashed line.

The other arrow connects the claim of d with the reservation of ego . This arrow is attached to the left border of both its source and its target. Its label contains the variable x as the right border of the half-open interval $[3, x)$.

Now we have everything at hand to describe multi-lane traffic situations diagrammatically. For atomic situations, we need a lane sequence, possibly with distance arrows within the sequence, and surround it with a dashed or solid rectangle. The former denotes that only a part of the current situation on the road is described, while the latter enforces that the perceived part of the road is fully compatible with the diagram at hand. Furthermore, a shaded rectangle without a surrounding border denotes an arbitrary spatial situation. In addition, we allow for Boolean combinations of such descriptions and existential quantification on variables occurring in within topological situations or within labels of distance arrows.

Definition 5.4 (Spatial Diagram). *The language of spatial diagrams is given by the following inductive definition.*

- A shaded rectangle is an atomic spatial diagram.
- Let L be a lane sequence together with distance arrows within the sequence. Adding a solid (or dashed, respectively) rectangle around L yields an atomic spatial diagram, called a full (or partial, respectively) layer.
- Let S be a spatial diagram. Surrounding S with a hook on its left and top side results in the spatial diagram S' , the negation of S .
- Let S_1 and S_2 be spatial diagrams. Juxtaposing S_1 and S_2 and adding one dashed vertical line left of S_1 and one right of S_2 yields a spatial diagram, the conjunction of S_1 and S_2 .
- Let S be a spatial diagram. Adding a solid rectangle around S with the label $\exists x$ for a variable $x \in CVar \cup RVar$ on top yields a spatial diagram.

No other diagram is a spatial diagram.

Finally, to describe temporal aspects of traffic, we lift the given definition to sequences of spatial diagrams. The diagrams are connected by arrows defining the possible time elapsing or the different spatial transitions happening between the spatial situations. A solid arrow annotated with a spatial transition denotes that only this transition occurs between the specified snapshots. If the arrow is labelled with an interval, there has to be an evolution respecting the given interval between the snapshots. Recall that this means that no discrete transitions apart from changes in accelerations may happen. For this reason, solid temporal arrows will also be called *precise temporal arrows*. Dashed arrows,

however, may not carry a label. They allow for arbitrary transitions (i.e. an abstract transition) occurring between the snapshots, which is why they are called *faint temporal arrows*. If a solid arrow is annotated with the interval $[0, \infty)$, the label may be omitted. Again, we allow for logical connectives between sequences. This yields the language of Traffic Diagrams.

Definition 5.5 (Traffic Diagram). *The language of Traffic Diagrams is given by the following inductive definition.*

- *Each spatial diagram surrounded with a dotted rounded rectangle (a sequence) is an atomic Traffic Diagram.*
- *Let S be a spatial diagram and let T be a Traffic Diagram. Aligning S and T such that S is above T in vertical order and connecting S with T via an unlabelled dashed arrow or a solid arrow labelled with either a real-valued interval or an action and surrounding this whole sequence with a dotted rounded rectangle yields a temporal sequence. The arrow connecting S and T is called a temporal arrow. More specifically, a solid arrow is called a duration arrow if it is labelled with an interval and a discrete temporal arrow if it is labelled with an action.*
- *Let T be a Traffic Diagram. Surrounding T with a hook on its left and top side results in the Traffic Diagram T' , the negation of T .*
- *Let T_1 and T_2 be Traffic Diagrams. Juxtaposing T_1 and T_2 and adding a dashed vertical line left of T_1 and right of T_2 yields a Traffic Diagram, the conjunction of T_1 and T_2 .*
- *Let T be a Traffic Diagram. Adding a solid rectangle around T with the label $\exists x$ for a variable $x \in \text{CVar} \cup \text{RVar}$ on top results in the spatial diagram T' .*
- *No other diagram is a Traffic Diagram.*

We denote the set of all Traffic Diagrams by \mathfrak{D}

The notations for Boolean operators are inspired by the approach of Stapleton and Masthoff [SM07]. However, we chose to add two dashed vertical lines to denote conjunction to have a closer match between the abstract and concrete syntax. To express negation of an element, we use a line on top and on its left side. While Stapleton and Masthoff use only a line on top of diagrams, we use this extension to emphasise the difference between the negation of a single layer and a whole sequence including the connecting arrows. For an example, see Fig. 5.11. For existential quantification, we use the notation of visual first-order logic (VFOL) [Sta+05], i.e., the scope of the quantification is denoted by a rectangle and the quantor together with the quantified variable is inscribed on top of the scope.

5.2 Conveniences

In this section, we will remove redundancies and clutter from the formal diagrams, while still maintaining a mathematically distinct semantics.

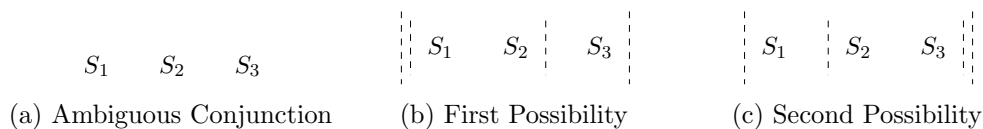


Figure 5.5: Omission of Notation for Conjunction

Typically, conjunction of diagrams would be denoted by juxtaposition, like, e.g. in visual first-order logic [Sta+05]. We decided to include a notation mimicking parenthesis to avoid ambiguities between abstract and concrete syntax, which would be created by omitting a distinguished syntax element for conjunction. In particular, conjunction is usually a binary operator, while juxtaposition in general is of arbitrary arity. Consider a diagram D consisting of three juxtaposed sequences S_i as in Fig. 5.5a.

This concrete diagram would have two representations in terms of Traffic Diagrams, as shown in Fig. 5.5b and 5.5c. Similar to rules for the omission of parenthesis for sentential logics, we assume that such ambiguities are resolved by the first possibility. That is, conjunction is left-associative. As we will see in Sect. 5.5, the semantics of the diagrams which are now drawn similarly are in fact the same.

The next simplification involves the notion of sequences. Within the concrete syntax, the dotted rounded rectangles denoting sequences often seem superfluous. For example, the sequence around an atomic Traffic Diagram is not needed to identify this case. However, the sequences are helpful to establish the existence of a unique representation for each Traffic Diagram in Sect. 5.4.

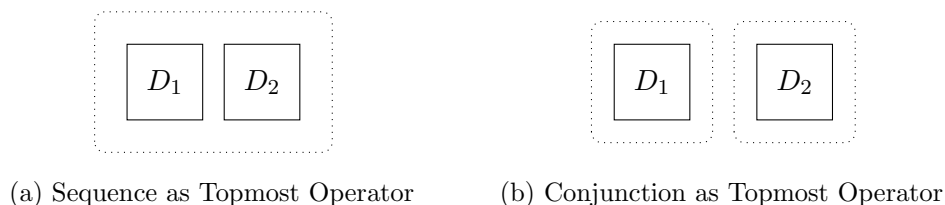


Figure 5.6: Ambiguities with Sequences

Consider the diagrams in Fig. 5.6a and Fig. 5.6b. Sequences themselves will not add any restrictions to the semantics, and hence both figures express exactly the same properties. We will omit the notations of sequences from the concrete depiction of Traffic Diagrams in the following sections and chapters. For the sake of well-definedness, however, we assume these diagrams to be defined as in Fig. 5.6a. Finally, we will often denote a point-like interval, i.e., an interval of the form $[x, x]$ by writing simply x . The context of this notation will always clarify, whether we mean the variable or such an interval.

5.3 Formalising Sanity

In this section, we formalise the sanity conditions of the abstract model with suitable Traffic Diagrams. We will present the informal semantics of the diagrams, so that the reader may get accustomed to the concrete syntax. Only afterwards, in Sect. 5.4 and 5.5, will we start to formalise both the abstract mathematical syntax, as well as the formal semantics of Traffic Diagrams.

These formalisations show typical forms of Traffic Diagrams. For example, all of these diagrams state invariance properties for all cars by negating the assertion of the existence of a car violating this property. Furthermore, some of the diagrams can be directly read as implications. Consider, e.g., the structure of Fig. 5.7e. Disregard the outer negation for the moment. Then, the diagram states that there is a situation, where the spatial diagram to the left holds, but the negated diagram to its right does not. That is, if we negate the whole diagram again, we get that there is no such situation. Hence, the left spatial diagram implies the diagram under the inner negation at all reachable traffic snapshots. These types of diagrams will again arise in Chap. 7, where we use the combination of EMLSL and Traffic Diagrams to define a specification for a safe lane-change controller. Furthermore, these diagrams state properties of the domain as a whole, thereby revealing these properties within the syntax of Traffic Diagrams. Similar formalisations within EMLSL have been found by Hilscher [Hil14].

The sanity conditions of Def. 3.2 are given for single snapshots only. For the axiomatic diagrams however, we use a notion corresponding to invariant properties along all possible transition sequences. That is, the correctness of these diagrams is a direct result of Lemma 3.1. In this sense, the diagrams are a stronger assertion than the sanity conditions, since they already incorporate their invariance along the transitions. However, the sanity conditions are defined for whole snapshots, whereas the diagrams may only state the conditions for the current view under consideration. Hence we can not use these diagrams to replace the sanity conditions.

The diagrams depicting the sanity conditions are given in Fig. 5.7. The first set of diagrams is relatively easy. The fact that the reservations and claims of a single car never overlap is shown in Fig. 5.7a. Fig. 5.7b and 5.7c define the maximal number of reservations and claims for a single car in a similar fashion. Note that the relative horizontal positions of the reservations (and claims) are not constrained, since the topological relations between cars on different lanes are unspecified. Furthermore, the wide lane separations allow for an arbitrary number of lanes in between the reservations (and claims). Also note that we allow for the non-existence of reservations for cars. On a first glance, this seems to contradict the sanity conditions 2 and 3, which explicitly require all cars to occupy at least one lane. However, since we interpret the diagrams on a view V , the reservations of cars outside of V are not visible. Because we interpret the quantifiers over *all* existing cars, whether or not they are visible in the current view, we have to allow for a car to not have any *visible* reservation.

The diagram depicted in Fig. 5.7d denotes that no claim may exist, whenever a car already reserves two lanes. Note that within the diagram, the layers have to be satisfied simultaneously and by the disjointness property of Fig. 5.7a, they may not coincide.

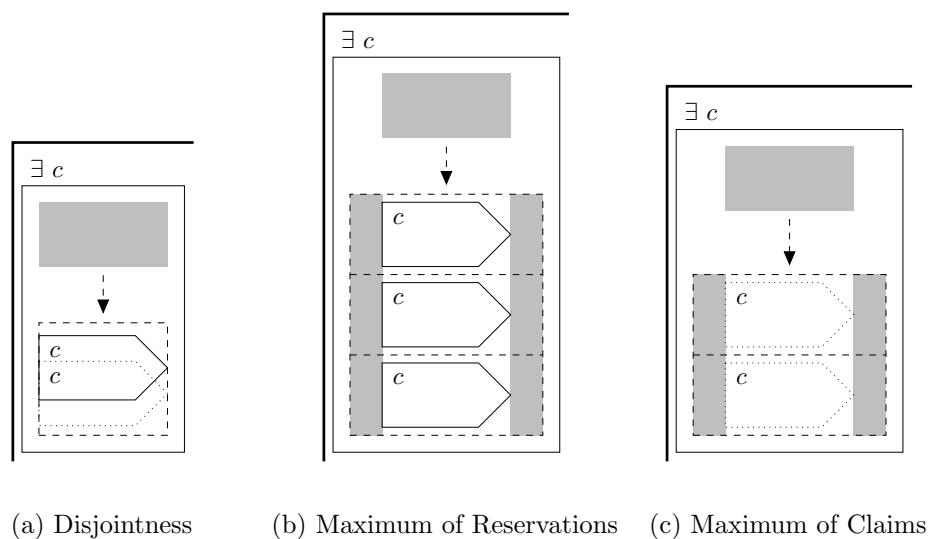
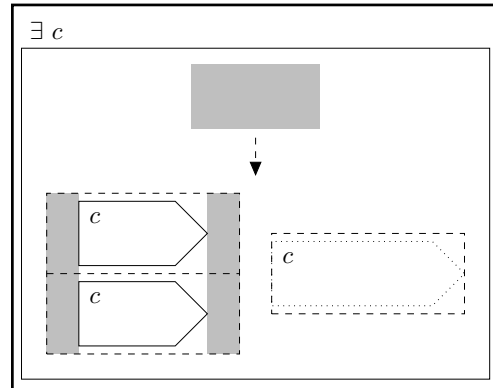


Figure 5.7: Sanity Conditions as Diagrams (I)

Hence a satisfying model has either less than three lanes, or at most one of the layers in the diagram can be satisfied. Together with the previous axiomatic diagrams, we get that either one reservation, one claim, or both one claim and exactly one reservation of a single car are visible in a view. However, outside of the current view, the car could possess an arbitrary number of claims and reservations.

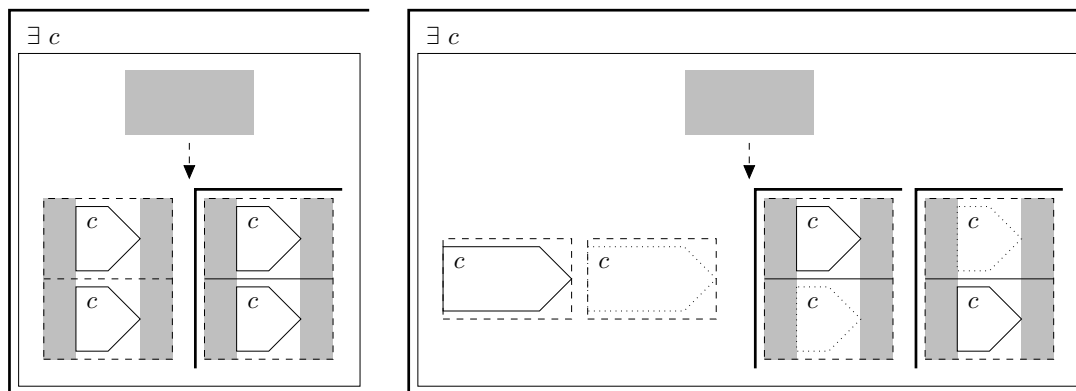
The diagrams in Fig. 5.7e and Fig. 5.7f denote that reservations and claims of a single car have to be on adjacent lanes. Both of these diagrams are essentially implications, where the parts under an odd number of negations state the premisses and the evenly diagrams give the conclusion. In Fig. 5.7e, observe that the mere existence of two reservations of the same car, as indicated by the wide lane separation and the unspecified space in front and in the back of the reservations, implies that these reservations occur on adjacent lanes, denoted by the exact lane separation. The horizontal positions of the reservations are not constrained in any way. The diagram defining the adjacency of claims and reservations is a bit more complicated, since we have to explicitly define the vertical possibilities for the claim. On the other hand, the existence of both the claim and the reservation is simpler by using two separate partial layers. Note that we can not specify Fig. 5.7e in a similar way, since two partial layers with a single reservation may refer to the same situation in the traffic snapshot.

In addition to the formalisation of the sanity conditions, we also define two properties which are only implicitly given by our model. First, we have to ensure that reservations and claims are connected. That is, for each car, there is at most one reservation on each lane. In the model, this is implied by the definition of len_V for a view V . For each car C , this function returns either the empty set, or exactly one interval giving the extension C occupies on the lanes it reserves. In the diagrams, we can state the existence of two reservations or claims of one car, where arbitrary space lies in between. Now the diagrams



(d) Maximum of Simultaneous Reservations and Claims

Figure 5.7: Sanity Conditions as Diagrams (II)



(e) Adjacency of Reservations

(f) Adjacency of Reservations and Claims

Figure 5.7: Sanity Conditions as Diagrams (III)

in Fig. 5.8 express that whenever we can find such a situation with the length x , we can also find a connected reservation or claim of this length. Since the quantifiers are negated, this holds in particular for the maximal possible value for x .

The last property of the models is partially formalised in Fig. 5.9. These diagrams state, that the start and end points of reservations on different lanes are the same. That is, the reservations lie directly next to each other and not shifted. Again we use distance arrows to relate the reservations on different lanes. This time, we do not only relate the lengths, but also restrict the start and endpoints of the reservations. Consider Fig. 5.9a. Whenever we find a reservation on the lower lane of length x , we can find a situation, where the reservation on the upper lane is at least of this length. In particular, if x is the maximal length of the lower reservation (which is one of the cases for which the diagram has to hold), then the upper reservation is of length $\geq x$ (since the reservation may also extend into the part denoted by the shaded rectangle). In conjunction with Fig. 5.9b, which states the inverse relationship between the reservations, we get that both

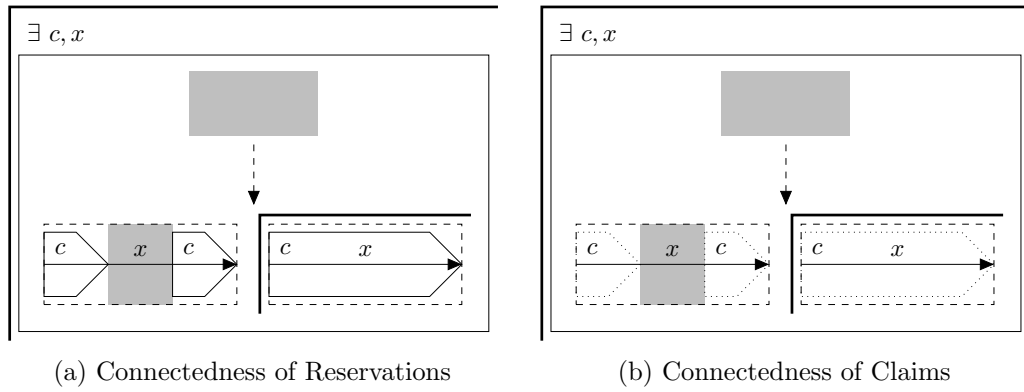


Figure 5.8: Connectedness of Cars

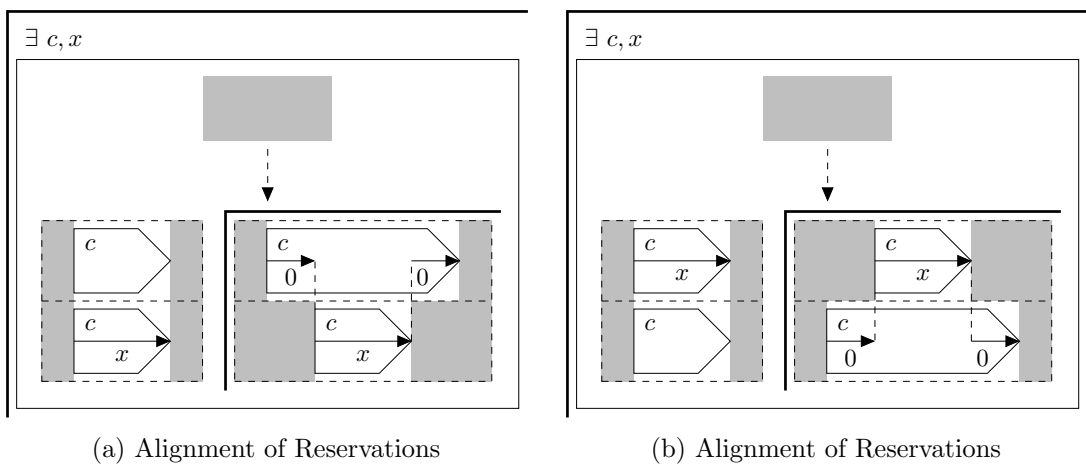


Figure 5.9: Alignment of Cars

reservations are of the same length, and have to start and end at the same points. For constraining the alignment of claims, we would have to draw a similar set of diagrams for the case when the claim resides below the reservation and vice versa. I.e., we need four diagrams, which are structurally similar to the ones presented in Fig. 5.9. We chose to omit these diagrams, since they are easily constructable from the given ones.

5.4 Abstract Syntax

Following the approach of Minas [Min00], we chose to define our abstract syntax in terms of graph rewriting systems, due to the generality of the approach and the relative simplicity of the generated graphs. In particular, the graph transformation system will create a hypergraph $G = (\mathcal{V}, \mathcal{E}, \tau, \theta, \iota)$ over a set of types T and of labels \mathbb{O} , which represents the concrete syntax of a diagram. Recall that τ assigns to each edge $e \in \mathcal{E}$ the sequence of nodes $\langle v_0 \dots v_n \rangle$ ($v_i \in \mathcal{V}$) e visits, where the length of $\tau(e)$ is determined by

$\theta(e)$, the type of e . Finally, recall that l denotes the labelling function of G .

Example 5.4. *To familiarise the reader with this notation and its connection with the visualisations of the graphs in this section, we present the formal hypergraph $G = (\mathcal{V}, \mathcal{E}, \tau, \theta, \mathsf{l})$ consisting of one of the cases shown in Tab. 5.2. Consider the edge representing an occupied topological situation. We call the vertices of the graph in this depiction v_1 to v_4 , i.e., $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$. The set of edges is only the singleton set $\mathcal{E} = \{e\}$. The type function associates the type `cars` with this edge, i.e., $\theta(e) = \text{cars}$. Furthermore $\tau(e) = \langle v_1, v_2, v_3, v_4 \rangle$. For the relation of the attachment function to the graph, we have to choose the order the mnemonic names occur in the sequence. Assume that s and t are the first two elements of the sequence, followed by l and r . That is, the s tentacle visits v_1 , the t tentacle v_2 , the l tentacle visits v_3 and v_4 is visited by the r tentacle of e . Finally, the labelling of e is given by $\mathsf{l}(e) = (R, C)$.*

Within the graph representing the syntax, edges either represent a diagrammatic element or the way an element is drawn, i.e., dashed or solid. The nodes serve as the attachment areas of these elements, e.g., an edge denoting an arrow is visiting nodes that are also visited by its source and target. Labels define the intervals arrows are annotated with, the labels on lanes, as well as which reservations or claims a car represents.

The set of types is distinguished into two disjoint sets of *terminal* T_T and *non-terminal* types T_{NT} . In the following, elements of the former set will be written in sans serif font, while non-terminal types will be capitalised.

The minimal requirement for a graph G to be eligible as a representation of a diagram, is that G contains only terminal edges. More specific structural properties for G will be given in Def. 5.7.

As shown in Tab. 5.2, we employ several terminal types to refer to the diagrammatic elements. On the one hand, we use `sequence`, `layer`, `true`, `lane`, `cars`, `free`, `totrue`, `tarrow` and `sarrow` to represent explicit elements within the concrete syntax. Despite the fact that duration and distance arrows are drawn alike, we distinguish both types on the level of the abstract syntax already by referring to duration arrows with the type `tarrow` and to distance arrows with `sarrow`. On the other hand, the types `partial` and `full` (faint and precise, wide and exact) define whether a layer (arrow, lane separation, respectively) is drawn dashed or solid. The Boolean operators are represented by the types `¬` and `∧`, while the existential quantifier is denoted by the type `∃`. The different types offer different tentacles. The p tentacle will be used to refer to the parent of the edge at hand. The tentacles named o (o_1 and o_2 , respectively) refer to the operands of the logical operators. Edge types with s and t tentacles, denoting the source and target of the edges, imply an order on the paths created by these edges of these types. That is, they either represent a lane sequence, a topological sequence or an arrow. The i tentacle of the types `layer` and `sequence` refers to the interior of the concrete elements the edges represent. The tentacles with the name `ty` are used to connect the edges representing layers or temporal arrows with their diagrammatic types, i.e., whether the layer is full or partial and whether the temporal arrow is precise or faint. The representation of distance arrows needs an auxiliary tentacle named c , which represents the layer which contains the arrows. This tentacle is especially useful for the definition of the semantics (cf. Sect. 5.5).

Table 5.2: Terminal Types of the Abstract Syntax
 $(\alpha \in \mathcal{I} \cup \mathcal{A}, i \in \mathcal{I}_{\text{Var}}$ and $R, C \subset \text{Var} \cup \{\text{ego}\})$

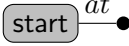
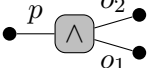
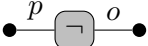
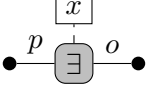
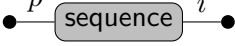
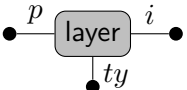
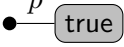
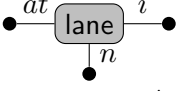
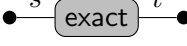
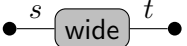
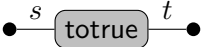
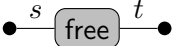
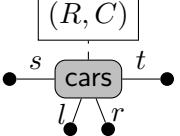
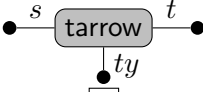
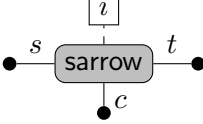
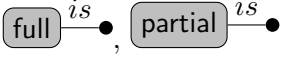
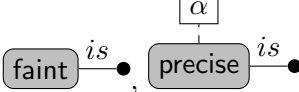
Terminal Type	Name
	Root of the Syntax Graph
	Conjunction
	Negation
	Existential Quantification
	Sequence
	Layer
	Unspecified Layer
	Lane
	Exact Lane Separation
	Wide Lane Separation
	Unspecified Topological Situation
	Free Topological Situation
	Occupied Topological Situation
	Temporal Arrow
	Distance Arrow
	Types for Full/Partial Layers
	Type for Faint/Precise Temporal Arrows

Figure 5.10 shows the abstract syntax of Fig. 5.1. The graph contains two edges of type `lane` representing both lanes of the layer. On each lane, there exists a sequence of edges of types `free` and `cars`, which model the spatial situations. Furthermore, two `sarrow` edges define the spatial constraints depicted as distance arrows in the concrete diagram.

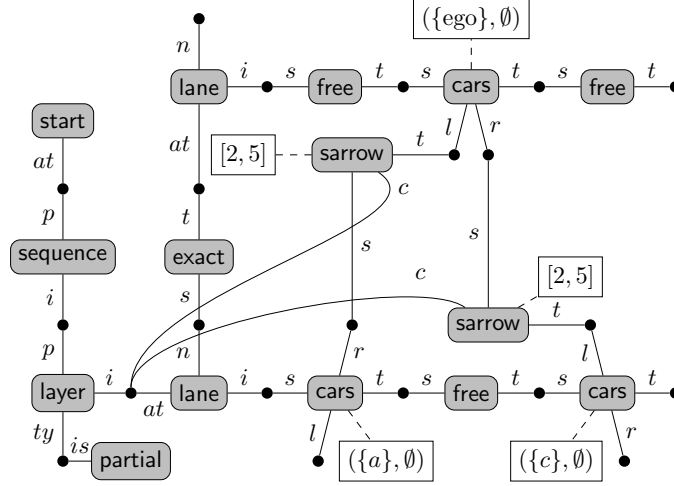


Figure 5.10: Abstract Syntax of Fig. 5.1

One detail of the abstract syntax is the representation of cars on the road. Instead of associating each car on a lane with a `cars` edge, we represent each part of a lane occupied by a different combination of cars on the road by a single edge with an appropriate label. Such a label consists of a tuple (R, C) of sets of variables. If c is a variable and $c \in R$ ($c \in C$), then the `cars` edge denotes that the reservation (claim) of c occupies the corresponding space. Hence, if c and d are elements of R , the edge denotes that the reservations of c and d coincide at this position. Hence only changes in the spatial configuration of the lane are reflected by the use of a new edge, either of type `cars`, `totrue` or `free`. For an example see Fig. 5.11.

Since distance arrows may connect two arbitrary cars on one layer, we must ensure that there exists a path from the arrow to both nodes visited by its source and target tentacles, which uses only edges of the following types: `lane`, `wide`, `exact`, `totrue`, `free` and `cars`. In particular, the path may not contain an edge of type `layer`, since otherwise the path would lead to representations of elements outside of the layer containing the arrow (see Fig. 5.10). Within Sect. 5.4.2, we will examine this requirement in detail.

