Carl von Ossietzky Universität Oldenburg

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

A Consistency Analysis Method for Traffic Sequence Charts

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften der Carl von Ossietzky Universität Oldenburg zur Erlangung des Grades und Titels

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

angenommene Dissertation

von Herrn Jan Steffen Becker
geboren am 29. Juli 1990 in Wilhelmshaven

Gutachter:	Herr Prof. Dr. Werner Damm
weitere Gutachtende:	Herr Prof. Dr. Martin Fränzle Frau Jun.-Prof. Dr. Maike Schwammberger
Tag der Disputation:	29. Oktober 2025

Abstract

With an increasing level of driving automation, new challenges for the development of highly-automated vehicles and driver assistance systems arise. Highly automated vehicles and advanced driver assistance systems are designed to take over some safety-critical driving tasks. They must function in a wide range of traffic situations and environmental conditions, which renders traditional development and verification methods unfeasible. Therefore, the trend in automotive industry goes towards a scenario-based process.

Typically, scenarios are used at different abstraction levels. So-called *concrete scenarios* describe a single evolution of a traffic simulation and are usually described by a sequence of actions and events. Because they allow for the reproducible replay of scenarios, concrete scenarios are commonly used for testing. *Logical scenarios* equip concrete scenarios with parameters, such as the distance kept during an overtaking maneuver, allowing a certain degree of freedom. *Abstract* and *functional scenarios* on the other hand focus on the relations between traffic participants, and thereby allow an even higher abstraction level. Therefore, they are well suited for clustering the scenario space. While functional scenarios are described by natural language, abstract scenarios use a formal syntax and semantics.

In the field of automated driving, Traffic Sequence Charts (TSCs) are a specification language for abstract traffic scenarios, which focuses on scenario-based requirements specification. The core idea behind TSCs is to provide a graphical representation of traffic situations, so-called spatial views. In a spatial view, participants and scenery (e.g., lane markings, traffic signs, and obstacles) are described by placing symbols for every relevant object. The relative placement of the symbols expresses the spatial relations between the objects they represent. Each spatial view encodes an invariant over time. Spatial views are assembled to TSCs using boolean operators (conjunction, disjunction, negation) and sequences. Additionally, charts can be annotated with timing constraints.

This thesis addresses the consistency problem for TSC-based requirements. Consistency means that requirements (or scenarios, in the case of TSCs) are free of contradictions. This includes contradictions within a requirement (or scenario) itself, as well as contradictions with other requirements in the same or related documents (e.g., associated standards).

Consistency of formal requirements has been addressed in the literature already. However, most of the existing work focuses on finite state systems and is not directly applicable to TSCs. Therefore, this thesis addresses the following two research questions:

1. How can consistency for traffic sequence charts be formally defined?

2. How can a consistency check, that uses these definitions of consistency, be automated?

First, the role of a consistency analysis in a V-model development process is investigated. This leads to some basic properties that a consistency analysis should have in relation to an evolving requirement specification.

Second, consistency of TSCs is formally addressed. First, two "ideal" consistency notions are defined that reflect the intuitive meaning of consistency: *Existential consistency*, which formalizes consistency of single TSCs based on existing work, and *strong existential consistency*, which generalizes existential consistency for sets of TSCs. Unfortunately,

transferring existing results about hybrid systems and interval logics to TSCs indicates that these consistency notions are undecidable. As a consequence, two decidable variants – *weak existential consistency* and *partial consistency* – are developed. In a development process, it is more important to avoid false warnings than to prove the consistency of a potentially incomplete specification. Therefore, the decidable variants are designed as weak approximations of the idealized ones. For instance, if the consistency analysis reveals a violation of partial consistency, we know that strong existential consistency is also violated. This thesis examines and formally proves the properties and relations between the different consistency notions.

The third contribution is the most technical one: a solution for checking decidable consistency notions on a set of TSCs. Current work has shown that SMT (satisfiability modulo theories) solvers can be efficiently used to check some form of consistency (sometimes called existential consistency) and refinement for pattern-based requirements. The consistency analysis for TSCs presented in this thesis also uses SMT solving. The idea is to reduce consistency to a finite set of satisfiability problems for so-called positive existential TSCs. A positive existential TSC describes an abstract scenario and is called satisfiable if there exists at least one corresponding concrete scenario. This thesis describes a semi-decision procedure that translates positive existential TSCs to decidable SMT problems. The translation process consists of two parts: one that handles the temporal aspects (i.e. the chart structure) of TSCs and one that handles spatial properties. The latter is developed in two variants, yielding necessary and sufficient conditions for satisfiability of positive existential TSCs. The former uses only information from the TSCs and the underlying domain ontology, and is used to formally prove unsatisfiability of existential TSCs. The latter also considers a simple but correct vehicle dynamics model and thereby can create concrete scenarios as witnesses for satisfiable TSCs. The correctness of these constructions is proven.

The developed methods are prototypically implemented on top of the TSC editor, a formal modeling tool for TSCs. Applicability, performance, and scalability are evaluated on a series of experiments. This includes three case studies in which regulatory requirements (traffic rules and a subset of UN regulation no. 157 on automated lane keeping systems) are formalized as TSCs and analyzed for consistency using the developed methods.

Zusammenfassung

Mit erhöhtem Abstraktionsgrad entstehen neue Herausforderungen für die Entwicklung hochautomatisierter Fahrzeuge und Fahrerassistenzsysteme. Hochautomatisierte Fahrzeuge und Fahrerassistenzsysteme werden entwickelt, um sicherheitskritische Fahraufgaben zu übernehmen. Sie müssen in vielen Verkehrssituationen und unter verschiedenen Umweltbedingungen funktionieren. Traditionelle Entwicklungsmethoden reichen hier nicht mehr aus. In der Automobilindustrie geht der Trend daher hin zu szenarienbasierten Prozessen.

Standardmäßig werden Szenarien auf unterschiedlichen Abstraktionsstufen benutzt. Sogenannte *konkrete Szenarien* beschreiben eine einzige Entwicklung einer Verkehrssimulation und werden für gewöhnlich durch eine Sequenz aus Aktionen und Ereignissen beschrieben. Da sie eine reproduzierbare Wiedergabe von Szenarien ermöglichen, werden konkrete Szenarien hauptsächlich zum Testen eingesetzt. *Logische Szenarien* fügen konkreten Szenarien Parameter hinzu, wie z.B. den Abstand während eines Überholmanövers, und ermöglichen so eine gewisse Flexibilität. *Abstrakte* und *funktionale Szenarien* hingegen konzentrieren sich auf die Relationen zwischen Verkehrsteilnehmern und erreichen dadurch einen noch höheren Abstraktionsgrad. Dadurch eignen diese sich besonders, um Szenarienräume zu beschreiben. Während funktionale Szenarien mit natürlicher Sprache ausgedrückt werden, verwenden abstrakte Szenarien eine formale Syntax und Semantik.

Im Bereich des automatisierten Fahrens stellen *Traffic Sequence Charts* (TSCs) eine Beschreibungssprache für abstrakte Szenarien dar, die sich auf szenarienbasierte Anforderungen konzentriert. Die Kernidee hinter TSCs ist eine graphische Darstellung von Verkehrssituationen bereitzustellen, sogenannte *Spatial Views*. In einer Spatial View werden Verkehrsteilnehmer und Szenerie (z.B. Fahrbahnmarkierungen, Verkehrsschilder und Hindernisse) dargestellt, indem für alle relevanten Objekte Symbole platziert werden. Die relative Anordnung der Symbole repräsentiert die räumlichen Beziehungen der dargestellten Objekte. Jede Spatial View stellt eine Invariante mit zeitlicher Ausdehnung dar. Spatial Views werden durch Bool'sche Operatoren (Konjunktion, Disjunktion und Negation) sowie einem Sequenzoperator zu TSCs zusammengesetzt. Zusätzlich können TSCs mit Zeitannotationen versehen werden.

Die vorliegende Arbeit adressiert das Konsistenzproblem für TSC-basierte Anforderungen. Konsistenz bedeutet, dass Anforderungen (oder Szenarien, im Fall von TSCs) frei von Widersprüchen sind. Dies schließt sowohl Widersprüche innerhalb einer Anforderung sowie Widersprüche zu anderen Anforderungen im selben oder verknüpften Dokumenten (z.B. Standards) ein.

Konsistenz formaler Anforderungen wird in der Literatur bereits behandelt. Allerdings betrachten die meisten existierenden Arbeiten endliche Zustandssysteme und sind nicht direkt auf TSCs übertragbar. Daher betrachtet die vorliegende Arbeit die folgenden Forschungsfragen:

- Wie kann Konsistenz für Traffic Sequence Charts formal definiert werden?
- Wie kann eine Konsistenzprüfung, die diese Definitionen von Konsistenz nutzt, automatisiert werden?

Als erstes wird die Rolle einer Konsistenzanalyse in einem V-Modell-Entwicklungsprozess untersucht. Dies führt zu einer Reihe von Eigenschaften, die eine Konsistenzanalyse im Bezug auf eine sich entwickelnde Anforderungsspezifikation haben sollte.

Als zweites wird Konsistenz formal adressiert. Zunächst werden zwei „ideale“ Konsistenzbegriffe definiert, die die intuitive Bedeutung von Konsistenz widerspiegeln: *Existentielle Konsistenz*, die, basierend auf existierenden Arbeiten, Konsistenz für einzelne TSCs beschreibt, und *starke existentielle Konsistenz*, die diese auf Mengen von TSCs verallgemeinert. Leider folgt aus existierenden Ergebnissen zu hybriden Systemen und Intervallogik, dass diese Konsistenzbegriffe unentscheidbar sind. Als Antwort darauf werden zwei entscheidbare Varianten – *schwache existentielle Konsistenz* und *partielle Konsistenz* – entwickelt. In einem Entwicklungsprozess ist es wichtiger falsche Warnungen zu vermeiden, als die Konsistenz einer potentiell unvollständigen Spezifikation zu beweisen. Daher werden die entscheidbaren Varianten als schwache Approximationen für die idealen Konsistenzbegriffe entwickelt. Wenn z.B. die Konsistenzanalyse eine Verletzung der partiellen Konsistenz aufdeckt, ist klar, dass starke Konsistenz ebenfalls verletzt ist. Die vorliegende Arbeit untersucht und beweist die Eigenschaften und Beziehungen der verschiedenen Konsistenzbegriffe formal.

Der dritte Beitrag ist der am meisten technische: eine Lösung, um entscheidbare Konsistenzbegriffe für TSCs zu überprüfen. Aktuelle Arbeiten haben gezeigt, dass SMT (Satisfiability modulo Theories)-Solver effektiv genutzt werden können, um Formen von Konsistenz und Refinement für patternbasierte Anforderungen zu überprüfen. Die Konsistenzanalyse für TSCs, die in der vorliegenden Arbeit vorgestellt wird, benutzt ebenfalls SMT-Solving. Die Idee ist Konsistenz auf eine endliche Menge an Erfüllbarkeitsproblemen für sogenannte positive existentielle TSCs zu reduzieren. Ein positives existentielles TSC beschreibt ein abstraktes Szenario und ist erfüllbar, wenn ein entsprechendes konkretes Szenario existiert. Die vorliegende Arbeit beschreibt ein Semi-Entscheidungsverfahren, bei dem positive existentielle TSCs in entscheidbare SMT-Probleme übersetzt werden. Der Übersetzungsprozess besteht aus zwei Teilen: einer beschreibt die temporalen Aspekte (d.h., die Chartstruktur) der TSCs und der andere die räumlichen Relationen. Der letztere Teil kommt in zwei Varianten, wodurch eine notwendige und eine hinreichende Bedingung für die Erfüllbarkeit positiver existentieller TSCs erhalten werden. Die notwendige Bedingung nutzt nur informationen aus dem TSC und der zugrundeliegenden Domänen-Ontologie und wird verwendet, um Unerfüllbarkeit existentieller TSCs zu zeigen. Die hinreichende Bedingung nutzt zusätzlich ein einfaches aber korrektes Fahrzeugdynamikmodell und kann dadurch konkrete Szenarien als Belege für erfüllbare TSCs erzeugen. Die Korrektheit dieser Konstruktionen wird gezeigt.

Die entwickelten Methoden werden prototypisch im TSC-Editor umgesetzt, einem formalen Modellierungswerkzeug für TSCs. Anwendbarkeit, Performanz und Skalierbarkeit werden durch eine Reihe an Experimenten evaluiert. Diese enthalten drei Fallstudien, in denen regulatorische Anforderungen (Verkehrsregeln und Teile von UN-Regulation 157 für automatisierte Spurhaltesysteme) als TSCs formalisiert und mittels der entwickelten Methoden auf Konsistenz überprüft werden.

Contents

List of Figures

List of Tables

List of Definitions

List of Assumptions and Theorems

List of Constructions

List of Algorithms

1 Introduction

Requirements-driven development is state of the art in embedded systems engineering. Normative standards define how safety shall be assessed during development of cyber-physical systems. For example, ISO 26262 [76] defines a requirements-driven safety process for the development of automotive systems [24, 89]. This process begins with a hazard and risk analysis (HARA), where possible malfunctions of the system are identified and analyzed. The results are used to derive safety requirements which are to be incorporated into the system design.

With an increasing level of driving automation, new challenges arise. Highly automated vehicles and advanced driver assistance systems are designed to take over some safety-critical driving tasks. They must function in a wide range of traffic situations and environmental conditions. Because they are expected to run unattended by a human driver for at least some limited time, it is not possible to rely on the driver to intervene timely in case of malfunctions. In addition to hardware failures in the scope of ISO 26262, functional deficiencies of the system must be considered. These include perception faults, misinterpretation of traffic situations, and unexpected behavior of the system. Due to the low statistical probability – yet high relevance – of some critical events, it is impossible to base a safety argument solely on field experience [117]. Together with the immense amount and complexity of the to-be-considered driving situations, this renders traditional development and verification methods infeasible [84]. Therefore, the trend in the automotive industry is towards a scenario-based process [106]. The key idea is to cluster the different traffic situations and scenarios into a manageable set of scenario classes [90]. Benefits of scenario-based methods have been identified in many phases of the safety process required by ISO 26262 [90].

In a scenario-driven development process, scenarios are used at different abstraction levels. So-called *concrete scenarios* describe a single evolution of a traffic simulation and are usually described by a sequence of actions and events. Because they allow for the reproducible replay of scenarios, concrete scenarios are commonly used for testing. *Logical scenarios* equip concrete scenarios with parameters, such as the distance kept during an overtaking maneuver, allowing a certain degree of freedom. *Abstract* and *functional scenarios* on the other hand focus on the relations between traffic participants, and thereby allow an even higher abstraction level. Therefore, they are well suited for clustering the scenario space. While functional scenarios are described by natural language, abstract scenarios use a formal syntax and semantics. In the field of automated driving, Traffic Sequence Charts (TSCs) [38] are a specification language for abstract traffic scenarios, that also focuses on scenario-based requirements specification. The core idea behind TSCs is to provide a graphical representation of traffic situations, so-called spatial views. In a spatial view, participants and scenery (e.g., lane markings, traffic signs, and obstacles) are described by placing symbols for every relevant object. The relative placement of the symbols expresses the spatial relations between the objects they represent. Each spatial view encodes an invariant over time. Spatial views are assembled to TSCs using boolean operators (conjunction, disjunction, negation) and sequences. Additionally, TSCs can be annotated with timing constraints.

TSCs are used for different tasks. Some of the most relevant use cases are:

- Specification of safety requirements for driving functions [33]
- Specification of functional requirements (behavior in certain traffic situations) for autonomous transportation systems [11, 33, 36]
- Specification of abstract test cases [40]
- Building databases of abstractions of real-world observations [39]

ISO 26262, and good engineering practices in general, mandate that requirements must comply with the three big Cs: Correctness, Completeness and Consistency [123]. Whether a specification is complete can usually only be defined relatively, for example in relation to the outcome of the HARA [87]. Completeness can be addressed by using systematic approaches to identify all relevant scenario classes [93, 101]. Consistency means that requirements (or scenarios, in the case of TSCs) are free of contradictions. This includes contradictions within a requirement (or scenario) itself, as well as contradictions with other requirements in the same or related documents (e.g., associated standards) [76]. In the above use cases, unsatisfiable test cases are useless and systems with contradictory requirements cannot be built. In most cases, inconsistencies are caused by faults in the specifications. These faults are likely to be found in later phases of the development process, but at the cost of increased time and expense [54]. A specification is considered correct if it is both complete and consistent [123].

This thesis tackles the consistency problem for TSC-based requirements. Consistency of formal requirements has been addressed in the literature already (e.g. [1, 10, 48, 56, 77, 98]). However, most of the existing work focuses on finite state systems and is not directly applicable to TSCs. Therefore, this thesis addresses the following two research questions:

1. How can consistency for traffic sequence charts be formally defined?
2. How can a consistency check that uses these definitions of consistency be automated?

The research contributions of this thesis are structured as shown in Figure 1.1.

First, the role of a consistency analysis in a V-model development process [89] is investigated. This leads to some basic properties that a consistency analysis should have in relation to an evolving requirement specification (i.e., adding new requirements). It is also expected that the granularity of the domain and environment model descriptions (i.e., their granularity) change during the development process. This needs to be considered when interpreting consistency results in the context of the overall process.

With the defined use cases in mind, consistency of TSCs is formally addressed. First, two "ideal" consistency notions are defined that reflect the intuitive meaning of consistency: *Existential consistency*, which formalizes consistency of single TSCs based on existing work, and *strong existential consistency*, which generalizes existential consistency for sets of TSCs. Unfortunately, transferring existing results [22, 75] about hybrid systems and interval logics to TSCs (cf. Section 3.3) indicates[1] that these consistency notions are undecidable. In order to deal with the undecidability (and the to be expected complexity in decidable cases), two decidable variants – *weak existential consistency* and *partial consistency* – are developed. In a development process, it is more important to avoid false warnings than to prove the consistency of a potentially incomplete specification. Therefore, the decidable variants are designed as weak approximations of the idealized ones. For instance, if the

[1] A formal undecidability proof for TSCs is not part of this thesis.