Definition 5.6 (Abstract Syntax of Traffic Diagrams). *The graph transformation system $\mathcal{G} = (T_{NT} \uplus T_T, S, P)$ defining the abstract syntax of Traffic Diagrams consists of the axiom S , a single hyperedge without any tentacles and the rules given in Figures 5.12, 5.13, 5.15, 5.18, 5.19, 5.20, 5.21, 5.22 and 5.23.*

In the depictions of the rules, we will omit all tentacles which need not be taken into account for the definition of the rule. For example, we do not show the t tentacles of

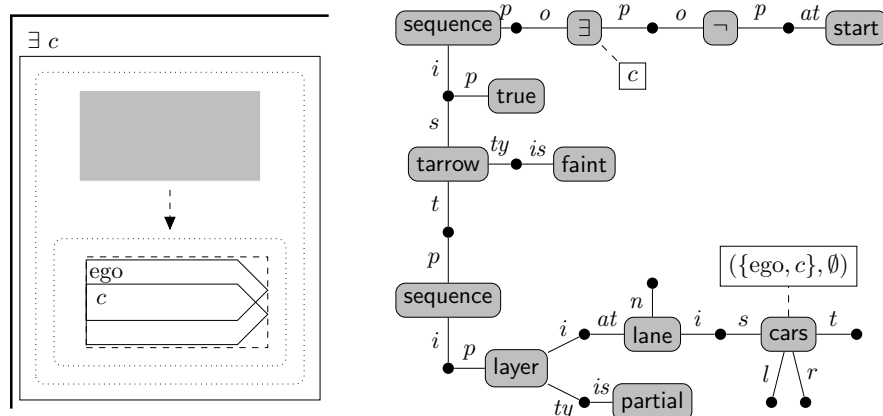


Figure 5.11: Concrete and Abstract Syntax of a Diagram with Duration Arrows

the cars edges in Fig. 5.18. The non-terminal types of \mathcal{G} will be described and presented during the discussion of the rules they are involved in.

To show that the rules of \mathcal{G} are sensible, we need to define, what it means for a Traffic Diagram to be represented by a hypergraph. Then we can show, that the hypergraphs created by \mathcal{G} are representations of Traffic Diagrams, and that for each Traffic Diagram D , there is a (up to isomorphism) unique hypergraph created by \mathcal{G} , which represents D .

Definition 5.7. A graph $G = (\mathcal{V}, \mathcal{E}, \tau, \theta, \iota)$ represents a Traffic Diagram D iff we can find an injective mapping h from the visual elements of D to G such that

1. each overlap of a set of reservations R and claims C is mapped to an edge e such that $\theta(e) = \text{cars}$ and $\iota(e) = (R, C)$,
2. each occurrence of blank space within a topological sequence is mapped to an edge with $\theta(e) = \text{free}$,
3. each occurrence of undefined space within a topological sequence is mapped to an edge with $\theta(e) = \text{totrue}$,
4. a situation s within a topological sequence is adjacent to the left of a topological situation s' if and only if the t tentacle of $h(s)$ and the s tentacle of $h(s')$ visit the same node,
5. the s tentacle of the image $h(s)$ of the left-most topological situation s of each topological sequence and the i tentacle of a unique edge e with $\theta(e) = \text{lane}$ visit the same node,
6. each exact (wide) lane separation is mapped to an edge e with $\theta(e) = \text{exact}$ ($\theta(e) = \text{wide}$),
7. if a topological sequence T_1 is drawn directly beneath a topological sequence T_2 and s is the lane separation drawn in between T_1 and T_2 , then the s tentacle of $h(s)$

visits the same node as the n tentacle of the lane edge connected to the image of the left-most topological situation in T_1 and the t tentacle of $h(s)$ visits the same node as the at tentacle of the lane edge connected to the image of the left-most topological situation in T_2 ,

8. each wholly unspecified space (i.e., a shaded rectangle without a surrounding layer) is mapped to an edge e with $\theta(e) = \mathbf{true}$,
9. each layer l is mapped to an edge e with $\theta(e) = \mathbf{layer}$ and if l is full (partial), the ty tentacle of e visits the same node as the is tentacle of an edge e' with $\theta(e') = \mathbf{full}$ ($\theta(e') = \mathbf{partial}$, respectively),
10. if T is the lowest lane sequence, then the edge e of type **lane** which is connected to the representation of T visits a node v via its at tentacle, which is also visited by the i tentacle of the edge representing the layer T resides in,
11. each spatial arrow A labelled with an interval i is mapped to an edge e with $\theta(e) = \mathbf{sarrow}$ and $l(e) = i$, where
 - a) the s tentacle of e visits the same node as the l of the image of the topological situation S where A starts, if A starts at the left border of S ,
 - b) the s tentacle of e visits the same node as the r of the image of the topological situation S where A starts, if A starts at the right border of S ,
 and similarly for the t tentacle of e and the end of A ,
12. if A is a faint temporal arrow, then h maps A to an edge e with $\theta(e) = \mathbf{tarrow}$, whose ty tentacle visits the same node as the is tentacle of an edge with type **faint**.
13. if A is a precise temporal arrow labelled with α , then h maps A to an edge e with $\theta(e) = \mathbf{tarrow}$, whose ty tentacle visits the same node as the is tentacle of an edge e' with $\theta(e') = \mathbf{precise}$ and $l(e) = \alpha$.
14. if N is symbol for negation denoting the negation of D' in D , then h maps N to an edge e with $\theta(e) = \mathbf{\neg}$ the o tentacle of e visits the same node as the p tentacle of the edge representing the topmost element in D' ,
15. if A are two vertical dashed lines denoting the conjunction of D_1 and D_2 in D , then h maps A to an edge e with $\theta(e) = \mathbf{\wedge}$ and the o_i tentacle of e visits the same node as the p tentacle of the edge representing the topmost element in D_i (for $1 \leq i \leq 2$),
16. if E is a box with the label $\exists x$ on top around D' , then h maps E to an edge e with $\theta(e) = \mathbf{\exists}$ and $l(e) = x$ and the o tentacle of e visits the same node as the p tentacle of the edge representing the topmost element in D' ,
17. if L is spatial diagram and D' a diagram such that the temporal arrow A starts at L and ends at D' , then the s tentacle of $h(A)$ visits the same node as the p tentacle of the edge representing the topmost element in L and similarly for the t tentacle of $h(A)$ and the p tentacle of the edge representing the topmost element in D' .

18. the p tentacle of the representation of the topmost element of the traffic diagram D visits the same node as the at tentacle of the unique edge of the type `start`.

By abuse of language, we will also say that, e.g., a topological sequence T is represented by a graph G if the conditions only concerning topological sequences hold for T and G .

Recall Convention 2.1 for the naming of graph rewriting rules. According to this convention, the rule creating the representation of a wide lane separation in Fig. 5.12 will, e.g., be denoted by R_{SEP}^2 .

5.4.1 Topological Sequences and Lane Sequences

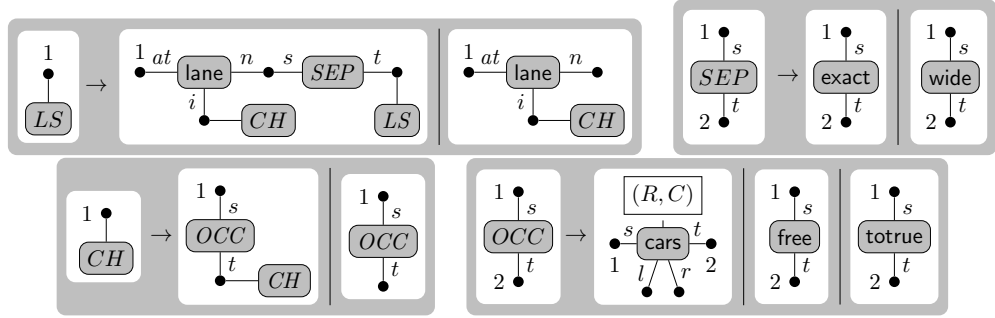


Figure 5.12: Rule Sets R_{LS} , R_{SEP} , R_{CH} and R_{OCC} : Lane Sequences and Topological Sequences ($R, C \subset \text{CVar} \cup \{\text{ego}\}$)

Similar to the description of the concrete syntax, we start with the representations of topological and lane sequences. The corresponding graph rewriting rules are shown in Fig. 5.12. Each lane of a lane sequence is represented by an edge of type `lane`. The separations are denoted by `wide` and `exact`, depending on whether they are wide or exact separations. The topological sequences are denoted by `cars`, `free` and `totrue` edges denoting the spatial situation on each lane. The l and r tentacles of edges of type `cars` represent the left and right borders of cars, while the s and t tentacles of `cars`, `free` and `totrue` edges denote the source and target of the edges, yielding a sequence of cars on a lane.

Lemma 5.1. Let T be a topological sequence and H be a graph of the form $H = (\{v\}, \{e\}, \tau, \theta, \mathfrak{l})$ with $\tau(e) = \langle v \rangle$ and $\theta(e) = CH$ and $\mathfrak{l} = \emptyset$. Then there is a graph G such that $H \Rightarrow^* G$ and G represents T . Furthermore, G is unique up to isomorphism.

Proof. We proceed by induction on the length n of T . If $n = 1$, we can apply the second alternative for `CH` and to get a graph consisting of two nodes and a connecting `OCC` edge. Depending on whether T is free space, an overlap of reservations and claims or undefined space, we can choose the suitable rule for `OCC` to get the right graph G . Observe that for each of the three possibilities for T , there is exactly one derivation $H \Rightarrow^* G$, i.e. G is uniquely defined.

Assume that for all topological sequences with length $\leq n$ the lemma holds. Now we have to prove the lemma for a topological sequence T of length $n + 1$. So T consists of a

sequence T_1 of length n and an additional sequence of length 1. For T_1 , we know by the induction hypotheses that there exists a unique representing graph G_1 with $H \Rightarrow^* G_1$. Within this derivation, there is exactly one application of the second alternative of R_{CH} , i.e. $H \Rightarrow^* G_3 \Rightarrow_{R_{CH}^2} G_2 \Rightarrow^* G_1$. Now insert an additional application R_{CH}^1 to G_3 and get $H \Rightarrow^* G_3 \Rightarrow_{R_{CH}^1} G'_3 \Rightarrow_{R_{CH}^2} G'_2 \Rightarrow^* G_1$. All applications of rules in $G_2 \Rightarrow^* G_1$ are still possible starting with G'_2 , since G_2 is a subgraph of G'_2 , i.e., $H \Rightarrow^* G_3 \Rightarrow_{R_{CH}^1} G'_3 \Rightarrow_{R_{CH}^2} G'_2 \Rightarrow^* G_1$. Then G'_1 is isomorphic to a path of length $n + 1$, where the last edge is of the type OCC and the first n edges represent T_1 . Similar to the induction base, we can apply exactly one of the rules R_{OCC}^i ($1 \leq i \leq 3$) depending on the type of the last situation in T to derive graph G representing T . \square

Lemma 5.2. Let L be a lane sequence and H be a graph $H = (\{v\}, \{e\}, \tau, \theta, \mathfrak{l})$ with $\tau(e) = \langle v \rangle$ and $\theta(e) = LS$ and $\mathfrak{l} = \emptyset$. Then there is a graph G such that $H \Rightarrow^* G$ and G represents L . Furthermore, G is unique up to isomorphism.

Proof. By induction on the length of L and applying Lemma 5.1. \square

5.4.2 Spatial Diagrams

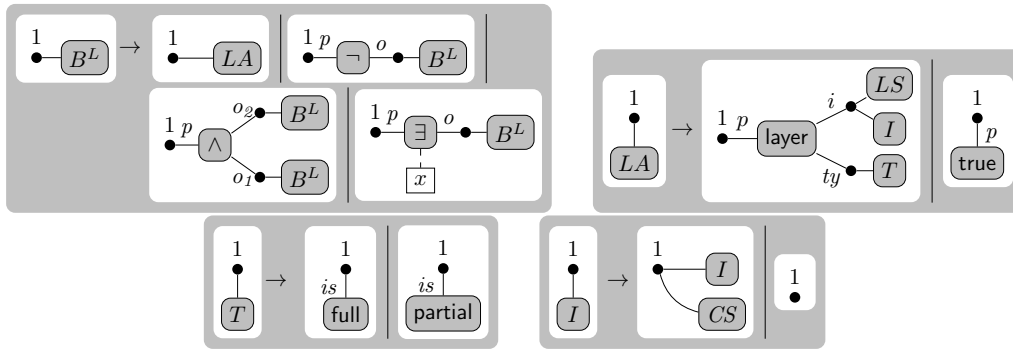


Figure 5.13: Rule Sets R_{BL} , R_{LA} , R_T and R_I : Structure of Spatial Diagrams ($x \in \text{Var}$)

The sets of rules in Fig. 5.13 define the abstract syntax for the qualitative parts of spatial diagrams. The rules for the non-terminal B^L describe the structure of the edges representing first-order connectives between atomic spatial diagrams. The graphs created by this set of rules are binary trees, where the leaves are non-terminals of the type LA . Edges of this type can either be replaced by an edge of type `true` to denote a shaded rectangle or an edge of type `layer` to represent a full or partial layer. All edges created by these sets of rules have a tentacle p , which denotes the parent of the represented element like in a typical syntax graph. The logical connectives furthermore employ either one tentacle o or two tentacles o_1 and o_2 for the connections to their operands. The edges of type `layer` in contrast use a tentacles called i to describe the interior of the layer and a tentacle called ty to denote its type, i.e., whether it describes the whole view or just a part. The nodes visited by the ty tentacles are then also visited by the single tentacle of

either an edge of type **full** or of type **partial**. The non-terminal I allows for the creation of arbitrary many non-terminals CS , which will be used for the creation of distance arrows within one layer.

Lemma 5.3. Let S be a spatial diagram and H be a graph $H = (\{v\}, \{e\}, \tau, \theta, \mathfrak{l})$ with $\tau(e) = \langle v \rangle$ and $\theta(e) = B^L$ and $\mathfrak{l} = \emptyset$. Then there is a graph G such that $H \Rightarrow^* G$ and the unique graph resulting from G by removing all edges of the types CS represents S , with the omission of the distance arrows within S .

Proof. By induction on the structure of spatial diagrams. Let S be a shaded rectangle denoting unspecified space. Then we can use the following derivation to get a suitable graph G .

$$H \Rightarrow_{R_{BL}^1} H_1 \Rightarrow_{R_{LA}^2} G.$$

In G the only node is visited by the p tentacle of the only edge of type **true**.

Now let S be a partial layer containing the lane sequence L . Then we use the following derivation

$$H \Rightarrow_{R_{BL}^1} H_1 \Rightarrow_{R_{LA}^1} G_1 \Rightarrow_{R_T^2} G_2.$$

By Lemma 5.2, we can derive a graph G_L from the edge of type LS such that G_L uniquely represents the lane sequence L . This derivation is applicable to G_2 , resulting in the graph G . Now the only remaining non-terminal is of the type I , which we remove by applying R_I^2 to get the graph G' where we can map S to the edge e with $\theta(e) = \text{layer}$, whose ty tentacle visits the same node as the is tentacle of the **partial** edge. Hence if we remove all pending edges of the type CS from G' (which may have been created, before the application of R_I^2), we get the unique graph representing S with the omission of the distance arrows within S . The case for a full layer is similar.

Now assume that for the spatial diagrams S_1 and S_2 the lemma holds. Let S be the negation of S_1 . Then we use

$$H \Rightarrow_{R_{BL}^2} H_1.$$

Within H_1 there exists exactly one edge e of the type B^L . Call the subgraph consisting of this edge and the node it is visiting G_H . By the induction hypothesis we know that there is a suitable derivation $G_H \Rightarrow^* G_1$ such that G_1 represents S_1 , if we remove all CS edges from G_1 . All the derivation steps can be directly applied to H_1 , i.e., we get a derivation

$$H \Rightarrow_{R_{BL}^2} H_1 \Rightarrow^* G,$$

where G_1 is a subgraph of G . In G , the o tentacle of the edge representing the negation visits the same node as the p tentacle of the representation of the topmost operator in S_1 . Since G_1 is contained as a subgraph in G , removing all edges of the type CS from G results in the unique representation of S , with the omission of all distance arrows within S . The cases for the conjunction of S_1 and S_2 as well as for the existential quantification are similar. \square

Distance Arrows

Figure 5.15 to Fig. 5.22 contain the most complicated rules, which create the hyperedges representing distance arrows, i.e., which replace edges of the type CS . In contrast to the other rules, which replace non-terminal edges in a context-free manner in the sense that only one hyperedge is replaced by an application of a rule without taking the environment of the edge into account, these productions employ application conditions. Fig. 5.14 shows the grammar for replacing a non-terminal P by a path between the nodes labelled as 1 and 2.

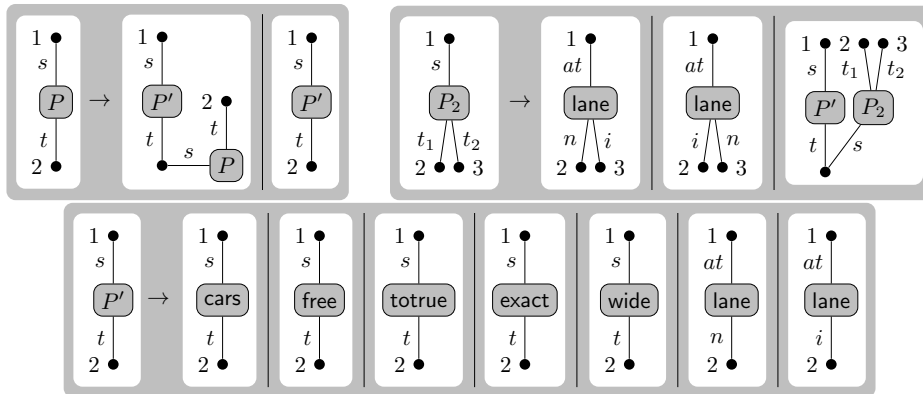


Figure 5.14: Hyperedge Replacement Grammar for the Definition of Paths

The edge P may be replaced in two ways. First, by two edges: the first of type P' connecting 1 and an intermediate node i , and the second again of type P , connecting i and 2. The second possibility is by a single edge of type P' , connecting 1 and 2. The edges typed with P' may then be replaced by edges of the following terminal types: cars, free, totrue, lane, exact and wide.

Furthermore, Fig. 5.14 shows the replacement rules for the non-terminal P_2 , which is used for defining a path splitting into two at a lane edge (cf. Fig. 5.18 to Fig. 5.22). This grammar also uses P' for the representation of the common part of the paths. Note that similarly to the grammar for P , P_2 allows for the creation of a path of arbitrary length ending at the s tentacle of P_2 .

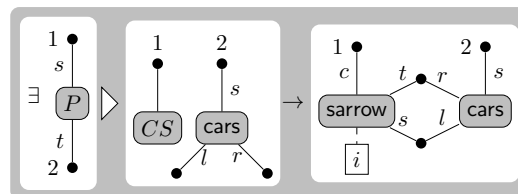


Figure 5.15: Rule R_{CS}^{sing} : Distance Arrow from the Left to the Right Border of a Single Topological Situation ($i \in \mathcal{I}_{Var}$)

The rules depicted in Fig. 5.18 to Fig. 5.22 define the creation of distance arrows

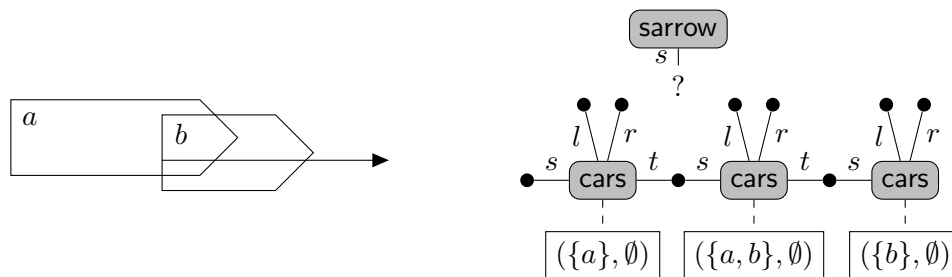


Figure 5.16: Different Possibilities for Distance Arrows

between the tentacles of *cars* edges. For this, we have to distinguish between four cases. Consider for simplicity the content of a single lane with a single distance arrow like in Fig. 5.16. The source of the distance arrow is the right border of the part, where only car *a* occupies the lane. However, in the abstract syntax, two possibilities could describe this situation. Both the *r* node of the leftmost *cars* edge and the *l* node of the middle *cars* edge refer to the same border. If we allowed for both possibilities, such situations would not have a unique type. Hence we chose to attach the source tentacle of *sarrow* edges to *l* nodes of *cars* edges whenever possible. A similar situation comes up for the target tentacle of distance arrows, which we chose to attach to *r* nodes. The rules for these standard cases are given in Fig. 5.15 and Fig. 5.18.

Figure 5.15 can be read as follows. If there is a non-terminal *CS* visiting the node labelled 1 and there exists a path from 1 to the representation of a topological situation representing an occupied topological situation, i.e., an edge of type *cars*, then *CS* can be replaced to represent a distance arrow constraining the length of this situation. This rule is a special case of the rule depicted in Fig. 5.18. This special case is needed, since we use injective graph conditions, i.e., the nodes labelled 2 and 3 in Fig. 5.18 have to be distinct.

In the rules we furthermore have to distinguish two cases. The first case is that both *cars* edges are part of the same lane sequence. This is captured in the upper rules of Fig. 5.18 to Fig. 5.20 and Fig. 5.21. The *HR** condition ensures, that both edges are reachable via a single path from the node, the non-terminal *CS* is visiting. In terms of the concrete syntax, this means that both topological situations are part of a single topological sequence within one layer.

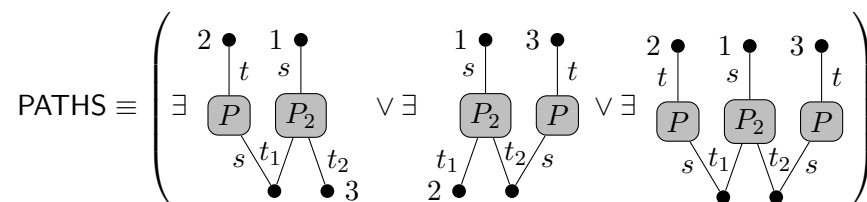


Figure 5.17: Application Condition for Distance Arrows between Different Lanes

The second case is that the edges are part of different topological sequences within the same layer. That is, there is a path from the node the *CS* edge is visiting to the

node 1 a lane is visiting with its *at* tentacle, and from this lane edge, a path to node 2 and another path to node 3. Either the source of the arrow lies on a lower topological sequence than the target or vice versa. Intuitively, the lane edge represents this lower topological sequence. One of the paths defined by the *P* edges then defines the path to the situation on this sequence, while the other incorporates all lane separations in addition to all topological situations to the left of the upper topological situation. Due to its size, we present the corresponding HR* condition in Fig. 5.17.

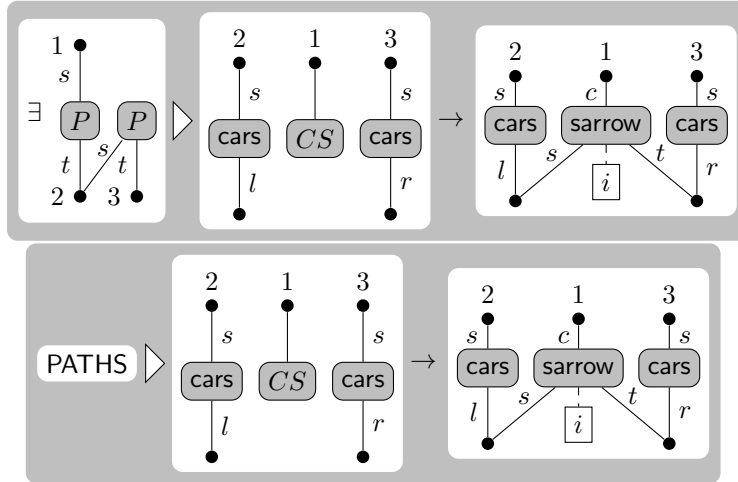


Figure 5.18: Rules R_{CS}^1 and R_{CS}^2 : Distance Arrow between Cars ($i \in \mathcal{I}_{Var}$)

There are still situations, where we want to attach the target of a distance arrow the *l* node of *cars* edges however. For example, if the arrow *A* ends at the left border of situation, whose left neighbour is blank space, we are *forced* to allow for such an attachment, since otherwise there would not exist a type for this situation. This is due to the fact that in the abstract syntax, there is no other *cars* edge attached to the source node of the edge representing the target of *A*. This case is depicted in Fig. 5.19. Note how the application condition not only ensures the existence of the according paths, but also prohibits another *cars* edge to be attached to the node labelled 3.

A similar situation arises for the sources of distance arrows. However, the standard attachment of *cars* edges in this case is the *l* node. If there is no *cars* edge to the right of the target edge, the arrow may also connect to the *r* node. This is shown in Fig. 5.20.

Finally, we have to consider the case, where both conditions as discussed above apply. That is, the arrow to be represented starts at the right side of an overlap of reservations and claims and ends at the left side of such a situation, where neither to the right of the source, nor to the left of the target there are such topological situations. The rules for these cases are depicted in Fig. 5.21 and Fig. 5.22.

Lemma 5.4. Let *A* be a distance arrow within a spatial diagram *D*, G_D the graph, where removing all edges of the type *I* and *CS* results in a graph representing *D* with omission of the distance arrows and *H* be a subgraph of G_D of the form $H = (\{v\}, \{e\}, \tau, \theta, \iota)$

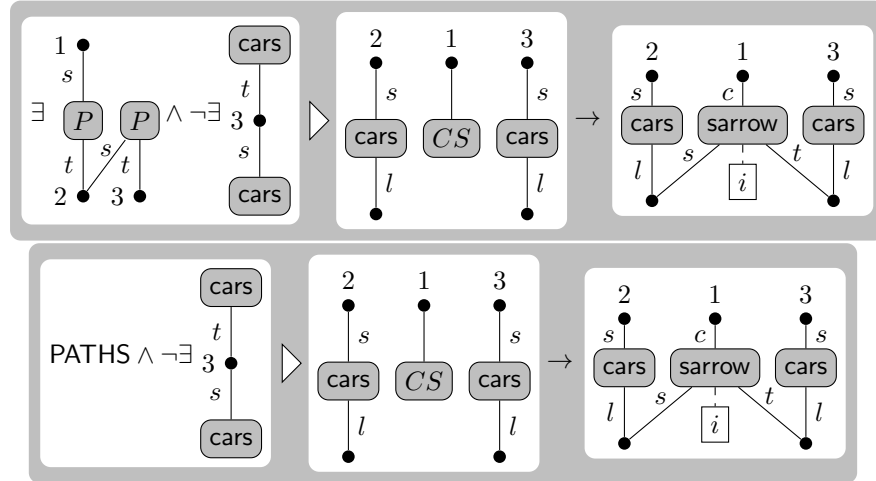


Figure 5.19: Rules R_{CS}^3 and R_{CS}^4 : Distance Arrow attached to the Left Side of the Target ($i \in \mathcal{I}_{\text{Var}}$)

with $\tau(e) = \langle v \rangle$ and $\theta(e) = CS$ and $l = \emptyset$ where v is visited by the i tentacle of the edge representing the layer A resides in. Then there is a graph G such that $G_D \Rightarrow^* G$ and G represents D together with A , with the omission of all other distance arrows within D . Furthermore, G is unique up to isomorphism.

Proof. We have to consider several cases, most of which can be handled similarly. Assume that h is the injective mapping showing that G_D is representing D .

First, let A connect the left and right border of a single topological situation S within the layer L . Let e_S be the edge representing S . If e_S is the leftmost situation within its topological sequence, its s tentacle visits the same node as the i tentacle of a unique edge e_{LS} with $\theta(e_{LS}) = \text{lane}$. Otherwise, we know that its s tentacle visits the same node as the t tentacle of the edge representing its left predecessor situation. So, using only edges of the type *cars*, *free* and *totrue*, we finally reach the unique edge e_{LS} . Now again, if the sequence of edges e_{LS} is connected to represents the lowermost topological sequence, then the at tentacle of e_{LS} visits the node which is also visited by the i tentacle of the representation of L . Otherwise there is an edge of type *exact* or *wide* whose t tentacle visits this node and whose s tentacle visits the node which is connected to the n tentacle of another *lane* edge representing the next topological sequence. That is, in a similar manner to topological sequences, we finally reach the unique edge representing the lowermost topological sequence, which is connected to the node visited by the i tentacle of the representation of L . In other words, we can find a path P_S leading from this node to the node visited by the s tentacle of e_S . Now, if we substitute P_S for P in the application condition of R_{CS}^{Sing} , the condition is satisfied. That is, we may replace the non-terminal CS with an edge e with $\theta(e) = \text{sarrow}$ which visits the nodes visited by the l and r tentacle in the correct way to be a representation of A . Furthermore we may choose the label of e to be the interval labelling A . Hence the lemma holds in this case.