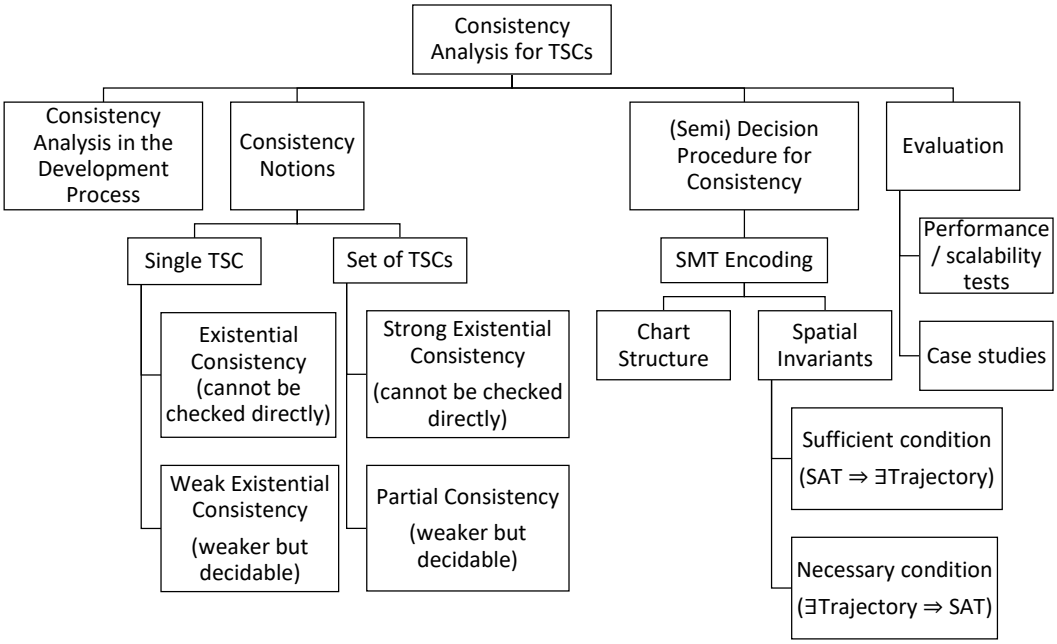


Figure 1.1: Structure of the thesis

consistency analysis revels a violation of partial consistency, we know that strong existential consistency is also violated. This thesis examines and formally proves the properties and relations between the different consistency notions.

The third contribution is the most technical one: a solution for checking decidable consistency notions on a set of TSCs. Current work [10, 13, 48, 56] has shown that SMT (satisfiability modulo theories) solvers [8] can be efficiently used to check consistency and refinement for pattern-based requirements. The consistency analysis for TSCs presented in this thesis also uses SMT solving. The consistency notions presented rely on checking satisfiability of TSCs, i.e., determining whether the behavior described by a TSC is physically possible. Since this is generally undecidable, this thesis describes a semi-decision procedure that translates a certain sub-class of TSCs (so-called positive existential TSCs) to decidable SMT problems. The translation process consists of two parts: one that handles the temporal aspects (i.e. the chart structure) of TSCs and one that handles spatial properties. The latter is developed in two variants, yielding necessary and sufficient conditions for the satisfiability of positive existential TSCs.

The thesis builds upon the author's following published results:

[10] In Becker (2018): "Analyzing Consistency of Formal Requirements", the idea of *partial consistency* is presented for the first time.

[11] In Becker (2020): "Partial Consistency for Requirement Engineering with Traffic Sequence Charts", first consistency notions for TSCs are presented, including an earlier version of *partial consistency* for pairs of TSCs. Applicability of the method is demonstrated on a toy example.

[15, 18] In Becker et al. (2022): "Simulation of Abstract Scenarios: Towards Automated Tooling in Criticality Analysis", and Borchers et al. (2025): "TSC2CARLA: An abstract scenario-based verification toolchain for automated driving systems", the

sufficient conditions for satisfiability of existential TSCs are used for generating concrete scenarios out of abstract ones.

[12] The encoding of the single track vehicle dynamics into linear satisfiability problems (see Chapter 7) is presented in detail in Becker (2024): "Safe Linear Encoding of Vehicle Dynamics for the Instantiation of Abstract Scenarios". Some of the experimental results are reproduced in Section 8.4.

[14] Becker and Neurohr (2025): "Correct-by-Construction Instantiation of Abstract Scenarios" is an extended version of the above work [12] which additionally presents reduced versions of the translation of the temporal structure of TSCs (without timing annotations) and spatial views into SMT. Furthermore, it gives more details on the performed experiments and further applications of the translation to SMT.

An overview on this thesis has been presented in the VEHITS'24 Doctoral Consortium[2] [9].

The structure of the thesis is as follows. First, the technical background (TSCs, SMT-solving, bounded model checking) is introduced. In Chapter 3 related work is discussed which also motivates the choice of SMT solving and chosen encoding techniques. Then, the main contributions of the thesis follow: Chapter 4 places consistency analysis in a V-development process context and takes a closer look at different world models. The different consistency notions considered in the thesis are then formally introduced in Chapter 5, followed by their implementation (inclusive correctness proof) in Chapter 6. Due to its complexity, encoding of the sufficient conditions has been moved to a separate Chapter 7. The evaluation follows in Chapter 8. Chapter 9 closes the thesis with a conclusion and outlook.

[2] `https://vehits.scitevents.org/DoctoralConsortium.aspx?y=2024`, accessed on 2026-02-28

2 TSC Syntax & Semantics, Notations

This chapter gives an overview on the theoretical foundations of this work. The first section introduces technological bricks for the consistency analysis method: multi-sorted signatures, satisfiability modulo theories (SMT) solving [8] and bounded model checking (BMC). Multi-sorted signatures are a building block for both TSC formal semantics and SMT/BMC; therefore they are described first. The second section then covers TSCs and their detailed semantics, including a conceptualization of world models and a formal definition of multi-sorted real-time logic (MSRTL). The latter forms the semantic basis for TSCs.

2.1 Satisfiability Modulo Theories

Satisfiability modulo theories [8] is an extension of classical satisfiability solving that considers, in addition to propositional logic, formulae over some multi-sorted signature.

Definition 1 (Multi-sorted Signature)**.** *A* multi-sorted signature $\Sigma = (\Gamma, \Pi, \Upsilon)$ *defines finite sets of sorts (Γ), n-ary predicate symbols (Π), and function symbols (Υ).*

Signatures can build on top of each other. We write $\Sigma \subseteq \Sigma'$ if $\Gamma \subseteq \Gamma'$, $\Pi \subseteq \Pi'$, and $\Upsilon \subseteq \Upsilon'$.

An SMT formula over some signature Σ is an expression build up from the predicate and function symbols in Σ, i.e., formulas (constraints ϕ and terms t) follow the grammar

$$\begin{aligned} \phi &::= p_n(t_1, \ldots, t_n) \mid \mathit{true} \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \exists X \in \mathcal{S} \bullet \phi \\ t &::= f_n(t_1, \ldots, t_n) \mid (\textbf{if } \phi \textbf{ then } t_1 \textbf{ else } t_2) \end{aligned}$$

where $p_n \in \Pi$ and $f_n \in \Upsilon$ are n-ary predicate rsp. function symbols (including constants), and $\mathcal{S} \in \Gamma$ a sort from Σ. For better readability, SMT formulae are written down in this thesis using standard mathematical conventions. For instance, binary predicate and function applications are written in infix notation (e.g., $a + b$ instead of $+(a, b)$) if appropriate. Furthermore, standard abbreviation symbols are used such as $\phi_1 \Rightarrow \phi_2$ for $\neg\phi_1 \vee \phi_2$, and $\phi_1 \Leftrightarrow \phi_2$ for $(\phi_1 \wedge \phi_2) \vee (\neg\phi_1 \wedge \neg\phi_2)$.

A Σ-*model* $\mathcal{M}$ assigns an interpretation $\underline{p}$ to every predicate symbol p and the same for function symbols and sorts. Formally, sorts are mapped to (finite and infinite) subsets of some universe $\mathcal{U}$ of values, and function and predicate symbols are mapped to functions and predicates over $\mathcal{U}$, respecting typing. The interpretation $[\![\phi]\!]\,\mathcal{M}$ of a formula ϕ in $\mathcal{M}$ is defined as usual, in particular [42]:

$$[\![f_n(t_1, \ldots, t_n)]\!]\,\mathcal{M} := \underline{f}_n([\![t_1]\!]\,\mathcal{M}, \ldots, [\![t_n]\!]\,\mathcal{M})$$

$$[\![(\textbf{if } \phi \textbf{ then } t_1 \textbf{ else } t_2)]\!]\,\mathcal{M} := \begin{cases} [\![t_1]\!]\,\mathcal{M} & \text{if } [\![\phi]\!]\,\mathcal{M} = \mathit{true} \\ [\![t_2]\!]\,\mathcal{M} & \text{else} \end{cases}$$

$$[\![\exists \mathtt{x} \in \mathcal{S} \bullet \phi]\!]\,\mathcal{M} :\Leftrightarrow \exists x \in \underline{\mathcal{S}} : [\![\phi]\!]\,\mathcal{M}[\mathtt{x} \mapsto x] = \mathit{true}$$

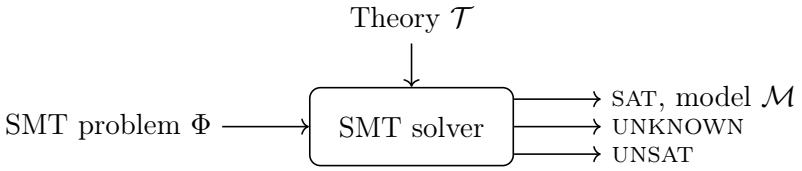


Figure 2.1: Functionality of an SMT solver

$$[\![p_n(t_1, \ldots, t_n)]\!]\,\mathcal{M} :\Leftrightarrow \underline{p}_n([\![t_1]\!]\,\mathcal{M}, \ldots, [\![t_n]\!]\,\mathcal{M})$$

We write $\mathcal{M} \models \phi$ if $[\![\phi]\!]\,\mathcal{M} = \mathit{true}$. We write $\phi_1 \Rrightarrow \phi_2$ if $\mathcal{M} \models \phi_1 \Rightarrow \mathcal{M} \models \phi_2$ for all Σ-models $\mathcal{M}$. For a set Φ of constraints, $\mathcal{M} \models \Phi$ holds if $\mathcal{M} \models \phi$ for all $\phi \in \Phi$.

In general, one is interested in models belonging to a certain theory $\mathcal{T}$. Given some signature Σ, a Σ-theory is a (possibly infinite) set of Σ-models that fixes the interpretation of sorts, functions, and predicates. A set Φ of constraints over Σ is $\mathcal{T}$-satisfiable if there exists a Σ-model $\mathcal{M} \in \mathcal{T}$ such that $\mathcal{M} \models \Phi$.

If all symbols in a formula belong to a given theory's signature Σ, it is called a *ground term*. Usually, constraints that form an SMT problem are not only ground terms. They also contain *uninterpreted* function and predicate symbols not belonging to Σ. Uninterpreted 0-ary function or predicate symbols are also called uninterpreted constants and take the role of free variables. Together with the original symbols from Σ they form a signature $\Sigma' \supseteq \Sigma$. In this case, $\mathcal{T}$-satisfiability asks for existence of a model $\mathcal{M}'$ that is an *expansion* of a model $\mathcal{M} \in \mathcal{T}$. This means, $\mathcal{M}'$ uses the same universe and same interpretations for symbols from Σ as $\mathcal{M}$ (and assigns new interpretations to uninterpreted functions and predicates from $\Upsilon' \setminus \Upsilon$, resp. $\Pi' \setminus \Pi$.

In this thesis, the only used theory is mixed Boolean and linear arithmetic over the reals, sometimes also called LinSAT. It uses decimal numbers as interpreted constants, and the standard comparison operators $(<, \leq, =)$ as predicate symbols. Pre-defined interpreted functions are the arithmetic operations addition, subtraction, and multiplication. Division is only allowed if the divisor is a ground term.

An SMT solver (Figure 2.1) takes a set of constraints Φ as input and tries to find a satisfying model in a supported theory[1] $\mathcal{T}$. Due to the undecidability of many theories, or implementation limitations, the solver may also answer with UNKNOWN if it cannot determine satisfiability. Because we apply SMT solving only to a decidable theory (linear arithmetic over the reals with quantifiers) in this thesis, UNKNOWN will only be returned in case of timeouts.

2.2 Bounded Model Checking

Bounded model checking (BMC) [27] is a technique that is commonly used on top of SMT solving. Bounded model checking targets reacheability problems for transition systems. The idea is simple: The initial and final (also called accepting) states of the system are characterized by constraints I resp. F over the state variables. We unroll the transition relation k times, introducing $k + 1$ copies of the state variables. The possible transitions are thereby encoded as a constraint T that relates the state variables between consecutive states.

[1] Or a combination of supported theories

Definition 2 (BMC problem)**.** *A BMC problem over a set* $\mathbf{X}$ *of variables is a tuple* $BMC = (I, T, F)$ *where* I *and* F *are constraints over* $\mathbf{X}$ *and* T *is a constraint over* $\mathbf{X} \cup \mathbf{X}'$*. The set* $\mathbf{X}'$ *contains a fresh variable* x' *for each* $x \in \mathbf{X}$*. Unrolling the BMC problem for* k *steps leads to an SMT problem* BMC_k *(over fresh copies* $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_k$ *of* $\mathbf{X}$*) which is*

$$BMC_k :\equiv I[\mathbf{X} \backslash \mathbf{X}_0] \wedge \bigwedge_{i=0}^{k-1} T[\mathbf{X} \backslash \mathbf{X}_i, \mathbf{X}' \backslash \mathbf{X}_{i+1}] \wedge F[\mathbf{X} \backslash \mathbf{X}_k] \ .$$

In each unrolling step, variables are replaced (denoted by squared brackets) by the instances that refer to the current state's state vector, and primed variables by references to the next step's state vector. Typically, one starts with a minimum unrolling depth (probably zero), and increases unrolling steps until a solution is found or a maximum unrolling depth reached. Algorithm 1 presents a possible implementation skeleton based on an SMTLib [28] compliant solver. Such a solver maintains a stack of *assertions*. An assertion is a constraint that is part of the SMT problem; the SMT problem to be solved is formed by the set of all assertions currently on the stack when the `(check-sat)` instruction is executed (which triggers the (semi-)decision process). With `(push)` and `(pop)`, assertions can be put on or removed from the stack. Using this stack instead of a new solver session for every unrolling step enables the solver to reuse learned information from previous calls to `(check-sat)` (i.e., doing incremental solving). With `(declare-fun` *`name`* `(`*`parameter-sorts`*`)` *`result-sort`*`)`, new uninterpreted functions are declared. A function with an empty parameter list is a constant. Analogously, `(define-fun` *`name`* `(`*`parameters`*`)` *`result-sort`* *`body`*`)` declares an interpreted function that resolves to an instance of the *`body`* term.

2.3 Traffic Sequence Charts for Requirements

The contents of this sections have been largely taken from the author's published work [11].

Traffic Sequence Charts are a graphical specification language for traffic scenarios. They enable a wide range of applications in the development process, because they have an intuitive, yet formal semantics. In [36], a number of applications (including the use as a requirement specification language) have already been identified. Recent work [40] exploits the use of TSCs in testing. Another work [39] describes a discovery process for abstract scenarios – represented as TSCs – that exposs the violation of a safety goal. In the following, the structure and semantics for TSCs is introduced.

2.3.1 An Overview on TSCs

Figure 2.2 shows a TSC that specifies an overtaking scenario. Intuitively, the TSC specifies the following behavior of a highly automated vehicle (adapted from [11]):

> *Whenever you are in the right of two lanes and there is another vehicle in front of you, closer than 50 meters and slower than you, perform the following actions within 40 seconds as long as the other vehicle remains in the right lane and does not accelerate:*
>
> 1. *Move to the left lane, and accelerate until you are at least 20 km/h faster than the other vehicle, but stay behind it.*

Algorithm 1: Bounded Model Checking

Input: BMC problem $BMC = (I, T, F)$ over variables $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$, minimum and maximum unrolling depth $\ell \leq u$

Output: A model $\mathcal{M} \models BMC_k$ ($\ell \leq k \leq u$) or UNSAT or UNKNOWN

```
1  (define-fun p_I (x_1 x_2 ... x_n) Bool I);
2  (define-fun p_T (x_1 x'_1 x_2 x'_2 ... x_n x'_n) Bool T);
3  (define-fun p_F (x_1 x_2 ... x_n) Bool F);
4  for each x_i ∈ X do
5  |  (declare-fun x_{i,0} () σ(x_i));
6  end
7  (assert (p_I x_{1,0} x_{2,0} ... x_{n,0}));
8  result := UNSAT;
9  for k = 0, 1, ... u do
10 |  if k ≥ ℓ then
11 |  |  (push);
12 |  |  (assert (p_F x_{1,k} x_{2,k} ... x_{n,k}));
13 |  |  switch (check-sat) do
14 |  |  |  case SAT do return SAT, the model produced by the solver;
15 |  |  |  case UNKNOWN do result := UNKNOWN;
16 |  |  end
17 |  |  (pop);
18 |  end
19 |  for each x_i ∈ X do
20 |  |  (declare-fun x_{i,k+1} () σ(x_i));
21 |  end
22 |  (assert (p_T x_{1,k} x_{1,k+1} x_{2,k} x_{2,k+1} ... x_{n,k} x_{n,k+1}));
23 end
24 return result;
```

2. *Pass the other vehicle using the left lane.*
3. *When you are at least 50 meters in front of the vehicle, move back to the right lane.*

The basic building block of TSCs are the so-called *nodes*, which are the boxes numbered (1) to (6) in Figure 2.2. All nodes in the example are so-called *invariant nodes* that each contain graphical representations of traffic situations, called *spatial views*[2] Nodes are assembled to so-called *basic charts*[3].

Within this thesis, two types of TSCs are used: TSCs for requirement specification, and so-called *existential* TSCs. A TSC of the former type consists of three basic charts: the *history*, the *future* and the *consequence*. Informally, the semantics of such a TSC is as follows: Whenever the TSC's *history* held from some point in the past until now, and the

[2] In earlier (before $\approx$ 2020) publications, TSCs used a different terminology. No distinction between invariant nodes and the containing spatial view was made. The combination of invariant node and spatial view was called *snapshot*. It turned out that the term snapshot is misleading because it suggests a time instant rather than a state invariant. Therefore, the old terminology has been replaced by the terminology used in this thesis.

[3] called *snapshot charts* in earlier literature

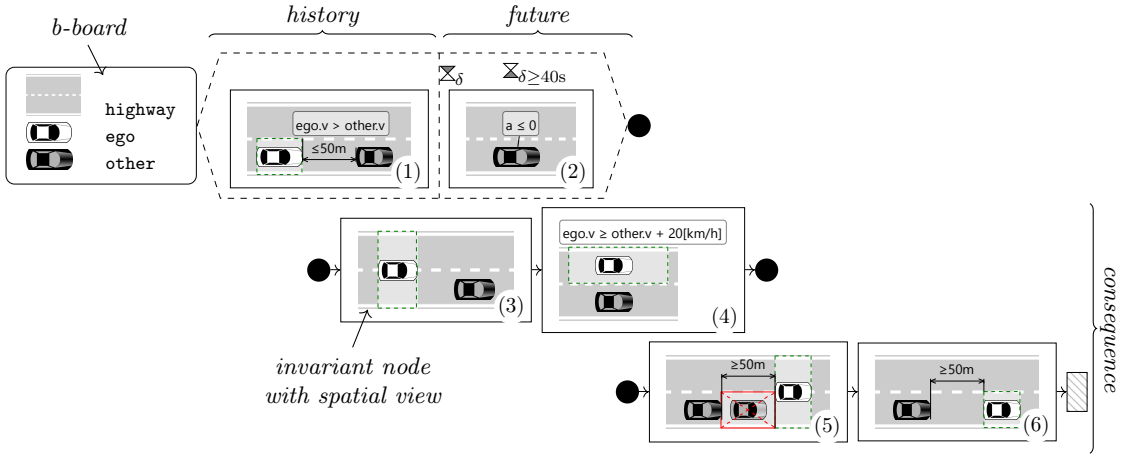
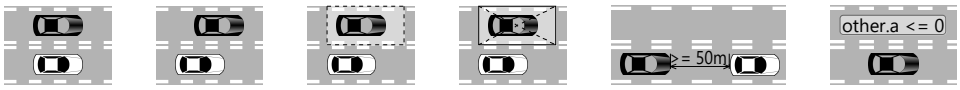


Figure 2.2: TSC "Overtaking" (adapted from [11])

TSC's *future* will hold from now till some point in the future, then also the *consequence* shall hold from now on. History and future form the so-called *pre-chart* of a TSC. The pre-chart is drawn as a bisected dashed hexagon, where the history is in the left, and the future is in the right part (in Figure 2.2, history and future consist of one invariant node each). Everything that follows the pre-chart (i.e. the second and third row of Figure 2.2) is part of the consequence. The rounded rectangle in Figure 2.2 is the *bulletin board* of the TSC and explained later.

In contrast to TSCs for requirement specification, existential TSCs do not have a pre-chart, but solely consist of a bulletin board and a consequence. They are used to describe abstract scenarios.

In a spatial view, symbols are placed to describe the spatial relations of their real-world counterparts. For instance, the spatial views (a) and (b) in Figure 2.3 specify that the `ego`-vehicle (the HAV under development, white car) is in a lane, and that there exists a second lane with another vehicle (black car) to the left of `ego`. The relative placement of symbols induces somewhat strict constraints: Spatial view (a) is only satisfied in traffic situations where front and rear of the two vehicles are aligned exactly. Spatial view (b) is satisfied only if the front of `ego` is strictly between the front and rear of the other vehicle. *Somewhere-boxes*, as in spatial view (c), can be used to relax some of the constraints. The somewhere box in spatial view (c) means that there is a vehicle somewhere on the left lane, next to or in front of `ego`. Note that the left border of the box aligns with the left (i.e., rear) border of the `ego` symbol. Therefore, the box does not match cars in the left lane behind `ego`. To this end, spatial views speak only about the presence of objects at certain positions, but do not restrict the existence of other objects. For example, there could be additional cars around the ego vehicle in spatial views (a) to (c). To specify absence, *nowhere-boxes* are used. They are used analogous to somewhere-boxes, but with the opposite meaning. For example, the nowhere-box in spatial view (d) states that there must not be another vehicle on the left lane next to or in front of `ego`. The graphical placement of symbols in a spatial view can be complemented by textual annotations. Textual annotations can be placed on distance lines or attached to one or more symbols. For instance, the distance line in spatial view (e) states that the other vehicle is at least 50 meters behind `ego`. Note, that a distance line does not imply free space between the connected objects – free space has to be made explicit by placing a nowhere-box with a symbol denoting the excluded



(a) (b) (c) (d) (e) (f)

Figure 2.3: Example spatial views

object type, as done in Figure 2.2, invariant (5). The textual constraint shown in spatial view (f) states that the acceleration of `other` is negative. For better readability, invariant nodes with empty spatial views are drawn as hatched rectangles ▨. Empty spatial views are always satisfied.

Basic charts combine invariant nodes with the operations *sequence*, *concurrency*, *choice*, and *negation* shown in Figure 2.4a–e. Furthermore, we allow multiple spatial views being combined inside one invariant node using Boolean operators. For example, Figure 2.4g shows an invariant node that combines spatial views A, B, and C to $(\neg A \vee B) \wedge C$. We use rounded rectangles (e.g. (*A*)) as placeholders for arbitrary sub-charts. A sequence of two invariant nodes means that the first node has to hold, followed by the second. Invariant nodes in a sequence follow each other directly; there is no time span between the first and the second one. Concurrency means that both nodes (resp. basic charts) have to hold simultaneously, and choice means that at least one of the charts has to hold, respectively. Negation means the chart must *not* hold. There are two kinds of timing constraints in TSCs. *Hour glasses* constrain the duration of a chart. A full hour glass (⧗) starts a clock, and the empty hour glass (⧖) ends a clock and specifies a clock constraint, i.e., the duration between start and end. For example, the hour glasses in Figure 2.4e state that the sequence A → B → C → D is completed in at most 5 s. Another form of timing constraints are *time pins* (🔎), which synchronize different time points in a chart. They are either put above a sequence arrow or above a chart. In the latter case, multiple time pins can be set in a row imposing an ordering among the time points. Time pins with the same label refer to the same time point, and hence synchronize charts.

In this thesis, we restrict ourselves to basic charts that do not use chart negation (negation of spatial views is allowed[4]). We call this subset *positive (existential/requirement) TSCs*. Because a negated basic chart allows almost any behavior, including unwanted behavior, negation is barely needed.

The *bulletin board* binds symbols in the spatial views to certain objects. A symbol that is contained in the bulletin board refers to the same object in every spatial view. The bulletin board (rounded box) in Figure 2.2 binds the white car symbol to the ego vehicle, and the lane symbols to certain lanes of the road where ego is on[5]. Also, the black car symbol is bound to a vehicle that is the same in all spatial views; if the black car would not be in the bulletin board, it could be a different vehicle in any spatial view.

[4]Note that there is a semantic difference between negating an invariant node, and negating the spatial view inside an invariant node. These two forms of negation differ in time quantification. See the formal definition of TSC semantics in Section 2.4.

[5]That the lanes belong to the road used by ego is a consequence of the spatial relations expressed in the spatial views, and not part of the bulletin board. Therefore, lane symbols can also be used to model, e.g., incoming lanes at an intersection.

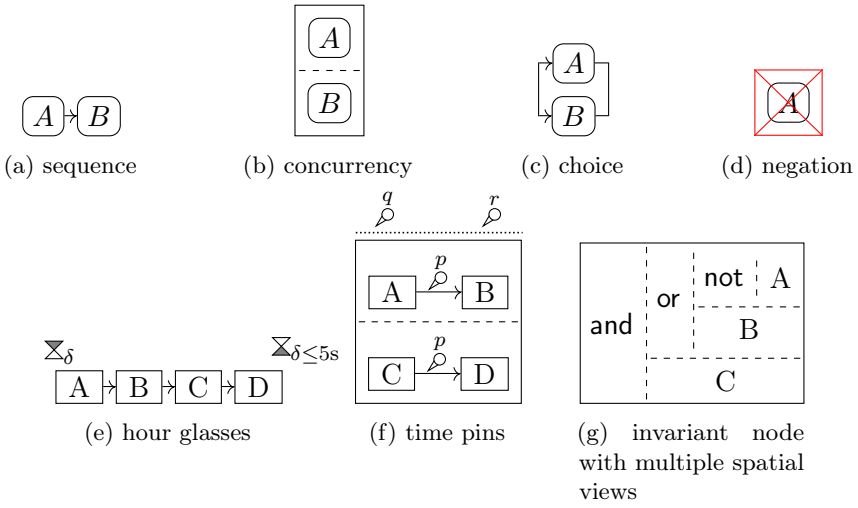


Figure 2.4: Operations on basic charts

2.3.2 World Models and Symbol Dictionaries

A TSC is always interpreted over some *world model*. As its name suggests, the world model models the different objects that a TSC can speak about. First of all, a world model specifies a hierarchy of object types (e.g., obstacles, lanes, cars, pedestrians, ...) and their attributes. Furthermore, the world model defines their dynamics, for example in form of hybrid automata [3]. World models can be defined on different abstraction levels. Especially for analysis of requirements it might be desirable to start the development process with a very abstract world model that imposes only minimal restrictions on the dynamics of the system (e.g., that vehicles have a minimum size), and use more detailed world models in later steps. As discussed later in Section 5.5, a simpler world model lowers the complexity of an analysis and allows complete model-checking. However, for some use cases such as, e.g., test-case generation, a more exact world model is needed.

The link between the world model and a TSC is defined in form of a so-called *symbol dictionary*. The symbol dictionary links the graphical symbols used in the TSCs to object classes in the world model (whereas the bulletin board binds symbols to certain objects). In this thesis, very simple symbol dictionaries and world models are used that consist of only of a few symbols and types. Figure 2.5 shows the world model that is used in the examples in this chapter (the case studies presented in Chapter 8 use slightly different world models that refine this one as needed). It defines the type `Physical Object` as the base for both `Dynamic Objects` and `Static Objects`. Commonly, an obstacle can be any object on a vehicle's path that cannot be passed. Although this also may include dynamic objects such as vehicles, in general, we consider only static obstacles (such as barriers or construction sites) here. Other static objects (roads, lanes, crossings, traffic signs) describe the road network. `Physical Object` declares six attributes $\underline{x}$, x, $\overline{x}$, $\underline{y}$, y, and $\overline{y}$ that describe a reference point (x, y) of the object, and its axis aligned bounding box. In general, attribute values may change over time (e.g., when objects move). Therefore, `Static Object` refines them explicitly to be constant. For `Dynamic Objects`, instead, we declare attributes for speed (v) and acceleration (a). The dynamics of the world model are not relevant for understanding the examples, and therefore omitted here.

The symbol dictionary is depicted in Table 2.1. In this thesis, only symbols for types

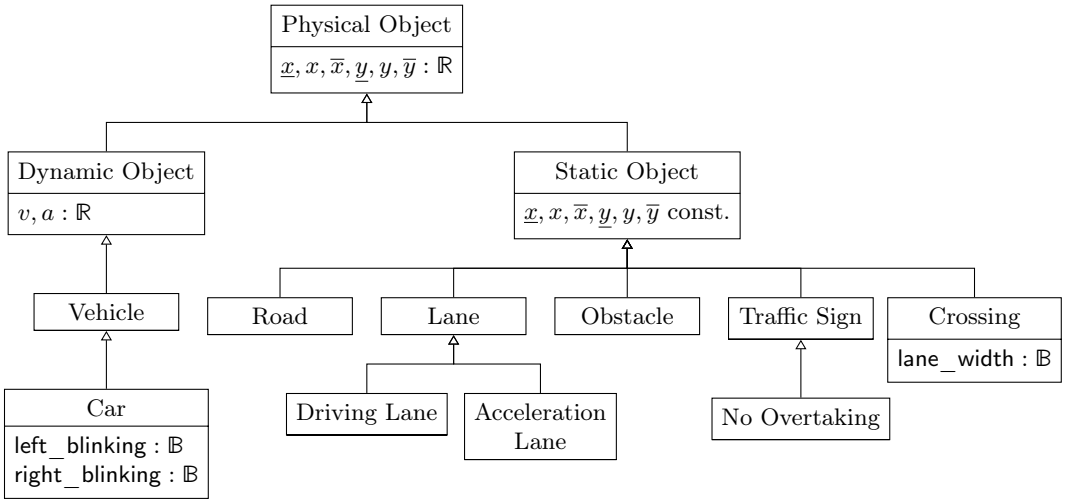


Figure 2.5: World model

at the leaves of the refinement hierarchy are used (the only exception is a symbol for the `Lane` type). To be able to distinguish between different objects of the same type, the visual representation of a symbol may be varied. For instance, the car symbol is used with different colors. Along the graphical representation of the symbols, the symbol dictionary defines so-called *anchors* on each symbol, and how they map to the attributes of the object. The anchors are used later on to derive spatial relations from a spatial view. For most objects, the anchors map to the corners of the axis-aligned bounding box, except for traffic signs where the reference point is used. Note that in the examples and case studies in this thesis, two possible ways to model two lane roads are used: either as single objects using the road symbol given in Table 2.1, or as two separate lanes (the `Driving Lane` symbol or a variant of it). This is caused by some of the case studies and examples being taken from the author's previous work [9, 11, 12, 15], where both ways are prominent.

2.4 TSC Formal Semantics

In [37, 38] the semantics of TSCs are formally defined. In the following, we recall the semantics in a condensed form. We refer the reader to the related literature for the complete semantics.

2.4.1 Semantics of Charts

Every chart BC can be translated to a *multi-sorted real-time logic (MSRTL) formula* φ_{BC} that is interpreted on some trajectory π over some *world model* WM.

For the definition of MSRTL semantics, we consider signatures $\Sigma = (\Gamma, \Pi, \Upsilon)$ that define at least a sort $\mathbb{R}$ for real numbers along with the usual arithmetic operations (addition and multiplication) and the natural ordering relation. Furthermore, we split the set of sorts into disjoint sets of *class symbols* and primitive type symbols. For simplicity, we assume that $\mathbb{R}$ and $\mathbb{B}$ are the only primitive type symbols[6]. We use real numbers also as the time domain.

[6]For the definition of MSRTL semantics, this assumption can easily be relaxed by introducing new sorts, which is a natural extension of the presented definitions. Extending the SMT encoding method described in Chapter 6 is also straight forward, as long as the SMT solver backend supports additional theories.

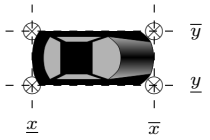
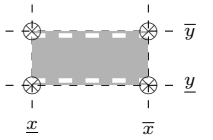
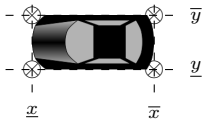
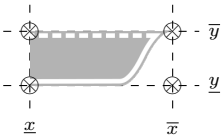
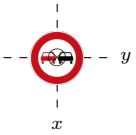
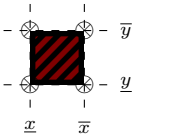
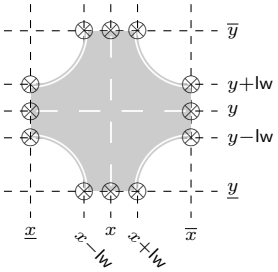
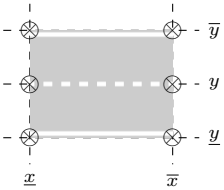
Symbol	Object Type	Symbol	Object Type
	`Car`		`Driving Lane`
	`Car` (other direction)		`Acceleration Lane`
	`No Overtaking`		`Obstacle`
	`Crossing`		`Road`

Table 2.1: Symbol dictionary; lane_width is abbreviated by lw

For now, a world model provides a universe of objects (e.g., vehicles, pedestrians, and roads), and an interpretation $\mathcal{I}$ that maps some object types from the alphabet Γ to classes of objects, each having a defined set of attributes (in the context of MSRTL, object attributes may be seen as unary function symbols). Furthermore, $\mathcal{I}$ assigns interpretations to the predicate and function symbols (excluding attributes, which are interpreted later on for a given trajectory). A world model also defines a (typically infinite) set $\mathcal{T}(\mathsf{WM})$ of trajectories over the attributes of objects. This set contains any possible behavior of the objects within the world model. For example, the world model trajectories may be seen as generated from a network of hybrid automata, as suggested in [38]. A more detailed formal definition of world models is postponed to Section 5.5.

In this work, we consider trajectories that are *piecewise continuous*. The following definition is based on a definition by Alur et al. [3].

Definition 3 (Piecewise continuous function)**.** *A function* $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ *is* piecewise continuous *if every closed interval* $I \subset \mathbb{R}_{\geq 0}$ *can be partitioned into a finite sequence* $I_1, I_2, \ldots$ *of intervals such that* f *coincides with some continuous function* $f_i : \mathrm{Cl}(I_i) \to \mathbb{R}$ *on each interval* I_i *(i.e.,* $f(t) = f_i(t)$ *for all* $i = 1, 2, \ldots$ *and* $t \in I_i$*).*

This definition implies that f has only a finite number of discontinuities on every closed interval. Furthermore, the f_i are required to be defined and continuous on the closure $\mathrm{Cl}(I_i)$ of each interval, i.e., the smallest closed interval containing I_i (the partition $I_1, I_2, \ldots$ naturally contains both open and closed intervals). As a consequence, all discontinuities are "jumps" where (different) limits of f exists from both sides. The definition naturally extends to functions with a higher dimensional range.

Definition 4 (Trajectory)**.** *A* trajectory *over an (ordered) set* $\mathbf{X}$ *of variables is a piecewise continuous function* $\pi\colon \mathbb{R}_{\geq 0} \to \mathbb{R}^{\mathbf{X}}$*. A trajectory over some finite set* O_π *of objects from some world model* WM *is a trajectory over* $\mathbf{X} \supseteq \mathcal{A}(O_\pi)$*, containing variables for all attributes of objects in* O_π*. We write* $\pi(t)$ *for the state vector at time* t *and* $\pi(t)[obj.a]$ *to access the value of the attribute* a *of object* $obj \in O_\pi$ *at time* t*. The universe of trajectories over* $\mathbf{X}$ *is denoted* $\mathcal{T}(\mathbf{X})$*.*

The term π^{t_0} *denotes a trajectory* π *time-shifted by* t_0*, i.e.,* $\pi^{t_0}(t) = \pi(t + t_0)$*, which is a suffix of* π*. The set of all suffixes is defined as* $\mathit{Suffix}(\pi) := \{\pi^t \mid t \in \mathbb{R}_{\geq 0}\}$ *for a single trajectory, and as* $\mathit{Suffix}(\Pi) := \bigcup_{\pi\in\Pi} \mathit{Suffix}(\pi)$ *for a set of trajectories. We use* $\pi{\downarrow_t}$ *to denote a prefix of* π*, i.e. a reduction of* π *to the domain* $[0, t]$*. The projection of* π *to a subset* $\mathbf{Y} \subseteq \mathbf{X}$ *of variables is written as* $\pi{\downarrow_{\mathbf{Y}}}$*.*

By interpreting the Boole sort as a subset $\mathbb{B} = \{0, 1\} \subset \mathbb{R}$ of the reals, the above definition also covers Boolean attributes of objects[7].

In contrast to related work, we do not consider scenarios during which objects are created or destroyed. An attribute *o.alive* (as introduced in [37, 38]) that indicates if object o is currently existing is not required. Instead, by convention a trajectory defines values for all existing objects (and vice versa, an object exists in the trajectory if values for its attributes are defined).

Definition 5 (Multi-sorted Real-time Logic [37, 38])**.** *An MSRTL formula* φ *is of the form*

$$\begin{aligned}\varphi &::= p(\psi_1, \ldots, \psi_n) \mid \exists\tau : \varphi \mid \exists o \in \mathsf{O} : \varphi \mid \exists x \in \mathbb{R} : \varphi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \\ &\quad \mid \Box_{[\tau_1,\tau_2)}\varphi \mid \tau_1 - \tau_2 \bowtie X \\ \psi &::= f(\psi_1, \ldots, \psi_n) \mid X \mid o.a \mid x\end{aligned}$$

where p *is an* n*-ary predicate symbol,* f *an* n*-ary function symbol,* o *is an object variable,* a *is an attribute,* τ, τ_1, τ_2 *are time variables,* x *is a primitive variable,* $\bowtie \in \{<, \leq, =, \geq, >\}$*, and* $X \in \mathbb{R}$ *is a constant. An MSRTL formula that does not contain temporal operators or time variables (i.e., it uses none of* $\exists\tau : \varphi \mid \Box_{[\tau_1,\tau_2)}\varphi \mid \tau_1 - \tau_2 \bowtie X$*) is called an MSRTL predicate.*

Furthermore, the usual abbreviations are used: $\forall\tau : \varphi$ is the usual abbreviation for $\neg\exists\tau : \neg\varphi$, as well as $\forall o \in \mathsf{O} : \varphi$ abbreviates $\neg\exists o \in \mathsf{O} : \neg\varphi$. Implication $\varphi_1 \Rightarrow \varphi_2$ stands for $\neg\varphi_1 \vee \varphi_2$. Also, we simply write $\tau_1 \bowtie \tau_2$ instead of $\tau_1 - \tau_2 \bowtie 0$.