Now let A connect the left border of a topological situation S_1 with the right border

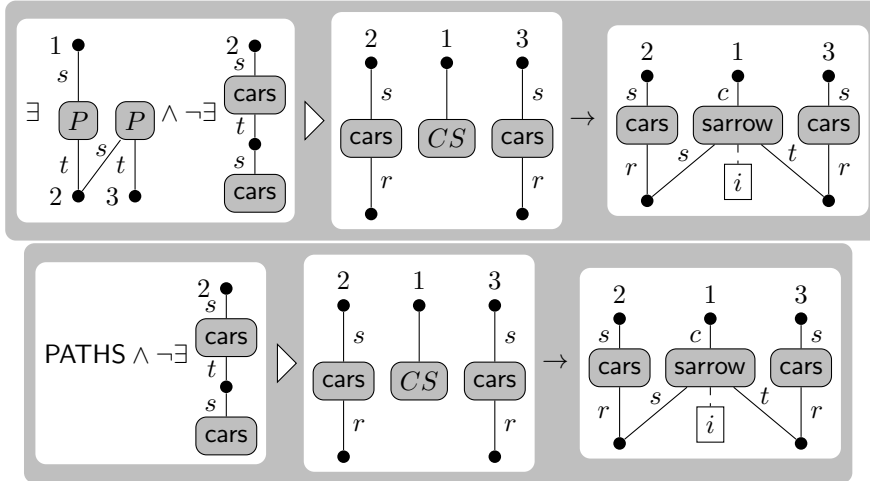


Figure 5.20: Rules R_{CS}^5 and R_{CS}^6 : Distance Arrow attached to the Right Side of the Source ($i \in \mathcal{I}_{\text{Var}}$)

of situation S_2 where S_1 and S_2 lie within the same topological sequence T and S_1 is to the left of S_2 . Furthermore assume that there are topological situations represented by `cars` edges both to the right of S_1 and to the left of S_2 . Now by arguments similar to the first case, since G represents the qualitative aspects of L , we can find a path from the node 3 visited by the s tentacle of $h(S_2)$ to the node 2 visited by the s tentacle of $h(S_1)$. Furthermore, we get that there is a path from 2 to the node 1 visited by the i tentacle of $h(L)$. That is, we can replace substitute the variables P with these paths and hence get that the rule R_{CS}^1 is applicable. The application of this rule yields an edge representing A . Observe that none of the rules R_{CS}^i for $1 < i \leq 8$ are applicable, since either they require the path to split at one edge (for $i \in \{2, 4, 6, 8\}$) or they require the absence of at least one of the `cars` edges representing the topological situation to the right of S_1 and to the left of S_2 .

Now assume that A connects the left border of a topological situation S_1 with the right border of situation S_2 where S_1 lies in the topological sequence T_1 and S_2 lies within T_2 . Similarly to the previous case, we assume that there are topological situations represented by `cars` edges both to the right of S_1 and to the left of S_2 . Without loss of generality, let T_1 be lower than T_2 . Since G_D represents D , we have edges e_{S_1} and e_{S_2} representing S_1 and S_2 respectively. Furthermore, we find a path P_d from e_{S_1} to a node v_1 visited by the i tentacle of a unique edge e_{T_1} with $\theta(e_{T_1}) = \text{lane}$ and similarly a path P'_u from e_{S_2} to a node v_2 which is visited by the i tentacle of the unique edge e_{T_2} with $\theta(e_{T_2}) = \text{lane}$. Since T_1 is lower than T_2 , there is a path P''_u from v_2 to the node visited by the n tentacle of e_{T_1} . Let P_u be the concatenation of P''_u and P'_u . Again, like in the first case, we find a path P_s from e_{T_1} to the node visited by the i tentacle of $h(L)$. Let both P_d be non-empty, i.e. S_1 is not the left-most topological situation of T_1 (P_u can not be empty since P''_u contains at least one edge). Now consider the application condition PATHS (Fig. 5.17). If we substitute P_d for the edge of type P connected to the node 2,

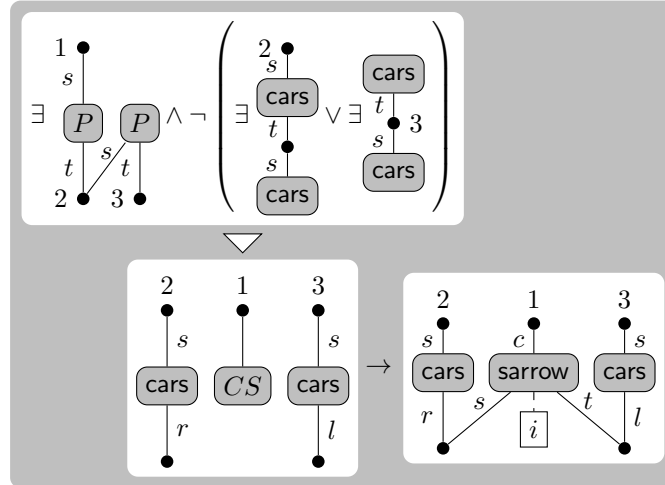


Figure 5.21: Rule R_{CS}^7 : Distance Arrow between Cars on the Same Lane attached to the Right Side of the Source and the Left Side of the Target ($i \in \mathcal{I}_{\text{Var}}$)

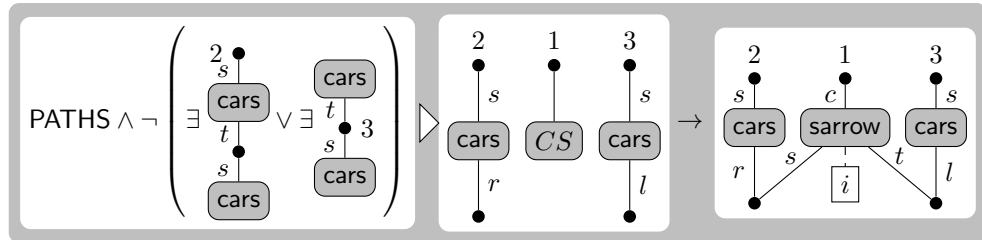


Figure 5.22: Rule R_{CS}^8 : Distance Arrow between Cars on Different Lanes attached to the Right Side of the Source and the Left Side of the Target ($i \in \mathcal{I}_{\text{Var}}$)

P_u for the other edge connected to node 3 and P_s for the node of type P_2 , the application condition for rule R_{CS}^2 is satisfied. Hence we can apply this rule and get a representation of A . Similar to the case above, none of the other rules in the set R_{CS} is applicable.

The proof for the rest of the cases are similar. Observe that the additional application conditions restrict the paths created by the HR grammar such that the rules are only applicable, whenever the representation of the situation right to the source of A is not an edge of type `cars` or similarly for the situation left to the target of A or both. Hence the application conditions for all these rules are mutually exclusive, i.e. for all possibilities for the source and targets of A , there is exactly one type of rule applicable. By that we know that the representation of A is uniquely defined. \square

5.4.3 Temporal Sequences

Finally, the rules creating the representations for Traffic Diagrams as a whole are given in Fig. 5.23. The rule set R_{BD} is structurally similar to the set R_{BL} with the exception of R_{BD}^1 , which creates the start of a temporal sequence. The set of rules R_L allows for

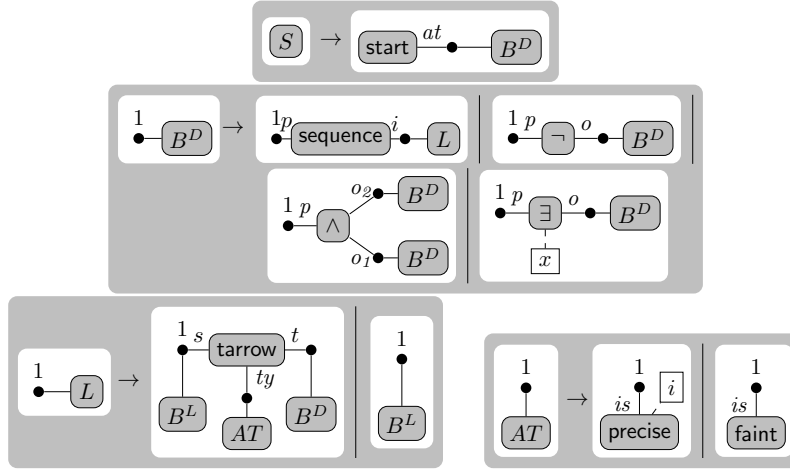


Figure 5.23: Rule Sets R_S , R_{B^D} , R_L and R_{AT} : Structure of Traffic Diagrams ($i \in \mathcal{I} \cup \mathcal{A}$)

the creation of the internal parts of a temporal sequence, i.e. a spatial diagram connected by a temporal arrow to a new temporal sequence. Similar to the type of layers, the R_{AT} rules define, whether a temporal arrow is precise or faint. The non-terminal S serves to have a clearly defined start for the derivation of abstract syntax graph for a Traffic Diagram.

Lemma 5.5. Let D be a traffic diagram and H be a graph $H = (\{\}, \{e\}, \tau, \theta, \mathfrak{l})$ with $\tau(e) = \langle \rangle$ and $\theta(e) = S$ and $\mathfrak{l} = \emptyset$. Then there is a graph G such that $H \Rightarrow^* G$ and G represents D .

Proof. By structural induction on traffic diagrams. For the induction base, let D be a spatial diagram D' surrounded by a rounded dotted rectangle denoting a temporal sequence. Then we can use the following derivation

$$H \Rightarrow_{R_S} H_1 \Rightarrow_{R_{B^D}^1} H_2 \Rightarrow_{R_L^2} H_3 .$$

The graph H_3 consists of a path starting at the single edge of the type start and ending at non-terminal edge of the type B_L . By Lemma 5.3, we can derive a graph G_L from H_3 which represents the qualitative aspects of D' . Within this derivation, there is exactly one application of R_L^2 , i.e., the rule removing the the single edge of the type I . Without loss of generality, we can assume, that it was the last rule to be applied., i.e.,

$$H_3 \Rightarrow^* G'_L \Rightarrow_{R_L^2} G_L .$$

Consider now two cases for G'_L . Let the number m of CS edges within G'_L be greater than the number n of distance arrows within D' . Then the rule R_L^1 was applied m times within the derivation. However, all other rule applications within $H_3 \Rightarrow^* G'_L$ are independent of the existence of these edges. Hence we can simply omit $m - n$ applications of R_L^1 and still get a representation of the qualitative aspects of D' , but containing exactly n edges of

the type CS . Now let the number of CS edges be smaller than n , the number of distance arrows within D' . We can use the rule R_1^1 n -times to create n instances of non-terminal edges of the type CS .

By now we have a representation of D' , where the number of CS edges agrees with the number of distance arrows within D' . By abuse of notation, we will still call this concrete representation G'_L . By Lemma 5.4, each of the CS edges can be used to derive a representation of one of the distance arrows of D' , i.e.,

$$H \Rightarrow_{R_S} H_1 \Rightarrow_{R_{BD}^1} H_2 \Rightarrow_{R_L^2} H_3 \Rightarrow^* G_L \Rightarrow_{R_1^1}^n G'_L \Rightarrow_{R_1^2} G''_L \Rightarrow_{Lem. 5.4}^n G .$$

The topmost operator in D is the sequence, which is represented in G by the unique edge e of type **sequence**. The p tentacle of e visits the same node as the edge of type **start**. Hence, G represents D .

For the induction hypothesis, assume that for the traffic diagrams D_1 and D_2 , the lemma holds. The cases for the logical operators are similar to the proof of Lemma 5.3. Hence we proceed with the case, where D consists of a temporal arrow connecting a spatial diagram D_S with the traffic diagram D_1 . By the induction hypothesis there is a derivation

$$H \Rightarrow^* G_1$$

such that G_1 represents D_1 . Observe that the first applied rule within this derivation has to be R_S . I.e.,

$$H \Rightarrow_{R_S} H_1 \Rightarrow^* G_1 .$$

Now we modify the beginning of the derivation as follows.

$$H \Rightarrow_{R_S} H_1 \Rightarrow_{R_{BD}^1} H_2 .$$

Depending on whether the temporal arrow in D is precise or faint, we apply the rule R_{AT}^1 or R_{AT}^2 , respectively.

$$H \Rightarrow_{R_S} H_1 \Rightarrow_{R_{BD}^1} H_2 \Rightarrow_{R_{AT}^i} H_3 .$$

Observe that H_3 contains a subgraph H_S of the form as needed for the application of Lemma 5.3. Similar to the induction base, we find a derivation $H_S \Rightarrow^* G_S$ such that G_S represents D_S . The rules used in this derivations are all applicable, consider H_3 itself to get H_4 . Similarly, the derivation $H_1 \Rightarrow^* G_1$ is applicable to H_4 , i.e., to the subgraph attached to the node visited by the t tentacle of the created **tarrow** edge e . Hence

$$H \Rightarrow_{R_S} H_1 \Rightarrow_{R_{BD}^1} H_2 \Rightarrow_{R_{AT}^i} H_3 \Rightarrow^* H_4 \Rightarrow^* G .$$

Then the s tentacle of e visits the same node as the p tentacle of the representation of D_S , and similarly the t tentacle of e visits the same node as the p tentacle of the representation of D_1 . Since the type of the temporal arrow is represented by the edge visiting the same node as the ty arrow as e , we have found a representation for the diagram D . The rules to be apply were uniquely defined by the type of the temporal arrow, hence G is also uniquely defined. \square

5.5 Semantics

Now having a clearly defined mathematical model to describe the syntax of traffic diagrams, we turn our attention to the definition of their semantics. Similarly to EMLSL, we use the abstract road model, i.e., traffic snapshots and views, introduced in Chap 3 as a basis for the semantics. We have to slightly extend the notion of a valuation given in Chap. 4, Definition 4.3 to also evaluate the variables used within intervals.

Definition 5.8 (Interval Valuation). *Let ν be a valuation, i.e., a function $\nu: \text{Var} \rightarrow \mathbb{I} \cup \mathbb{R}_+ \cup \mathbb{N}$ respecting the sorts of variables. Similar to Definition 4.3, we lift ν to a function ν_I evaluating intervals, where variables and ego are interpreted as in ν and for an interval $i = [a, b]$, we define $\nu(i)$ to be $[\nu(a), \nu(b)]$ and similarly if i is (half-) open. We use the notation $\nu \oplus \{v \mapsto \alpha\}$ for the modification of ν which is identical to ν for all variables different from v , and which maps v to α .*

Observe that even though the range of ν includes \mathbb{N} , the definition of traffic diagrams does not include any variables whose type needs them to be evaluated to a natural number. However, this definition slightly eases the combination of EMLSL formulas and Traffic Diagrams as defined in Chap. 6.

For the definition of the satisfaction relation \models , observe that the abstract syntax of a diagram up to the level of layers is essentially a tree. Hence we can traverse the abstract syntax, starting at the single edge of type `start`, to define \models almost in a standard way. Like for EMLSL, we define the semantics of a whole diagram with respect to a traffic snapshot \mathcal{TS} , a view V and a valuation ν . To keep track of the position within the tree, we furthermore have to use a node of the hypergraph. This node will be important for the semantics of distance arrows. For one of the semantic definitions to be applicable to a syntax graph G , we require the existence of an embedding of the depicted subgraph into G at the labelled nodes.

We chose to present the semantics based on the graphical representation of the abstract syntax for purposes of readability. In this way, the presentation can be directly compared to the definition of the abstract syntax in the previous section.

In the depictions of the semantics, some of the cases are not immediately mutually exclusive. Consider for example Fig. 5.27. If a syntax graph contains the representation of a layer with two lanes connected by an exact lane separation, then both the first and the second case may be applied. In these cases, we require that the maximal possible occurrence of a graph has to be chosen.

We distinguish two types of variables in the semantics. First and most important, all variables occurring in the syntax of diagrams are called *explicit variables*. *Implicit variables* refer to auxiliary variables only occurring in the semantics. (See Fig. 5.28). These variables are used to refer to the length of topological situations. They give us the possibility to connect the qualitative semantics of a diagram, i.e., the order and type of topological situations with the quantitative semantics, i.e., the constraints on these lengths imposed by the distance arrows. Formally, we use an injective, partial function mapping all `cars`, `free` and `totrue` edges of an abstract syntax graph to implicit variables.

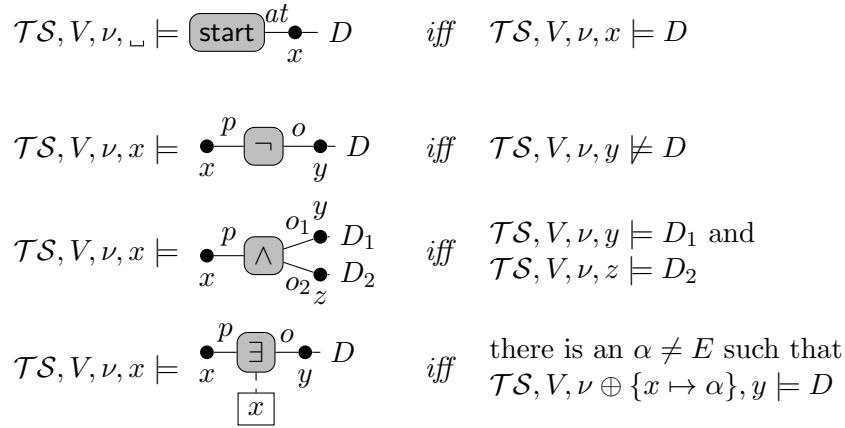


Figure 5.24: Semantics for Logical Connectives

Definition 5.9 (Implicit Variables and Implicit Lengths Function). *Let $G = (\mathcal{V}, \mathcal{E}, \tau, \theta, \iota)$ be the abstract syntax graph of a Traffic Diagram D and let $\text{Var}_I \subset \text{RVar}$ be an infinite subset of real-valued variables called implicit variables, which are fresh for D . Then the implicit lengths function $\chi: \mathcal{E} \rightarrow \text{Var}_I$ is an injective partial function given by*

$$\chi(e) = \begin{cases} v_e & \text{if } \theta(e) \in \{\text{cars, free, totrue}\} \\ \text{undef.} & \text{otherwise} \end{cases}.$$

Within the semantics, all implicit variables are existentially quantified (see Fig. 5.26), since we want the values of these variables to be independent of the valuation. They should reflect the spatial situation, and hence the satisfaction of a diagram should not depend on whether an implicit variable is given the right value by the valuation. There are two types of constraints on these variables. On the one hand, the semantics of the topological situations will ensure that the values of implicit variables reflect the length of the respective situation. On the other hand the semantics of distance arrows will require the values of these variables to respect the intervals the arrows are annotated with.

Definition 5.10 (Satisfaction of Traffic Diagrams). *The satisfaction relation \models defines the semantics of Traffic Diagrams as shown in Figures 5.24, 5.25, 5.26, 5.27, 5.28 and 5.29. We say that a traffic snapshot \mathcal{TS} , a view $V = (L, X, E)$ and a valuation ν with $\nu(\text{ego}) = E$ satisfy a Traffic Diagram D , if $\mathcal{TS}, V, \nu, \sqsubseteq \models D$, denoted by $\mathcal{TS}, V, \nu \models D$.*

The semantics of the first-order elements of Traffic Diagrams are given in Fig. 5.24. Most of these definitions are not surprising. The only non-standard aspect lies in the semantics of the existential quantifier. We require the quantified variables to be different than the owner of the current view. This is the only possibility to explicitly distinguish cars with Traffic Diagrams, since we do not have a diagrammatic representation of equality.

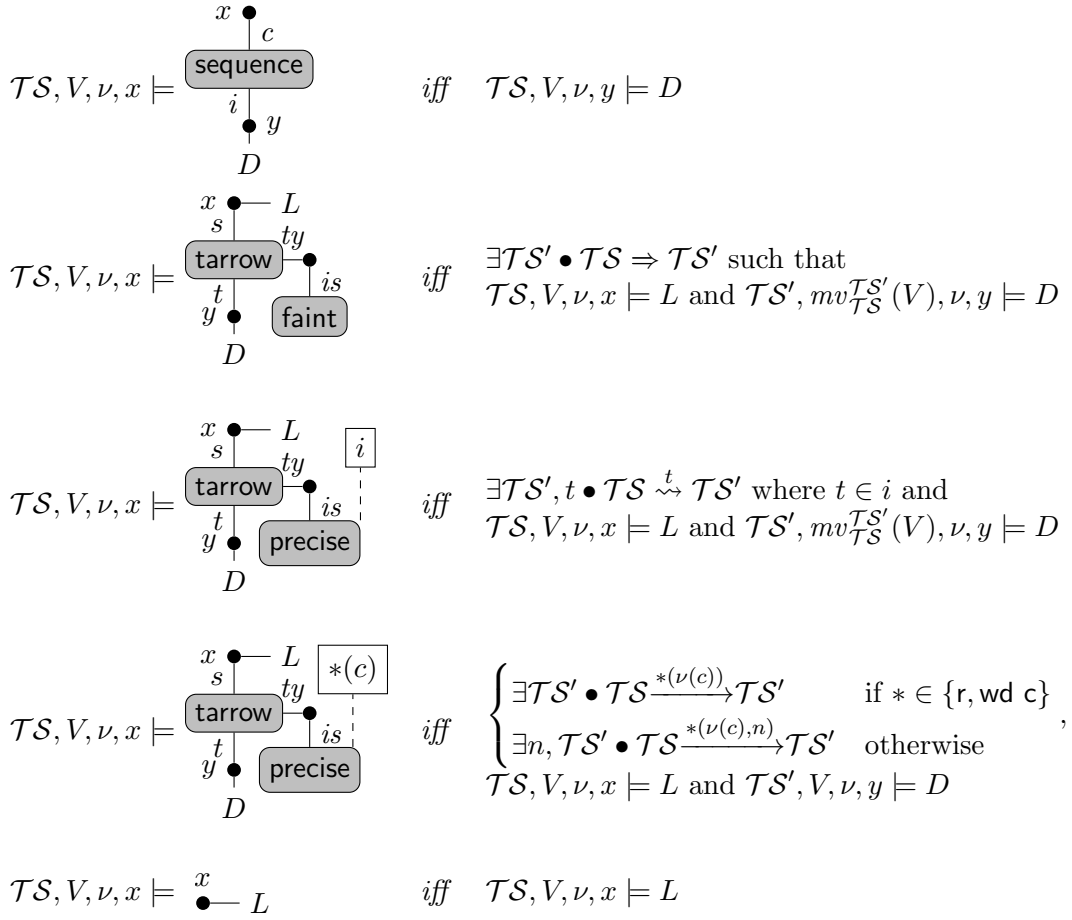


Figure 5.25: Semantics of Sequences (where $i \in \mathcal{I}$ and $* \in \{r, c, wd\}$)

Yet, often inequalities are implicit due to restrictions of the positions of reservations and claims as given by the semantic model (cf. Chap. 3).

Figure 5.25 shows the semantics of sequence, i.e., it describes how the temporal arrows are evaluated. Faint arrows only require the existence of an abstract transition to the next traffic snapshot, while precise arrows explicitly restrict the type of transition to be considered. Similar to the modalities of EMLSL, we abstract from the concrete lanes the transitions refer to.

In Fig. 5.26, the satisfaction relation is split up into the relations \models_S and \models_M , defining the *spatial semantics* and the *metric semantics* respectively. The former defines the topological structure of the snapshot, i.e., in which order the lanes occur, and how the cars are arranged on each lane. The relative positions between cars on different lanes are not restricted in any way. The metric semantics constrains space according to distance arrows. For the distinction between full and partial layers, we use subviews as in Definition 3.6.

Figure 5.27 describes the semantics of lane sequences. This resembles the vertical chop

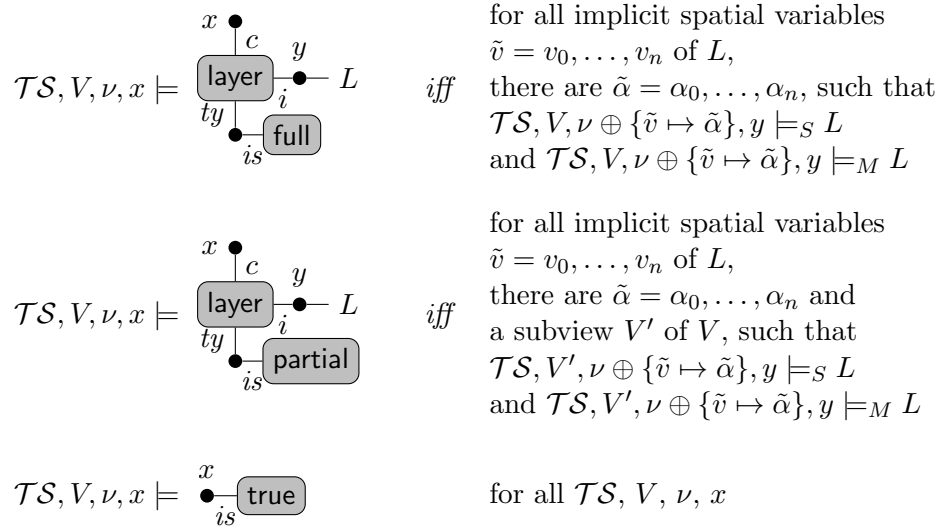


Figure 5.26: Semantics of Layers

modality of EMLSL. The existence of a lane edge in a layer ensures the existence of at least one lane in the view, even if the lane is filled with only one edge `totrue` and the layer is partial. This differs from the somewhere modality $\langle \varphi \rangle$, since $\langle \top \rangle \equiv \top$ does not require the view to contain a lane. Hence an partial layer denotes a stronger assertion than $\langle \varphi \rangle$. The diagrammatic element corresponding to a single \top of MLSL would be the shaded rectangle.

The difference of wide and exact lane separations is rather simple. While an exact lane separation divides a view into exactly two parts, one containing a single lane and the other one containing the rest, a wide lane separation may omit several lanes in between.

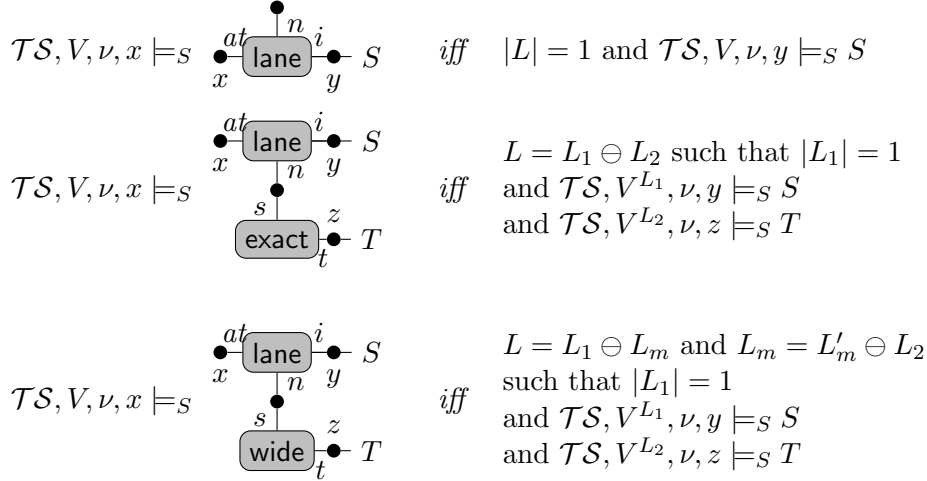
The spatial semantics of a single topological sequence as shown in Fig. 5.28 subsumes the horizontal chop of EMLSL as well as its atoms. On the one hand, the semantics of all these edges chop the given view along the horizontal extension into two intervals X_1 and X_2 , where `free` and `cars` edges additionally require X_1 to have a length greater than zero. Then, a `free` edge asserts the absence of any car, while a `cars` edge needs all the reservations and claims defined in its label to be present on the interval. Each of the edges of types `free`, `cars` and `totrue` are associated with their distinguished implicit variable holding the length of the corresponding interval. These variables will be used again in the semantics of distance arrows.

In the semantics of the `cars` edges, we use some abbreviations to enhance readability. For a set of car variables C and a view $V = (L, X, E)$, these are

$$\mathcal{R}[C, V] \equiv \text{for all } c \in C \bullet \text{res}_V(\nu(c)) = L \text{ and } \text{len}_V(\nu(c)) = X$$

and

$$\mathcal{C}[C, V] \equiv \text{for all } c \in C \bullet \text{clm}_V(\nu(c)) = L \text{ and } \text{len}_V(\nu(c)) = X.$$

Figure 5.27: Semantics of Lane Sequences, where $V = (L, X, E)$

Like the semantics of EMLSL atoms, these formulas ensure that the interval is filled completely by the reservations (claims) of each variable. Furthermore, the semantics only constrains cars referred to in the label. This implies that there may be reservations or claims of other cars present within the view. Only the edge representing free space restricts the presence of all cars. Note that in contrast to EMLSL, we are not able to define free space as an abbreviation. We can neither use negation nor horizontal chops arbitrarily within the diagrams and hence the abbreviation given in Sect. 4.1 is not definable as a Traffic Diagram. This comparison will be made more clearly in Sect. 6.1.