In the following, a valuation is a function that maps variables to values. We write a valuation that assigns values $v_1, v_2, \ldots$ to variables $x_1, x_2, \ldots$ as $\{x_1 \mapsto v_1, x_2 \mapsto v_2, \ldots\}$. Given a valuation μ, the term $\mu \cup \{x \mapsto v\}$ denotes valuation μ extended by mapping $x \mapsto v$ (i.e., the valuation μ' with $\mu'(x) = v$ and $\mu'(x') = \mu(x')$ for $x' \neq x$).

Definition 6 (MSRTL semantics [37, 38])**.** *Consider a world model* WM *for a signature* Σ *and an MSRTL formula* φ *over some signature* $\Sigma' \subseteq \Sigma$*. Then,* satisfaction *of* φ *by a*

[7] For efficiency, however, a practical implementation (including the consistency analysis prototype) would use a dedicated Boole type

trajectory π over objects O_π from WM *and a valuation μ is defined as follows:*

$$
\begin{array}{lcl}
[\![o.a]\!]_{\mathsf{WM}}(\pi(0),\mu) & := & \pi(0)[\mu(o).a] \\
[\![X]\!]_{\mathsf{WM}}(\pi(0),\mu) & := & X \text{ for constants } X \\
[\![x]\!]_{\mathsf{WM}}(\pi(0),\mu) & := & \mu(x) \text{ for variables } x \\
[\![f(\psi_1,\ldots,\psi_n)]\!]_{\mathsf{WM}}(\pi(0),\mu) & := & \mathcal{I}(f)([\![\psi_1]\!]_{\mathsf{WM}}(\pi(0),\mu),\ldots,[\![\psi_n]\!]_{\mathsf{WM}}(\pi(0),\mu)) \\
\pi,\mu \models_{\mathsf{WM}} p(\psi_1,\ldots,\psi_n) & :\Leftrightarrow & \mathcal{I}(p)([\![\psi_1]\!](\pi(0),\mu),\ldots,[\![\psi_n]\!](\pi(0),\mu)) \\
\pi,\mu \models_{\mathsf{WM}} \Box_{[\tau_1,\tau_2)}\varphi & :\Leftrightarrow & \forall \tau \in [\mu(\tau_1),\mu(\tau_2)) : \pi^\tau,\mu \models_{\mathsf{WM}} \varphi \\
\pi,\mu \models_{\mathsf{WM}} \exists \tau : \varphi & :\Leftrightarrow & \exists v \in \mathbb{R}_{\geq 0} : \pi,\mu \cup \{\tau \mapsto v\} \models_{\mathsf{WM}} \varphi \\
\pi,\mu \models_{\mathsf{WM}} \exists x \in \mathbb{R} : \varphi & :\Leftrightarrow & \exists v \in \mathbb{R} : \pi,\mu \cup \{x \mapsto v\} \models_{\mathsf{WM}} \varphi \\
\pi,\mu \models_{\mathsf{WM}} \exists o \in \mathsf{O} : \varphi & :\Leftrightarrow & \exists obj \in \mathcal{I}(\mathsf{O}) \cap O_\pi : \pi,\mu \cup \{o \mapsto obj\} \models_{\mathsf{WM}} \varphi \\
\pi,\mu \models_{\mathsf{WM}} \tau_1 - \tau_2 \bowtie X & :\Leftrightarrow & \mu(\tau_1) - \mu(\tau_2) \bowtie X
\end{array}
$$

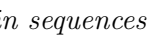
Boolean operators are defined as usual.

Definition 7 (Scope of time pins and hour glasses)**.** *The scope of a time pin or hour glass is determined by considering a TSC as a whole. A hour glass or time pin x is implicitly* introduced by *some sub-chart* BC *of the TSC if either* BC *is the only chart that is annotated with x, or* BC *is the innermost basic chart that contains* all *sub-charts that are annotated with x.*

Now, we are ready to define TSC semantics in terms of MSRTL.

Definition 8 (Translation of basic charts)**.** *A chart* BC *translates to a MSRTL formula φ_{BC} with free time variables b_{BC} and e_{BC} that stand for begin and end of the chart in a trajectory.*

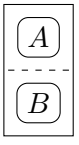
For each time pin or hour glass x, introduce a time variable τ_x. Then, each timing annotation ta translates to a constraint ψ_{ta} as follows:

- *Hour glass starts $\boxtimes_{hg}$ are translated to $\psi_{ta} :\equiv \tau_{hg} = b_{\mathsf{BC}}$.*
- *Hour glass ends $\boxtimes_{hg} \bowtie X$ are translated to $\psi_{ta} :\equiv e_{\mathsf{BC}} - \tau_{hg} \bowtie X$.*
- *Time pin sequences*  *are translated to $\psi_{ta} :\equiv b_{\mathsf{BC}} \leq \tau_{p_1} < \cdots < \tau_{p_n} \leq e_{\mathsf{BC}}$.*

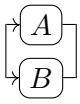
A basic chart BC*, annotated with timing annotations $ta_1,\ldots,ta_n$ and introducing time pins and hour glasses $x_1,\ldots,x_m$, is translated as*

$$\varphi_{\mathsf{BC}} :\equiv \exists \tau_{x_1},\ldots,\tau_{x_m} : \varphi'_{\mathsf{BC}} \wedge \psi_{ta_1} \wedge \cdots \wedge \psi_{ta_n} \ ,$$

where φ'_{BC} is the semantics of the unannotated chart as defined below. Square brackets $[x\backslash y]$ denote substitution of variable x by variable y.

- *If* BC *is an invariant node, then $\varphi'_{\mathsf{BC}} :\equiv b_{\mathsf{BC}} < e_{\mathsf{BC}} \wedge \Box_{[b_{\mathsf{BC}},e_{\mathsf{BC}})}\phi$ where ϕ is the translation of the node's spatial view contents into a predicate (see Section 2.4.2 for the translation of spatial views).*
- *Concurrency: If* $\mathsf{BC} =$  *then*

$$\varphi'_{\mathsf{BC}} :\equiv \varphi_A[b_A\backslash b_{\mathsf{BC}}, e_A\backslash e_{\mathsf{BC}}] \wedge \varphi_B[b_B\backslash b_{\mathsf{BC}}, e_B\backslash e_{\mathsf{BC}}]$$

- *Choice: If* $\mathsf{BC} =$  *then*

$$\varphi_{\mathsf{BC}} :\equiv \varphi_A[b_A\backslash b_{\mathsf{BC}}, e_A\backslash e_{\mathsf{BC}}] \vee \varphi_B[b_B\backslash b_{\mathsf{BC}}, e_B\backslash e_{\mathsf{BC}}]$$

- *Negation: If* $\mathsf{BC} =$  *then*

$$\varphi_{\mathsf{BC}} := \neg\varphi_A[b_A\backslash b_{\mathsf{BC}}, e_A\backslash e_{\mathsf{BC}}]$$

- *Sequence: If* $\mathsf{BC} =$  *(optionally with time pin, i.e.*  *) then*

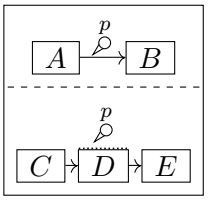
$$\varphi_{\mathsf{BC}} :\equiv \exists\tau : \varphi_A[b_A\backslash b_{\mathsf{BC}}, e_A\backslash\tau] \wedge \varphi_B[b_B\backslash\tau, e_B\backslash e_{\mathsf{BC}}] \underbrace{\wedge\, \tau_p = \tau}_{\text{only if time pin } p \text{ is present}}$$

If the contents of an invariant node is a Boolean combination of spatial views $sv_1, \ldots, sv_n$*, then* $\phi :\equiv p_n(\phi_{sv_1}, \ldots, \phi_{sv_1})$*. Here,* p_n *is the* n*-ary predicate formed by the Boolean operators that combine the spatial views.*

For example, the combination of spatial views A, B, C depicted in Figure 2.4g forms the predicate $\phi \equiv (\neg\phi_{\mathrm{A}} \vee \phi_{\mathrm{B}}) \wedge \phi_{\mathrm{C}}$.

Note, that in the above inductive definition, the substitutions $[x\backslash y]$ only replace free variables. The resulting MSRTL formula φ_{BC} has two dedicated free time variables b_{BC} and e_{BC}. All other free time variables denote time pins and hour glasses (note, that the variable τ introduced for a sequence is existentially bound and not free).

Example 1. *The basic chart* BC *given below translates to the MSRTL formula:*



$$\exists\tau_p \bullet \bigwedge \left\{ \begin{array}{c} \exists\tau_1 \bullet \bigwedge \left\{ \begin{array}{c} b_{\mathsf{BC}} < \tau_1 \wedge \Box_{[b_{\mathsf{BC}},\tau_1)}\phi_A \\ \tau_1 < e_{\mathsf{BC}} \wedge \Box_{[\tau_1,e_{\mathsf{BC}})}\phi_B \\ \tau_1 = \tau_p \end{array} \right\} \\ \exists\tau_2 \bullet \bigwedge \left\{ \begin{array}{c} 0 < \tau_2 \wedge \Box_{[0,\tau_1)}\phi_C \\ \exists\tau_3 \bullet \bigwedge \left\{ \begin{array}{c} \tau_2 < \tau_3 \wedge \Box_{[\tau_3,\tau_4)}\phi_D \\ \tau_2 \leq \tau_p \leq \tau_3 \\ \tau_3 < e_{\mathsf{BC}} \wedge \Box_{[\tau_4,e_{\mathsf{BC}})}\phi_E \end{array} \right\} \end{array} \right\} \end{array} \right\}$$

Definition 9 (TSC Semantics)**.** *A trajectory* $\pi \in \mathcal{T}(\mathsf{WM})$ *satisfies a TSC* TSC *with history* H*, future* F*, and consequence* C *in the world model* WM*, written* $\pi \models_{\mathsf{WM}} \mathsf{TSC}$*, if*

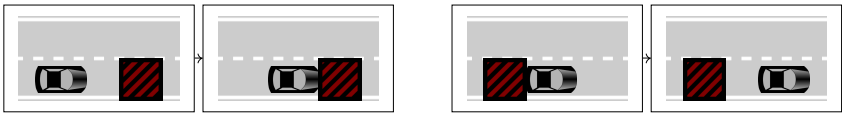
$$\pi \models_{\mathsf{WM}} \forall o_1 \in \mathsf{O}_1, o_2 \in \mathsf{O}_2, \ldots \forall b, m, e : \\ (\varphi_H[b_H\backslash b, e_H\backslash m] \wedge \varphi_F[b_F\backslash m, e_F\backslash e]) \Rightarrow \varphi_C[b_C\backslash m, e_C\backslash e] \ .$$

Here, $o_1 \in \mathsf{O}_1, o_2 \in \mathsf{O}_2, \ldots$ *is the list of free object variables in* φ_H*,* φ_F*,* φ_C *originating from the b-board.*

A trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ *satisfies an existential TSC with consequence* BC *if*

$$\pi \models_{\mathsf{WM}} \exists o_1 \in \mathsf{O}_1, o_2 \in \mathsf{O}_2, \ldots \exists b, e : \varphi_{\mathsf{BC}}[b_{\mathsf{BC}}\backslash m, e_{\mathsf{BC}}\backslash e] \ .$$

We say some TSC TSC *is* satisfiable *in a world model* WM*, written* $\mathit{SAT}_{\mathsf{WM}}(\mathsf{TSC})$ *if there exists some trajectory* $\pi \in \mathcal{T}(\mathsf{WM})$ *with* $\pi \models_{\mathsf{WM}} \mathsf{TSC}$*.*



(a) Satisfiable sequence (b) Unsatisfiable sequence

Figure 2.6: Examples of a satisfiable (left) and an unsatisfiable (right) sequence of invariant nodes

Definition 10 (Equivalence of charts)**.** *A chart A* implies *another chart B, written $A \Rrightarrow B$, if, for all trajectories π and valuations μ (in any world model where both A and B are defined):*

$$(\pi, \mu \models \varphi_A[b_A \backslash b, e_A \backslash e]) \Longrightarrow (\pi, \mu \models \varphi_B[b_B \backslash b, e_B \backslash e]) \ .$$

Here, b and e are fresh time variables. Two charts A and B are equivalent*, written $A \equiv B$, if both $A \Rrightarrow B$ and $B \Rrightarrow A$.*

A note on the semantics of invariants. This thesis sticks to the original TSC semantics given by Damm et al. [37, 38] in the sense that invariant nodes are always evaluated on a left-closed, right-open interval. While this definition is quite intuitive and uncomplicated at first sight, it has some practical implications, as shown by the following example.

Example 2 (Infinitissimal discontinuity)**.** *Considering the two sequences depicted in Figure 2.6, the left sequence is usually satisfiable, while the right one is not. Assuming that the distance d between the car and the obstacle is a continuous function, the left sequence requires that there exists some time points $b < m < e$ such that $d(t) > 0$ for all $b \leq t < m$, and $d(t) = 0$ for all $m \leq t < e$. This is satisfied for example if $d(t) = (m - t)^2$ for $t \leq m$ and $d(t) = 0$ for $t \geq m$. The right sequence, however, requires $d(t) = 0$ for $b \leq t < m$ and $d(t) > 0$ for $m \leq t < e$. This is not possible with a continuous d because, however small $d(m) > 0$ is, there would be some $b < t < m$ with $d(t) > 0$, though.*

This problem may be resolved by changing the semantics of invariant nodes (respectively time-bounded globally in MSRTL) from "satisfied *everywhere* on $[b, e)$" to "satisfied *almost everywhere* on $[b, e]$". A similar semantics is applied for Duration Calculus (cf. Section 3.3), which uses Riemann integrals to evaluate state invariants. Improving the semantics of TSCs, however, is future work and outside the scope of this thesis.

2.4.2 Semantics of Spatial Views

In this section, we have a closer look to the semantics of spatial views. Figure 2.9 shows a conceptual meta model for spatial views. According to the semantics definition in [37, 38], a spatial view is made up from *frames*. Frames contain *spatial elements* which are *symbol occurrences* and *boxes*[8]. The spatial view itself is a frame, and each *box* inside the spatial view is also a frame. Spatial elements are equipped with anchors, which form the basis for the semantics of spatial views.

Definition 11 (Elements of a frame [37, 38])**.** *We adopt the notion from [37, 38] and define for some frame f:*

- $\mathcal{S}_f^0$ *is the set of* symbol occurrences *within f*

[8] [37, 38] use the term *symbol* for both symbol occurrences and boxes. In this thesis the term *spatial element* is used to avoid confusion.

- $\mathcal{S}_B \subseteq \mathcal{S}_f^{\mathtt{O}}$ *is the set of symbol occurrences with reference to the bulletin board*
- $\mathcal{S}_f^{\mathtt{SB}}$ *is the set of* somewhere boxes *within* f
- $\mathcal{S}_f^{\mathtt{NB}}$ *is the set of* nowhere boxes *within* f
- $\mathcal{S}_f^{\mathtt{B}} := \mathcal{S}_f^{\mathtt{SB}} \cup \mathcal{S}_f^{\mathtt{NB}}$ *is the set of* boxes *in* f
- $\mathcal{S}_f := \mathcal{S}_f^{\mathtt{O}} \cup \mathcal{S}_f^{\mathtt{B}}$ *is the set of* spatial elements *in* f
- A_f *is the set of* attached predicates *in* f
- L_f *is the set of* distance arrows *in* f

Furthermore, a symbol occurrence $s \in \mathcal{S}_f^{\mathtt{O}}$ *belongs to a* symbol definition $s.\mathfrak{s}$ *and has some type* $\mathsf{type}(s.\mathfrak{s}) \in \mathcal{C}$ *defined in the world model.*

Definition 12 (Spatial view semantics [37, 38])**.** *A frame* f *containing* $n \in \mathbb{N}$ *symbol occurrences and* $m \in \mathbb{N}$ *boxes is translated to an MSRTL formula*

$$
\begin{aligned}
\phi_f :\equiv\ & \exists o_{s_1.id} \in \mathsf{type}(s_1.\mathfrak{s}) \cdots \exists o_{s_n.id} \in \mathsf{type}(s_n.\mathfrak{s}) \\
& \exists \underline{x}_{b_1.id}, \overline{x}_{b_1.id}, \underline{y}_{b_1.id}, \overline{y}_{b_1.id} \in \mathbb{R} \cdots \exists \underline{x}_{b_m.id}, \overline{x}_{b_m.id}, \underline{y}_{b_m.id}, \overline{y}_{b_m.id} \in \mathbb{R} \\
& \quad \bigwedge_{s \in \mathcal{S}_f^{\mathtt{B}}} (\underline{x}_{s.id} \leq \overline{x}_{s.id} \wedge \underline{y}_{s.id} \leq \overline{y}_{s.id}) \\
& \quad \wedge \bigwedge_{s \in \mathcal{S}_f^{\mathtt{O}}} \mathtt{O}_{s.\mathfrak{s}}(o_{s.id}) \\
& \quad \wedge \mathtt{topologicalRelations}(f) \\
& \quad \wedge \bigwedge_{l \in A_f} \mathtt{P}(l) \wedge \bigwedge_{l \in L_f} \mathtt{D}(l) \\
& \quad \wedge \bigwedge_{s \in \mathcal{S}_f^{\mathtt{SB}}} \phi_s \wedge \bigwedge_{s \in \mathcal{S}_f^{\mathtt{NB}}} \neg \phi_s
\end{aligned}
$$

where $s_1, \ldots, s_n \in \mathcal{S}_f^{\mathtt{O}} \setminus \mathcal{S}_B$ *(with* $n = |\mathcal{S}_f^{\mathtt{O}} \setminus \mathcal{S}_B|$*) range over all symbol occurrences in* f*, except those from the bulletin board, and* $b_1, \ldots, b_m \in \mathcal{S}_f^{\mathtt{B}}$ *(with* $m = |\mathcal{S}_f^{\mathtt{B}}|$*) range over all somewhere and nowhere boxes in* f*.*

The predicates `O`, `P`, `D`, and `topologicalRelations` are defined in the following.

Definition 13 (Visualized state)**.** *A symbol definition visualizes the state of an object, such as a driving direction, or the state of turning indicators. The state that is expressed within some symbol* $s \in \mathcal{S}_f^{\mathtt{O}}$ *is defined as part of its symbol definition* $s.\mathfrak{s}$ *and translated as the predicate* $\mathtt{O}_{s.\mathfrak{s}}(o_{s.id})$*.*

For example, we can visually indicate the state of a car's indicators in the car symbol as shown in Figure 2.7.

Definition 14 (Semantics of attached predicates)**.** *An attached predicate* $l \in A_f$ *is attached to* $n \in \mathbb{N}$ *spatial elements* $l.s_1, \ldots, l.s_n$ *and labeled with a predicate* $\mathtt{label\text{-}predicate}_l$*. It is translated to*

$$\mathtt{P}(l) :\equiv \mathtt{label\text{-}predicate}_l(o_{(l.s_1).id}, \ldots, o_{(l.s_n).id}) \ .$$

left_blinking = 1 left_blinking = 0 right_blinking = 1 right_blinking = 0

Figure 2.7: Visualization of car indicator state

For translating spatial constraints, the *anchors* of the spatial elements are used. Each spatial element s has some set $\mathcal{A}(s)$ of anchors. Each anchor $s.\alpha \in \mathcal{A}(s)$ has a position $(s.\alpha.x, s.\alpha.y) \in \mathbb{R} \times \mathbb{R}$ in the spatial view. This is the position where the anchor is placed *visually* when drawing a spatial view. Opposed to that, the anchor has a *semantical position* $\langle \mathtt{A}_x(s.\alpha), \mathtt{A}_y(s.\alpha) \rangle$. Note, that the anchors itself are not shown in the spatial views; their position is inferred from the symbol dictionary.

Definition 15 (Semantical position $\langle \mathtt{A}_x(s.\alpha), \mathtt{A}_y(s.\alpha) \rangle$ of anchors)**.** *For object symbols, the symbol dictionary* `sdic` *maps some anchor $s.\mathfrak{z}.\alpha$ to a pair $\langle \mathtt{sdic}(s.\mathfrak{z}.\alpha.x), \mathtt{sdic}(s.\mathfrak{z}.\alpha.y) \rangle$ of expressions over the object attributes. When translating a frame, these expressions are instantiated for the symbol's object variable, i.e., $\mathtt{A}_x(s.\alpha) := \mathtt{sdic}(s.\mathfrak{z}.\alpha.x)(o_{s.id})$ and $\mathtt{A}_y(s.\alpha) := \mathtt{sdic}(s.\mathfrak{z}.\alpha.y)(o_{s.id})$.*

In case of a box, $\mathtt{A}_x(s.\alpha)$ and $\mathtt{A}_y(s.\alpha)$ translate to the variables $\underline{x}_{s.id}$, $\overline{x}_{s.id}$, $\underline{y}_{s.id}$, and $\overline{y}_{s.id}$, depending on the anchor α. Per definition, boxes have four anchors, one in each corner, as indicated in Figure 2.8.

Definition 16 (Semantics of distance lines)**.** *A distance line l connects two anchors $l.s_1.a_1$ and $l.s_2.a_2$ and is labeled with a distance predicate $\mathtt{dist\text{-}pred}_l$. The semantics of a* horizontal *distance line l is*

$$\begin{aligned}\mathtt{D}(l) &:\equiv \mathtt{dist\text{-}pred}_l(|\mathtt{A}_x(l.s_1.\alpha_1) - \mathtt{A}_x(l.s_2.\alpha_2)|) \\ &\equiv \mathtt{dist\text{-}pred}_l(\mathtt{A}_x(l.s_1.\alpha_1) - \mathtt{A}_x(l.s_2.\alpha_2)) \vee \mathtt{dist\text{-}pred}_l(\mathtt{A}_x(l.s_2.\alpha_2) - \mathtt{A}_x(l.s_1.\alpha_1)) \ .\end{aligned}$$

The semantics of a verical *distance line l is*

$$\begin{aligned}\mathtt{D}(l) &:\equiv \mathtt{dist\text{-}pred}_l(|\mathtt{A}_y(l.s_1.\alpha_1) - \mathtt{A}_y(l.s_2.\alpha_2)|) \\ &\equiv \mathtt{dist\text{-}pred}_l(\mathtt{A}_y(l.s_1.\alpha_1) - \mathtt{A}_y(l.s_2.\alpha_2)) \vee \mathtt{dist\text{-}pred}_l(\mathtt{A}_y(l.s_2.\alpha_2) - \mathtt{A}_y(l.s_1.\alpha_1)) \ .\end{aligned}$$

Definition 17 (Topological relations)**.** *The topological relations in a frame f are defined as*

$$\mathtt{topologicalRelations}(f) :\equiv \bigwedge_{(s,s') \in \mathcal{S}_f \times (\mathcal{S}_f \cup \mathcal{S}_{spans(f)})} \mathtt{topologyBetween}(s, s')$$

where `topologyBetween` *is defined in Algorithm 2 below and $\mathcal{S}_{spans(f)} = \{f\}$ if f is a box and $\mathcal{S}_{spans(f)} = \emptyset$ if f is the top-level spatial view.*

Example 3 (Semantics of a spatial view)**.** *Consider the spatial view depicted in Figure 2.10. We denote the road object by s, the white car by a, the black car by b, and the gray car by c. Assuming that none of the object symbols is contained in the bulletin board, the spatial view translates to:*

$$\phi :\equiv \exists s \in \mathtt{Road} : \exists \underline{x}_1, \overline{x}_1, \underline{y}_1, \overline{y}_1, \underline{x}_2, \overline{x}_2, \underline{y}_2, \overline{y}_2 \in \mathbb{R} :$$

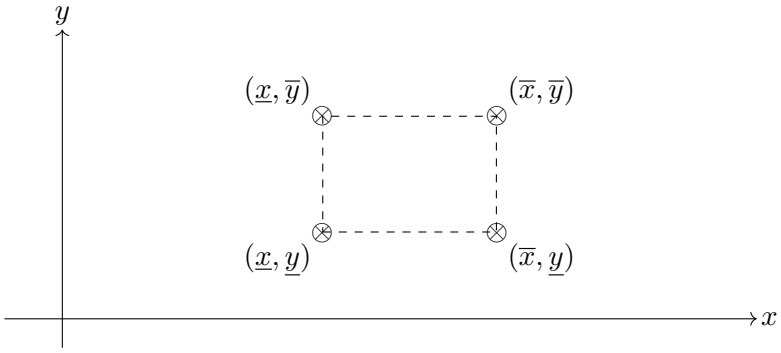


Figure 2.8: Box anchors

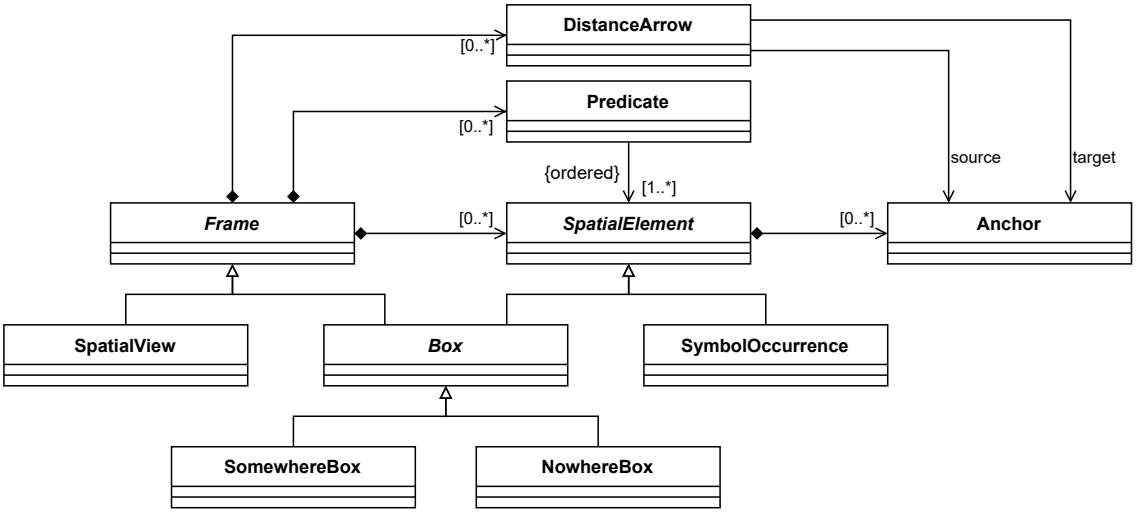


Figure 2.9: Conceptual meta model for spatial views

Algorithm 2: `topologyBetween`(s, s') (cf. [37, 38])

Input: Symbol occurrences s, s'
Output: A constraint $\mathtt{T}_{s,s'}$

$\mathtt{T}_{s,s'} := \mathit{true}$;
for each $\bowtie \in \{<, \leq, \geq, >\}$ **do**
  for each $(s.\alpha, s'.\alpha') \in \mathscr{A}(s) \times \mathscr{A}(s')$ **do**
    if $s.\alpha.x \bowtie s'.\alpha'.x$ **then**
      $\mathtt{T}_{s,s'} := \mathtt{T}_{s,s'} \wedge \mathtt{A}_x(s.\alpha) \bowtie \mathtt{A}_x(s'.\alpha')$
    end
  end
  for each $(s.\alpha, s'.\alpha') \in \mathscr{A}(s) \times \mathscr{A}(s')$ **do**
    if $s.\alpha.y \bowtie s'.\alpha'.y$ **then**
      $\mathtt{T}_{s,s'} := \mathtt{T}_{s,s'} \wedge \mathtt{A}_y(s.\alpha) \bowtie \mathtt{A}_y(s'.\alpha')$
    end
  end
end
return $\mathtt{T}_{s,s'}$

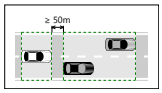


Figure 2.10: Spatial view for Example 3

$$
\begin{aligned}
& s.\underline{x} < \underline{x}_1 < \overline{x}_1 < \underline{x}_2 < \overline{x}_2 < s.\overline{x} && \textit{(outer frame, x-axis)} \\
\wedge\ & \underline{y}_1 = \underline{y}_2 = s.\underline{y} < \overline{y}_1 = \overline{y}_2 = s.\overline{y} && \textit{(outer frame, y-axis)} \\
\wedge\ & |\underline{x}_2 - \overline{x}_1| \geq 50\,\mathrm{m} && \textit{(distance line)} \\
\wedge\ & \phi_1 \wedge \phi_2
\end{aligned}
$$

$\phi_1 :\equiv \exists a \in \texttt{Car} :$

$$
\begin{aligned}
& \underline{x}_1 = a.\underline{x} < a.\overline{x} = \overline{x}_1 && \textit{(left sw-box, x-axis)} \\
\wedge\ & \underline{y}_1 < a.\underline{y} < a.\overline{y} < \overline{y}_1 && \textit{(left sw-box, y-axis)}
\end{aligned}
$$

$\phi_2 :\equiv \exists b, c \in \texttt{Car} :$

$$
\begin{aligned}
& \underline{x}_2 = b.\underline{x} < b.\overline{x} < c.\underline{x} < c.\overline{x} = \overline{x}_2 && \textit{(right sw-box, x-axis)} \\
\wedge\ & \underline{y}_2 < b.\underline{y} < b.\overline{y} < c.\underline{y} < c.\overline{y} < \overline{y}_2 && \textit{(right sw-box, y-axis)}
\end{aligned}
$$

Note, that some of the inequalities produced by `topologicalRelations` *(e.g.,* $b.\underline{x} < c.\overline{x}$*) have been omitted because they are implied by the remaining inequalities. The quantified variables of type* $\mathbb{R}$ *can be eliminated from the formula. This simplifies* ϕ *to:*

$\phi :\equiv \exists s \in \texttt{Road} : a, b, c \in \texttt{Car} :$

$$
\begin{aligned}
& s.\underline{x} < a.\underline{x} < a.\overline{x} \wedge b.\underline{x} < b.\overline{x} < c.\underline{x} < c.\overline{x} < s.\overline{x} && \textit{(x-axis)} \\
\wedge\ & b.\underline{x} - a.\overline{x} \geq 50\,\mathrm{m} && \textit{(distance arrow)} \\
\wedge\ & s.\underline{y} < a.\underline{y} < a.\overline{y} < s.\overline{y} && \textit{(y-axis from left sw-box)} \\
\wedge\ & s.\underline{y} < b.\underline{y} < b.\overline{y} < c.\underline{y} < c.\overline{y} < s.\overline{y} && \textit{(y-axis from right sw-box)}
\end{aligned}
$$

An optimized variant of Algorithm 2 is given later on in Section 6.4. This optimized version of the algorithm omits transitive relations as done in Example 3 above.

3 Related Work

In this chapter, a summary about related work is presented. This thesis is related to four major fields of research:

- Consistency analysis of requirements
- Scenario-driven development in the automotive industry
- Model checking/verification of hybrid systems
- Trajectory generation/instatiation of abstract scenarios

The third one, hybrid systems verification, also includes work about model checking Duration Calculus [23], a well-known interval logic for specifying real-time systems. Satisfiability of Duration Calculus has been extensively studied and the results can be transferred to TSCs. The following sections provide brief summaries of related work in each of these fields. The first three sections cover work that this thesis directly build upon: Consistency, scenario-driven development, and Duration Calculus. The remaining two sections report on alternative and related approaches to hybrid systems verification and trajectory generation.

3.1 Consistency

In general, consistency of requirements can be defined as "no two or more requirements in a specification contradict each other" [123]. Often, consistency is not only applied to requirements but also to other artifacts such as design models. According to an overview presented by Kamalrudin and Sidek [79], the predominant approaches to check consistency can be categorized in traceability, heuristic, and formal analysis. Traceability supports consistency by identifying existing and missing links between development artifacts [79]. This may be supported by a formal analysis of the artifacts, as presented by Cârlan et al. [21] or Egyed [47]. Heuristic analysis is mostly applied to semi-formal specifications. This can be done, e.g., by checking pre-defined consistency rules [6, 47]. The Vitruvius approach [53, 105] builds on the paradigm of view-based development in order to maintain consistency. It ensures maintainability and scalability by using a so-called virtual single underlying models (V-SUM) which combines different meta-models by consistency preserving rules. Šenkýr and Kroha [112] use natural language processing to check consistency between UML models and their textual requirements. Wan et al. [118] use natural language processing and a domain ontology to check consistency of formalized traffic rules with their natural language counterparts.

In this thesis, a formal approach to consistency analysis is followed. Early formal approaches date back to the '90s [71, 77]. Here, consistency has been defined for automata-like specification languages. A specification is consistent in terms of [71, 77], if, in any state, there is at most one enabled transition.

More recent definitions of consistency can be found in [7, 10, 17, 43, 48, 56, 82]. Here, consistency is reduced to satisfiability of a temporal locics formula (the conjunction of

all the requirements in a specification) [7, 17, 43, 56, 82] or equivalently existence of an accepting run for the cross-product of automata [10, 48]. Some approaches [10, 48, 56] take dedicated means to exclude trivial solutions that cannot be used as an argument for the conflict-freeness of requirements. Among these, *partial consistency* [10] is of special interest for this thesis. Although partial consistency has been developed for the simplified universal pattern [16], the core idea is expected to be applicable for other specification languages with a similar structure of triggers and actions. The trigger is some enabling condition of a requirement (similar to the pre-chart of a TSC) and the action defines the behavior that shall occur in response. The basic idea of partial consistency is to

1. identify critical combinations (including timing) of the triggers of different requirements
2. check if they are possible in practice, and
3. if they are, check if the whole set of requirements has at least one satisfying run under each trigger combination.

Jéron et al. [78] generalize partial consistency to deterministic timed automata. A first attempt to adapt partial consistency to TSCs is made by the thesis author himself in earlier work [11]. In Chapter 5 of this thesis, partial consistency for TSCs is reworked in order to improve scalability and to align with the rest of the developed framework.

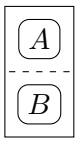
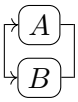
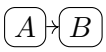
3.2 Scenario-driven Development

Ulbrich et al. [115] define a scenario as a sequence of scenes, and a description of the evolution between them. A scene is a "snapshot of the environment including the scenery and dynamic elements" [115]. Menzel et al. [90] identified three abstraction levels for scenarios: functional, logical, and concrete scenarios. Functional scenarios are natural language descriptions of scenarios at a high level of abstraction, while concrete scenarios describe a specific sequence of state values that results in a reproducible sequence of scenes. Logical scenarios allow for state parameter ranges, and are at an intermediate abstraction level between functional and concrete scenarios. Later on, abstract scenarios [94] have been added to this hierarchy. Abstract scenarios can be seen as formal language descriptions of functional scenarios.

A wide range of different modeling languages for traffic scenarios has been developed. For example, Bach et al. [6] present a scenario language that comes with a graphical modeling environment, while Zhang et al. [122] propose a textual scripting language. An overview of the different approaches is presented by Ma et al. [88]. At time writing this thesis, the most prominent scenario language is OpenSCENARIO XML[1] [104]. It emerged to a widely accepted standard and is supported by many industrial and academic simulators such as ESMINI, Vires VTD, and CARLA. While OpenSCENARIO XML addresses concrete and logical scenarios, OpenSCENARIO DSL [104] adds support for abstract scenarios. Other languages for abstract scenarios include Zone graphs [20], Scenic [64], and TSCs. These languages follow complementary concepts. Zone graphs provide finite state abstractions of abstract scenarios. The static part of a scene (e.g., a crossing with traffic lights) is modeled as a graph of map regions, called zones. This leads to a discretization of traffic situations

[1] Formerly known as OpenSCENARIO 1.x

Table 3.1: Comparison of basic charts and Duration Calculus

Basic Chart	MSRTL	Duration Calculus	
	$\varphi_A[b_A\backslash b, e_A\backslash e] \wedge \varphi_B[b_B\backslash b, e_B\backslash e]$	$A \wedge B$	(conjunction)
	$\varphi_A[b_A\backslash b, e_A\backslash e] \vee \varphi_B[b_B\backslash b, e_B\backslash e]$	$A \vee B$	(disjunction)
	$\neg\varphi_A[b_A\backslash b, e_A\backslash e]$	$\neg A$	(negation)
	$\exists m : \varphi_A[b_A\backslash b, e_A\backslash m] \wedge \varphi_B[b_B\backslash m, e_B\backslash e]$	$A \frown B$	(chop)
	$\varphi_A[b_A\backslash b, e_A\backslash e] \wedge e - b \sim X$	$A \wedge \ell \sim X$	(duration)
	$b < e \wedge \Box_{[b,e)}\phi_a$	$[a]$	(state)

that allows to cluster the decision space of an HAV into a finite set of equivalence classes. Schwammberger [111] follows a similar discretization approach for modeling traffic scenarios at urban intersections. Scenic and OpenSCENARIO DSL are both textual languages that script scenarios based on pre-defined libraries of driving maneuvers (e.g. lane changes, following). Fremont et al. [64] present an execution engine for Scenic that uses guided simulation and supports different simulators.

3.3 Duration Calculus

Duration Calculus [23] is a well-studied formalism for real-time systems. It is an interval logic for dense time and has many similarities to TSCs: Conjunction, disjunction, negation, and sequences of states are the core elements of Duration Calculus. Furthermore, it is possible to specify the duration of states. In DC, a state predicate (as in state expressions $[s]$) is a propositional formula over some set P of (propositional) state variables. DC formulae are evaluated on bounded time intervals $[b, e]$ on traces over P. As a notable difference to MSRTL semantics, DC state expressions $[s]$ are evaluated with the help of a Riemann integral[2]. This makes a distinction between open and closed time intervals unnecessary and avoids problems such as seen in Example 2 in Section 2.4. Table 3.1 compares basic charts/MSRT formulae to equivalent DC formulae. DC assumes *finite variability* of traces, i.e., for every trace π, state variable $p \in P$, and time points $b < e$, there exists a finite sequence $t_0 = b < t_1 < \cdots < t_n = e$ of time points such that, for all $i = 1, 2, \ldots n$, either $\pi \models \Box_{[t_{i-1},t_i)}p$ or $\pi \models \Box_{[t_{i-1},t_i)}\neg p$. Under this assumption, the equivalences between DC and charts in Table 3.1 hold (i.e., the MSRTL formula is satisfied if, and only if, the DC formula holds on the interval $[b, e]$).