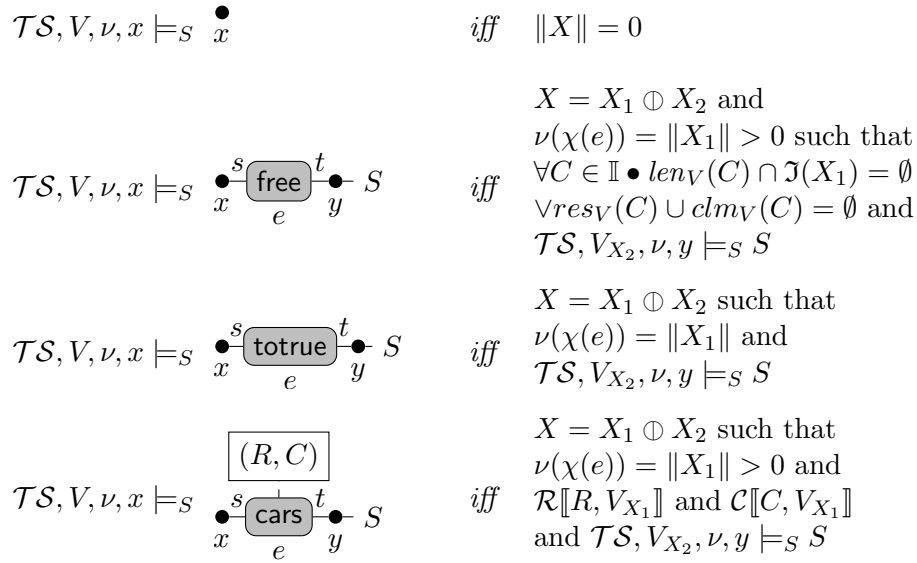
Finally, the metric semantics of distance arrows of a layer is shown in Fig. 5.29. For this, we need an order on cars within a lane. Therefore, we define a *precedence* relation \prec on edges.

Definition 5.11 (Order \preceq on Cars). *Let c and d be edges of the types cars, free or true. Then c is the predecessor of d , denoted by $c \prec d$, if and only if the s tentacle of d visits the same node as the t tentacle of c .*

Furthermore, for two edges c and d of these types, we write $c \preceq d$, if and only if there exists a sequence c_0, \dots, c_n such that $c = c_0$, $c_n = d$ and $c_{i-1} \prec c_i$ for all $i \in \{1, \dots, n\}$, or $c = d$.

The relation \preceq is a partial order on each sequence of edges denoting a single lane. This partial order will be used to sum up the distance of the left border of a view up to the cars the distance arrows connects. The difference between these sums is then constrained to lie within the interval the arrow is labelled with.

Since more than one arrow can be present inside of a single layer, we enumerate these distance arrows, as shown in the first case of Fig. 5.29, and require the layer to satisfy each. If an arrow is connected to the left side of a car, only the distance from the border of the view to the car without the length of the car itself is considered. Otherwise, the length of the car is included in the sum. For distance arrows connecting the left and


 Figure 5.28: Semantics of Topological Sequences, where $V = (L, X, E)$

right borders of a single car, the semantics is defined similarly.

Example 5.5. We briefly recall the traffic snapshot and view defined in Chap. 3 (Examples 3.1 to 3.4). The view is given by $V = (\{1, 2\}, [12, 42], E)$ and the corresponding restrictions of \mathcal{TS} are

$$\begin{array}{llll}
 \text{res}_V(A) = \{1, 2\} & \text{res}_V(B) = \{1\} & \text{res}_V(C) = \emptyset & \text{res}_V(E) = \{2\} \\
 \text{clm}_V(A) = \emptyset & \text{clm}_V(B) = \emptyset & \text{clm}_V(C) = \emptyset & \text{clm}_V(E) = \{1\} \\
 \text{len}_V(A) = [28, 39] & \text{len}_V(B) = [12, 15] & \text{len}_V(C) = \emptyset & \text{len}_V(E) = [14, 27]
 \end{array}$$

Furthermore, let ν be defined by $\nu(a) = A$, $\nu(b) = B$, $\nu(c) = C$ and $\nu(\text{ego}) = E$.

Now we will examine, whether this model satisfies the diagram shown in Fig. 5.30. The diagram itself carries no deeper meaning, but still incorporates the main elements of Traffic Diagrams. Hence it is a suited example to show the calculation of the semantics. The sequences and the hyperedge denoting the start of the diagram can be safely omitted. Then, the first question is, whether \mathcal{TS} , V and ν satisfy the upper layer in the diagram. Therefore, we have to find a suitable subview V_{sub} of V consisting of one lane. Let $V^{\text{sub}} = (\{2\}, [16, 20], E)$. The first part of the semantics of the edge of the type lane is satisfied, since $|\{2\}| = 1$. We then have to check, whether the semantics of the cars edge is fulfilled. For that, we chop the extension of V^{sub} into $[16, 20] = [16, 20] \oplus [20, 20]$. We have $\|[16, 20]\| = 4 > 0$ and both

$$\begin{aligned}
 \text{res}_{V_{[16,20]}^{\text{sub}}}(\nu(\text{ego})) &= \text{res}_{V_{[16,20]}^{\text{sub}}}(E) = \{2\} \\
 (\text{len}_{V_{[16,20]}^{\text{sub}}}(\nu(\text{ego}))) &= [16, 20] .
 \end{aligned}$$

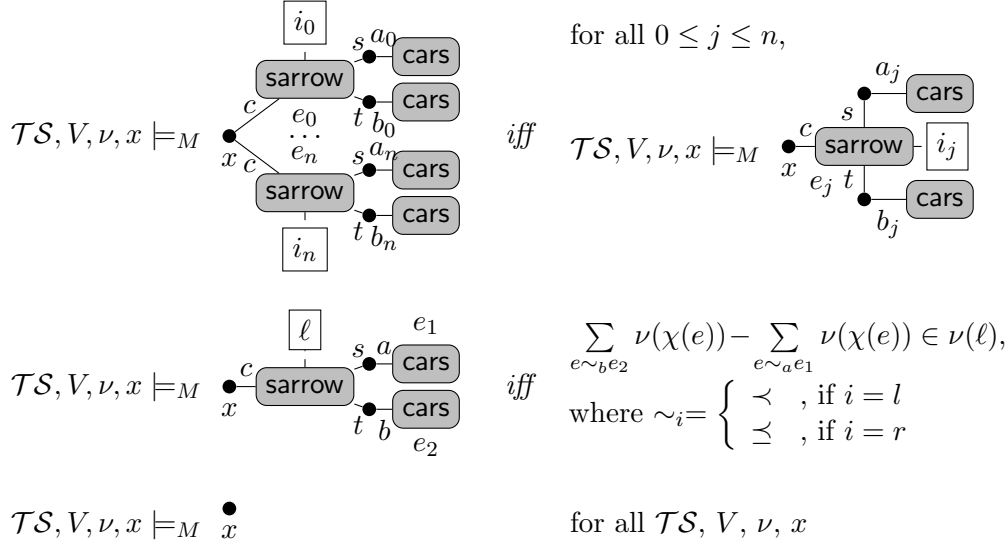


Figure 5.29: Semantics of Distance Arrows

Since $\| [20, 20] \| = 0$, this subview satisfies the semantics of the whole topological sequence.

Now we have to consider the transitions starting in \mathcal{TS} , where $\nu(a) = A$ withdraws one of its reservations. Consider the transition $\mathcal{TS} \xrightarrow{\text{wd } r(A,2)} \mathcal{TS}'$, i.e., in \mathcal{TS}' , we have $\text{res}'(A) = \{2\}$, while for the rest of \mathbb{I} , res' coincides with res . Again, we have to find a suitable subview $V^{\text{sub}'}$ of V which satisfies the topological sequence in the lower layer. Let $V^{\text{sub}'} = (\{2\}, [16, 30], E)$. Furthermore, we now have to consider the implicit spatial variables of the abstract syntax. Let e_i be the variable of the i -th edge in the topological sequence, i.e. e_1 belongs to the left cars edge, e_2 to the free edge and e_3 to the right cars edge. Similar to the case above, the semantics of the lane edge is satisfied. For the topological sequence itself, we consider the following chops of the extension:

$$\begin{aligned} [16, 30] &= [16, 27] \oplus [27, 30] \\ [27, 30] &= [27, 28] \oplus [28, 30] \\ [28, 30] &= [28, 30] \oplus [30, 30] \end{aligned}$$

Similar to the case above, $V_{[16,27]}^{\text{sub}'}$ satisfies the semantics of the first cars edge and $V_{[28,30]}^{\text{sub}'}$ does the same for the second edge of this type (for that, we let $\nu(e_1) = 3$ and $\nu(e_3) = 2$. We can chose these values, since the variables are existentially quantified for the evaluation of the layer). Consider now $V_{[27,28]}^{\text{sub}'}$ and the free edge. If we examine $\text{res}'_{V_{[27,28]}^{\text{sub}'}}$, $\text{clm}'_{V_{[27,28]}^{\text{sub}'}}$ and $\text{len}'_{V_{[27,28]}^{\text{sub}'}}$, we see that for the cars C and B in the traffic snapshot, both $\text{clm}'_{V_{[27,28]}^{\text{sub}'}}$ and $\text{res}'_{V_{[27,28]}^{\text{sub}'}}$ return \emptyset . For A , we get that $\text{len}'_{V_{[27,28]}^{\text{sub}'}}(A) = [28, 28]$, which intersected with $(27, 28)$ results in the empty set. Similar reasoning holds for E . Hence, if we let $\nu(e_2) = 1$, the semantics of this edge is fulfilled.

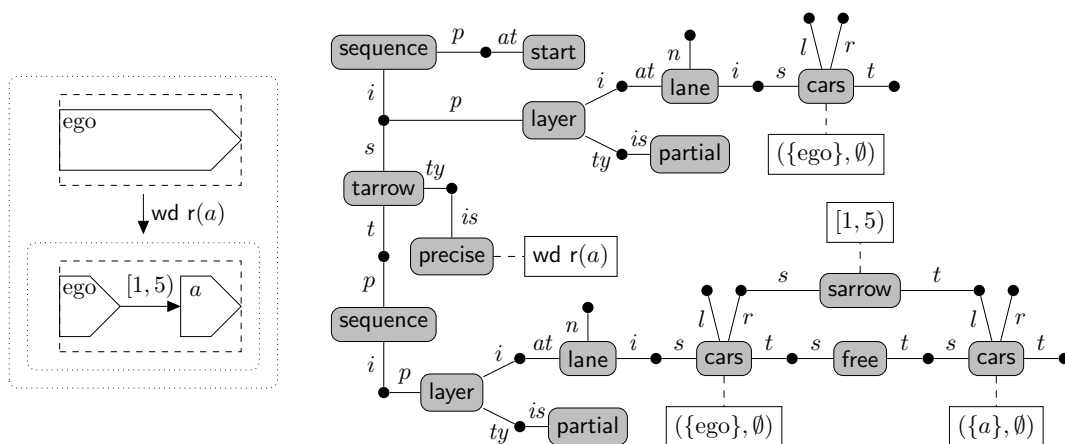


Figure 5.30: Concrete and Abstract Syntax of an Example Diagram

Now we turn to the distance arrow. Since the edge visits the same node as the r tentacle of its source as well as the l tentacle of its target, we have to check, whether the following holds:

$$((\nu(e_1) + \nu(e_2)) - \nu(e_1)) \in [1, 5]$$

This is true, since $\nu(e_2) = 1$. Hence, \mathcal{TS} , V and ν satisfy the diagram.

5.6 Decidability of Spatial Traffic Diagrams

In this section, we sketch a decision procedure for satisfiability of atomic spatial diagrams according to Def. 5.4. For that, we translate an atomic spatial diagram S into two independent formulas $F_X(S)$ and $F_L(S)$: $F_X(S)$ is a formula of linear arithmetic over real numbers which describes the possible relations of the cars along the extension of views. $F_L(S)$ is a formula of Presburger arithmetic which defines the lanes the cars may occupy, either with their reservations or with their claims. Then, S is satisfiable if and only if both $F_X(S)$ and $F_L(S)$ are satisfiable.

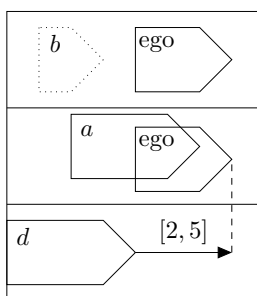


Figure 5.31: Example of a Spatial Diagram

We will not give a formal translation of this procedure, but illustrate the main ideas by an example. Consider the spatial diagram in Fig. 5.31. To each car c (i.e., each variable and ego) referenced in the diagram, we associate the following five variables:

1. c_l and c_r denote the left and right border of the extension of c ,
2. c_R^1 and c_R^2 refer to the lanes reserved by c , and
3. c_C refers to the lane possibly claimed by c .

In addition to these variables, we introduce four variables describing the view V itself.

1. v_l and v_r describe the borders of the extension of V .
2. \bar{v} and \underline{v} denote the upper and lower lane, respectively.

The formula $F_X(S)$ will only refer to variables of the form c_l and c_r for cars c and to the corresponding variables v_l and v_r for the borders of the view. Similarly, $F_L(S)$ will use the variables of the form c_R^1 , c_R^2 and c_C for cars as well as the variables \bar{v} and \underline{v} for the lowest and uppermost lanes in the view.

Translating the Relations of Positions of Cars The first part of the translation of Fig. 5.31 is simply a formalisation of Convention 3.2, i.e., the perceived lengths of the cars and the extension of the view are non-empty:

$$\text{conv}(S) \equiv a_l < a_r \wedge b_l < b_r \wedge d_l < d_r \wedge \text{ego}_l < \text{ego}_r \wedge v_l < v_r .$$

Then we have to relate the borders of the cars to the extension of the view:

$$\begin{aligned} \text{relative}(S) \equiv & a_l > v_l \wedge a_r < v_r \wedge b_l > v_l \wedge b_r < v_r \\ & \wedge d_l \leq v_l \wedge d_r < v_r \wedge \text{ego}_l > v_l \wedge \text{ego}_r < v_r . \end{aligned}$$

Now recall that the relations between cars on different lanes are not constrained at all, as long as no distance arrows connect such cars. Hence we can treat each lane separately. The lower lane only contains the car d , i.e., no new constraints arise for this lane. For the lane in the middle, we have to relate the cars a and ego. By inspection of the semantics of Traffic Diagrams, we see that the diagram only ensures that an overlap between a and e exists. It would still be satisfied, if a and ego completely overlap (this is due to the semantics of the cars edges, which only require the mentioned cars to be present, but do not prohibit other cars to reside at these parts of the lane). Hence we get the following translation for the upper lanes:

$$\begin{aligned} \text{topo}_2(S) \equiv & a_l \leq \text{ego}_l \wedge \text{ego}_r \geq a_r , \\ \text{topo}_3(S) \equiv & b_r < \text{ego}_l . \end{aligned}$$

The distance arrow can be translated by referring to the corresponding borders of the cars:

$$\text{dist}(S) \equiv 2 \leq \text{ego}_r - d_r \wedge \text{ego}_r - d_r \leq 5 .$$

So finally, we arrive at the formula $F_X(S)$:

$$F_X(S) \equiv \text{conv}(S) \wedge \text{relative}(S) \wedge \text{topo}_2(S) \wedge \text{topo}_3(S) \wedge \text{dist}(S) .$$

Translating what Lanes are Occupied by Cars We stipulate the following conventions: The value 0 corresponds to the empty set. That is, we will require the lowest lane of the view to have at least the value 1. With this convention, the constraints for the lanes of the view are as follows:

$$\text{lanes}(S) \equiv \underline{v} \geq 1 \wedge \bar{v} = \underline{v} + 2 .$$

This ensures, that the view contains exactly three lanes. Now we can encode the properties that we can derive from the diagram in the formula

$$\text{discrete}(S) \equiv a_R^1 = \underline{v} + 1 \wedge b_C = \bar{v} \wedge d_R^1 = \underline{v} \wedge \text{ego}_R^2 = \bar{v} \wedge \text{ego}_R^2 = \text{ego}_R^1 + 1 .$$

Note that some of the variables, like for example a_R^2 , are not constrained in any way. Thus, the formula still allows for many possible valuations of these variables, which do not correspond to a valid model. To constrain the remaining variables, we translate the sanity conditions. For all cars c , we use the following formula:

$$\begin{aligned} \text{san}(c) \equiv & c_R^1 \neq c_R^2 \rightarrow c_R^2 = c_R^1 + 1 \\ & \wedge c_R^1 \neq c_R^2 \rightarrow c_C = 0 \\ & \wedge c_C \neq 0 \rightarrow ((c_C = c_R^1 + 1 \vee c_R^1 = c_C + 1 \vee c_R^1 < \underline{v} \vee c_R^1 > \bar{v}) \wedge c_R^1 = c_R^2) . \end{aligned}$$

So the full formula $F_L(S)$ is given by

$$F_L(S) \equiv \text{lanes}(S) \wedge \text{discrete}(S) \wedge \text{san}(a) \wedge \text{san}(b) \wedge \text{san}(d) \wedge \text{san}(e) .$$

Deciding Satisfiability of Atomic Spatial Diagrams On the one hand, $F_X(S)$ is a formula over the reals, which should be checked for satisfiability in conjunction with the axioms of an ordered, real closed field, where \geq is a total order. Furthermore, $F_X(S)$ only defines *linear constraints*, for which the satisfiability problem is decidable [FR75]. On the other hand, the formula $F_L(S)$ is a formula of integer arithmetic without multiplication. Satisfiability for these kinds of formulas is also decidable [Coo72].

Hence, to decide the satisfiability of an atomic spatial diagram S , we construct the formulas $F_X(S)$ and $F_L(S)$ and check for satisfiability of the formulas independently. This is possible, since the properties defined in the formulas are not interdependent.



Figure 5.32: Problematic Example for Decidability

However, we cannot extend this approach directly to Boolean combinations of spatial diagrams. Consider for example the diagram S , consisting of the negation of the two negated atomic spatial diagrams S_1 and S_2 (cf. Fig. 5.32). If we use the translation naively, we would construct $F_X \equiv \neg(\neg F_X(S_1) \wedge \neg F_X(S_2))$ and $F_L \equiv \neg(\neg F_L(S_1) \wedge \neg F_L(S_2))$. Then we would check for satisfiability of each of these formulas on its own and return

“Satisfiable”, if both are. But the diagram states in effect, that S_1 or S_2 has to be true, for the whole diagram to be true. Now assume that $F_L(S_1)$ is satisfiable, but $F_X(S_1)$ is not and vice versa for S_2 , i.e., $F_L(S_2)$ is not satisfiable, but $F_X(S_2)$ is. Then the diagram is not satisfiable, since neither S_1 nor S_2 is. But our decision procedure would in effect decide whether $(F_X(S_1) \vee F_X(S_2)) \wedge (F_L(S_1) \vee F_L(S_2))$ is satisfiable, which is indeed true. Hence this naive procedure yields wrong results. For a correct procedure, the connection between the satisfiability results would have to mimic the Boolean structure between the atomic diagrams. We will not explore this problem more deeply here.

5.7 Related Work

Even though diagrams have often been used for informal argumentations during mathematical proofs, the formalisation of diagrammatic reasoning has only been carried out in the recent past. The main focus often lies on very abstract diagrammatic systems, like Euler and Venn diagrams. Shin defined a sound and complete formal deduction system for Venn Diagrams [Shi95], which is equally expressive as monadic first-order logic. The existential graphs of Peirce and their deductions rules have been formalised and proven to be sound, complete and equivalent to propositional logic by Hammer [Ham96]. Mineshima et al. compared a natural deduction style proof system for Euler diagrams with a proof system based on resolution for Venn diagrams [MOT10].

Even though these types of diagrams make extensive use of spatial properties in their presentation, spatial reasoning itself is not the intended application. Most work on spatial properties and diagrams copes with the representation of arbitrary relations (like set inclusion) in terms of spatial relations in diagrams. One of the few exceptions is given by Erwig and Schneider, who presented a diagrammatic system as a query language for spatio-temporal databases [ES99]. Their approach abstracts from extent, form and exact location of an object, only topological relations are considered. The movement of objects is indicated by trajectories in vertical direction.

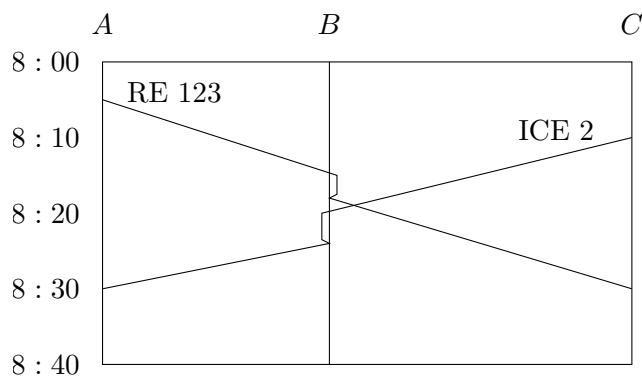


Figure 5.33: A Time-Distance Diagram for Railways

For the definition and evaluation of railway schedules, diagrams are often used to visualise the dependencies between occupied tracks and the time a train takes to leave the

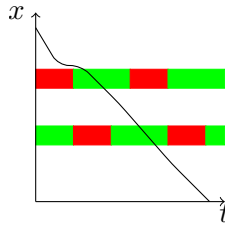


Figure 5.34: A Time-Space Diagram

track [HP08]. There are different types of diagrams, used for different purposes. In the following, we will describe diagrams that are used to describe dependencies on the tracks between stations. Consider the example in Fig. 5.33. The horizontal axis is labelled with the names of the stations A , B and C , while the vertical axis describes the flow of time. Trains are denoted by diagonal lines going from one station to another. The slope of the trains define whether they go from left to right or the other way round. For example, the train named “RE 123” drives from station A to C , while the train “ICE 2” goes from C to A . Both the trains stop at station B . Such diagrams allow for the examination whether a schedule is possible for the single track segments. For example, the lines of both trains intersect on the segment between B and C . That is, both drive on this segment at the same time, and hence it has to be a double track. If this is not the case, the schedule has to be adapted appropriately.

In the context of describing situations of vehicular traffic, diagrams are often used, but with imprecise or even undefined semantics. E.g., time-space diagrams are used to analyse the consequences of different timing behaviours of traffic lights [Koo+08]. The horizontal axis of such a diagram denotes time, while the vertical axis denotes space. Horizontal blocks of alternating green and red represent the state of traffic lights over time. Cars driving along a street are denoted by trajectories. For example, in Fig. 5.34 a single car has to wait at the first intersection at a red light. Afterwards it drives such that it reaches the next intersection during the green phase. These diagrams may then, e.g., describe queuing of several cars due to inappropriate choices for the intervals of traffic lights.

To the best of our knowledge, the only other approach to visually reason explicitly about traffic situations with formal semantics is due to Kemper and Etzien [KE14]. The main application of their language is for the specification and simulation of *advanced driver assistance systems* (ADAS). Within the visual logic, they depict spatial situations to define, what situations are considered. The situations are constrained qualitatively by implying topological relations, i.e., in what order and on which lanes. the cars drive on the road. For quantitative constraints, they use arrows annotated with intervals. For the semantics, the spatial situations are translated to linear constraints over the positions of the cars. Hence the main idea of this part of both the visual language and its semantics is similar to our approach. However, the diagrams differ in the second part of the visual language, which is basically a reduced live sequence chart (see below) connecting the traffic situation with messages passed within the ADAS. They use this visual logic to

analyse whether runs of a simulation satisfy the logical constraints. Due to the lack of logical operators, their approach is not suited to reason about safety properties of traffic.

Since we abstract from the hybrid dynamics of cars in our setting, the systems under consideration may also be regarded as a special kind of real-time systems, i.e., systems which are required to react within certain time intervals. Due to the broadness of the UML framework [RJB04], it is not surprising that a subset suited for specifying real-time system accompanied with tool support [Fab+11] has been developed. While in this approach the overall structure and discrete behaviour may be defined diagrammatically by using class diagrams, component diagrams and statecharts [Har88], the preconditions and effects of method executions are defined in Object-Z [Smi00], while the real-time aspects themselves are given as Duration Calculus (DC) [ZHR91] formulas. For verification purposes, such a specification can be translated into a formal specification language [Hoe06].

In contrast, Kleuker introduced Constraint Diagrams (CD) [Kle00] (not to be confused with the Constraint Diagrams of Kent [Ken97]), which are designed to specify real-time aspects of systems diagrammatically. Their abstract syntax is not constructed within a general formalism but as a mathematical structure specific to the peculiarities of CDs. Constraint Diagrams are given a semantics in terms of Duration Calculus formulas and hence can be used in conjunction with DC by definition.

Live sequence charts (LSCs) [DH01], both an extension and formalisation of message sequence charts (MSCs), are capable of describing real-time aspects of different interacting objects. Objects may synchronise and communicate via messages. The semantics of a LSC is a symbolic transition system induced by the structure of the diagram. Therefore, a partial order on the messages sent and received by each object and the events (e.g. execution of methods) occurring during the existence of an object is defined. As long as only events occur, i.e., no communication between objects takes place, the order in which these events occur is only restricted within one object. This resembles the independence of lanes in traffic diagrams, where the spatial relations between cars on different lanes are only constrained, if distance arrows imply a certain order.

While all of these approaches to real-time systems are suitable for the specification of such systems, none are initially capable of taking spatial aspects into account. While LSCs and the UML diagrams may be augmented with a treatment of space by introducing suited variables and domains, such approaches will lack in intuition and clarity.

Several different approaches to define the abstract syntax of diagrams have been developed, but all fall into two distinct categories, ad-hoc approaches, i.e., mathematical structures which are designed to fit to a specific language, and instances of a general framework. Both approaches have their own benefits and disadvantages. Ad-hoc definitions are well-suited for a diagrammatic language at hand, but are often initially hard to grasp and to define. Examples for this type of abstract syntax are Concept Graphs with negation as defined by Dau [Dau04a] and the Constraint Diagrams of Kleuker [Kle00].

Using a general framework implies on the one hand the need to understand the framework first, before attempting to understand the abstract syntax. Furthermore, the generality may cause unaesthetic parts within the syntax, to overcome the quirks of the framework. On the other hand, defining the syntax within a framework eases relating the diagrams to other languages given in the same framework. In addition, tools to

analyse the frameworks may also be used to examine the abstract syntax. Examples of suited frameworks for the definition of abstract syntax of diagrams are hypergraph rewriting systems [Min00], triple graph grammars [AER98; RS95] or simply the definition of relations of diagrammatic elements in terms of logical formulas [Haa98].

We defined Traffic Diagrams in terms of the general framework of hypergraph rewriting systems. The use of hypergraphs enables us to easily use extensions of rewriting systems in the form of HR^* conditions for the more complicated rules.

For parsing purposes, a typical distinction within the syntax of diagrams using a framework of graph rewriting systems is done between the *spatial relations graphs* (SRG) and the *abstract relations graph* (ARG) [RS95; Min00]. The SRG is a direct abstraction of the concrete syntax, insofar as its edges directly reflect the form of the elements, and not their intended use. This part of the syntax is determined by the ARG. That is, the edges in the SRG are denoting, *how* elements are depicted (e.g. as circles), while the edges in the ARG refer to *what* is meant by the elements (e.g. states of an automaton).

In this presentation, the main focus lies on the definition of the language itself, and not on general parsing, hence we chose to omit a definition for the spatial relations graphs. If we wanted to make use of existing parsing approaches, the tool DiaGen [MV95] based on Minas' work is the most promising candidate to create a parser and editor for Traffic Diagrams. It uses hypergraphs as the underlying representation structure of the diagrams and makes use of hyperedge replacement rules for the creation and analysis of diagrams. Unfortunately, even though DiaGen can use some application conditions for the rewriting rules, it is currently not able to cope with the complex conditions we need for our abstract syntax.

Another example of a language with an abstract syntax defined by graph rewriting systems on hypergraphs is the *graphical security protocol modelling language* (GSPML) [McD05; MA08], intended to specify security protocols. The semantics of GSPML is a labelled transition system. While the graphical syntax of GSPML incorporates inclusion of elements as a spatial aspect, this is only used to define the scopes of parallelism operators. Even though many different production rules are needed for the definition of the syntax, they only rely on embedding contexts as side conditions. In Traffic Diagrams, most of the spatial aspects are also very easily defined, like inclusion in layers, or sequences. However, for the definition of the distance arrows, i.e., the diagrammatic elements with a distinguished spatial interpretation, we need the more complex conditions. This highlights the difficulties spatial interpretations of spatial relations impose not only on the semantics, but also on the abstract syntactic elements and the ways they may possibly be related.