[2]The semantics of $[s]$ on a trace π and interval $[b, e]$ is $\int_b^e \mathbb{1}(\pi^t \models s)\mathrm{d}t = e - b > 0$ over the indicator function of satisfaction of s at time t on π.

A DC formula is called *valid* if it holds on every non-empty interval on every trace. So, validity of DC-formulae corresponds to unsatisfiability of negated equivalent MSRTL formulae. The subset of DC that consists only of the operators used in Table 3.1 is commonly referred to as the $[s], \ell = X$ subset[3] of DC [22]. It has been shown that validity of DC formulae of this subset is undecidable in general [22], but decidable for the following smaller subsets:

1. If DC is used on an integer time domain [22]
2. If DC formulae do not use duration constraints $\ell \sim X$ [22]
3. If the DC formula has a single outermost negation [57, 60]

In cases 1 and 2, DC formulae can be translated to equivalent finite automata that have an empty language if, and only if, the DC formula is valid [22]. Case 3 has first been shown by Fränzle [57] for a fragment of DC[4]. In a later work [60], Fränzle and Hansen show how positive DC formula (i.e., without negations[5], but with general duration constraints) can be translated to equisatisfiable LinSat[6] constraints. The resulting constraint is satisfiable if, and only if, the DC formula has a bounded model (i.e., is satisfied on some trace and interval) [60]. Despite the small differences between the formal semantics of DC state expressions $[s]$ and TSC invariant nodes, using the equivalences from Table 3.1, the undecidability proofs for Duration Calculus apply to TSCs as well.

A practical approach for bounded model checking negation-free dense-time duration calculus is explained in [113]. In combination with [57, 60], also a minimum unrolling depth can be derived that guarantees completeness of the results. The approach reduces satisfiability of a Duration Calculus formula to LinSat and is easy to implement on top of recent SMT solvers, but results in a polynomial ($\mathcal{O}(n^4)$) blowup of the formula. Another approach is presented in [65] which translates a similar subset of integer-time DC into symbolic automata which can also be addressed with bounded model checking. This work inspired the encoding technique for basic charts presented in Section 6.3. Various other approaches (e.g. [19, 68, 91]) address model checking of hybrid systems against different variants of DC, but do not seem to be appropriate for consistency analysis of TSCs.

3.4 Other Approaches to Hybrid Systems Verification

Another related field to the consistency analysis of TSCs is verification of hybrid systems in general. This field encompasses research from the past three decades (first popular works are probably by Alur and Dill, e.g. [3]) and it is not possible to cover this research completely here. A survey can be found for example in [44]. Given a formal description of a hybrid system, e.g., in the form of a hybrid automaton [72], verification techniques typically aim to prove or disprove whether the system fulfills some safety property. This substantially differs from consistency analysis, which asks whether a fulfilling hybrid system exists. Nevertheless, some of the existing solutions could serve as a basis for alternative

[3] I.e., it does not allow generalized duration constraints of the form $\sum_i a_i \int s_i \sim X$. Duration inequalities $\ell \geq X$ can be expressed in this subset as $\ell = X \frown true$.

[4] where state durations only appear in the form $[s]$ or $\ell \sim X$

[5] Including the aforementioned outermost negation. However, negations are still allowed inside state predicates.

[6] satisfiability of boolean combinations of linear inequalities

approaches or further improvements to the consistency analysis of TSCs. The following briefly summarizes some interesting work.

3.4.1 Hybrid Games

As an alternative to the different forms of existential consistency proposed in Section 3.1, consistency can be seen from a game-theoretic perspective as a two-player game between the SUD and the environment: A specification is consistent, if there exists a strategy for the SUD to satisfy all requirements, or alternatively, if there exists no strategy for the environment to violate a requirement. A similar consistency notion has been developed in [1]. For example, Henzinger et al. [74] as well as Quesel and Platzer [103] investigate into solving hybrid games.

3.4.2 Hybrid Automata

In the formal definition of TSCs [37, 38], it is assumed that the world model defines a system of hybrid automata [72]. It is expected that at least for a fragment of TSCs a translation to a hybrid automaton is possible. Bouajjani et al. [19] present a translation from a dialect of Duration Calculus into hybrid automata. A wide number of model checking tools for hybrid automata exist. Among the most popular ones are HyTech [73], PHAVer [62], its successor SpaceEx [63], the work by Eggers et al. [46], and by Chen and Sankaranarayanan [25]. Of special interest for this thesis may be [2], because the research focuses on hybrid systems with large discrete state spaces (which is probably the case for simulations of multiple TSCs and objects in the world model). Unfortunately, [2] is limited to linear hybrid automata.

Another interesting approach used in the (Timed)HyDice tools presented by Plaku et al. [96, 97]. The authors combine finite state LTL model checking with simulation techniques for hybrid systems, being able to produce precise witness trajectories for LTL property violation. The approach requires a sound discrete state approximation of the system state space, which must be provided by the user of the method. In brief, the LTL formula and user-provided abstraction are used to build up a search tree that in turn guides the simulation. The authors evaluate their work on a scalable vehicle motion planning problem and claim good scalability. However, performance and correctness clearly depends on the quality of the user-provided discrete approximation.

3.4.3 Robust Interpretations

Robust interpretations of (hybrid) systems and temporal logics investigate satisfaction and/or reachability under (infinitesimal) small perturbations in trajectories and constraints. They rely on providing both upper and lower discrete approximations of the reachable state space, which, for robust systems, converge for fine discretizations. This makes satisfiability/reachability questions decidable in the end. For example, Damm, Pinto and Ratschan [41] present a decision procedure for robust satisfaction of LTL properties on discrete-time hybrid systems, which uses a successive discrete approximation of the state space. Similarly, Fränzle and Hansen [59] define a robust interpretation of duration calculus. The authors present a time-wise discretization of the DC formula, which, by iterative refinement, enables to semi-decide validity.

3.4.4 Small Model Theorems

Another problem in determining the satisfiability of TSCs is the large number of objects in the world that must be considered in the analysis. In theory, one must consider an unlimited number of objects, such as vehicles and obstacles. Clearly, this renders the satisfiability of a spatial view undecidable in general. In any case, considering more objects increases the size of a system state. However, in practice, it is often possible to limit the number of objects while preserving the soundness and completeness of the satisfiability results. Damm et al. [35] present a small model theorem for families of linear hybrid automata, where each automaton communicates with a bounded number of neighbors. Fränzle et al. [61] extend multi-lane spatial logic (MLSL) by so-called scopes which limit the range of quantified variables to a finite set of objects. The authors show that validity of MLSL formulae with scopes is decidable. Furthermore, they show how this can be used to derive a semi-decision procedure for a subset of MLSL without scopes.

The consistency checking approach presented in this thesis abstain from applying small model theorems á la Fränzle et al. [61] or Damm et al. [35], because they may result in a quite large number of potential neighbors and suffer from combinatorial explosion. Instead, an abstraction is encoded that asks for the possibility of neighbor existence. However, in future work, small model theorems may be used to derive adequate numbers of assumed neighbors and thereby improve completeness.

3.5 Trajectory Planning and Generation

Describing vehicle trajectories with Bézier splines, as presented in Chapter 7, is not a new idea. Bézier splines [99] describe parametric curves, such as a vehicle's trajectory, uniquely by a finite set of control points. They are commonly applied in trajectory planning and optimization (e.g., [26, 67, 102]). In contrast to the approach followed in Chapter 7, where a Bézier curve describes both the path and the speed profile, related work uses Bézier curves for path planning only, and determines speed profiles separately (apparently this is easier if distinct speed and steering controllers are used).

Other related work [45, 81, 107] utilizes constraint solving/optimization to generate trajectories from abstract scenarios. Klischat and Althoff [81] use mixed-integer quadratic optimization (MIQP) to find trajectories in a two-phase approach. Traffic scenarios are described as propositional constraints that assign vehicles to so-called lanelets[7]. These predicates are then converted to MIQP constraints that are optimized subject to vehicle dynamics and a cost function. The second phase then calculates concrete trajectories. Esterle et al. [49, 52, 80] describe an encoding of vehicle dynamics into linear constraints which they also use as part of an MIQP problem for maneuver planning. Similar to the approach described in Chapter 7 of this thesis, they use a discretization of the vehicle heading into intervals in order to encode the vehicle dynamics into linear constraints. Compared to the Bézier spline encoding used in this thesis, Esterle et al. evaluate planning constraints (such as collision avoidance) only at discrete time points and therefore require a finer step size and discretization.

Eggers et al. [45, 107] use SMT solving for trajectory generation. The used scenario specifications are quite similar to TSCs, but support only a reduced set of temporal (i.e.

[7] A lanelet is a segment of a driving lane, allowing to separate complex road layouts and crossings into a set of regions.

sequences only) and spatial operators (relative distances and speed). The authors use a custom solver that uses interval propagation techniques. None of [45, 81, 107] uses Bézier curves to describe trajectories.

Plaku and Karaman [95] give a survey on motion planning under temporal (i.e., LTL) constraints.

4 Consistency in the Development Process

This chapter goes into more detail about the purpose of the consistency analysis for TSCs. Section 4.1 gives an overview of the development process for highly automated vehicles, and locates the consistency analysis within this process. Section 4.2 then takes a closer look at automated driving functions itself, and which part of their specification is targeted by the methods being developed in this thesis. This leads to a set of goals for the development of the TSC consistency analysis which are collected in Section 4.3.

4.1 Development Process and Scenarios

Development of safety-critical cyber-physical systems is traditionally organized in a V-process [24]. The V-process dates back to the '90s [24] and has been adopted by the safety standards in different transportation domains such as ISO 26262 for the automotive domain, ARP 4754 for aerospace, EN 50128 for railway systems, and ISO 17894 for the maritime domain. The process model is named after the shape of the upper-case letter V. The process starts at the top left corner of the V, sliding down the left leg, and climbing up the right leg. The left leg describes the system design and development phases which are carried out top down. Typically, the process begins with system definition and a risk assessment, followed by a hierarchical decomposition into (sub-)components. The purpose of the risk assessment is to identify potential risks emerging from malfunctions of the product. This is the basis for deriving appropriate safety requirements and assigning safety integrity levels, which in turn define acceptable maximum failure rates and implementable safety measures. The software implementation marks the point of the V. On the right leg of the V, the developed parts are integrated bottom-up into the overall system, with an emphasis on validation and verification. This typically begins with unit testing, followed by integration testing, and finally acceptance testing.

The above-mentioned safety standards ISO 26262, EN 50128, and ISO 17894 target so-called *functional safety*. Functional safety means to ensure that the system always operates correctly with respect to its specification. This is done by systematically identifying possible causes of malfunctions, rating their impact on overall safety, and defining adequate mitigation strategies that are to be implemented as part of the development process. Hereby, functional safety focuses on malfunction that is caused by random hardware faults. These are physical factors that may cause malfunctions of computation hardware or electro-mechanical components. Thereby it is assumed that a correct implementation of the system specification on fault-free hardware will result in a safe system. Opposed to that, *Safety of the intended functionality* (SOTIF) deals with safety risks that are caused by limitations of the correctly functioning system. These are challenges for the development of HAV that come on top of functional safety. Therefore, SOTIF is addressed by a dedicated normative standard ISO 21448. SOTIF considers aspects such as:

- Sufficiency of the specified sensors and algorithms in order to identify traffic situations correctly.

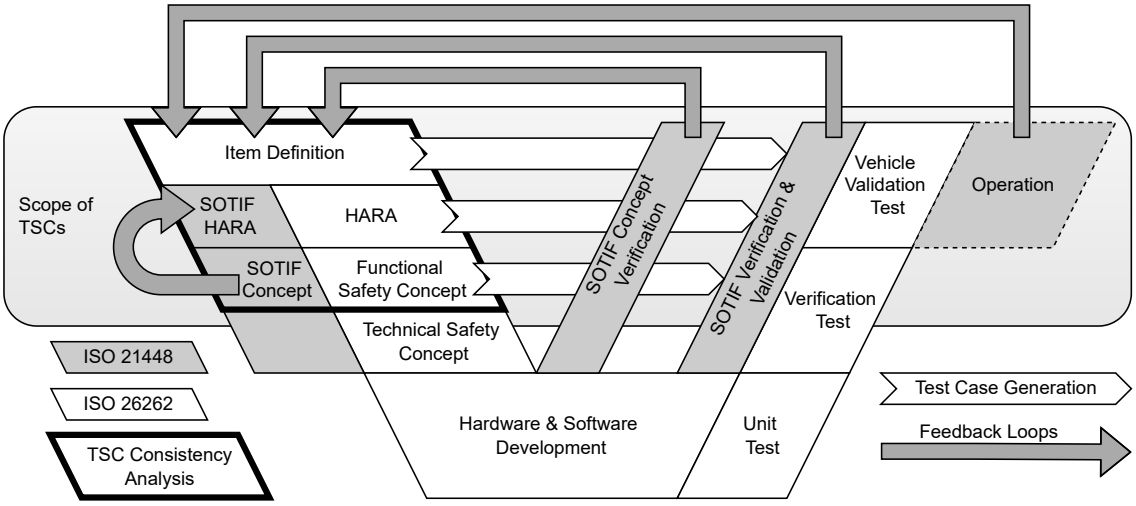


Figure 4.1: The ISO 26262 and ISO 21448 V-processes overlaid (based on a figure from [89]).

- Completeness of the system specification with respect to all traffic situations and conditions that the system may be confronted with.

The ISO 21448 standard proposes the use of scenario-based methods. Especially, it clusters the overall scenario space to be considered for SOTIF into a 2×2 matrix considering hazardous versus non-hazardous scenarios on one axis, and known versus unknown scenarios on the other axis. Hence, an important element of SOTIF is the identification of criticalities, with a special emphasis on completeness. One approach is to systematically derive critical scenarios from the (a-priori known) operational scenarios and known driving maneuvers [86, 93, 101]. Furthermore, automotive industry tries to build up scenario catalogs from expert knowledge and databases of recorded data [5, 100]. Due to HAVs having to operate in an open world, it is expected that field experiences need to be fed back into the development process [34]. Extending the V-process with this feedback loop from the operations phase results in a systems engineering dev-ops process [29, 85].

Menzel et al. [90] identify applications of driving scenarios in the ISO 26262 V-process. In the concept phase, which marks the start of the automotive V-process, the operational scenarios of the HAV are defined. This phase also contains the hazard analysis and risk assessment (HARA), which forms the basis for defining safety requirements. Later on in the development process, functional and technical requirements are derived that implement these safety requirements. At the right side of the V, scenarios are used to define test cases for safety validation, unit, and integration testing. The test scenarios will be based on the operational scenarios, identified critical scenarios, and requirements [90, 94].

Figure 4.1 overlays the different phases of the ISO 26262 and ISO 21448 reference processes, based on an overview by Mariani [89]. Scenario refinement cycles due to feedback loops in the process are also highlighted in the figure. TSCs are best suited for describing requirements at the concept level, where the behavior of the HAV in different traffic situations is specified. The outcome of HARA activities leads to both a functional safety and a SOTIF concept, which extend the initial specification. During the Verification & Validation phases, as well as the operations phase, new information is gathered. This leads to an iterative refinement of requirements and TSCs in the development process.

ISO 26262 [76] states that consistency, beneath completeness, is a mandatory property of

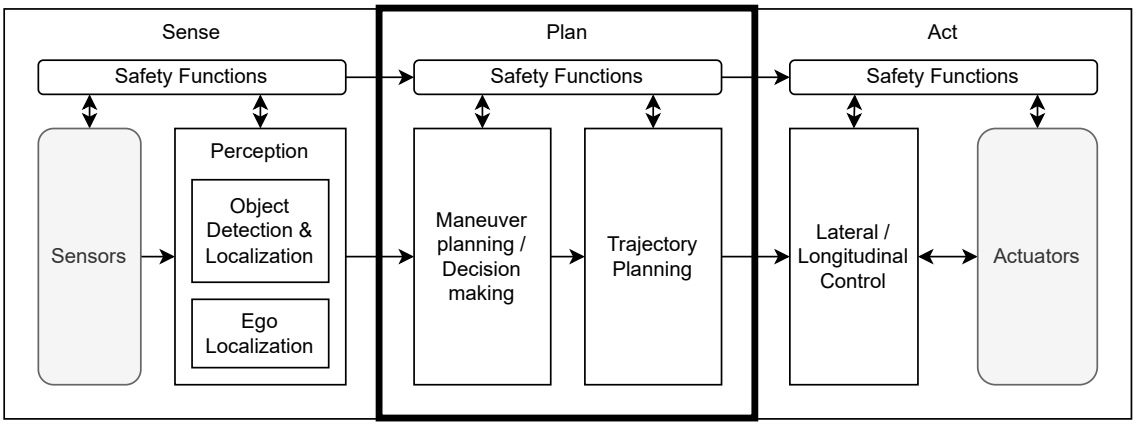


Figure 4.2: The sense-plan-act design pattern for HAVs. TSCs are best suited for specification of the *Plan* part.

any requirements specification. According to this standard, consistency means that a set of requirements is free of contradictions. This includes that each requirement neither contains a contradiction within itself, nor contradicts other requirements in the same specification, or related documents (e.g., other normative standards) [76]. Detecting inconsistencies early in the development process prevents implementation faults and saves time and money [54]. Therefore, an automated consistency analysis is beneficial in all phases in which scenario-based requirements specification takes place, including the item definition, SOTIF concept definition, and functional safety concept definition.

Note, that this thesis focuses on consistency between scenario-based requirements only. Some related work, as outlined in Section 3.1, has a more holistic view on the consistency problem. It considers consistency not only between requirements, but between different kinds of artifacts – such as textual requirements, models, or test cases – in the development process. This leads to multi-dimensional consistency frameworks such as the one described by Reussner et al. [105] and Feichtinger et al. [53]. The integration of the TSC consistency analysis in such a framework is left to future work.

4.2 Worldmodel Aspects and Refinement

As explained in Section 2.3, TSCs are always used together with a world model. Damm et al. [33] see world models as networks of communicating hybrid automata, each belonging to some automaton class. Hence, world models define both the different types of entities that are interacting in a scenario, as well as their possible behavior. The former is part of the domain ontology which defines the concepts (including their relations) that are used in the requirements specification.

To determine which aspects of a world model are important for the consistency analysis, we take a closer look at the requirements specifications to be analyzed. In the previous section, the SOTIF and functional safety concepts have already been identified as the targeted abstraction level of the HAV, respectively the driving function under design.

Typically, the design of an automated driving function follows the sense-plan-act pattern depicted in Figure 4.2. The *Sense* part covers sensors such as cameras, LiDAR, radar, telemetry, and GPS which capture the current driving situation and locate the HAV within the scene. The *Plan* part contains the logic that chooses the next driving maneuver and

calculates a trajectory for the HAV. The latter is finally executed in the control loop of the *Act* part. As part of the SOTIF and safety concept, all parts include safety functions which monitor correct functioning of the components and, in case of failures, maintain a safe state. The *Plan* part is the one in focus of the TSC consistency analysis: In this part, traffic rules and other behavioral requirements are evaluated that work on information contained in a scene.

In the following, we examine the expected content of a domain ontology. This gives insight on required capabilities of the consistency analysis when it comes to world models.

Koopman [83] gives a long – but still not exhaustive – list of different aspects that need to be considered for a complete safety argumentation. Here, four "axes"[1] of the validation space are identified. Koopman formulates them from a safety argumentation perspective. Because a safety argumentation also requires a completeness argument, the axes give an idea about the different aspects that we can expect to be addressed by a world model. The four axes are:

- *Operational Design Domain (ODD)* which describes aspects of the environment (road types, terrain, infrastructure) and environmental conditions under that the system under development shall operate.
- *Object and Event Detection and Response (OEDR)* which includes types of traffic participants, temporary obstacles, expected behavior of the environment, and the perception thereof.
- *Faults* including imprecise or wrong perception, hardware failures (e.g., loss of sensors or actuators, including degraded modes), capacity variation (e.g., vehicle being overload), and wrong or missing external information.
- *Maneuvers* of the system under design.

A majority of concepts originating from the ODD and OEDR axes are already quite well covered in automotive domain ontologies such as OpenXOntology [4], A.U.T.O [119], or the Operational World Model Ontology [30, 31]. A good overview about the high-level structure gives the popular 6-layer model [109] that structures urban traffic ontologies into road network, roadside structures, temporal modifications thereof, dynamic objects (e.g. vehicles), environmental conditions, and digital information (e.g. vehicle-to-vehicle communication messages).

When using TSCs, the *maneuvers* are usually not described by the world model, but by the set of TSCs that describes the HAV and its ODD (cf. [39]). The world model has to cover the features of the vehicles (e.g., turning indicator state, movement direction, velocity) that are needed to describe them. While the world model may make assumptions about possible vehicle behavior (e.g, assuming a single driving direction on highways), it still must allow all possible maneuvers within the ODD.

Besides the HAV under development, the TSC instantiates roads (with a specific type), nearby traffic participants, and obstacles. The underlying participant types, road types, and associated parameter ranges (e.g., road width and allowed speed) are defined within the world model. Environmental conditions and known failure modes may be specified both in the TSC (e.g., in form of observed abnormal behavior) or in the world model (e.g.,

[1]The term *axes* that is used in [83] may be misleading in the sense that it suggests existence of a natural ordering along each dimension, which is not the case.

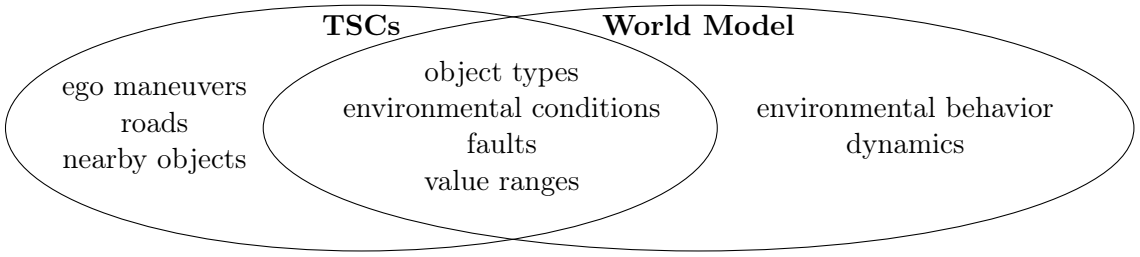


Figure 4.3: Validation space dimensions in TSCs and world model (examples)

in form of a state variable). In general, a TSC can restrict all the value ranges (e.g speed, braking force) specified by the world model, and may restrict object variables to be bound to objects of a specific type, so these dimensions are part of both the TSCs and the world model. All these aspects may be specific to the system under test and its design domain.

However, the purpose of a consistency analysis is to identify conflicts in the requirements specification. It is not a safety validation of the system under design. Hence, some of the aspects described by Koopman are outside the scope of a consistency analysis. In the context of this thesis, the following are explicitly not considered for consistency analysis.

Failure modes. Care must be exercised when modeling failure modes. A safety mechanism can deduce the presence of failures only from observed behaviors. This also applies to perception faults. TSCs, in their current form, do not allow to distinguish between the real situation and the perceived situation. In general, perfect knowledge is assumed. Therefore, perception faults are excluded from this thesis. Also, it cannot be expected that the consistency analysis "knows" and examines malfunction behavior. This is subject to a dedicated risk analysis and out of scope for this thesis. Instead, we assume that malfunction behavior is explicitly modeled, e.g., in form of a requirement TSC that describes what the system shall do if the HAV exceeds the speed limit for a certain period of time. However, it is totally fine to model detected faults as status variables in the world model. For example, the system under design may have a status variable indicating detected vehicle overload. A requirement TSC then may specify that safety distances must be increased.

Adverse conditions. Similar to failure models, it is not part of the consistency analysis to prove that the required maneuvers are feasible in presence of, e.g., aggressive drivers. The same holds for environmental effects. The world model can, and certainly will, contain information about weather conditions and road type, but again, they refer to information present to the (ego) vehicle. However, it is outside scope of a consistency analysis to evaluate the effect of these environmental conditions on the system under design.

4.3 Requirements for a TSC Consistency Analysis

The characterization of the application area of a TSC consistency analysis given in the previous sections leads to the following requirements. They form the basis for the development of a TSC consistency analysis and therefore are entitled *thesis goals*.

Approaches to automated consistency analysis, as described in the literature, aim to identify conflicting behavior specifications in discrete state systems, such as LTL (e.g. [56, 98]) or (timed) automata (e.g. [48, 71]). For this kind of systems, sound and complete

finite state space abstractions are well known. This enables analysis methods to give a definite answer whether a specification is consistent or not. TSCs, in contrast, describe hybrid systems, for which reachable states are much harder to compute. They require precise knowledge about the dynamics of the system and its environment. Therefore, in order to provide a tool that is both practicable and reliable, this thesis aims at a provably correct semi-decision procedure for consistency. Section 4.1 describes the assumed scenario-based development process in which the consistency analysis shall be applied. Due to the iterative structure coming from feedback loops, the requirements specifications are initially incomplete and successively refined. Furthermore, it has to be assumed that the domain knowledge captured in the world model is incomplete as well, i.e., abstracting from physical details. Therefore, the consistency analysis shall focus on conflicts that can be formally proven with available world model knowledge.

Thesis Goal 1. *The consistency analysis shall produce only valid inconsistency results, and the correctness of the identified inconsistencies shall be prioritized over completeness.*

Considering the feedback loops depicted in Figure 4.1 and the iterative nature of the development process in general, it is worth exploiting knowledge from earlier iterations for the consistency analysis of an updated specification.

Thesis Goal 2. *It shall be possible to reuse consistency analysis results about a subset of requirements to evaluate a refined set of requirements.*

In order to be able to fix found inconsistencies, an engineer must be able to trace the cause of an inconsistency. A prominent tool here is the automatic calculation of minimal inconsistent sets.

Thesis Goal 3. *The consistency analysis shall be able to point out the source of an identified inconsistency, e.g., by producing minimal inconsistent subsets.*

Furthermore, the requirements specification forms the basis for test case generation. Test cases are typically specified using concrete scenarios, which encompass vehicle trajectories. The following goal highlights this connection.

Thesis Goal 4. *The consistency analysis shall be able to produce trajectories as witnesses.*

Section 4.2 has a closer look at the world models that form the basis for the TSC specifications. It turns out that the used world models are highly dependent on the HAV's ODD and will evolve during the development process. Therefore, it does not make sense to limit the consistency analysis to a single world model in advance.

Thesis Goal 5. *The consistency analysis shall support user-defined world models containing custom entity types and attributes.*

A central concept in road traffic ontologies are vehicles. Due to our Goal 4 above and the focus of TSCs on spatio-temporal relations including distance constraints, a simple model of vehicle dynamics is required at least. This dynamics model shall be capable of describing the driving maneuvers that are to be expected in a HAV's ODD. The NHTSA identified a list of driving maneuvers related to HAVs [114]. Similarly, OpenXOntology defines a set of basic maneuvers such as lane changes, turns, or standstill.

Thesis Goal 6. *The consistency analysis shall use a simple vehicle dynamics model that is capable of generating sound trajectories for abstract scenarios involving:*

- *acceleration from/to standstill*
- *lane change*
- *right/left turns*
- *velocity, acceleration, and distance constraints*

Achievement of these goals is evaluated later on in Section 8.5.

5 Consistency Notions for TSCs

Before we can decide whether a set of requirements is consistent, we have to define what consistency means. As mentioned previously, different consistency notions have been proposed in the literature. In this chapter, consistency notions dedicated to requirement TSCs are developed. The first one – *existential consistency* – wraps up a popular idea from related work [10, 48, 56] that works well for single requirements. However, when adopted for sets of TSCs, applying it for a set does not add value compared to checking each TSC individually. Therefore, the second notion – *strong existential consistency* – strengthens the first one for sets of TSCs. Because this one, however, is intractable by SMT solving, *partial consistency* is developed as a trade-off between the former two notions.

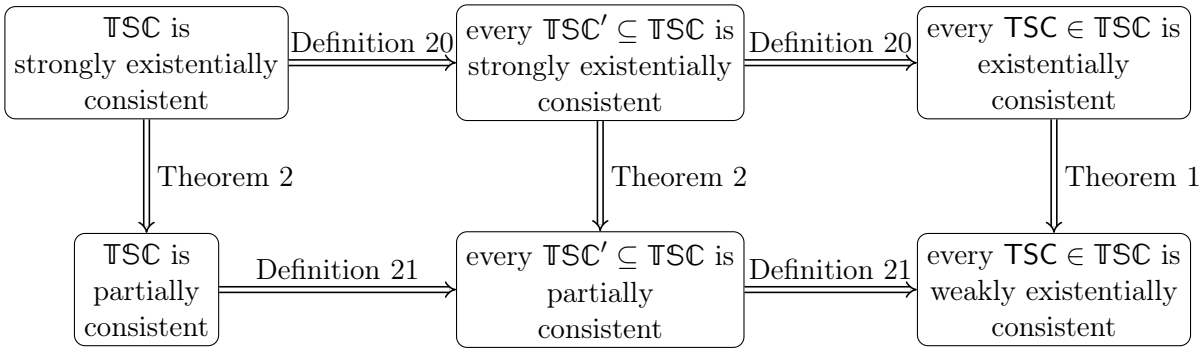


Figure 5.1: Relations between different consistency notions. The arrows indicate implications between the statements.

Figure 5.1 shows how the different consistency notions are related. The horizontal implications directly follow from the definitions of strong existential consistency (Definition 20) and partial consistency (Definition 21) while the vertical implications are shown in the following sections in Theorems 1 and 2.

5.1 Existential Consistency

This thesis proposes a notion for consistency of TSCs based on previous work for pattern-based requirements [10, 48, 56] due to the following reasons:

- The considered pattern languages define the semantics of requirements in terms of unbounded system runs.
- They allow "don't care"s in the requirements.
- The considered requirements are structured in a premise/consequence scheme.

In this context, "don't care" means that a requirement only restricts certain system variables, allowing for a high degree of freedom. For example, a TSC usually describes the behavior of traffic participants in terms of distance intervals. Within the interval, any behavior is

valid. Recall that requirement TSCs consist of two parts: the pre-chart (history and future) and the consequence. On any sub-trajectory, where the history is followed by the future, the consequence must hold in parallel to the future. The requirements tackled in related work [10, 48, 56] are structured similar: They specify some consequence (called action for pattern-based requirements [10, 48]) that shall occur in response to some premise (called trigger for pattern-based requirements [10, 48]).

The authors of [48, 56] define consistency as follows: A set of requirements is consistent, if there exists at least one trajectory that satisfies all the requirements in the set. This is called *existential consistency* in [48]. Lifting existential consistency to TSCs means the following: A set of TSCs is existentially consistent, if there exists at least one trajectory that satisfies all the TSCs. In the following, we restrict ourselves to a set consisting of a single TSC and consider the case of multiple TSCs later. By Definition 9, a trajectory satisfies a TSC if, for any time points $b < m < e$, such that the history holds from b to m and the future holds from m to e, the consequence holds from t to e. This is problematic for two reasons: First, the property is difficult to check, because there are typically infinitely many triples (b, t, e) on a trajectory where history and future hold. Second, we may choose a trajectory for witness, where history and future are never satisfied at all, in which case the consistency argument is not of practical use. The authors of [48, 56] solve the second issue by requiring that the premise of the requirements be satisfied at least once on the witness trajectory. When speaking about TSCs, this means that we search for a trajectory $\pi \in \mathcal{T}(\mathsf{WM})$, valuation μ, and time points $0 < b < m < e$ such that

$$\pi, \mu \models \varphi_H[b_H \backslash b, e_H \backslash m] \wedge \varphi_F[b_F \backslash m, e_F \backslash e] \tag{5.1}$$

where H and F are the history and the future of the TSC, respectively. We call a TSC *existentially consistent* if the trajectory in addition satisfies the whole TSC.

Definition 18 (Existential Consistency)**.** *A TSC* TSC *with history* H *and future* F *is* existentially consistent *in a world model* WM *if there exists a trajectory* $\pi \in \mathcal{T}(\mathsf{WM})$ *such that both* $\pi \models_{\mathsf{WM}} \mathsf{TSC}$ *and* $\pi \models_{\mathsf{WM}} \mathsf{HF}_{\mathsf{TSC}}$ *hold, where the existential TSC* $\mathsf{HF}_{\mathsf{TSC}}$ *is defied as*

$$\mathsf{HF}_{\mathsf{TSC}} = \boxed{} \rightarrow (H) \rightarrow (F)$$

and has the same bulletin board as TSC*.*

5.2 Weak Existential Consistency

In the following, we weaken existential consistency defined above and derive a consistency notion that we can easier address with SMT solving. Weak existential consistency avoids to search for a satisfying trajectory for the requirement TSC and instead reduces consistency to satisfiability of an existential TSC. As a consistency argument, we would like to see that it is at least possible to satisfy the consequence C of the TSC in parallel to the future. This brings us to our second consistency notion for TSCs which we call *weak existential consistency for TSCs*.

Definition 19 (Weak Existential Consistency)**.** *A TSC with history* H*, future* F*, and consequence* C *is* weakly existentially consistent *wrt. the world model* WM *if the existential*

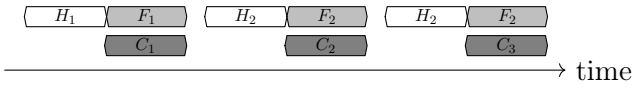


Figure 5.2: Timing diagram for naive existential consistency check of three TSCs

TSC

$$\mathsf{HFC}_{\mathsf{TSC}} = \quad H \quad \begin{matrix} F \\ \hline C \end{matrix} \ ,$$

with the same bulletin board as the original TSC, is satisfiable in WM.

Fact 1 (Existential approximation)**.** *Assume some TSC* TSC *with history H, future F and consequence C that is interpreted over some world model* WM*. A trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ that satisfies both $\pi \models_{\mathsf{WM}} \mathsf{TSC}$ and $\pi \models_{\mathsf{WM}} \mathsf{HF}_{\mathsf{TSC}}$, also satisfies $\pi \models_{\mathsf{WM}} \mathsf{HFC}_{\mathsf{TSC}}$.*

This fact can be derived from Definitions 5 to 9 using some basic algebra. This fact implies that weak existential consistency does not produce false warnings. In other words, if a TSC is weakly existentially inconsistent (i.e., $\mathsf{HFC}_{\mathsf{TSC}}$ is unsatisfiable within the world model), then either the pre-chart can never occur, or its occurrence directly violates the TSC's consequence. Note that the opposite is not true. A trajectory that satisfies $\mathsf{HFC}_{\mathsf{TSC}}$ does not necessarily satisfy the TSC. However, using this under-approximation of a satisfying trajectory we get rid of the all-quantifier in the formal definition of satisfaction of a trajectory by a TSC. Also, we can restrict ourselves to search for a prefix of a trajectory, instead of a complete trajectory, which would require some kind of unbounded model checking techniques in a consistency analysis.

Theorem 1 (Existential Consistency ⇒ Weak Existential Consistency)**.** *A TSC that is existentially consistent within some world model is also weakly existentially consistent within the same world model.*

Proof. This follows directly from Fact 1. □

5.3 Strong Existential Consistency

The most intuitive way of lifting existential consistency from one to a set of TSCs is probably building a conjunction of the consistency condition (Equation (5.1)) for the individual TSCs in the set, i.e.,

$$\exists \pi \in \mathcal{T}(\mathsf{WM}), \text{valuation } \mu\colon \pi, \mu \models \bigwedge_{\mathsf{TSC} \in \mathbb{TSC}} (\exists b, m, e \bullet 0 < b < m < e \\ \wedge \varphi_H[b_H \backslash b, e_H \backslash m] \wedge \varphi_F[b_F \backslash m, e_F \backslash e] \wedge \varphi_{\mathsf{TSC}}) \ . \quad (5.2)$$

However, this formula is also satisfied by trajectories where pre-charts do not overlap, as illustrated in the timing diagram in Figure 5.2. Especially for encoding techniques (cf. Chapter 6) that approximate dynamics by very weak constraints only, the solver finds a solution for Equation (5.2) above whenever it finds a solution for existential consistency for each of the TSCs. As indicated in Figure 5.2, this solution can be assembled by "gluing" witness trajectories for the individual TSCs together. The following example

shows an inconsistency between two TSCs that is not found with the naive definition of existential consistency from Equation (5.2), but found with the definition of strong existential consistency later on in this section.

Example 4. *Assume a specification with two TSCs stating the following:*

TSC 1 *Whenever the velocity of* `ego` *is below the speed limit,* `ego` *shall accelerate.*

TSC 2 *Whenever* `ego` *is being overtaken by another vehicle,* `ego` *must not accelerate.*

Intuitively, the TSCs are inconsistent, because the require contradictory behavior when `ego` *is being overtaken and at the same time below the speed limit. However, the following scenario describes a trajectory satifying Equation* (5.2)*:*

1. `other` *is on the same lane behind* `ego`*; the velocity of* `ego` *is below the speed limit*
2. `other` *stays behind* `ego` *while* `ego` *accelerates to the speed limit*
3. `other` *is overtaking* `ego` *(exceeding the speed limit) which keeps its velocity at the speed limit.*

The idea that brings us closer to solving this problem is quite simple: Instead of searching for a single, unconstrained—and therefore trivial—witness combining all the TSCs, we should check all possible combinations of the TSCs. We construct these combinations by constraining the time points $b < e$ that mark begin and end of the TSC consequences. A first approach is to check *all* combinations of time points:

$$\forall b_1, e_1, \ldots, b_n, e_n \in \mathbb{R}_{\geq 0} \mid b_1 < e_1, b_2 < e_2, \ldots, b_n < e_n : \\ \exists \pi \in \mathcal{T}(\mathsf{WM}), \text{valuation } \mu \colon \pi, \mu \models \bigwedge_{i=1}^{n} (\mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i) \wedge \varphi_{\mathsf{TSC}_i}) \quad (5.3)$$

where

$$\mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i) :\equiv \exists t : \varphi_{H_i}[b_{H_i} \backslash t, e_{H_i} \backslash b_i] \wedge \varphi_{F_i}[b_{F_i} \backslash b_i, e_{F_i} \backslash e_i]$$

All-quantifying the time points $b_1, e_1, \ldots, b_n, e_n$ ensures that (besides trajectories that combine TSCs sequentially as illustrated above) parallel compositions (i.e., $[b_i, e_i) \cap [b_j, e_j) \neq \emptyset$ for some i, j) are included. However, with the above formula also combinations are checked where not even pre-charts are satisfiable, e.g., because

- the chosen time span $|e_i - b_i|$ of one of the TSCs future is shorter (or longer) than possible,
- two TSCs with overlapping time intervals have mutually exclusive futures, i.e., describe different developments of the situation, or
- two pre-charts cannot occur in the same trajectory, e.g., because they apply to different global environmental conditions (e.g., weather).