6

Combining Text and Diagrams

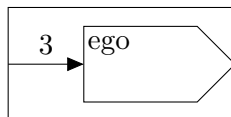
Contents

6.1	Comparison of Expressivity	111
6.2	Verification Framework	121
6.3	Related Work	122

In the previous chapters, we defined the logic EMLSL and the diagrammatic language of Traffic Diagrams, which are both evaluated on the same semantic model. In Sect 6.1, we will discuss the expressive power of Traffic Diagrams in comparison with EMLSL. As it turns out, Traffic Diagrams can be translated to equivalent formulas, but there are properties definable with EMLSL but not with diagrams. We will then sketch in Sect. 6.2 how formulas and diagrams can be used in combination to allow for the exploitation of the advantages of both. For that, we integrate EMLSL formulas into the abstract syntax graph of Traffic Diagrams and define, how they are then evaluated semantically.

6.1 Comparison of Expressivity

While the language of traffic diagrams seems very expressive, it is not as expressive as EMLSL in several ways. Consider for example the formula $\ell = 3 \wedge free \wedge re(ego)$, i.e., there is a free extension behind the reservation of ego with the length 3. Intuitively, an equivalent Traffic Diagram should be



But this is not a syntactically correct Traffic Diagram, since Definition 5.3 requires distance arrows to connect borders of reservations or claims, while the arrow in the diagram above starts at the border of the layer. More generally, Traffic Diagrams are not able to state properties concerning quantitative properties of the whole view under consideration. They may only relate the lengths of and space between reservations and claims.

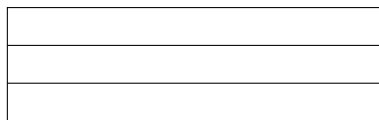


Figure 6.1: Schema of a Spatial Diagram

Furthermore, the diagrams do not allow for arbitrary chops. Consider the schema of a spatial diagram S as given in Fig 6.1 consisting of three topological sequences within a full layer. If we compute the semantics of S with respect to a traffic snapshot \mathcal{TS} , a view V and a valuation ν , we first have to compute the semantics of the lane sequence within S . That is, V will be vertically chopped two times to V_1 , V_2 and V_3 which all have to consist of exactly one lane. Only afterwards do the V_i get horizontally chopped. In general, each atomic spatial diagram consists of a sequence of lanes, each of which is divided horizontally. That is, it is not possible in general to find a corresponding Traffic Diagram for a formula φ , where the horizontal chop is the topmost operator, e.g., if φ is of the form

$$\varphi = \left(\begin{array}{c} \phi_1 \\ \phi_2 \end{array} \right) \frown \left(\begin{array}{c} \psi_1 \\ \psi_2 \end{array} \right).$$

Another restriction of Traffic Diagrams is the lack of a general equality predicate. It is not possible to distinguish two cars C and D , which occupy the same space, except if one of them is the car denoted by *ego*. Then the definition of existential quantification ensures that the bound variable is evaluated to a car different from the owner of the current view. In EMSLS however, we can distinguish between an arbitrary finite number of cars in every formula.

Translation of Traffic Diagrams to EMLSL. Each diagram can be transformed into an equivalent EMLSL formula. Similar to the definition of the semantics of diagrams, we distinguish between the qualitative and the quantitative aspects of layers. For the former, we define the *qualitative transformation function* and for the latter the *metric transformation function*. These transformations are then combined within the complete *transformation function* $\cdot^{\mathcal{T}} : \mathfrak{D} \rightarrow \Phi$. We start with the definition of the metric transformation.

The metric translation mimics the semantics of distance arrows as given in Sect. 5.5. It uses the implicit variables of a diagram by making them explicit. As an example,

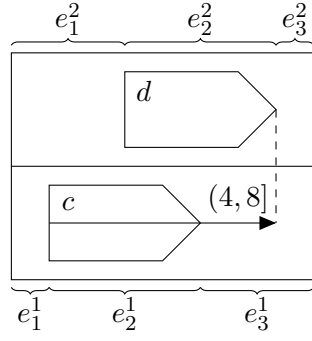


Figure 6.2: Example for the Metric Transformation

consider the Traffic Diagram in Fig. 6.2. Let A be the distance arrow in that diagram. The main idea of the translation is to sum up the length left to the source of A and left of the target of A , and then subtract the former from the latter. Recall that we defined an order \preceq within topological sequences in Sect. 5.5, Def. 5.11. We will denote the edges representing the topological sequences in the lower lane by e_i^1 and similarly e_i^2 for the upper lane as indicated in the figure. Furthermore, let χ be the implicit length function, i.e., the mapping of edges to implicit variables. Then we want the representation of A to be

$$4 < (\chi(e_1^2) + \chi(e_2^2) - \chi(e_1^1)) \wedge (\chi(e_1^2) + \chi(e_2^2) - \chi(e_1^1)) \leq 8 .$$

That is, we sum up the implicit variables of all edges preceding the reservation of c (since A is connected to the left border of this situation) and similarly, we add up the implicit variables of the edges preceding the reservation of d , but include the topological situation labelled d (since A is connected to the right border of it). This approach is formalised in the following definition.

Definition 6.1 (Metric Transformation Function). *Let L be the representation of a lane sequence and S be the set of all hyperedges representing distance arrows within L , i.e., $\forall d \in S \bullet \theta(d) = \text{sarrow}$. Furthermore, let χ be the implicit lengths function of the abstract syntax graph of L and ι its labelling function. For an edge $d \in S$, we will refer to the edge visiting the same node as the s tentacle of d as $s(d)$. Similarly, we use $t(d)$ for the edge visiting the same node as the t tentacle of d . The metric transformation $\cdot^{\mathcal{M}}: \mathfrak{D} \rightarrow \Phi$ is defined by*

$$L^{\mathcal{M}} = \bigwedge_{d \in S} \left(i_l \sim_l \left(\sum_{e \sim_a s(d)} \chi(e) - \sum_{e \sim_b t(d)} \chi(e) \right) \wedge \left(\sum_{e \sim_a s(d)} \chi(e) - \sum_{e \sim_b t(d)} \chi(e) \right) \sim_r i_r \right),$$

where $\sim_a = \prec$ if the tentacle incident with the s tentacle of the edge d is labelled with l and $\sim_a = \preceq$ otherwise (and similar for \sim_b). Furthermore, let $i = \iota(d)$ and i_l (i_r) refer to

the left (right, respectively) border of i . Then $\sim_l = <$ if i is left-open, otherwise $\sim_l = \leq$ and similar for \sim_r .

We use the following definition of the qualitative transformation function to translate the qualitative spatial aspects to suitable EMLSL formulas. The lane separations correspond to vertical chops, either a single one to represent an exact lane separation, or a double chop with an additional formula \top to simulate the semantics of wide lane separations. The contents of topological sequences are translated one after another and connected by horizontal chops. Here, the semantics of the atoms, as well as the definition of *free* ensures, that the parts where these topological sequences are evaluated are non-empty intervals. Additionally, the end of a topological sequence constrains the formula to be evaluated on a single lane.

Definition 6.2 (Qualitative Transformation Function). *Let L and L' be lane sequences, T a topological sequence. Recall that the abstract syntax graph of a topological sequence is an ordered sequence of edges of the types cars, free and true. In the translation, the name of the edge under consideration will be denoted by e . Let the function associating implicit variables with hyperedges be denoted by χ . The qualitative transformation function $\cdot^{\mathcal{Q}} : \mathcal{D} \rightarrow \Phi$ is given by the following definition.*

$$\begin{aligned}
 \left(\begin{array}{c} L \\ \text{---} \\ T \end{array} \right)^{\mathcal{Q}} &= \frac{L^{\mathcal{Q}}}{T^{\mathcal{Q}}} \\
 \left(\begin{array}{c} L \\ \text{---} \\ T \end{array} \right)^{\mathcal{Q}} &= \frac{L^{\mathcal{Q}}}{\top} \\
 \left(\begin{array}{c} \bullet \xrightarrow{s} \text{cars} \xrightarrow{t} \bullet \xrightarrow{s} T \\ \vdots \\ \boxed{(R, C)} \end{array} \right)^{\mathcal{Q}} &= ((\bigwedge_{r \in R} re(r)) \wedge (\bigwedge_{c \in C} cl(c)) \wedge \ell = \chi(e)) \frown T^{\mathcal{Q}} \\
 \left(\begin{array}{c} \bullet \xrightarrow{s} \text{free} \xrightarrow{t} \bullet \xrightarrow{s} T \end{array} \right)^{\mathcal{Q}} &= (free \wedge \ell = \chi(e)) \frown T^{\mathcal{Q}} \\
 \left(\begin{array}{c} \bullet \xrightarrow{s} \text{totrue} \xrightarrow{t} \bullet \xrightarrow{s} T \end{array} \right)^{\mathcal{Q}} &= \ell = \chi(e) \frown T^{\mathcal{Q}} \\
 (T)^{\mathcal{Q}} &= \ell = 0 \wedge \omega = 1, \text{ if } T \text{ is the empty sequence}
 \end{aligned}$$

The complete transformation function is based on the qualitative and metric transformations for the translation of layers. For the translation of implicit variables, we denote the set of implicit variables used in a layer L by $\chi(L)$ and for a set of variables S , we use the notation $\exists S \bullet \varphi$ as an abbreviation for the existential quantification of all elements of S .

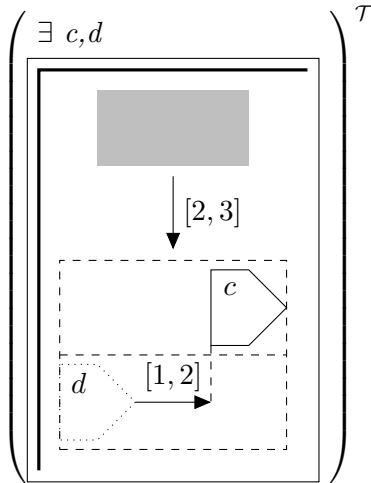
Definition 6.3 (Transformation Function). *In the following, let D, D_1, D_2 be spatial or Traffic Diagrams, S a spatial diagram and L a lane sequence. Furthermore, let χ be the*

implicit lengths function of the abstract syntax. The transformation function $\cdot^{\mathcal{T}} : \mathfrak{D} \rightarrow \Phi$ translating Traffic Diagrams into formulas of EMLSL is given by the following definition.

$$\begin{aligned}
 \left(\boxed{D} \right)^{\mathcal{T}} &= \neg(D^{\mathcal{T}}) \\
 \left(\begin{array}{|c|c|} \hline D_1 & D_2 \\ \hline \end{array} \right)^{\mathcal{T}} &= D_1^{\mathcal{T}} \wedge D_2^{\mathcal{T}} \\
 \left(\begin{array}{|c|} \hline \exists x \\ \hline \boxed{D} \\ \hline \end{array} \right)^{\mathcal{T}} &= \exists x \bullet (x \neq \text{ego} \wedge D^{\mathcal{T}}) \\
 \left(\boxed{L} \right)^{\mathcal{T}} &= \exists \chi(L) \bullet (L^{\mathcal{Q}} \wedge L^{\mathcal{M}}) \\
 \left(\blacksquare \right)^{\mathcal{T}} &= \top \\
 \left(\begin{array}{|c|} \hline L \\ \hline \end{array} \right)^{\mathcal{T}} &= \exists \chi(L) \bullet (\langle L^{\mathcal{Q}} \rangle \wedge L^{\mathcal{M}}) \\
 \left(\begin{array}{c} S \\ \downarrow * \\ D \end{array} \right)^{\mathcal{T}} &= S^{\mathcal{T}} \wedge \diamond_* (D^{\mathcal{T}}) \\
 \left(\begin{array}{c} S \\ \vdots \\ D \end{array} \right)^{\mathcal{T}} &= S^{\mathcal{T}} \wedge \mathbf{F} (D^{\mathcal{T}})
 \end{aligned}$$

That is, on the Boolean operators the translations acts simply as a homomorphism. Each precise temporal arrow is translated to the diamond-like modality referring to the transition the arrow is labelled with. Similarly, a faint arrow corresponds to the use of the finally modality \mathbf{F} . Observe that for quantification, we ensure that the bound variable has a different value than ego, as the semantics in Sect. 5.5 require. Before we prove the soundness of the translation, we give a short example.

Example 6.1. In the translation of the following diagram, we denote the i -th edge in the abstract syntax graph of the j -th topological sequence by e_i^j .



Proposition 6.1. The translation of Traffic Diagrams to EMLSL is sound, i.e., for all traffic snapshots \mathcal{TS} , views V , valuations ν and Traffic Diagrams D ,

$$\mathcal{TS}, V, \nu \models D \quad \text{if and only if} \quad \mathcal{TS}, V, \nu \models D^{\mathcal{T}}.$$

Before we can prove this proposition, we have to show the equivalence of the qualitative properties of the diagram and the equivalence of the interpretation of distance arrows. For proving the equivalence of the qualitative properties, we will employ some intermediate lemmas, similar to the proofs for the uniqueness of abstract syntax graphs given in Chap. 5. That is, we first prove the equivalence for a single topological sequence, and afterwards the equivalence for a lane sequence. Recall that \models_S is the consequence relation for the qualitative aspects of spatial diagrams.

Lemma 6.1. Let T be a topological sequence. Then for all traffic snapshots \mathcal{TS} , views V and valuations ν , we have

$$\mathcal{TS}, V, \nu \models_S T \quad \text{iff} \quad \mathcal{TS}, V, \nu \models T^{\mathcal{Q}}.$$

Proof. By induction on the length of the topological sequence. Let T be a topological sequence consisting of one topological situation and let $G = (\mathcal{V}, \mathcal{E}, \tau, \theta, \mathfrak{l})$ be its representation. I.e., within G , there is exactly one edge e_t of the type lane whose i tentacle visits the same node as the s tentacle of an edge e_t of the type totrue, free or cars.

1. Let e_t be of the type totrue. According to the semantics of lane sequences (Fig. 5.27) and topological sequences (Fig. 5.28), the semantics of G with respect to a traffic snapshot \mathcal{TS} , a view $V = (L, X, E)$ and a valuation ν is

$$\|L\| = 1 \wedge X = X_1 \oplus X_2 \wedge \nu(\chi(e_t)) = \|X_1\| \wedge \|X_2\| = 0.$$

On the other hand, the translation of T is

$$T^{\mathcal{Q}} \equiv \ell = \chi(e_t) \wedge \ell = 0 \wedge \omega = 1.$$

A brief calculation of the semantics of $T^{\mathcal{Q}}$ yields exactly the same conditions for \mathcal{TS} , V and ν .

2. Let e_t be of the type free. The qualitative transformation of T is given as follows:

$$T^{\mathcal{Q}} \equiv \text{free} \wedge \ell = \chi(e_t) \wedge \ell = 0 \wedge \omega = 1.$$

Expanding the abbreviation *free* yields

$$T^{\mathcal{Q}} \equiv \ell > 0 \wedge \omega = 1 \wedge \forall c \bullet \square_{\ell} \neg (cl(c) \vee re(c)) \wedge \ell = \chi(e_t) \wedge \ell = 0 \wedge \omega = 1,$$

and similarly, expressing \square_{ℓ} in terms of horizontal chops and removing the redundant conjunct $\omega = 1$ gives us

$$T^{\mathcal{Q}} \equiv \ell > 0 \wedge \forall c \bullet \neg (\top \wedge (cl(c) \vee re(c)) \wedge \top) \wedge \ell = \chi(e_t) \wedge \ell = 0 \wedge \omega = 1.$$

Observe that this formula requires a view $V = (L, X, E)$ to be chopped horizontally, i.e., $V = V_1 \oplus V_2$, such that the extension X_2 of V_2 has the measure 0 and the extension X_1 of V_1 both greater than zero and exactly the value of $\chi(e_t)$. Now, the semantics of G is

$$\begin{aligned} |L| = 1 \wedge X = X_1 \oplus X_2 \wedge \nu(\chi(e_t)) = \|X_1\| \wedge \|X_1\| > 0 \wedge \|X_2\| = 0 \\ \wedge \forall C \in \mathbb{I} \bullet \text{len}_V(C) \cap \mathfrak{J}(X_1) = \emptyset \vee \text{res}_V(C) \cup \text{clm}_V(C) = \emptyset . \end{aligned}$$

Observe that for the extensions of V_1 and V_2 as well as the set of lanes, this formula expresses similar restrictions as the translation. Moreover, it also requires the measure of X_1 to have the same value as the value of $\chi(e_t)$.

Hence we only have to show the equivalence between the universally quantified conjuncts. Let \mathcal{TS} be a traffic snapshot, V a view and ν a valuation. Note that $\|X_2\| = 0$ implies $X = X_1$. I.e., we will reason about X , to avoid unnecessary subscripts. Assume that there is a car C such that $\text{len}_V(C) \cap \mathfrak{J}(X) \neq \emptyset$ and $\text{res}_V(C) \cup \text{clm}_V(C) \neq \emptyset$. According to Convention 3.2, $\text{len}_V(C)$ cannot be a singleton set, i.e., the only possible difference between $\text{len}_V(C)$ and $\text{len}_V(C) \cap \mathfrak{J}(X)$ may be the lack of (one or both) borders of the latter. However, within $\text{len}_V(C) \cap \mathfrak{J}(X)$ there exists a closed non-empty interval X_C . Now let us consider the subview $V_C = (X_C, L, E)$ of V , i.e., its restriction to the extension X_C . Then $\text{len}_{V_C}(C) = X_C$. Because the set of lanes is shared between V and V_C , we also know $\text{res}_{V_C}(C) \cup \text{clm}_{V_C}(C) \neq \emptyset$, i.e., at least one of the sets $\text{res}_{V_C}(C)$ and $\text{clm}_{V_C}(C)$ is non-empty. From these two observations, we get that on the subview V_C of V_1 either $\mathcal{TS}, V_1', \nu \oplus \{c \mapsto C\} \models \text{re}(c)$ or $\mathcal{TS}, V_1', \nu \oplus \{c \mapsto C\} \models \text{cl}(c)$ holds. Hence $\mathcal{TS}, V_1, \nu \models \neg \forall c \bullet \neg(\top \wedge (\text{cl}(c) \vee \text{re}(c)) \wedge \top)$. The other direction is similar.

3. Let e_t be of the type cars and let $\mathfrak{l}(e_t) = (R, C)$. The semantics of G is

$$\begin{aligned} |L| = 1 \wedge X = X_1 \oplus X_2 \wedge \nu(\chi(e_t)) = \|X_1\| \wedge \|X_1\| > 0 \wedge \|X_2\| = 0 \\ \wedge \forall c \in R \bullet \text{res}(\nu(c)) = L \wedge \text{len}_{V_1}(\nu(c)) = X_1 \\ \wedge \forall c \in C \bullet \text{clm}(\nu(c)) = L \wedge \text{len}_{V_1}(\nu(c)) = X_1 , \end{aligned}$$

where $V_1 = (L, X_1, E)$. The translation of T yields

$$\left(\left(\bigwedge_{r \in R} \text{re}(r) \right) \wedge \left(\bigwedge_{c \in C} \text{cl}(c) \right) \wedge \ell = \chi(e_t) \right) \wedge \ell = 0 \wedge \omega = 1 .$$

This formula ensures that V can be chopped into two subviews, where the second one is of length 0. Since $\omega = 1$ is horizontally rigid, both views have to consist of exactly one lane, i.e. V also satisfies $|L| = 1$. Furthermore, the atoms $\text{re}(c)$ and $\text{cl}(c)$ both require the view V_1 to have a non-empty extension, i.e. $\|X_1\| > 0$, and the length constraint ensures $\|X_1\| = \nu(\chi(e_t))$. The remaining constraints are also part of the semantics of the atoms.

Now assume that for all sequences T_s with a length smaller or equal to n , we have

$$\mathcal{TS}, V, \nu \models_S T_s \quad \text{iff} \quad \mathcal{TS}, V, \nu \models T_s^{\mathcal{Q}} .$$

Consider a topological sequence T of length $n + 1$ such that $\mathcal{TS}, V, \nu \models_S T$. Let e_t be the last edge of T . Removing e_t yields a sequence T_s , of length n . The semantics of each of the edges within a topological sequence needs the view under consideration to be chopped horizontally into two subviews, i.e., we get that there have to be two views V_1 and V_2 such that $V = V_1 \oplus V_2$ and $\mathcal{TS}, V_1, \nu \models_S T_s$ and $\mathcal{TS}, V_2, \nu \models_S e_t$.

By the induction hypothesis, we know that $\mathcal{TS}, V_1, \nu \models T_s^{\mathcal{Q}}$. This formula requires V_1 to consist of exactly one lane, and hence implies that also V and V_2 consist of this lane. Now similar to the induction base, we can calculate the value of the transformation function for e_t and compare this formula to the semantics of e_t to get the equivalence. \square

Lemma 6.2. Let L be a lane sequence. Then for all traffic snapshots \mathcal{TS} , views V and valuations ν , we have

$$\mathcal{TS}, V, \nu \models_S L \quad \text{iff} \quad \mathcal{TS}, V, \nu \models L^{\mathcal{Q}} .$$

Proof. By induction on the length of lane sequences, i.e., the number of topological situations within the sequence. If L has the length 1, it consists of only one topological sequence T , for which Lemma 6.1 yields the equivalence. Now assume that for all lane sequences L_s of lengths smaller or equal to n , the equivalence holds. Consider L of length $n + 1$ such that $\mathcal{TS}, V, \nu \models_S L$ with $V = (L, X, E)$. Then L consists of a lane sequence L_s of length n , an additional topological sequence L_t and an edge e_s , which connects L_t with the last sequence of L_s and represents the lane separation.

1. Let $\theta(e_s) = \text{exact}$. Then by the semantics of L , there are two sets of lanes such that $L = L_1 \oplus L_2$ and $\mathcal{TS}, V^{L_1}, \nu \models_S L_s$ and $\mathcal{TS}, V^{L_2}, \nu \models_S L_t$. Since L_t is a single topological sequence, we get by Lemma 6.1 that $\mathcal{TS}, V^{L_2}, \nu \models L_t^{\mathcal{Q}}$. The induction hypothesis yields $\mathcal{TS}, V^{L_1}, \nu \models L_s^{\mathcal{Q}}$. So overall we get that there have to exist L_1 and L_2 such that $L = L_1 \oplus L_2$, $\mathcal{TS}, V^{L_1}, \nu \models L_s^{\mathcal{Q}}$ and $\mathcal{TS}, V^{L_2}, \nu \models L_t^{\mathcal{Q}}$. That is, we have that $\mathcal{TS}, V, \nu \models_S L$ is equivalent to

$$\mathcal{TS}, V, \nu \models \begin{array}{c} L_t^{\mathcal{Q}} \\ L_s^{\mathcal{Q}} \end{array}$$

which is by definition equivalent to $\mathcal{TS}, V, \nu \models L^{\mathcal{Q}}$.

2. Let $\theta(e_s) = \text{wide}$. This case is similar to the previous one, with the exception that we have to have sets of lanes such that $L = L_1 \oplus L_m$, $L_m = L'_m \oplus L_2$ with $\mathcal{TS}, V^{L_1}, \nu \models L_s^{\mathcal{Q}}$ and $\mathcal{TS}, V^{L_2}, \nu \models L_t^{\mathcal{Q}}$. Since all views and snapshots satisfy \top , we then get that $\mathcal{TS}, V, \nu \models_S L$ is equivalent to

$$\mathcal{TS}, V, \nu \models \begin{array}{c} L_t^{\mathcal{Q}} \\ \top \\ L_s^{\mathcal{Q}} \end{array}$$

which is by definition equivalent to $\mathcal{TS}, V, \nu \models L^{\mathcal{Q}}$.

Hence in both cases the lemma holds. \square

Now we can proceed to prove Prop. 6.1. For the qualitative spatial aspects of Traffic Diagrams we use the preceding lemmas. Then, we mainly need to concentrate on the metric transformation and the translation of the temporal arrows.

Proof of Prop. 6.1. We proceed by induction on the structure of Traffic Diagrams. In the induction base we will prove the equivalence of the satisfaction of an atomic spatial diagram with its translation to EMLSL. First observe that the metric semantics of a single distance arrow is similar to the metric transformation of just this one arrow. The only difference is that in the diagram semantics the distance is required to be an element of the interval defined by the label, while the metric transformation explicitly states the relations between the borders of the interval and the result of the difference. Hence both formalisations are equivalent.

Now let S be an atomic spatial diagram. Then S consists of either a full or partial layer together with its contents, or of a single shaded rectangle. For the last case, the proposition is easily verified. So we now consider the other cases.

1. Let $\mathcal{TS}, V, \nu \models S$, where S consist of a full layer. Furthermore let L be the lane sequence within S . Then by the semantics of S , we know that for all implicit variables $\tilde{v} = v_0, \dots, v_n$, of L , there are elements $\tilde{\alpha} = \alpha_0, \dots, \alpha_n$, such that $\mathcal{TS}, V, \nu \oplus \{\tilde{v} \mapsto \tilde{\alpha}\} \models_S L$ and $\mathcal{TS}, V, \nu \oplus \{\tilde{v} \mapsto \tilde{\alpha}\} \models_M L$. By the remark at the beginning of the proof concerning the metric transformation, we have $\mathcal{TS}, V, \nu \oplus \{\tilde{v} \mapsto \tilde{\alpha}\} \models_M L$ iff $\mathcal{TS}, V, \nu \oplus \{\tilde{v} \mapsto \tilde{\alpha}\} \models L^M$ and by Lemma 6.2, we have $\mathcal{TS}, V, \nu \oplus \{\tilde{v} \mapsto \tilde{\alpha}\} \models L^Q$. So, all in all, we have $\mathcal{TS}, V, \nu \oplus \{\tilde{v} \mapsto \tilde{\alpha}\} \models L^Q \wedge L^M$. By the semantics of existential quantification, this is equivalent to $\mathcal{TS}, V, \nu \models \exists \chi(L) \bullet (L^Q \wedge L^M)$, which is by definition equivalent to $\mathcal{TS}, V, \nu \models S^T$.
2. This case is similar to the previous one, except that the semantics of S require a subview of V to satisfy the metric and qualitative aspects of the lane sequence L . Observe that the formula L^M is rigid. Hence it is not important, whether we interpret it on V or a subview of V . That L^Q is satisfied by a subview V' of V means that we can both vertically and horizontally chop V to get V' . That is $\mathcal{TS}, V', \nu \models L^Q$ is equivalent to $\mathcal{TS}, V, \nu \models \langle L^Q \rangle$. Hence the equivalence holds also for this case.

For the induction hypothesis, assume that the proposition holds for the Traffic Diagrams D_1 and D_2 as well as for the spatial diagram S . In the induction step, the cases for the Boolean operators and quantification are standard. Let D be the Traffic Diagram consisting of a precise temporal arrow A connecting S and D_1 and assume $\mathcal{TS}, V, \nu \models D$. Let A be a precise arrow labelled with a real-valued interval i . The semantics of D yields that $\mathcal{TS}, V, \nu \models S$ and that there exists another snapshot \mathcal{TS}' and a duration $t \in \mathbb{T}$ such that $\mathcal{TS} \xrightarrow{t} \mathcal{TS}'$ with $t \in i$ and $\mathcal{TS}', V', \nu \models D_1$, where $V' = mv_{\mathcal{TS}'}^{\mathcal{TS}}(V)$. By the induction hypothesis, we get that $\mathcal{TS}, V, \nu \models S^T$ and $\mathcal{TS}', V', \nu \models D_1^T$. That is, $\mathcal{TS}, V, \nu \models S^T \wedge \diamond_i(D_1^T)$, which is by definition equivalent to $\mathcal{TS}, V, \nu \models D^T$. The cases for the other types of temporal arrows are similar. \square

6.2 Verification Framework

In this section we will describe how the logic and the diagrams defined in Chap. 4 and 5 can be combined to gain a *heterogeneous reasoning system*. Furthermore, we describe in what ways we assume this combined system to be helpful to reasoners about traffic situations.

We base our heterogeneous reasoning system on the concrete and abstract syntax of Traffic Diagrams defined in Sect. 5.1 and 5.4. Formulas may be written at any place, where an atomic diagram or a temporal sequence may be drawn according to Definition 5.5. For the abstract syntax, the set of types Θ gets extended by a new terminal type *formula*, which denotes the existence of a formula at this position. Therefore, we require that for each edge e with $\theta(e) = \text{formula}$, the labelling function l returns the corresponding formula φ , i.e., $l(e) = \varphi$. To achieve that these edges may appear at the correct positions within the abstract syntax graph, we need a new rewriting rule. Recall that the rules R_{BD} given in Fig. 5.23 create the logical structure of Traffic Diagrams, where the rule inserting edges of the type *sequence* define the place, where either an atomic diagram or a temporal sequence resides. Hence we introduce a new rule for the replacement of the non-terminal B^D as shown in Fig. 6.3.

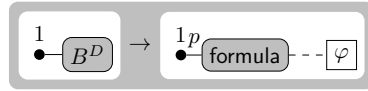


Figure 6.3: New Rule R_{BD}^5 : Formulas ($\varphi \in \Phi$)

The satisfaction relation for Traffic Diagrams as given in Def. 5.10 has to be extended to cope with edges of this new type. However, the extension is straightforward. A traffic snapshot \mathcal{TS} , a view V and a valuation ν satisfy the graph consisting of the edge e with $\theta(e) = \text{formula}$ and $l(e) = \varphi$ iff $\mathcal{TS}, V, \nu \models \varphi$.

In this way, we can exploit the advantages of formulas and diagrams alike. While the diagrams allow for a very intuitive specification of the spatial relations of cars (quantitative as well as qualitative), formulas let us express more involved properties as well as assertions about identities of cars. For example, while a constraint for the endpoints of reservations of cars on different lanes is easily expressible with Traffic Diagrams, the length of the whole view can be measured with the EMLSL formula $\ell = x$. Since Traffic Diagrams and formulas may use the same variables, this measurement may also be used within Traffic Diagrams.

An example combining an EMLSL formula within a Traffic Diagram is given in Fig. 6.4. The diagrammatic part constrains that one reservation of c lies in front of one reservation of ego, and the EMLSL formula that no reservation of another car overlaps with c 's reservation. In general, the main advantage of this combination is due to the equality constraints of EMLSL, and the arbitrary possibilities to chop and nest the behavioural modalities.

We now describe the method how the combination of Traffic Diagrams and EMLSL

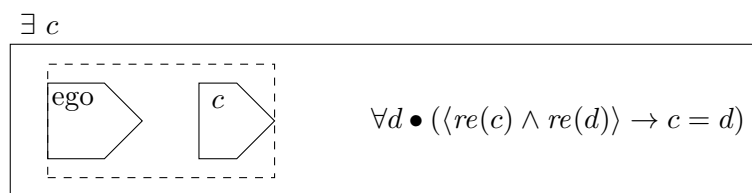


Figure 6.4: Example of a Combination of EMLSL and Traffic Diagrams

formulas can and should from our point of view be used. For that, recall that we only gave a proof system for EMLSL, and no similar system for Traffic Diagrams. Hence, to explicitly and syntactically prove that a system possesses a certain property, the system has to be given as formulas of EMLSL.

Our proposed methodology is as follows. First, describe a minimal system S and prove that this system fulfils the verification property P . Then, augment this system with suitable Traffic Diagrams and formulas to further constrain the system behaviour, resulting in the system S' . If P is a safety property, these additional constraints cannot harm the validity of the proof done in the first step. In this way, the core specification S can be gradually refined to get the more realistic system S' , while still retaining the safety of S .

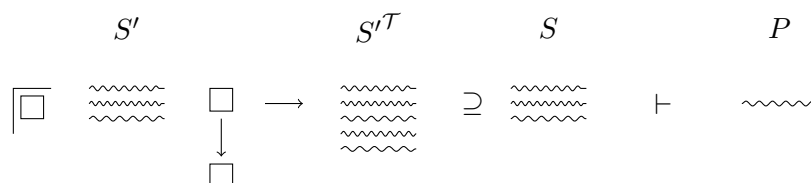


Figure 6.5: Proving Safety with EMLSL and Traffic Diagrams

Figure 6.5 visualises this method. The wiggly lines denote formulas of EMLSL, while the boxes, hooks and vertical arrows represent Traffic Diagrams. Observe that the diagrams in S' can be translated into equivalent EMLSL formulas. Hence by the addition of these diagrams, we get a set S'^T of EMLSL formulas, which is a superset of the set S used to derive P . Since provability is monotone with respect to finite sets of assumptions, we can still find a suitable derivation of P .

6.3 Related Work

The ancestor of all heterogeneous logical systems (i.e., logical systems which use more than one language) is the *Hyperproof* system [BE92; BE96], which uses Tarski's world as an example for teaching purposes. It integrates first-order logic with predicates over the spatial relations and properties of the elements within the world with a (possibly incomplete) diagrammatic representation of a world. The student then has to draw

conclusions from both the logical statements as well as the graphical representation to finally give a concrete world which satisfies both specifications. The *Openproof* system [Bar+08] is a generalisation of Hyperproof since it is not restricted to a particular application but allows for using arbitrary textual and diagrammatic representations of the underlying semantics. Neither of these approaches uses a so-called *interlingua*, i.e., a language into which both representations are translated for reasoning purposes. The system presented in this chapter is similar, insofar as EMLSL is given in terms of textual definitions and Traffic Diagrams are based on an abstract syntax of hypergraphs. However, as shown in Sect. 6.1, we could translate each diagram into EMLSL, which would then serve as an interlingua. Even though we did not define proof rules for Traffic Diagrams, this approach would directly enable the user to use the proof system of Sect. 4.2 for the verification of heterogeneous specifications.

The combination of Traffic Diagrams and EMLSL is not strictly a heterogeneous reasoning system in the sense of Barwise and Etchemendy [BE96; Shi04], since it lacks not only proof rules for Traffic Diagrams but also transformation rules, which allow for the transfer of information from EMLSL to Traffic Diagrams and vice versa. However, the similar semantical model of both languages make an analysis in the terms of situation theory, as used by Barwise and Etchemendy [BE90], worthwhile. In their paper, instead of using a model of truth, they introduced a model of “information” in terms of distributive lattices to justify the heterogeneous proof rules within Hyperproof. Since they explicitly refer to partial information, it would be interesting to compare their approach to the views and different sensor implementations of the abstract model (c.f. Chap. 3).

As discussed in Sect. 4.4, an implementation of this proof system within the context of Isabelle seems promising. For further integration of Traffic Diagrams into such an implementation, an examination of *Diabelli* [UJ12] is worthwhile. *Diabelli* gives the possibility to use spider diagrams within Isabelle [Pau94] by providing an integration with a diagrammatic theorem prover called Speedith [Urb+12]. Urbas and Jamnik further generalised the approach of *Diabelli* [UJ14] to almost arbitrary combinations of sentential and diagrammatic theorem provers. They distinguish between *master* and *slave* reasoners. The master reasoner provides the more meta-theoretical aspects, e.g., the concept of a proof, while the slave reasoner only has to provide inference rules. For our purposes, using simple diagrammatic inference rules in combination with a more general implementation of the proofs system for EMLSL in Isabelle would be probably appropriate.

All these heterogeneous systems concentrate on the derivation of theorems from a set of assumptions. An heterogeneous proof system defined explicitly for verification purposes is the system of *ribbon proofs* [WDP13; Bea06]. The goal of this system is to ensure program correctness [Hoa69] based on separation logic [Rey02]. While the definition of ribbon proofs is mainly diagrammatic, the current implementation in Isabelle needs textual inputs to prove intermediate steps. However, even within the definition, excerpts describing the current heap structure similar to the notions in separation logic are used, hence ribbon proofs are more heterogeneous than purely graphical in our opinion. Similar to Traffic Diagrams, they employ a relatively involved graph structure for syntax, i.e., nested hypergraphs.

7

Case study

Contents

7.1	Controller Specification	126
7.2	Safety Proof	127
7.3	Refining the Specification	133

The main motivation for the definition of EMLSL and Traffic Diagrams was the derivation of a safe protocol to perform lane change manoeuvres. In our previous work, we presented a specification as an extended timed automaton [Hil+11], where guards and invariants could also contain formulas of (E)MLSL. However, the semantics of these automata and the safety proof were only given informally. Only recently, within her Master’s thesis, Schwammberger took steps towards a completely formal semantics for these kinds of automata [Sch14].

In this chapter, we take a denotational approach. That is, we give a specification for controllers as several EMLSL formulas. While the automata mentioned above only specify the behaviour of a single car, we explicitly require all cars to behave similarly. Furthermore, our presentation here is much more abstract. We only constrain some of the possible transitions, in particular the creation of reservations. All other transitions may occur almost without limitations. The only other constrain we impose is on the dynamical behaviour. In contrast, a specification using automata would explicitly state which transitions can be taken by a car according to the current state of the controller. Hence, the controller specification given in Sect. 7.1 is less restrictive. In Sect. 7.2, we use the proof system presented in Chap. 4 to derive the desired safety predicate. Finally, we use the diagrams given in Chap. 5 to present a specification which is closer to possible implementations. For that we combine the specification of Sect. 7.1 with a set of diagrams and get a heterogeneous specification in the sense of Chap. 6.

7.1 Controller Specification

We proceed in three steps. First, we present the safety predicate $\text{safe}(e)$, i.e., the formula whose validity the controllers shall ensure during all possible transition sequences. Intuitively, it states that there is no car c , whose reservation overlaps anywhere with the reservation of the car e :

$$\text{safe}(e) \stackrel{df}{\Leftrightarrow} \neg \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle .$$

The safety predicate used in our previous work was $\text{safe}(\text{ego})$, i.e., there we only proved safety with respect to the special car ego . However, since we assumed all cars to be alike, we could use an informal symmetry argument to derive the safety predicate for all cars. In the present form however, we will be more concrete about the invariant. First, we will show that with the appropriate controllers, $\mathbf{G} \text{safe}(e)$ holds, i.e., $\text{safe}(e)$ is an invariant along all transition sequences. Since we will not make any additional assumptions on e , we can then derive that $\mathbf{G} \text{safe}(e)$ holds for all cars e , i.e., we get $\forall e \bullet \mathbf{G} \text{safe}(e)$.

In the specification of the controllers, we need to be aware of spatial situations which may result in unsafe behaviour. In particular, we have to prohibit the cars to mutate their claims to reservations if these claims already overlap with any parts of other cars. We call these situations *potential collisions*, and the following formula $pc(c, d)$ the *potential collision check*:

$$pc(c, d) \stackrel{df}{\Leftrightarrow} c \neq d \wedge \langle cl(d) \wedge (re(c) \vee cl(c)) \rangle .$$

The definition of the potential collision check is the same as for the informal safety proof [Hil+11]. Observe that this definition is a bit more restrictive than needed. We could omit the disjunction with $cl(c)$, and still get a safe controller specification, due to our use of interleaving semantics. If the claims of two different cars overlap, one car has to be first to change its claim to a reservation. Then, the other car could identify the resulting situation as a potential collision and react appropriately. However, we still chose to use this slightly more restrictive definition, since it does not complicate the proofs very much, and will also ensure safety if we assume synchronous creation of reservations.

Now we proceed with the specification of what we call the *distance controller*. To that end, recall that $\Box_\tau \phi$ is an abbreviation for $\Box_{[0, \infty)} \phi$ (cf. Convention 4.1). This formula ensures that safe situations will be preserved under all possible evolutions, i.e., as long as only time passes and accelerations are changed:

$$\text{DC} \stackrel{df}{\Leftrightarrow} \mathbf{G} \forall c, d \bullet c \neq d \rightarrow (\neg \langle re(c) \wedge re(d) \rangle \rightarrow \Box_\tau \neg \langle re(c) \wedge re(d) \rangle) .$$

This definition is a direct formalisation of the very important assumption A2 of the informal proof:

Assumption A2. [Hil+11] Every car C is equipped with a *distance controller* that keeps the safety property invariant under time and acceleration transitions, i.e., for every transition $\mathcal{TS} \xrightarrow{t} \mathcal{TS}'$ and $\mathcal{TS} \xrightarrow{\text{acc}(C, a)} \mathcal{TS}'$ if \mathcal{TS} is safe also \mathcal{TS}' is safe.

The premises of DC state that the traffic snapshot \mathcal{TS} is safe. Then, the conclusion ensures that under all possible evolutions, i.e., transition sequences, where only the dynamics of cars may change and time may pass in between, the safety property is preserved. The invariance modality at the front of the formula guarantees that the distance controller keeps working during all transition sequences. Note that if spatial transitions occur, DC does not ensure safety of the resulting snapshot.

Finally, the *lane change controller* restricting the discrete transitions is given by the following formula LC. We employ the potential collision check to prohibit the creation of unsafe reservations:

$$\text{LC} \stackrel{df}{\Leftrightarrow} \mathbf{G} \forall d \bullet (\exists c \bullet pc(c, d) \rightarrow \Box_{r(d)} \perp) .$$

This formula defines that whenever the claim of a car denoted by d overlaps with the claim or reservation of another car, d is not allowed to mutate its claim into a reservation. For that we use the subformula $\Box_{r(d)} \perp$. It states that after for all transitions creating the reservation of d , the resulting traffic snapshot, view and valuation have to satisfy \perp . Since no model satisfies \perp , no such transition is possible. Observe that this formula neither prohibits any other transitions nor does it enforce the withdrawal of the claim which is responsible for the potential collision.

7.2 Safety Proof

We want to show that under the right circumstances, the formula $\forall e \bullet \mathbf{safe}(e)$ is an invariant along all transition sequences. In particular, we want that whenever we start with a safe situation and all cars use a controller implementation respecting the specification shown in Sec 7.1, $\forall e \bullet \mathbf{G} \mathbf{safe}(e)$ is true. Using the system of Labelled Natural Deduction (cf. Sect. 4.2), this means we have to prove

$$(\{ts, v: \text{DC}, ts, v: \text{LC}, ts, v: \forall e \bullet \mathbf{safe}(e)\}, \emptyset) \vdash ts, v: \forall e \bullet \mathbf{G} \mathbf{safe}(e) .$$

Since e does not occur free in any element of $\Gamma = \{ts, v: \text{DC}, ts, v: \text{LC}, ts, v: \forall e \bullet \mathbf{safe}(e)\}$, we can proceed to prove

$$(\Gamma, \emptyset) \vdash ts, v: \mathbf{G} \mathbf{safe}(e)$$

and apply $\forall\text{I}$ afterwards. To apply the invariance rule, we have to show that $\mathbf{safe}(e)$ is derivable under all possible transitions, i.e., we have to prove the existence of the following five derivations.

$$\begin{aligned}
& (\Gamma \cup \{ts_1, v_1 : \text{safe}(e)\}, \{ts, v \Rightarrow ts_1, v_1, ts_1, v_1 \xrightarrow{r(d)} ts'_1, v'_1\}) \vdash ts'_1, v'_1 : \text{safe}(e) \\
& (\Gamma \cup \{ts_2, v_2 : \text{safe}(e)\}, \{ts, v \Rightarrow ts_2, v_2, ts_2, v_2 \xrightarrow{c(d)} ts'_2, v'_2\}) \vdash ts'_2, v'_2 : \text{safe}(e) \\
& (\Gamma \cup \{ts_3, v_3 : \text{safe}(e)\}, \{ts, v \Rightarrow ts_3, v_3, ts_3, v_3 \xrightarrow{\text{wd } r(d)} ts'_3, v'_3\}) \vdash ts'_3, v'_3 : \text{safe}(e) \\
& (\Gamma \cup \{ts_4, v_4 : \text{safe}(e)\}, \{ts, v \Rightarrow ts_4, v_4, ts_4, v_4 \xrightarrow{\text{wd } c(d)} ts'_4, v'_4\}) \vdash ts'_4, v'_4 : \text{safe}(e) \\
& (\Gamma \cup \{ts_5, v_5 : \text{safe}(e)\}, \{ts, v \Rightarrow ts_5, v_5, ts_5, v_5 \xrightarrow{t} ts'_5, v'_5\}) \vdash ts'_5, v'_5 : \text{safe}(e)
\end{aligned}$$

We will now prove these propositions one after the other. The main idea is always the same: For the i -th proposition, we assume that there is an overlap between two reservations on the world ts'_i, v'_i and show that this results in a contradiction with the assumptions. Formally, we assume $ts'_i, v'_i : \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle$ and derive $ts'_i, v'_i : \perp$. Then we can use the derived rule $\neg\text{I}$ (cf. Sect.2.3) to get $ts'_i, v'_i : \text{safe}(e)$.

In all of the following proofs, we will use $\Delta_{v'_i}$ as a shorthand for the locality formulas $v'_i = v'_L \oplus v'_m, v'_m = v''_m \oplus v'_R, v''_m = v'_D \ominus v'''_m$ and $v'''_m = v_i^{\text{sub}} \ominus v'_U$. These are needed to denote the part of v_i , where the overlap between cars occurs. Note that no problem arises by the use of similar names v'_L, \dots within the different derivations in the following proofs, since all these assumptions will be eliminated before we apply the invariance rule.

The following derivation, denoted by $\Pi_i^{*,j}$ where $*$ \in $\{r(d), c(d), \text{wd } r(d), \text{wd } c(d)\}$, is used as an auxiliary proof tree to relate the world of our assumptions to the subview where $re(e) \wedge re(c)$ holds. This is possible, since the discrete transitions do not change the view (cf. Sect. 4.2.1). Within the tree, $i \in \{1, \dots, 4\}$ is the index of the traffic snapshots and views, while j denotes the rule used to eliminate the assumptions.

$$\frac{\frac{\frac{ts_i, v_i \xrightarrow{*} ts'_i, v'_i}{ts_i, v'_i \xrightarrow{*} ts'_i, v'_i} [v'_i = v'_L \oplus v'_m]_j}{ts_i, v'_m \xrightarrow{*} ts'_i, v'_m} [v'_m = v''_m \oplus v'_R]_j}{ts_i, v''_m \xrightarrow{*} ts'_i, v''_m} [v''_m = v'_D \ominus v'''_m]_j}{ts_i, v'''_m \xrightarrow{*} ts'_i, v'''_m} [v'''_m = v_i^{\text{sub}} \ominus v'_U]_j}{ts_i, v_i^{\text{sub}} \xrightarrow{*} ts'_i, v_i^{\text{sub}}}$$

Lemma 7.1. The safety predicate is preserved if some car creates a reservation, i.e.,

$$(\Gamma \cup \{ts_1, v_1 : \text{safe}(e)\}, \{ts, v \Rightarrow ts_1, v_1, ts_1, v_1 \xrightarrow{r(d)} ts'_1, v'_1\}) \vdash ts'_1, v'_1 : \text{safe}(e) .$$

Proof. We start with the root of the proof tree.

$$\frac{\frac{[ts'_1, v'_1 : \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_7}{ts'_1, v'_1 : \perp} \exists\text{E}_6}{\frac{[ts'_1, v'_1 : c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_6}{ts'_1, v'_1 : \langle re(e) \wedge re(c) \rangle} \Pi_{\perp}} \langle \text{E}_5}{ts'_1, v'_1 : \perp} \neg\text{I}_7$$

For the derivation Π_{\perp} , we make a first case distinction between the cases where $d = e$ and $d \neq e$, i.e., whether the car denoted by e is creating the reservation or not. In principle, both cases are very similar, which is why we concentrate on the first case. That is, we will not specify the proof tree $\Pi_{d \neq e}$.

$$\frac{\frac{ts_1, v_1^{sub} : d = e \vee d \neq e}{\frac{\frac{\frac{\Pi_{\leftarrow} \quad \Pi_{re(e)} \quad \Pi_{cl(e)}}{ts_1', v_1^{sub} : re(e) \vee cl(e)} \quad ts_1', v_1' : \perp \quad ts_1', v_1' : \perp}{\vee E_3} \quad \Pi_{d \neq e}}{ts_1', v_1' : \perp} \vee E_4}}{ts_1', v_1' : \perp}}$$

The derivation Π_{\leftarrow} is basically an application of the backwards activity rule for the creation of a reservation.

$$\frac{\frac{[ts_1', v_1^{sub} : re(e) \wedge re(c)]_5}{ts_1', v_1^{sub} : re(e)} \quad \Pi_1^{r(d),5} \quad \frac{ts_1, v_1^{sub} \xrightarrow{r(d)} ts_1', v_1^{sub} \quad [ts_1, v_1^{sub} : d = e]_4}{ts_1', v_1^{sub} : re(e) \vee cl(e)}}{ts_1', v_1^{sub} : re(e) \wedge re(c)}$$

The main part of the proof lies in the derivations $\Pi_{re(e)}$ and $\Pi_{cl(e)}$. In both cases, we have to derive a contradiction. In the former, we will use the assumption $ts_1, v_1 : \mathbf{safe}(e)$, and in the latter the specification of the lane change controller. We start with $\Pi_{re(e)}$. The beginning of this derivation is another case distinction. This time we have to distinguish the cases $d = c$ and $d \neq c$. The first case is a contradiction to the assumption that $c \neq e$.

$$\frac{\frac{ts_1, v_1^{sub} : d = c \vee d \neq c}{\frac{\frac{\Pi_{trans} \quad [ts_1', v_1' : c \neq e \wedge (re(e) \wedge re(c))]_6}{ts_1', v_1' : c = e \quad ts_1', v_1' : c \neq e}{\vee E_1} \quad \Pi_{d \neq c}}{ts_1', v_1' : \perp} \vee E_1}}{ts_1', v_1' : \perp}}$$

In the tree Π_{trans} , we derive $c = e$ by transitivity, symmetry and rigidity of equality.

$$\frac{\frac{[ts_1', v_1^{sub} : d = c]_1}{ts_1', v_1^{sub} : c = d} \quad [ts_1', v_1^{sub} : d = e]_4}{ts_1', v_1^{sub} : c = e}}{ts_1', v_1^{sub} : c = e}}$$

Now we proceed with the case that $d \neq c$, i.e., $\Pi_{d \neq c}$.

$$\frac{\frac{ts_1, v_1 \xrightarrow{r(d)} ts_1', v_1' \quad \frac{\Pi_{\langle re(e) \rangle} \quad [ts_1', v_1' : c \neq e \wedge (re(e) \wedge re(c))]_6}{ts_1, v_1' : \langle re(e) \wedge re(c) \rangle} \quad \frac{ts_1', v_1' : c \neq e}{ts_1, v_1' : c \neq e}}{ts_1, v_1 : \langle re(e) \wedge re(c) \rangle} \quad \frac{ts_1, v_1 : c \neq e \wedge \langle re(e) \wedge re(c) \rangle}{ts_1, v_1 : \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle}}{ts_1, v_1 : \mathbf{safe}(e)} \quad \frac{ts_1, v_1 : \perp}{ts_1', v_1' : \perp}}$$

We can derive the existence of an overlap by using the backwards stability rule for the creation of a reservation, since we know that c is not the car creating the reservation.

$$\frac{\frac{[ts'_1, v_1^{sub}: re(e) \wedge re(c)]_5}{ts'_1, v_1^{sub}: re(c)} \quad \frac{\Pi_1^{r(d),5}}{ts_1, v_1^{sub} \xrightarrow{r(d)} ts'_1, v_1^{sub}} \quad [ts_1, v_1^{sub}: d \neq c]_1}{ts_1, v_1^{sub}: re(c)} \quad \frac{[ts_1, v_1^{sub}: re(e)]_3}{ts_1, v_1^{sub}: re(e) \wedge re(c)}}{[\Delta_{v'_1}]_5} \quad \frac{ts_1, v'_1: \langle re(e) \wedge re(c) \rangle}{ts_1, v'_1: \langle re(e) \wedge re(c) \rangle}$$

We are now finished with the case where a reservation of e was present on ts_1, v_1^{sub} and turn our attention to the presence of a claim at this world, i.e., we create the derivation $\Pi_{cl(e)}$. In this part of the derivation, we will finally make use of our lane change controller specification. Like for the other case, we begin with a case distinction between $d = c$ and $d \neq c$. The first case is derived similarly as before, and hence omitted. The idea for the other case is to derive an overlap between the claim of e and a reservation or claim of c and employ the controller specification to derive $\Box_{r(e)} \perp$. Then we can employ Subst_{ri} to replace e with d and get the desired contradiction.

$$\frac{\frac{ts_1, v_1^{sub}: d = c \vee d \neq c}{ts_1, v_1^{sub}: d = c \vee d \neq c} \quad \frac{\Pi_{d=c}}{ts'_1, v'_1: \perp}}{ts'_1, v'_1: \perp} \quad \frac{\frac{\Pi_{LC}}{ts_1, v_1: \Box_{r(e)} \perp} \quad \frac{[ts'_1, v_1^{sub}: d = e]_4}{ts_1, v_1: d = e}}{ts_1, v_1: \Box_{r(d)} \perp} \quad \frac{ts_1, v_1 \xrightarrow{r(d)} ts'_1, v'_1}{ts'_1, v'_1: \perp} \quad \vee E_2}{ts'_1, v'_1: \perp}$$

In the derivation Π_{LC} , we use the controller specification and the fact that we could derive that the potential collision check pc is satisfied.

$$\frac{\frac{\Pi_{pcc}}{ts_1, v_1: \exists c \bullet c \neq e \wedge \langle cl(e) \wedge (re(c) \vee cl(c)) \rangle}}{ts_1, v_1: \Box_{r(e)} \perp} \quad \frac{\frac{ts_1, v_1: LC \quad ts, v \Rightarrow ts_1, v_1}{ts_1, v_1: \forall d \bullet (\exists c \bullet pc(c, d) \rightarrow \Box_{r(d)} \perp)}}{ts_1, v_1: \exists c \bullet pc(c, e) \rightarrow \Box_{r(e)} \perp}}{ts_1, v_1: \Box_{r(e)} \perp}$$

Now we can turn our attention to the derivation of the potential collision Π_{pcc} . This is similar to the case, where a reservation of e resides at the subview v_1^{sub} , i.e., the final goal is to use the stability rule. But first, we have to introduce the existential quantifier, the somewhere modality and replace v'_1 by v_1 .

$$\frac{\frac{\frac{[ts'_1, v'_1: c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_6}{ts'_1, v'_1: c \neq e}}{ts_1, v_1: c \neq e} \quad \frac{\frac{[ts_1, v_1^{sub}: cl(e)]_3}{ts_1, v_1^{sub}: cl(e) \wedge (re(c) \vee cl(c))} \quad \frac{\Pi_c}{ts_1, v_1^{sub}: re(c)}}{ts_1, v_1^{sub}: re(c) \vee cl(c)} \quad \frac{[ts_1, v_1^{sub}: cl(e)]_3}{ts_1, v_1^{sub}: cl(e) \wedge (re(c) \vee cl(c))} \quad \frac{ts_1, v_1 \xrightarrow{r(d)} ts'_1, v'_1}{ts_1, v_1: \langle cl(e) \wedge (re(c) \vee cl(c)) \rangle}}{ts_1, v_1: \langle cl(e) \wedge (re(c) \vee cl(c)) \rangle}}{ts_1, v_1: c \neq e \wedge \langle cl(e) \wedge (re(c) \vee cl(c)) \rangle} \quad \frac{ts_1, v_1: c \neq e \wedge \langle cl(e) \wedge (re(c) \vee cl(c)) \rangle}{ts_1, v_1: \exists c \bullet c \neq e \wedge \langle cl(e) \wedge (re(c) \vee cl(c)) \rangle}}$$

Finally, we arrive at the application of the stability rule in the derivation Π_c .

$$\frac{\frac{[ts'_1, v_1^{sub}: re(e) \wedge re(c)]_5}{ts'_1, v_1^{sub}: re(c)} \quad \frac{\Pi_1^{r(d),5}}{ts_1, v_1^{sub} \xrightarrow{r(d)} ts'_1, v_1^{sub}} \quad [ts_1, v_1^{sub}: d \neq c]_2}{ts_1, v_1^{sub}: re(c)}}{ts_1, v_1^{sub}: re(c)}$$

Now we constructed a well formed proof tree, where the only open assumptions are either elements of Γ or of $\{ts, v \Rightarrow ts_1, v_1, ts_1, v_1 \xrightarrow{r(d)} ts'_1, v'_1, ts_1, v_1 : \text{safe}(e)\}$. \square

Lemma 7.2. The safety predicate is preserved if some car creates a claim, i.e.,

$$(\Gamma \cup \{ts_2, v_2 : \text{safe}(e)\}, \{ts, v \Rightarrow ts_2, v_2, ts_2, v_2 \xrightarrow{c(d)} ts'_2, v'_2\}) \vdash ts'_2, v'_2 : \text{safe}(e) .$$

Proof. We begin like in the proof of Lemma 7.1 by presenting the root of the tree, to motivate the upper parts of the derivation.

$$\frac{\frac{[ts'_2, v'_2 : \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_5}{ts'_2, v'_2 : \perp} \quad \frac{\frac{[ts'_2, v'_2 : c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_4}{ts'_2, v'_2 : \langle re(e) \wedge re(c) \rangle} \quad \frac{ts_2, v_2 : \perp}{ts'_2, v'_2 : \perp} \Pi_1}{ts'_2, v'_2 : \perp} \exists E_4}{ts'_2, v'_2 : \neg \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle} \neg I_5 \quad \langle \rangle E_3$$

Now we have to show how to derive the contradiction on ts_2, v_2 . This is done in the tree Π_{\perp} . Note the use of the view invariance over discrete transitions in the rule IV.

$$\text{IV} \frac{\frac{ts_2, v_2 \xrightarrow{c(d)} ts'_2, v'_2}{ts_2, v_2 : \langle re(e) \wedge re(c) \rangle} \quad \frac{\Pi_{\langle \rangle} \quad [ts'_2, v'_2 : c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_4}{ts'_2, v'_2 : c \neq e} \quad \frac{ts_2, v_2 : \langle re(e) \wedge re(c) \rangle}{ts_2, v_2 : c \neq e}}{ts_2, v_2 : c \neq e \wedge \langle re(e) \wedge re(c) \rangle} \quad \frac{ts_2, v_2 : \text{safe}(e)}{ts_2, v_2 : \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle} \quad \frac{ts_2, v_2 : \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle}{ts_2, v_2 : \perp}$$

The tree $\Pi_{\langle \rangle}$ combines the results from the trees $\Pi_{re(e) \neq}$, $\Pi_{re(e) =}$ and $\Pi_{re(c)}$. We will only show the trees concerning the variable e in detail, since $\Pi_{re(c)}$ is similar.

$$\frac{\frac{ts_2, v_2^{sub} : d \neq e \vee d = e}{ts_2, v_2^{sub} : re(e)} \quad \frac{\Pi_{re(e) \neq} \quad \Pi_{re(e) =}}{ts_2, v_2^{sub} : re(e)} \vee E_1 \quad \frac{\Pi_{re(c)}}{ts_2, v_2^{sub} : re(c)}}{ts_2, v'_2 : re(e) \wedge re(c)} \quad \frac{ts_2, v'_2 : re(e) \wedge re(c)}{ts_2, v'_2 : \langle re(e) \wedge re(c) \rangle} [\Delta_{v'_2}]_3$$

The trees $\Pi_{re(e) \neq}$ and $\Pi_{re(e) =}$ are similar, with the exception that $ts_2, v_2^{sub} : d \neq e$ has to be substituted with $ts_2, v_2^{sub} : d = e$. The backwards stability rule for the creation of claims is still applicable, since reservations are not changed while creating claims.

$$\frac{\frac{[ts'_2, v_2^{sub} : re(e) \wedge re(c)]_3}{ts'_2, v_2^{sub} : re(e)} \quad \frac{\Pi_2^{c(d),3}}{ts_2, v_2^{sub} \xrightarrow{c(d)} ts'_2, v_2^{sub} [ts_2, v_2^{sub} : d \neq e]_1}}{ts_2, v_2^{sub} : re(e)}$$

The derivation tree $\Pi_{re(c)}$ is similar to the tree consisting of $\Pi_{re(e) =}$, $\Pi_{re(e) \neq}$ and the application of a disjunction elimination $\vee E_2$, where we substitute c for e . \square

Lemma 7.3. The safety predicate is preserved if some car withdraws one of its reservations, i.e.,

$$(\Gamma \cup \{ts_3, v_3 : \mathbf{safe}(e)\}, \{ts, v \Rightarrow ts_3, v_3, ts_3, v_3 \xrightarrow{\text{wd } r(d)} ts'_3, v'_3\}) \vdash ts'_3, v'_3 : \mathbf{safe}(e) .$$

Proof. The proof tree is structurally similar to the proof of Lemma 7.2, except that all indices are changed to 3, all occurrences of $ts_2, v_2 \xrightarrow{c(d)} ts'_2, v'_2$ are exchanged with $ts_3, v_3 \xrightarrow{\text{wd } r(d)} ts'_3, v'_3$ and all applications of $\leftarrow^{c(d)}\text{S}$ are substituted with applications of $\leftarrow^{\text{wd } r(d)}\text{S}$. \square

Lemma 7.4. The safety predicate is preserved if some car withdraws its claim, i.e.,

$$(\Gamma \cup \{ts_4, v_4 : \mathbf{safe}(e)\}, \{ts, v \Rightarrow ts_4, v_4, ts_4, v_4 \xrightarrow{\text{wd } c(d)} ts'_4, v'_4\}) \vdash ts'_4, v'_4 : \mathbf{safe}(e) .$$

Proof. Like the proof for Lemma 7.3, the proof tree is structurally similar to the proof of Lemma 7.2, with the appropriate changes. \square

Lemma 7.5. The safety predicate is preserved if time passes and accelerations change, i.e.,

$$(\Gamma \cup \{ts_5, v_5 : \mathbf{safe}(e)\}, \{ts, v \Rightarrow ts_5, v_5, ts_5, v_5 \xrightarrow{t} ts'_5, v'_5\}) \vdash ts'_5, v'_5 : \mathbf{safe}(e)$$

Proof. The root of the tree is similar to all the other derivations, i.e. we derive a contradiction on ts_5, v_5 .

$$\frac{\frac{\frac{\frac{\frac{\Pi_{\langle \rangle}}{ts_5, v_5 : \langle re(e) \wedge re(c) \rangle} \quad \frac{[ts'_5, v'_5 : c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_2}{ts'_5, v'_5 : c \neq e}}{ts_5, v_5 : c \neq e}}{ts_5, v_5 : c \neq e \wedge \langle re(e) \wedge re(c) \rangle} \quad ts_5, v_5 : \mathbf{safe}(e)}{ts_5, v_5 : \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle} \quad \frac{ts_5, v_5 : \perp}{ts'_5, v'_5 : \perp} \exists E_2}{ts'_5, v'_5 : \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_3}{ts'_5, v'_5 : \perp} \exists E_2}{ts'_5, v'_5 : \neg \exists c \bullet c \neq e \wedge \langle re(e) \wedge re(c) \rangle} \neg I_3$$

To derive the existence of an overlap on ts_5, v_5 in the tree $\Pi_{\langle \rangle}$, we use the definition of \Diamond_{τ} .

$$\frac{\frac{\frac{[ts'_5, v'_5 : c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_2}{ts'_5, v'_5 : \langle re(e) \wedge re(c) \rangle} \quad ts_5, v_5 \xrightarrow{t} ts'_5, v'_5}{ts_5, v_5 : \Diamond_{\tau} \langle re(e) \wedge re(c) \rangle} \equiv \quad \Pi_{DC}}{ts_5, v_5 : \neg \Box_{\tau} \neg \langle re(e) \wedge re(c) \rangle} \equiv \quad \frac{ts_5, v_5 : \perp}{ts_5, v_5 : \langle re(e) \wedge re(c) \rangle} \perp E_1}{ts_5, v_5 : \langle re(e) \wedge re(c) \rangle} \perp E_1$$

In the following derivation Π_{DC} , we use the abbreviations

$$DC' \equiv c \neq e \rightarrow (\neg \langle re(e) \wedge re(c) \rangle \rightarrow \Box_{\tau} \neg \langle re(e) \wedge re(c) \rangle)$$

and DC'_d for $DC'[e \mapsto d]$.

$$\frac{
 \frac{
 \frac{
 \frac{
 ts, v: DC \quad ts, v \Rightarrow ts_5, v_5
 }{
 ts_5, v_5: \forall c, d \bullet DC'_d
 }
 }{
 ts_5, v_5: \forall d \bullet DC'_d
 }
 }{
 ts_5, v_5: DC'
 }
 }{
 ts_5, v_5: \neg \langle re(e) \wedge re(c) \rangle \rightarrow \Box_{\tau} \neg \langle re(e) \wedge re(c) \rangle
 }
 }{
 ts_5, v_5: \Box_{\tau} \neg \langle re(e) \wedge re(c) \rangle
 }
 \quad
 \frac{
 [ts'_5, v'_5: c \neq e \wedge \langle re(e) \wedge re(c) \rangle]_2
 }{
 ts'_5, v'_5: c \neq e
 }
 }{
 ts_5, v_5: c \neq e
 }
 \quad
 [ts_5, v_5: \neg \langle re(e) \wedge re(c) \rangle]_1$$

Now we have constructed a well-formed derivation of the desired form. \square

We are now in the position to apply the invariance rule to get the desired result, i.e., the safety predicate is an invariant under all transitions allowed by our controllers DC and LC.

Theorem 7.1. The safety predicate is an invariant for all cars using the controller specification given in Sect. 7.1, i.e.,

$$(\{ts, v: DC, ts, v: LC, ts, v: \forall e \bullet \mathbf{safe}(e)\}, \emptyset) \vdash ts, v: \forall e \bullet \mathbf{G safe}(e) .$$

Proof. We can derive

$$(\Gamma, \emptyset) \vdash ts, v: \mathbf{safe}(e)$$

with a single application of $\forall E$. Then we can employ Lemma 7.1 to Lemma 7.5, to get the derivations needed for the application of \mathbf{GI} , i.e.,

$$(\Gamma, \emptyset) \vdash ts, v: \mathbf{G safe}(e) .$$

Observe that all application conditions of \mathbf{GI} are satisfied, since each world ts_i, v_i only occurs in the derivation of $ts'_i, v'_i: \mathbf{safe}(e)$ and d (the car referred to in the dynamic formulas) does not occur free in any element of Γ . As stated before, since e does not occur free in any element of Γ , we can use $\forall I$ to derive

$$(\Gamma, \emptyset) \vdash ts, v: \forall e \bullet \mathbf{G safe}(e) .$$

\square

7.3 Refining the Specification

Now that we defined which rules a safe controller must adhere to, we give several examples for a more refined controller specification. However, the formulas LC and DC will still be at the heart of each of these specifications. Our modelling language is the combination of traffic diagrams and EMLSL as presented in Chap. 6. Still, we will allow for many non-deterministic choices of an actual controller implementation. In the following, we assume that the controller specification is a conjunction of both LC and DC as well as the diagrams presented below.

First, we define several possible positive conditions for the creation of reservations. Note that the controller specification of Sect. 7.1 only restricts such transitions, but by no means ensures that any reservation will ever be created.

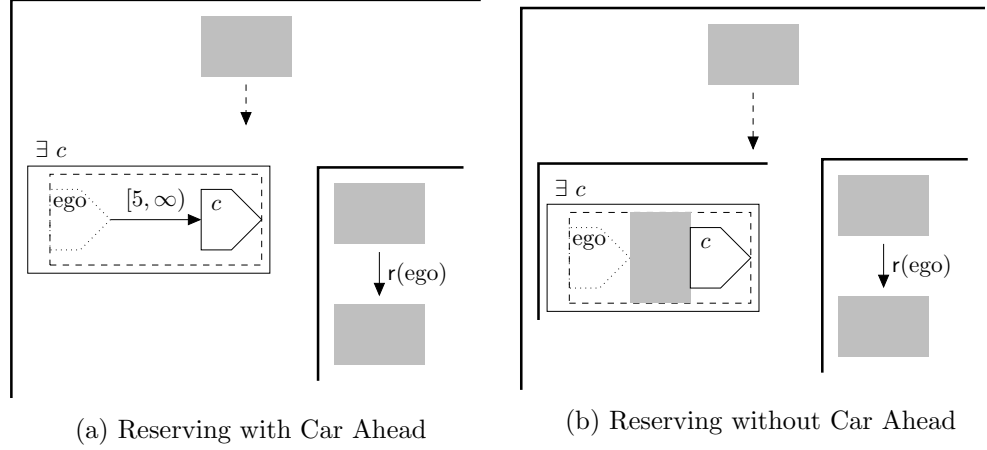
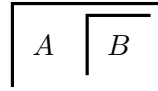


Figure 7.1: Positive Conditions for the Creation of Reservations

The diagram in Fig. 7.1a describes that the car ego may create a reservation, whenever the next car is at least 5 meters ahead. However, this condition will not ensure the creation of any reservation, when no car c is present. Hence we also introduce the diagram in Fig. 7.1b, which explicitly states the absence of a car ahead of ego as a precondition. For reading these diagrams, recall that a diagram of the form



is basically the implication $A \rightarrow B$. Furthermore, by definition we know that $\mathbf{F}\phi \equiv \neg\mathbf{G}\neg\phi$, i.e., $\neg\mathbf{F}\phi \equiv \mathbf{G}\neg\phi$. So, using this equivalence and thinking of the faint temporal arrow as an invariance rather than an existence statement, the subdiagrams after the arrow can be thought of as an implication, where the left diagram implies the right one. Observe that we do not need to state that there exist no potential collisions for ego. For if a potential collision existed as well as a situation satisfying e.g. Fig. 7.1a, the formula LC implies that the creation of a reservation results in a contradiction, i.e., the transition would still not be allowed.

Figure 7.2 defines a liveness property for the withdrawal of reservations. The overall structure is similar to the previous figures. However, the conclusion states only that there will finally be a transition withdrawing one of the reservations. It does not ensure that a actual lane change happened and it does not require the withdrawal to happen within a specified time frame.

For an upper bound for the withdrawal of a reservation, we would need a metric extension of the invariance modality. That is, the modality $\mathbf{F}_{[a,b]}\varphi$ would ensure that there is an abstract transition, in which time between a and b passes and the resulting snapshot satisfies φ . With the existing temporal modality \square_i or the precise temporal arrows, we require that no discrete transitions apart from changes of accelerations occur. Hence we would needlessly (and unjustifiably) restrict the behaviour of all cars on the freeway.

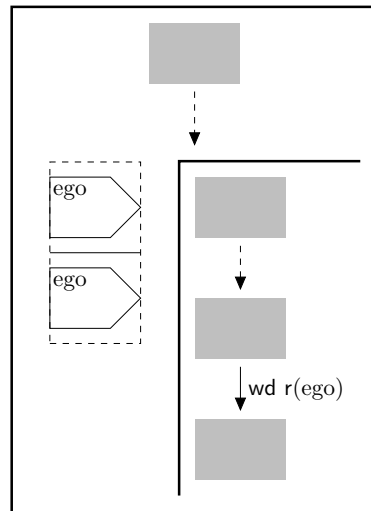


Figure 7.2: Withdrawal of Reservations

To specify that an actual lane change occurred, we would have to refer to the original lane of the car. In Fig. 7.2 however, we cannot distinguish anymore between the lane the car started the lane change on and the target lane. We could try to overcome this problem by adding a layer denoting, e.g., that the upper lane contained the claim of ego in between the unspecified space at the beginning of the sequence and the implication. However, the faint temporal arrows are not expressive enough to state that the described withdrawal is the *first* withdrawal of ego after the lane change. Still, the diagram describes a minimal liveness property every controller should satisfy.

We can also refine the behaviour of the distance controller for specific situations. The traffic diagrams in Fig. 7.3a and 7.3b describe behaviour of a distance controller which adds more restrictions on DC. In particular, the diagrams ensure that whenever there is free space in front of the car ego, this free space is preserved during time transitions. That is, similar to DC, the specifications of Fig. 7.3 do not restrict situations where overlaps occur due to the creation of claims or reservations. The specification has to be split into two diagrams concerning situations outside and during a lane change manoeuvre, since we cannot distinguish the lanes during such a manoeuvre. If we just used Fig. 7.3a and dropped the condition that the car is not using two lanes at once, the specification would still be satisfied if the free space on one lane was consumed, as long as the free space on the other lane is preserved. Using both diagrams ensures the preservation of space in both cases. Even though the diagrams seem to define a stronger controller at first sight, the property they ensure is weaker than DC. For example, if the reservation of a car c is directly adjacent to the reservation of ego, the diagrams do not guarantee anything, while DC still implies that the reservations will not overlap after an arbitrary time.

The diagrams given above only concretise the controller implemented in the car ego. For more concrete controller implementations for the rest of cars, we would have to repeat all these diagrams, replacing all occurrences of ego with a new variable d and

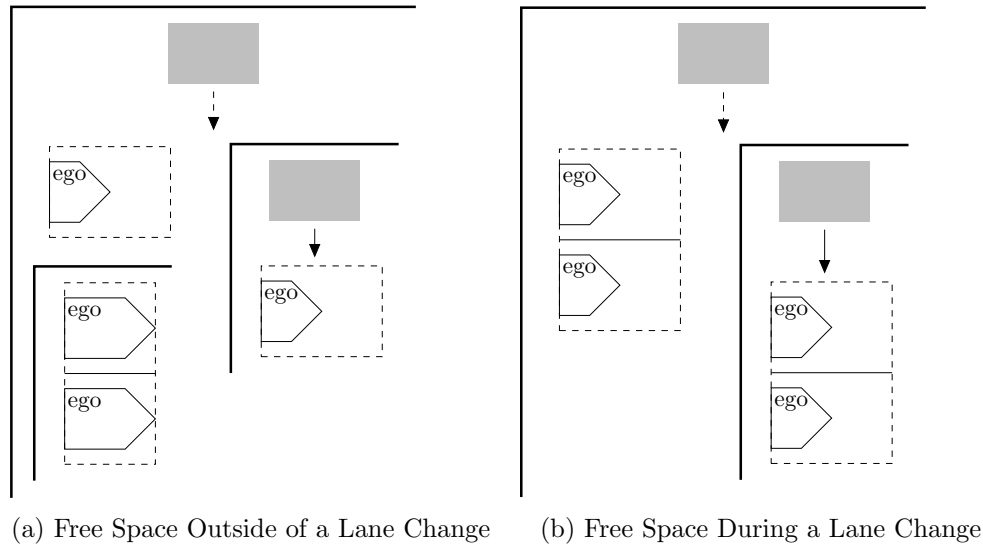


Figure 7.3: Distance Controller Preserving Free Space in Front of Cars

quantifying d existentially for the sequences under the outermost negations. To avoid this duplication, we could change the semantics of the quantifiers and remove the need for the cars to be different from ego. Such a change would decrease the expressive power of pure traffic diagrams, since ego could not be distinguished from other cars any more. However, in the combination with EMLSL, we could simply adjoin the inequality $c \neq \text{ego}$ for each quantification of a variable c .

8

Conclusion

Contents

8.1 Summary	137
8.2 Future Work	138

In this final chapter, we first summarise the content of the thesis in Sect. 8.1. Afterwards, in Sect. 8.2, we give several possibilities how the results of this work may be extended.

8.1 Summary

We presented an abstract model of freeway traffic, where space and its occupancy is the main focus. The dynamics of cars are mostly hidden within the model, and emphasis lies on the discrete changes cars may perform, in particular setting their turn signals and changing lanes. Furthermore, we defined how the models are reduced to a finite part of the freeway, to give a notion of locality as basic parts the models.

Instances of these models serve as the semantic structures for the extended multi-lane spatial logic EMLSL. This logic takes methods from interval logics and typical modal logic, to form a many-dimensional multi-modal logic, which may be used to reason about freeway traffic. Satisfiability of EMLSL formulas was shown to be undecidable, but nevertheless, we defined a proof system in the style of Natural Deduction, which was proven to be sound. Several derived rules enhance the usability of the proof system.

To make the approach of this thesis more feasible to end-users, we presented the visual language of Traffic Diagrams. To give an idea what types of properties are expressible with Traffic Diagrams, we gave exemplary diagrams which capture prominent features of the intended semantic model. The underlying structure of the depictions of Traffic Diagrams was formalised in terms of hypergraphs, which reflect both the diagrammatic elements, as well as important relations between these elements. We proved that each

diagram possesses a unique hypergraph as its type. Using these hypergraphs, we defined a semantics based on the same models as EMLSL. We sketched a possibility to decide satisfiability of a subclass of Traffic Diagrams.

Since both EMLSL and Traffic Diagrams share the same semantic model, they may be used in combination to exploit the advantages of each language. We formalised such a combination by incorporating the formulas into the hypergraphs representing the syntax of the diagrams. By showing the limits of Traffic Diagrams and defining a translation into EMLSL, the diagrams were shown to be less expressive than the formulas.

Finally, we used EMLSL to present minimal specifications for two controller. The first one avoids collisions while time passes, while the second one ensures a safe lane-change. This specification was proven to ensure safety for all possible transition sequences using the proof system for EMLSL. We then use different Traffic Diagrams to refine this specification, e.g., to guarantee a limited form of liveness.

8.2 Future Work

In this section, we describe possibilities to extend the work presented in the thesis. Even though most of these extensions affect the model itself, e.g., to incorporate notions of robustness or different topologies, other extensions concern tool support and the identification of new decidable subsets of the formalisms.

Different Models of Sensors As stated in Chap. 3, we only considered settings where every car knows both the position and safety envelope of all other cars within its view. Obviously, such a requirement needs perfect sensors installed in the cars. If we permit more realistic implementations of sensors, i.e., a situation where each car knows its own distance needed for an emergency braking and only the physical size of the other cars, our approach does not ensure safety of traffic anymore. Within such a setting explicit communication is needed for a safe lane-change protocol, as shown in previous work [Hil+11]. Hence to reason about safety in such a setting, we need to incorporate means of describing communications between several agents within our logic. Furthermore, we probably need methods to relate views of different owners to each other. That is, the current emphasis on a single, distinguished car has to be more explicit within the logic itself, with additional syntactic elements to switch views within a single formula.

Robustness Another possibility to increase the realism of our model is to allow for incorrect sensor values, within certain bounds. That is, instead of the exact satisfaction relation defined in Chap. 4, a more robust notion allowing for small deviances is needed. This also leads to the question, into which parts of the model such disturbances should be introduced. It seems reasonable to at least allow for errors in distances measurements, since these are dependent on sensor readings. Similarly, no car can possibly ensure to change its state at a single point in time, which implies that robustness within transitions may also be considered. In particular, that the discrete transitions of Chap. 3 do not take any time has to be questioned.

Synchronous Parallelism Even though traffic on freeways can be thought of as an inherently parallel system of multiple agents, our model is defined using an interleaving semantics. That is, no two transitions occur exactly simultaneously. In particular, in every trace, the order of consecutive discrete transitions is fixed, while the transitions may stem from different cars. However, in a system of multiple, autonomous agents, a model of true concurrency is more realistic. Hence the labelled transition system implied by our model could be replaced by either an event structure or a distributed transition system [Lod+92], which both allow for the simultaneous occurrence of transitions.

Connecting the Abstract Model with Dynamics We presented only a simple definition of the dynamics in Chap. 3. For concrete implementations, our approach has to be connected with more concrete specifications, e.g., in terms of hybrid automata or simply differential equations. A concrete model of the dynamical behaviour of cars has to fulfil several constraints. It has to be able to return the length needed for a safe emergency braking to the upper level (i.e., it has to compute the values of the sensor function at least for the current, distinguished car), and it has to ensure that each car only drives on the lanes it reserved. Furthermore, the dynamical model has to ensure, that the cars as well as the values of the sensor function only evolve continuously.

Increasing the Domain of Application To analyse different scenarios, the abstract model has to be changed severely. Even though, oncoming traffic may be incorporated rather easily [HLO13], modelling urban traffic scenarios, i.e., crossings, is more of a challenge. The topology of the model has to be changed drastically to allow for different streets to cross. However, vastly different topologies may also be of interest to describe other traffic scenarios, for example airborne traffic, where air planes reserve corridors in three-dimensional space. Furthermore, lanes which follow certain trajectories may be useful to reason about shipping traffic.

Decidability As shown in Chap. 4, our main tool of reasoning, the logic EMLSL, is undecidable in general. It is not directly obvious, what restrictions on the model or the syntax yield decidable subsets. A promising approach is to restrict the number of cars to be finite, to get a translation into a decidable subset of Schäfer's Shape Calculus [Sch07]. Restricting the number and nesting of chops is also sensible to use the "dove-tailing" approach of Gabbay [Gab98; Gab+03]. The satisfiability problem of atomic spatial diagrams probably is decidable, as sketched in Chap. 5. However, the procedure still has to be formally proven correct, and enhanced to cope with a larger subset of Traffic Diagrams.

Tool Support Several parts of the work in this thesis suggest the implementation of a tool, or rather a framework incorporating different tools. The proof system presented in Chap. 4 could be implemented within a general theorem prover. Isabelle [Pau94] seems like a suitable selection, since it is based on natural deduction, and since several labelled deductive systems for modal- and interval logics have already been implemented within

Isabelle [BMV98; Ras02; Vig00]. For the diagrammatic system of Chap. 5, a graphical editor is of inherent importance, to ease the use and creation of diagrams. With such an editor, an implementation of the translation given in Chap. 6 is sensible to gain a direct connection between the implementations of both. Then, techniques for heterogeneous reasoning as implemented in, e.g., the Diabelli system [UJ12] can probably be adapted to our setting.

Bibliography

- [AB96] G. Allwein and J. Barwise, eds. *Logical Reasoning with Diagrams*. Oxford University Press, 1996.
- [Abr10] J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.
- [AER98] M. Andries, G. Engels, and J. Rekers. “How to Represent a Visual Specification”. In: *Visual Language Theory*. Ed. by K. Marriott and B. Meyer. Springer, 1998.
- [All83] J. F. Allen. “Maintaining knowledge about temporal intervals”. In: *Communications of the ACM* 26.11 (1983), pp. 832–843.
- [APB07] M. Aiello, I. Pratt-Hartmann, and J. van Benthem, eds. *Handbook of Spatial Logics*. Springer, 2007.
- [Bar+08] D. Barker-Plummer, J. Etchemendy, A. Liu, M. Murray, and N. Swoboda. “Openproof—a flexible framework for heterogeneous reasoning”. In: *Diagrammatic Representation and Inference – International Conference on the Theory and Applications of Diagrams – DIAGRAMS 2008*. Ed. by G. Stapleton, J. Howse, and J. Lee. Vol. 5223. LNAI. Springer, 2008, pp. 347–349.
- [BB13] R. Banach and M. Butler. “Cruise Control in Hybrid Event-B”. In: *International Colloquium on Theoretical Aspects of Computing – ICTAC 2013*. Ed. by Z. Liu, J. Woodcock, and Huibiao Z. Vol. 8049. LNCS. Springer, 2013, pp. 76–93.
- [BB14] R. Banach and M. Butler. “A Hybrid Event-B Study of Lane Centering”. In: *International Conference on Complex Systems Design & Management – CSD&M 2013*. Ed. by M. Aiguier, F. Boulanger, D. Krob, and C. Marchal. Springer, 2014, pp. 97–111.
- [BE90] J. Barwise and J. Etchemendy. “Information, Infons, and Inference”. In: *Situation Theory and its applications*. Ed. by R. Cooper, K. Mukai, and J. Perry. Vol. 1. CSLI Lecture Notes Number 22. CSLI: Center for the Study of Language and Information, 1990, pp. 33–78.
- [BE92] J. Barwise and J. Etchemendy. “Hyperproof: Logical reasoning with diagrams”. In: *Working Notes of the AAAI Spring Symposium on Reasoning with Diagrammatic Representations*. 1992.

- [BE96] J. Barwise and J. Etchemendy. “Heterogeneous Logic”. In: *Logical Reasoning with Diagrams*. Ed. by G. Allwein and J. Barwise. Oxford University Press, 1996, pp. 179–200.
- [Bea06] J. M. L. Bean. “Ribbon Proofs: A Proof System for the Logic of Bunched Implications”. PhD thesis. 2006.
- [Ben86] E. Bencivenga. “Free Logics”. In: *Handbook of Philosophical Logic*. Ed. by D. M. Gabbay and F. Guenther. Vol. 166. Springer, 1986, pp. 373–426.
- [BKK03] P. Baldan, B. König, and B. König. “A Logic for Analyzing Abstractions of Graph Transformation Systems”. In: *International Symposium on Static Analysis – SAS 2003*. Ed. by R. Cousot. Vol. 2694. LNCS. Springer, 2003, pp. 255–272.
- [BMV98] D. Basin, S. Matthews, and L. Viganò. “Natural Deduction for Non-Classical Logics”. In: *Studia Logica* 60 (1 1998), pp. 119–160.
- [BRZ99] R. Barua, S. Roy, and Zhou C. “Completeness of Neighbourhood Logic”. In: *Symposium on Theoretical Aspects of Computer Science – STACS 99*. Ed. by C. Meinel and S. Tison. Vol. 1563. LNCS. Springer, 1999, pp. 521–530.
- [Bya+09] A. B. Byachkov, C. Cattani, E. M. Nosova, and M. P. Yushkov. “The Simplest Model of the Turning Movement of a Car with its Possible Sideslip”. In: *TECHNISCHE MECHANIK* 29.1 (2009), pp. 1–12.
- [CC01] L. Caires and L. Cardelli. “A spatial logic for concurrency (Part I)”. In: *International Symposium on Theoretical Aspects of Computer Software – TACS 2001*. Ed. by N. Kobayashi and B. C. Pierce. Vol. 2215. LNCS. Springer. 2001, pp. 1–37.
- [CC04] L. Caires and L. Cardelli. “A spatial logic for concurrency (Part II)”. In: *Theoretical Computer Science* 322.3 (2004), pp. 517–565.
- [CG98] L. Cardelli and A. D Gordon. “Mobile ambients”. In: *International Conference on Foundations of Software Science and Computation Structures – FoSSaCS 1998*. Ed. by M. Nivat. Vol. 1378. LNCS. Springer. 1998, pp. 140–155.
- [Coo72] D. C. Cooper. “Theorem proving in arithmetic without multiplication”. In: *Machine Intelligence* 7.91–99 (1972), p. 300.
- [CSS05] C. Caleiro, A. Sernadas, and C. Sernadas. “Fibring Logics: Past, Present and Future”. In: *We Will Show Them! Essays in Honour of Dov Gabbay, Volume 1*. Ed. by S. Artemov, H. Barringer, A. d’Avila Garcez, L. C. Lamb, and J. Woods. Colledge Publications, 2005, pp. 363–388.
- [CSS11] M. E. Coniglio, A. Sernadas, and C. Sernadas. “Preservation by fibring of the finite model property”. In: *Journal of Logic and Computation* 21.2 (2011), pp. 375–402.
- [Dau04a] F. Dau. *The Logic System of Concept Graphs with Negation*. Vol. 2892. LNAI. Springer, 2004.

- [Dau04b] F. Dau. “Types and Tokens for Logic with Diagrams”. In: *International Conference on Computational Science – ICCS 2004*. Ed. by M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra. Vol. 3038. LNCS. Springer, 2004, pp. 62–93.
- [Dau09] F. Dau. “The Advent of Formal Diagrammatic Reasoning Systems”. In: *International Conference on Formal Concept Analysis – ICFCA 2009*. Ed. by S. Ferré and S. Rudolph. Vol. 5548. LNCS. Springer, 2009, pp. 38–56.
- [DH01] W. Damm and D. Harel. “LSCs: Breathing Life into Message Sequence Charts”. In: *Formal Methods in System Design* 19 (Jan. 2001), pp. 45–80.
- [DHO06] W. Damm, H. Hungar, and E.-R. Olderog. “Verification of Cooperating Traffic Agents”. In: *International Journal of Control* 79.5 (2006), pp. 395–421.
- [DMR14] W. Damm, E. Möhlmann, and A. Rakow. “Component Based Design of Hybrid Systems: A Case Study on Concurrency and Coupling”. In: *International Conference on Hybrid Systems: Computation and Control – HSCC 2014*. Ed. by M. Fränzle and J. Lygeros. ACM, 2014, pp. 145–150.
- [Dut95] B. Dutertre. “Complete Proof Systems for First Order Interval Temporal Logic”. In: *IEEE Symposium on Logic in Computer Science – LICS 1995*. Ed. by D. C. Kozen. IEEE Computer Society, 1995, pp. 36–43.
- [EC82] E. A. Emerson and E. M. Clarke. “Using branching time temporal logic to synthesize synchronization skeletons”. In: *Science of Computer Programming* 2.3 (1982), pp. 241–266.
- [ES99] M. Erwig and M. Schneider. “Visual Specification of Spatio-Temporal Developments”. In: *IEEE International Symposium on Visual Languages – VL 1999*. IEEE Computer Society, 1999, pp. 187–188.
- [Fab+11] J. Faber, S. Linker, E.-R. Olderog, and J.-D. Quesel. “Syspect - Modelling, Specifying, and Verifying Real-Time Systems with Rich Data”. In: *International Journal of Software and Informatics* 5.1-2 (2011), pp. 117–137.
- [FG92] M. Finger and D. M. Gabbay. “Adding a temporal dimension to a logic system”. In: *Journal of Logic, Language and Information* 1.3 (1992), pp. 203–233.
- [FM98] M. Fitting and R. L. Mendelsohn. *First-Order Modal Logic*. Springer, 1998.
- [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.
- [Gab+03] D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyashev. *Many-dimensional modal logics: Theory and applications*. Vol. 148. Studies in logic and the foundations of mathematics. Elsevier, 2003.
- [Gab96] D. M. Gabbay. *Labelled deductive systems*. Vol. 1. Oxford Logic Guides. Oxford University Press, 1996.

- [Gab98] D. M. Gabbay. *Fibring logics*. Oxford Logic Guides. Oxford University Press, 1998.
- [Gen35] G. Gentzen. “Untersuchungen über das logische Schließen. I”. In: *Mathematische Zeitschrift* 39 (1 1935), pp. 176–210.
- [Haa98] V. Haarslev. “A Fully Formalized Theory for Describing Visual Notations”. In: *Visual Language Theory*. Ed. by K. Marriott and B. Meyer. Springer, 1998.
- [Ham96] E. Hammer. “Peircean Graphs for Propositional Logic”. In: *Logical Reasoning with Diagrams*. Ed. by G. Allwein and J. Barwise. Oxford University Press, 1996, pp. 129–147.
- [Har88] D. Harel. “On Visual Formalisms”. In: *Communications of the ACM* 31.5 (1988).
- [Hil+11] M. Hilscher, S. Linker, E.-R. Olderog, and A. P. Ravn. “An Abstract Model for Proving Safety of Multi-lane Traffic Manoeuvres”. In: *Formal Methods and Software Engineering – International Conference on Formal Engineering Methods – ICFEM 2011*. Ed. by Z. Qiu S. Qin. LNCS. Springer, 2011, pp. 404–419.
- [Hil14] M. Hilscher. Private communications. 2014.
- [HLO13] M. Hilscher, S. Linker, and E.-R. Olderog. “Proving Safety of Traffic Manoeuvres on Country Roads”. In: *Theories of Programming and Formal Methods – Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*. Ed. by Z. Liu, J. Woodcock, and Huibiao Z. Vol. 8051. LNCS. Springer, 2013, pp. 196–212.
- [Hoa69] C. A. R. Hoare. “An Axiomatic Basis for Computer Programming”. In: *Communications of the ACM* 12.10 (1969), pp. 576–580.
- [Hoe06] J. Hoenicke. “Combination of Processes, Data, and Time”. PhD thesis. Carl von Ossietzky Universität Oldenburg, 2006.
- [How+02] J. Howse, F. Molina, S.-J. Shin, and J. Taylor. “On Diagram Tokens and Types”. In: *Diagrammatic Representation and Inference – International Conference on the Theory and Applications of Diagrams – DIAGRAMS 2002*. Ed. by M. Hegarty, B. Meyer, and N. H. Narayanan. Vol. 2317. LNAI. Springer, 2002, pp. 146–160.
- [HP08] I. A. Hansen and J. Pachl, eds. *Railway Timetable & Traffic – Analysis, Modelling, Simulation*. Eurailpress, 2008.
- [HP09] A. Habel and K.-H. Pennemann. “Correctness of high-level transformation systems relative to nested conditions”. In: *Mathematical Structures in Computer Science* 19 (02 Apr. 2009), pp. 245–296.
- [HR10] A. Habel and H. Radke. “Expressiveness of graph conditions with variables”. In: *Electronic Communications of the EASST* 30 (2010).

-
- [HS91] J. Y. Halpern and Y. Shoham. “A Propositional Modal Logic of Time Intervals”. In: *Journal of the ACM* 38.4 (Oct. 1991), pp. 935–962.
- [Hsu+94] A. Hsu, F. Eskafi, S. Sachs, and P. Varaija. “Protocol design for an automated highway system”. In: *Discrete Event Dynamic Systems* 2.1 (1994), pp. 183–206.
- [ITU96] International Telecommunication Union (ITU). *Z.120. ITU-TS recommendation Z.120: Message Sequence Chart (MSC)*. 1996.
- [JKI99] H. Jula, E. B. Kosmatopoulos, and P. A. Ioannou. *Collision Avoidance Analysis for Lane Changing and Merging*. Tech. rep. UCB-ITS-PRR-99-13. California Partners for Advanced Transit and Highways (PATH), University of California at Berkeley, 1999.
- [KE14] S. Kemper and C. Etzien. “A Visual Logic for the Description of Highway Traffic Scenarios”. In: *International Conference on Complex Systems Design & Management – CSD&M 2013*. Ed. by M. Aiguier, F. Boulanger, D. Krob, and C. Marchal. Springer, 2014, pp. 233–245.
- [Ken97] S. Kent. “Constraint diagrams: visualizing invariants in object-oriented models”. In: *ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages & Applications – OOPSLA 1997*. Ed. by M. E. S. Loomis, T. Bloom, and A. M. Berman. ACM, 1997, pp. 327–341.
- [Kle00] C. Kleuker. “Constraint Diagrams”. PhD thesis. Carl von Ossietzky Universität Oldenburg, 2000.
- [Koo+08] P. Koonce, L. Rodegerdts, K. Lee, S. Quayle, S. Beaird, C. Braud, J. Bonneson, P. Tarnoff, and T. Urbanik. *Traffic Signal Timing Manual*. Tech. rep. U.S. Department of Transportation, Federal Highway Administration, 2008.
- [LGS98] J. Lygeros, D. N. Godbole, and S. Sastry. “Verified hybrid controllers for automated vehicles”. In: *IEEE Transactions on Automatic Control* 43.4 (1998), pp. 522–539.
- [LH13] S. Linker and M. Hilscher. “Proof theory of a multi-lane spatial logic”. In: *International Colloquium on Theoretical Aspects of Computing – ICTAC 2013*. Ed. by Z. Liu, J. Woodcock, and Huibiao Z. Vol. 8049. LNCS. Springer, 2013, pp. 231–248.
- [Lin07] S. Linker. “Natürliches Schließen für den Shape Calculus”. http://www.uni-oldenburg.de/fileadmin/user_upload/f2inform-csd/linker07.pdf. Minor thesis. Carl von Ossietzky Universität Oldenburg, 2007.
- [Lin10] S. Linker. “Diagrammatic Specification of Mobile Real-Time Systems”. In: *Diagrammatic Representation and Inference – International Conference on the Theory and Applications of Diagrams – DIAGRAMS 2010*. Ed. by A. K. Goel, M. Jamnik, and N. H. Narayanan. Vol. 6170. LNAI. Springer, 2010, pp. 316–318.

- [Lod+92] K. Lodaya, M. Mukund, R. Ramanujam, and P.S. Thiagarajan. “Models and logics for true concurrency”. In: *Sadhana (Academy Proceedings in Engineering Sciences)*. Vol. 17. 1. Indian Academy of Sciences. 1992, pp. 131–165.
- [LPN11] S. M. Loos, A. Platzer, and L. Nistor. “Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified”. In: *International Symposium on Formal Methods – FM 2011*. Ed. by M. Butler and W. Schulte. Vol. 6664. LNCS. Springer, 2011, pp. 42–56.
- [MA08] J. McDermott and G. Allwein. “A Formalism for Visual Security Protocol Modeling”. In: *Journal of Visual Languages & Computing* 19.2 (2008), pp. 153–181.
- [McD05] J. P. McDermott. *A Formal Syntax and Semantics for the GSPML Language*. Tech. rep. Naval Research Laboratory, Information Technology Division, 2005.
- [Min00] M. Minas. “Hypergraphs as a Uniform Diagram Representation Model”. In: *Selected Papers of the International Workshop on Theory and Application of Graph Transformations – TAGT 1998*. Ed. by H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg. Vol. 1764. LNCS. Springer, 2000, pp. 281–295.
- [Min67] M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, 1967.
- [Mos85] B. C. Moszkowski. “A Temporal Logic for Multilevel Reasoning about Hardware”. In: *Computer* 18.2 (1985), pp. 10–19.
- [MOT10] K. Mineshima, M. Okada, and R. Takemura. “Two Types of Diagrammatic Inference Systems: Natural Deduction Style and Resolution Style”. In: *Diagrammatic Representation and Inference – International Conference on the Theory and Applications of Diagrams – DIAGRAMS 2010*. Ed. by A. Goel, M. Jamnik, and N. H. Narayanan. Vol. 6170. LNCS. Springer, 2010, pp. 99–114.
- [MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer, 1995.
- [MPW92] R. Milner, J. Parrow, and D. Walker. “A Calculus of Mobile Processes, I”. In: *Information and Computation* 100.1 (Sept. 1992), pp. 1–40.
- [MV95] M. Minas and G. Viehstaedt. “DiaGen: a generator for diagram editors providing direct manipulation and execution of diagrams”. In: *IEEE International Symposium on Visual Languages – VL 1995*. Ed. by V. Haarslev. IEEE Computer Society, 1995, pp. 203–210.
- [MV97] M. Marx and Y. Venema. *Multi-dimensional modal logic*. Springer, 1997.
- [Pau94] L. Paulson. *Isabelle: A Generic Theorem Prover*. Springer, 1994.
- [Pla10a] A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010.

- [Pla10b] A. Platzer. “Quantified Differential Dynamic Logic for Distributed Hybrid Systems”. In: *EACSL Conference on Computer Science Logic – CSL 2010*. Ed. by A. Dawar and H. Veith. Vol. 6247. LNCS. Springer, 2010, pp. 469–483.
- [Pnu77] A. Pnueli. “The Temporal Logic of Programs”. In: *IEEE Symposium on Foundations of Computer Science – SFCS 1977*. IEEE Computer Society, 1977, pp. 46–57.
- [PQ09] A. Platzer and J.-D. Quesel. “European Train Control System: A Case Study in Formal Verification”. In: *Formal Methods and Software Engineering – International Conference on Formal Engineering Methods – ICFEM 2009*. Ed. by A. Cavalcanti and K. Breitman. Vol. 5885. LNCS. Springer, 2009, pp. 246–265.
- [Pra06] D. Prawitz. *Natural Deduction: A Proof-theoretical Study*. Dover, 2006.
- [Pri57] A. N. Prior. *Time and Modality*. Oxford University Press, 1957.
- [Rad13] H. Radke. “HR* Graph Conditions Between Counting Monadic Second-Order and Second-Order Graph Formulas”. In: *Electronic Communications of the EASST 61* (2013).
- [Ras+02] J. Rasga, A. Sernadas, C. Sernadas, and L. Viganò. “Fibring Labelled Deduction Systems”. In: *Journal of Logic and Computation* 12.3 (2002), pp. 443–473.
- [Ras01] T. M. Rasmussen. “Labelled Natural Deduction for Interval Logics”. In: *EACSL Conference on Computer Science Logic – CSL 2001*. Ed. by L. Fribourg. Vol. 2142. LNCS. Springer, 2001, pp. 308–323.
- [Ras02] T. M. Rasmussen. “Interval logic. Proof theory and theorem proving”. PhD thesis. Technical University of Denmark, 2002.
- [RCC92] D. A. Randell, Z. Cui, and A. G. Cohn. “A Spatial Logic based on Regions and Connection”. In: *International Conference on Principles of Knowledge Representation and Reasoning – KR 1992*. Ed. by B. Nebel, C. Rich, and W. Swartout. Morgan Kaufmann, 1992, pp. 165–176.
- [Rey02] J. C. Reynolds. “Separation Logic: A Logic for Shared Mutable Data Structures”. In: *IEEE Symposium on Logic in Computer Science – LICS 2002*. Ed. by G. D. Plotkin. IEEE Computer Society, 2002, pp. 55–74.
- [RJB04] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual (2nd Edition)*. Pearson Higher Education, 2004.
- [Roz97] G. Rozenberg, ed. *Handbook of graph grammars and computing by graph transformation: volume I. foundations*. World Scientific Publishing, 1997.
- [RS95] J. Rekers and A. Schürr. “A graph grammar approach to graphical parsing”. In: *IEEE International Symposium on Visual Languages – VL 1995*. Ed. by V. Haarslev. IEEE Computer Society, 1995, pp. 195–202.
- [Sch01] R. C. Schlör. “Symbolic Timing Diagrams: A Visual Formalism for Model Verification”. PhD thesis. Carl von Ossietzky Universität Oldenburg, 2001.

- [Sch05] A. Schäfer. “A calculus for shapes in time and space”. In: *International Colloquium on Theoretical Aspects of Computing – ICTAC 2004*. Ed. by Z. Liu and K. Araki. Vol. 3407. LNCS. Springer, 2005, pp. 463–478.
- [Sch07] A. Schäfer. “Axiomatisation and decidability of multi-dimensional Duration Calculus”. In: *Information and Computation* 205.1 (2007). Special Issue: TIME 2005, pp. 25–64.
- [Sch14] M. Schwammberger. “Semantik von Controllern für sicheren Fahrspurwechsel”. Master’s Thesis. Carl von Ossietzky Universität Oldenburg, 2014.
- [Sha+13] Z. Shao, J. Liu, Z. Ding, M. Chen, and N. Jiang. “Spatio-Temporal Properties Analysis for Cyber-Physical Systems”. In: *Engineering of Complex Computer Systems – ICECCS 2013*. IEEE. 2013, pp. 101–110.
- [Shi04] S.-J. Shin. “Heterogeneous Reasoning and its Logic”. In: *Bulletin of Symbolic Logic* 10.1 (2004), pp. 86–106.
- [Shi95] S.-J. Shin. *The logical status of diagrams*. Cambridge: Cambridge University Press, 1995.
- [Sim94] A. K. Simpson. “The Proof Theory and Semantics of Intuitionistic Modal Logic”. PhD thesis. University of Edinburgh, 1994.
- [SM07] G. Stapleton and J. Masthoff. “Incorporating Negation into Visual Logics: A Case Study Using Euler Diagrams”. In: *Visual Languages and Computing* (2007), pp. 187–194.
- [Smi00] G. Smith. *The Object-Z Specification Language*. Springer, 2000.
- [Sta+05] G. Stapleton, S. Thompson, A. Fish, J. Howse, and J. Taylor. “A New Language for the Visualization of Logic and reasoning”. In: *International Conference on Distributed Multimedia Systems – DMS 2005*. Ed. by P. Cox and T. Smedley. Knowledge Systems Institute, 2005, pp. 287–292.
- [Tob08] T. Toben. “Counterexample Guided Spotlight Abstraction Refinement”. In: *International Conference on Formal Techniques for Networked and Distributed Systems – FORTE 2008*. Ed. by K. Suzuki, T. Higashino, K. Yasumoto, and K. El-Fakih. Vol. 5048. LNCS. Springer, 2008, pp. 21–36.
- [UJ12] M. Urbas and M. Jamnik. “Diabelli: A heterogeneous proof system”. In: *International Joint Conference on Automated Reasoning – IJCAR 2012*. Ed. by B. Gramlich, D. Miller, and U. Sattler. Vol. 7364. LNAI. Springer, 2012, pp. 559–566.
- [UJ14] M. Urbas and M. Jamnik. “A Framework for Heterogeneous Reasoning in Formal and Informal Domains”. In: *Diagrammatic Representation and Inference – International Conference on the Theory and Applications of Diagrams – DIAGRAMS 2014*. Ed. by T. Dwyer, H. Purchase, and A. Delaney. Vol. 8578. LNAI. Springer, 2014, pp. 277–292.
- [UML12] *Unified Modeling Language (UML): Superstructure version 2.4.1*. formal/2012-05-07. 2012. URL: <http://www.omg.org>.

- [Urb+12] M. Urbas, M. Jamnik, G. Stapleton, and J. Flower. “Speedith: a diagrammatic reasoner for spider diagrams”. In: *Diagrammatic Representation and Inference – International Conference on the Theory and Applications of Diagrams – DIAGRAMS 2012*. Ed. by P. T. Cox, B. Plimmer, and P. Rodgers. Vol. 7352. LNAI. Springer, 2012, pp. 163–177.
- [Vig00] L. Viganò. *Labelled Non-Classical Logics*. Kluwer Academic Publishers, 2000.
- [WDP13] J. Wickerson, M. Dodds, and M. Parkinson. “Ribbon proofs for separation logic”. In: *Programming Languages and Systems – European Symposium on Programming – ESOP 2013*. Ed. by M. Felleisen and P. Gardner. Vol. 7792. LNCS. Springer, 2013, pp. 189–208.
- [WW07] B. Wachter and B. Westphal. “The spotlight principle”. In: *International Conference on Verification, Model Checking, and Abstract Interpretation – VMCAI 2007*. Ed. by B. Cook and A. Podelski. Vol. 4349. LNCS. Springer, 2007, pp. 182–198.
- [ZHR91] Zhou C., C. A. R. Hoare, and A. P. Ravn. “A Calculus of Durations”. In: *Information Processing Letters* 40.5 (1991), pp. 269–276.
- [ZHS93] Zhou C., M. R. Hansen, and P. Sestoft. “Decidability and undecidability results for duration calculus”. In: *Symposium on Theoretical Aspects of Computer Science – STACS 1993*. Ed. by P. Enjalbert, A. Finkel, and K.W. Wagner. Vol. 665. LNCS. Springer, 1993, pp. 58–68.

Symbol Index

- \oplus , 27, 37
 \ominus , 27, 37
acc, 21
 \square_* , 32
 \square_ℓ , 34
 \square_τ , 34
 clm_V , 28
CVar, 31
 \frown , 7, 32
clm, 21
cl, 32
 \models_M , 99
 \models (HR* condition), 15
 \models (diagrams), 98
 \models (formulas), 33
 \models (labelled formulas), 38
 \models (relational formulas), 38
 \models_S , 99
 c_C , 105
 c_R^i , 105
 c_l , 105
 c_r , 105
 Δ , 38
 $\Delta_{v_i'}$, 128
 \Rightarrow_p , 12
 \Rightarrow^* , 13
 \mathfrak{D} , 75
 \diamond_ℓ , 34
DC, 126
dri, 39
 \mathcal{E} , 11
ego, 32
 $[\cdot]$, 33
 $\exists(P \sqsupseteq C, c)$, 14
 $\exists(h, c)$, 14
 \mathbb{E} , 37
 F_X , 104
 F_L , 104
 \mathbf{F} , 34
 Φ , 32
ts, v: ϕ , 37
w: φ , 9
 ϕ^r , 35
 Γ , 38
 \mathbf{G} , 32
 \mathbb{G} , 11
halt, 62
hcf, 39
hri, 39
 \mathbb{I} , 20
init, 60
 \mathfrak{J} , 6
 \mathcal{I} , 6
 \mathcal{I}_{Var} , 6
 len_V , 28
 \mathbb{L} , 20
 L , 25
 \mathfrak{l} , 11
LC, 127
LVar, 32
 ℓ , 32
 λ , 38
marker, 60
 \mathcal{M} , 113
mv, 26
mutex, 60
 \mathbb{N} , 5
 \circ , 6
 \mathbb{O} , 11
 \preceq , 101
Per, 60
 Π_i^{*j} , 128

- \mathcal{P} , 5
 pc , 126
 periodic, 61
 pos , 21
 \vdash , 39

 $\cdot \mathcal{Q}$, 114

 \mathbb{R}_+ , 6
 res_V , 28
 R^* , 6
 \mathbb{R} , 6
 res , 21
 ρ , 37
 re , 32
 $RVar$, 32, 71
 ri , 39

 $[\cdot \mapsto \cdot]$, 9
 $safe$, 126
 $\langle \cdot \rangle$ (sequence), 6
 σ , 38
 $\langle \cdot \rangle$ (modality), 33
 spd , 21

 $\overset{t}{\rightsquigarrow}$, 23
 $\frac{acc(C,a)}{\rightarrow}$, 23
 $\overset{t}{\rightarrow}$, 23
 $\frac{c(C,n)}{\rightarrow}$, 22
 $\frac{r(C)}{\rightarrow}$, 22
 $\frac{wd\ c(C)}{\rightarrow}$, 22
 $\frac{wd\ r(C,n)}{\rightarrow}$, 23
 \mathcal{TS} , 21
 \mathbb{TS} , 21
 \mathbb{T} , 20
 \Rightarrow , 24, 37
 Θ , 32
 ts , 37
 \mathfrak{TG} , 37
 τ , 11
 θ , 32
 \leq , 37
 $\cdot \mathcal{T}$, 115

 $\frac{c(c)}{\rightarrow}$, 37
 $\frac{r(c)}{\rightarrow}$, 37
 $\frac{wd\ c(c)}{\rightarrow}$, 37
 $\frac{wd\ r(c)}{\rightarrow}$, 37
 $\overset{t}{\rightarrow}$, 37
 θ , 11
 T , 11
 T_T , 81
 T_{NT} , 81

 V , 25
 V^L , 25
 V_X , 25
 v , 37
 \bar{v} , 105
 \underline{v} , 105
 ν , 32
 ν_I , 97
 ν_V , 32
 Var , 32, 70
 vcf , 39
 \mathcal{V} , 11
 \mathfrak{V} , 37
 vri , 39
 v_l , 105
 v_r , 105

 ω , 32

 X , 25
 χ , 98
 $\oplus\{\cdot \mapsto \cdot\}$, 6

 Ω_E , 27

Subject Index

- advanced driver assistance system, 108
- application (graph rewriting), 12
- arithmetic
 - Presburger, 106
 - real, 106
- assumption elimination, 8
- atom
 - equality, 32
 - free, 35
 - spatial, 32
- attachment function, 11
- axiom (graph rewriting), 13

- car
 - action, 71
 - braking distance, 19
 - physical size, 19
 - safety envelope, 19, 27
- chop-free, 11
- claim, 21, 72
 - creation, 23
 - withdrawal, 23
- composition, 6

- derivation, 8, 39
- derivation (graph rewriting), 13
- diagram
 - spatial, 74
 - Traffic Diagram, 75
- distance arrow, 73
- distance controller, 126
- domain, 6

- edge, 11
- EMLSL, 31
- expressivity, 111

- flexible, 8, 35
- formula, 32
 - dynamic, 37
 - labelled, 9, 37
 - locality, 37
 - relational, 9, 37
 - timing, 37
- function, 6
 - modification, 6

- global contradiction, 9
- graph homomorphism, 11
- graph rewriting, 13
 - Traffic Diagrams, 83
- graph rewriting rule, 12
- graph substitution, 14

- HR* condition, 13
- heterogeneous reasoning, 121
- horizon, 25
- hyperedge replacement, 13
- hypergraph, 11

- image, 6
- implicit lengths, 98
- interlingua, 123
- interval, 6, 71
 - chop, 27
 - variable, 6, 71

- labelled natural deduction, 9
 - EMLSL, 36
- labelling algebra, 36
- lane change controller, 127
- lane separation, 73
- lane sequence, 73
- layer, 74
 - full, 74
 - partial, 74
- left-hand side, 12
- logic
 - computational tree logic (CTL), 7
 - interval temporal logic (ITL), 7
 - linear temporal logic (LTL), 7
 - signed interval logic (SIL), 10
 - temporal, 7

- match, 12

- measure, 26
- modality
 - chop, 7, 32
 - everywhere, 33
 - invariance, 32
 - somewhere, 33
- modification, 32
- natural deduction, 8
- nested condition, 13
- potential collision check, 126
- precedence, 101
- preimage, 6
- production, *see* graph rewriting rule
- proof, 8
- proof rule, *see* rule
- range, 6
- relation, 6
- representation, 84
- reservation, 20, 72
 - creation, 23
 - lemma, 55
 - withdrawal, 23
- right-hand side, 12
- rigid, 8, 36
 - dynamically, 36
 - horizontally, 36
 - vertically, 36
- rule, 8
 - activity, 46
 - backwards activity, 47
 - backwards stability, 47
 - elimination, 9
 - introduction, 9
 - invariance/induction, 52
 - stability, 46
- run, 59
- safety predicate, 126
- sane, *see* sanity conditions
- sanity conditions, 21
 - formalisation, 77
- satisfaction
 - HR* condition, 15
 - EMLSL, 33
 - labelled formula, 38
 - relational formula, 38
 - Traffic Diagram, 98
- semantics
 - HR* condition, 15
 - EMLSL, 33
 - metric, 99
 - spatial, 99
 - Traffic Diagrams, 98
- sensor function, 27
- sequence, 6
- single decomposition, 11
- soundness, 53
- space
 - free, 72
 - unspecified, 72
- subview, *see* view
- syntax
 - abstract (diagrams), 70
 - concrete (diagrams), 70
 - EMLSL, 32
 - substitution, 9
- temporal arrow
 - discrete, 75
 - duration, 75
 - faint, 75
 - precise, 74
- temporal sequence, 75
- tentacle, 11
- term, 32
 - timing, 37
- topological sequence, 72
- topological situation, 72
- Traffic Diagrams, 69
- traffic snapshot, 21
- transformation, 115
 - metric, 113
 - qualitative, 114
- transition, 22
 - abstract, 24
 - dynamic, 22

- evolution, 23
 - global, 22
 - local, 22
 - modality, 32
 - spatial, 22
- two-counter machine, 59
- type
 - non-terminal, 81
 - terminal, 81
- type function, 11
- typed hypergraph, *see* hypergraph
- valuation, 32
 - interval, 97
 - locality, 38
 - snapshot, 38
- variable, 32, 70
 - car, 31, 70
 - implicit, 98
 - lane, 32
 - length, 32, 71
 - timing, 37
 - traffic snapshot, 37
 - view, 37
- vertex, 11
- view, 25
 - extension, 25
 - lanes, 25
 - length, 32
 - moving, 26
 - owner, 25, 32
 - standard, 25
 - subview, 25
 - width, 32
- visiting a vertex, 11

Curriculum Vitae

20/02/2015	Defense of the dissertation
2011–2015	Research assistant in the group “Correct System Design” lead by Ernst-Rüdiger Olderog at the Carl von Ossietzky University
2009–2011	Member of the Research Training Group “TrustSoft” at the Carl von Ossietzky University of Oldenburg
2003–2008	Studies “Diplom Informatik” at the Carl von Ossietzky University of Oldenburg
2002–2003	Alternative civilian service
2002	Abitur
03/03/1983	Born in Uelzen, Germany