Obviously, none of the above indicates a specification fault. and therefore shall not be treated as inconsistencies. Rather, they describe cases in which only a subset of the TSCs apply, or in a different combination. Therefore, *strong existential consistency* defined below excludes these cases from the consistency check.

Definition 20 (Strong existential consistency)**.** *A set* $\mathbb{TSC}$ *of TSCs is* strongly existentially consistent *wrt. a world model* WM *if*

1. *all TSCs in* $\mathbb{TSC}$ *are* existentially consistent*, and*
2. *for every* $k < |\mathbb{TSC}|$*, subset* $\{\mathsf{TSC}_1, \ldots, \mathsf{TSC}_k\} \subseteq \mathbb{TSC}$ *and* $b_1, e_1, \ldots, b_k, e_k \in \mathbb{R}_{\geq 0}$ *with* $b_i < e_i$ *(*$1 \leq i \leq k$*) holds*

$$\begin{gathered}\Big(\exists \pi \in \mathcal{T}(\mathsf{WM}), \textit{valuation } \mu\colon \pi, \mu \models \textstyle\bigwedge_{i=1}^{k} \mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i)\Big) \\ \Downarrow \\ \Big(\exists \pi \in \mathcal{T}(\mathsf{WM}), \textit{valuation } \mu\colon \pi, \mu \models \textstyle\bigwedge_{i=1}^{k} (\mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i) \wedge \varphi_{\mathsf{TSC}_i})\Big) \quad .\end{gathered} \tag{CONS}$$

Definition 20 combines existential consistency with the idea from Equation (5.3) above. Again, the time points $b_1, e_1, \ldots, b_k, e_k$ are all-quantified, which includes parallel combinations of TSCs into the consistency check. The notable difference to Equation (5.3) is the insertion of the left-hand side of the implication (CONS) which filters out combinations with mutually exclusive pre-charts. The term "strong" has been chosen in order to distinguish it from a weaker consistency check, called *partial consistency*, which is discussed in the next section.

5.4 Partial Consistency

The problem with strong existential consistency as defined in the last section is the fact that all possible combinations of time points $b_i < e_i$ (for $i = 1, \ldots, n$) have to be checked. This results in a quantifier nesting. Although current SMT solvers are capable in resolving quantifiers in many cases, and the chart encoding (see Section 6.3.1) results mostly in linear arithmetic where quantifier elimination is possible theoretically, the resulting problems will become too complex to be solved in acceptable time.

The goal of *partial consistency* is to reduce the infinite number of combinations to a finite number of cases, and, at the same time, to avoid alternating quantifiers[1]. Each case $\mathcal{C}$ can be solved as a separate SMT problem and subsumes an infinite number of time point combinations $b_1 < e_1, \ldots, b_n < e_n$. Whenever condition (CONS) holds for all the subsumed combinations, then also $\mathcal{C}$ is satisfied. For deriving the cases, one TSC of each subset (we refer to it with the variable TSC) is selected as the primary TSC. The other TSCs from the set (we refer to them as T) are constrained (by using time pins p and q) such that the futures start before and end after the future of TSC. This is depicted in Figure 5.3. Informally, the other TSCs are placed "around" the primary TSC. Because the duration of a future may not be arbitrary, we cannot decide which TSC to use as a primary TSC in advance. As a consequence, we generate analysis cases with each TSC from the set being the primary TSC once. To filter out impossible combinations, we run each case once without the futures, and withdraw the case unless we know for sure that a trajectory exists with all the pre-charts satisfied (this is analogous to *strong existential consistency*, except that the number of cases is finite).

For each case $\mathcal{C} = (\mathsf{TSC}, T)$, two existential TSCs are derived and checked for satisfiability:

[1]The reader will note in Section 6.4 that quantifier elimination is actually necessary regardless of the consistency notion. However, this only concerns encoding of spatial constraints and can be performed in a pre-processing step. So, the actual consistency check is quantifier-free.

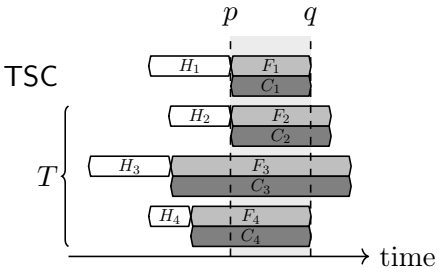
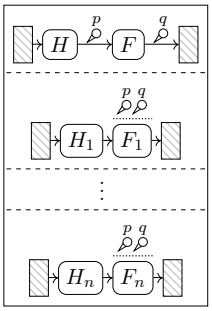
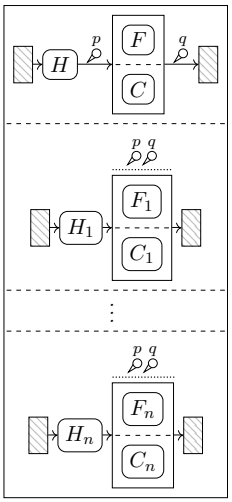


Figure 5.3: Timing diagram for a partial consistency case

- The first one, $\mathsf{BC}_1^{\mathsf{TSC},T}$, approximates the left-hand side of (CONS).
- The second one, $\mathsf{BC}_2^{\mathsf{TSC},T}$, approximates the right-hand side of (CONS),

The first one only contains pre-charts and filters cases in which not all TSCs are applicable. The second one checks for consistency in the remaining cases.

Definition 21 (Partial Consistency)**.** *For some TSC* $\mathsf{TSC} = \langle H, F, C\rangle$*, and a set of TSCs* $T = \{\langle H_1, F_1, C_1\rangle, \ldots, \langle H_n, F_n, C_n\rangle\}$*, construct the existential TSCs* $\mathsf{BC}_1^{\mathsf{TSC},T}$ *and* $\mathsf{BC}_2^{\mathsf{TSC},T}$ *as*

$\mathsf{BC}_1^{\mathsf{TSC},T} :=$  $\mathsf{BC}_2^{\mathsf{TSC},T} :=$ 

A set $\mathbb{TSC}$ *of TSCs is* partially consistent *wrt. a world model* WM *if*

- *all TSCs in* $\mathbb{TSC}$ *are weakly existentially consistent wrt.* WM*, and,*
- *for every* $\mathsf{TSC} \in \mathbb{TSC}$ *and non-empty set* $T \subseteq \mathbb{TSC} \setminus \{\mathsf{TSC}\}$ *holds*

$$SAT_{\mathsf{WM}}(\mathsf{BC}_1^{\mathsf{TSC},T}) \Longrightarrow SAT_{\mathsf{WM}}(\mathsf{BC}_2^{\mathsf{TSC},T}) \ . \tag{PCONS}$$

The combination of n different TSCs into two existential TSCs $\mathsf{BC}_1^{\mathsf{TSC},T}$ and $\mathsf{BC}_2^{\mathsf{TSC},T}$ requires to merge the bulletin boards of the individual TSCs into one. How this merge is performed is left to the implementation of the analysis and does not impact validity of the following theorems (as long the same merge is used for both $\mathsf{BC}_1^{\mathsf{TSC},T}$ and $\mathsf{BC}_2^{\mathsf{TSC},T}$). The prototype implementation developed as part of this thesis merges bulletin boards by unifying entries with same name. As the experimental evaluation (Section 8.3) shows, this rather simple and straight forward approach works sufficiently well if the TSCs follow a

well-defined naming scheme with predefined roles for the different objects. Section 6.2 sketches how partial consistency lifts to a general framework that allows for other means of combining requirement TSCs from a specification. For example, it could be considered that a traffic rule applies to all vehicles in a scenario and thereby may lead to some kind of deadlock situation. However, this requires to add the traffic rule from different perspectives and thereby blow up the number of to-be-checked cases. The current approach – considering a single, canonical, combination of TSCs in each subset – therefore is a trade-off between completeness and complexity.

Theorem 2 (Strong Existential Consistency ⇒ Partial Consistency)**.** *If a set* $\mathbb{TSC}$ *is strongly existentially consistent wrt. a world model* WM*, then it is partially consistent wrt the same world model.*

Proof. Assume that the set $\mathbb{TSC}$ is *strongly existentially consistent* wrt. some world model WM. By definition, all TSCs in $\mathbb{TSC}$ are existentially consistent. As a consequence of Theorem 1, they are also weakly existentially consistent. Choose some $\mathsf{TSC} = \langle H, F, C\rangle \in \mathbb{TSC}$ and set $T = \{\mathsf{TSC}_i = \langle H_i, F_i, C_i\rangle \mid i = 1, \ldots, k\} \subseteq \mathbb{TSC} \setminus \{\mathsf{TSC}\}$. By TSC semantics,

$$\begin{aligned}\varphi_{\mathsf{BC}_1^{\mathsf{TSC},T}}\left[b_{\mathsf{BC}_1^{\mathsf{TSC},T}}\backslash 0, e_{\mathsf{BC}_1^{\mathsf{TSC},T}}\backslash e\right] \equiv \\ \Big(\exists t_0, b_0, e_0 \bullet 0 < t_0 < b_0 < e_0 < e \wedge p = b_0 \wedge q = e_0 \\ \wedge \varphi_H[b_H\backslash t_0, e_H\backslash b_0] \wedge \varphi_F[b_F\backslash b_0, e_F\backslash e_0]\Big) \\ \wedge \bigwedge_{i=1}^{n}\Big(\exists t_i, b_i, e_i \bullet 0 < t_i < b_i < e_i < e \wedge b_i \leq p \wedge q \leq e_i \\ \wedge \varphi_{H_i}[b_{H_i}\backslash t_i, e_{H_i}\backslash b_i] \wedge \varphi_{F_i}[b_{F_i}\backslash b_i, e_{F_i}\backslash e_i]\Big)\end{aligned}$$

which in turn is equivalent to

$$\begin{aligned}F_1 :\equiv 0 < p < q < e \wedge \mathtt{Pre}(\mathsf{TSC}, p, q) \wedge \\ \bigwedge_{i=1}^{k}\Big(\exists b_i, e_i \bullet 0 < b_i \leq p \wedge q \leq e_i < e \wedge \mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i)\Big)\end{aligned}$$

Analogously, $\varphi_{\mathsf{BC}_2^{\mathsf{TSC},T}}\left[b_{\mathsf{BC}_2^{\mathsf{TSC},T}}\backslash 0, e_{\mathsf{BC}_2^{\mathsf{TSC},T}}\backslash e\right]$ is equivalent to

$$\begin{aligned}F_2 :\equiv 0 < p < q < e \wedge \mathtt{Pre}(\mathsf{TSC}, p, q) \wedge \varphi_C[b_C\backslash p, e_C\backslash q] \\ \wedge \bigwedge_{i=1}^{k}\Big(\exists b_i, e_i \bullet 0 < b_i < p \wedge q < e_i < e \wedge \mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i) \wedge \varphi_{C_i}[b_{C_i}\backslash b_i, e_{C_i}\backslash e_i]\Big)\end{aligned}$$

Assume there exist a trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ and a valuation μ such that $\pi, \mu \models F_1$. Then it is possible to find another valuation μ' such that

$$\pi, \mu' \models \mathtt{Pre}(\mathsf{TSC}, b_0, e_0) \wedge \bigwedge_{i=1}^{k} \mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i)$$

with $\mu'(b_0) = \mu(p) < \mu'(e_0) = \mu(q)$, and $\mu'(b_i) < \mu(p) < \mu(q) < \mu'(e_i)$ for $i = 1, \ldots, k$. So, μ' yields a set of time points that satisfies the left-hand side of (CONS) for the subset

$T \cup \{\mathsf{TSC}\}$ of TSCs. By assumption, the right-hand side of (CONS) is satisfied for the same set and time points, i.e., there exist $\pi' \in \mathcal{T}(\mathsf{WM})$ and valuation μ'' such that

$$\pi', \mu'' \models \mathtt{Pre}(\mathsf{TSC}, b_0, e_0) \wedge \varphi_{\mathsf{TSC}} \wedge \bigwedge_{i=1}^{k} (\mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i) \wedge \varphi_{\mathsf{TSC}_i})$$

By Fact 1, we know that $\mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i) \wedge \varphi_{\mathsf{TSC}_i} \Rrightarrow \mathtt{Pre}(\mathsf{TSC}_i, b_i, e_i) \wedge \varphi_{C_i}[b_{C_i} \backslash b_1, e_{C_i} \backslash e_i]$. So, finally, we can find a valuation μ''' such that $\pi', \mu''' \models F_2$. As a consequence, $\mathbb{TSC}$ is *partially consistent*. □

5.5 World Models

A worldmodel defines a type hierarchy and the universe of objects. In the following, we refine the definition of world models given earlier in Section 2.3. In particular, hierarchical class structures are introduced. Later on in the section, refinement relations between world models are considered.

Definition 22 (Type Hierarchy)**.** *A type hierarchy $(\mathcal{C}, \precsim)$ is a finite set of classes $\mathcal{C}$ and a partial order $\precsim$ on $\mathcal{C}$ that is the* refinement relation*. Each class $C \in \mathcal{C}$ defines a set of attributes $\mathcal{A}(C)$, and the possible behaviors $\mathcal{T}(C) \subseteq \mathcal{T}(\mathcal{A}(C))$ in form of traces over $\mathcal{A}(C)$.*

A class C_1 refines *class C_2 if $C_1 \precsim C_2$. We require that $C_1 \precsim C_2$ implies $\mathcal{A}(C_1) \supseteq \mathcal{A}(C_2)$ and $\mathcal{T}(C_1)\downarrow_{\mathcal{A}(C_2)} \subseteq \mathcal{T}(C_2)$.*

Definition 23 (World Model)**.** *Given some multi-sorted signature Σ, a world model $\mathsf{WM} = ((\mathcal{C}, \precsim), Obj, type, \mathcal{I})$ over Σ defines a class hierarchy $(\mathcal{C}, \precsim)$, a universe Obj of objects, a typing function $type\colon Obj \to \mathcal{C}$ and an interpretation $\mathcal{I}$ that maps all class symbols from Σ to classes from $\mathcal{C}$, and predicate and function symbols (except attribute symbols) to predicates and functions over $\mathbb{R}$. Furthermore, the class hierarchy must define an attribute for every attribute symbol from Σ.*

In the context of MSRTL, we can see attributes as a special form of unary function symbols. These function symbols are not covered by $\mathcal{I}$ because they are interpreted in the context of a trajectory.

Fact 2 (Well-defined semantics and refinement)**.** *For any two signatures $\Sigma \subseteq \Sigma'$, an MSRTL formula over Σ has a well-defined semantics in any world model over Σ'.*

This follows directly from the above definition which requires that all symbols have an interpretation. Therefore, we call every world model over $\Sigma' \supseteq \Sigma$ a Σ-world model.

Subclasses may add additional attributes, and restrict possible behaviors, but must not add new behavior. This is consistent with description logic, where any statement that can be derived from the superclass axioms must also be valid for instances of any subclass. As a consequence, any member of class C can be treated as a member of class $C' \succsim C$. Somewhat lazy, we write $o \in C$ if $type(o) \precsim C$. The attributes of an object $o \in Obj$ are defined as $\mathcal{A}(o) = \{o.a \mid a \in \mathcal{A}(type(o))\}$. We naturally extend this to sets of objects, i.e., $\mathcal{A}(O) := \bigcup_{o \in O} \mathcal{A}(o)$. The possible behavior in a given world model WM is defined as a usually infinite set $\mathcal{T}(\mathsf{WM})$ of trajectories over the attributes of finite subsets of Obj:

$$\mathcal{T}(\mathsf{WM}) \subseteq \bigcup_{O \subseteq Obj} \mathcal{T}(\mathcal{A}(O))$$

For convenience, we denote the set $O \subseteq Obj$, that some particular trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ is defined over, by O_π.

Definition 24 (World Model Refinement)**.** *A world model* $\mathsf{WM}_1 = ((\mathcal{C}_1, \precsim_1), Obj_1, type_1, \mathcal{I}_1)$ *over signature* Σ_1 refines *a world model* $\mathsf{WM}_2 = ((\mathcal{C}_2, \precsim_2), Obj_2, type_2, \mathcal{I}_2)$ *over signature* Σ_2*, written* $\mathsf{WM}_1 \precsim \mathsf{WM}_2$*, if* $\Sigma_2 \subseteq \Sigma_1$*,* $Obj_1 = Obj_2$*,* $\mathrm{Dom}(\mathcal{I}_1) \supseteq \mathrm{Dom}(\mathcal{I}_2)$*, there exists an injective mapping* $m\colon \mathcal{C}_2 \to \mathcal{C}_1$ *such that*

- $\mathcal{I}_1(\mathtt{O}) = m(\mathcal{I}_2(\mathtt{O}))$ *for all class symbols* $\mathtt{O} \in \Gamma \cap \mathrm{Dom}(\mathcal{I}_2)$*,*
- $type_1(o) \precsim C \Leftrightarrow type_2(o) \precsim m(C)$ *for all* $o \in Obj_1 = Obj_2$ *and* $C \in \mathcal{C}_2$
- $C \precsim_2 C' \Leftrightarrow m(C) \precsim_1 m(C')$ *for all* $C, C' \in \mathcal{C}_2$*,*
- $\mathcal{A}(C) \subseteq \mathcal{A}(m(C))$ *for all* $C \in \mathcal{C}_2$*,*

and $\mathcal{I}_1(f) = \mathcal{I}_2(f)$ *for all predicate and function symbols* f *in* Σ_2 *except attribute symbols.*

A refinement $\mathsf{WM}_1 \precsim \mathsf{WM}_2$ *is* conservative *if* $\mathcal{T}(\mathsf{WM}_1)\!\downarrow_{\mathcal{A}_2(Obj_2)} = \mathcal{T}(\mathsf{WM}_2)\!\downarrow_{\mathcal{A}_2(Obj_2)}$ *and* monotone *if* $\mathcal{T}(\mathsf{WM}_1)\!\downarrow_{\mathcal{A}_2(Obj_2)} \subseteq \mathcal{T}(\mathsf{WM}_2)\!\downarrow_{\mathcal{A}_2(Obj_2)}$*.*

Note that a refinement $\mathsf{WM}_1 \precsim \mathsf{WM}_2$ always implies $\mathcal{A}_2(Obj_2) \subseteq \mathcal{A}_1(Obj_1)$. Intuitively, we allow to refine a world model by introducing new classes and refining existing classes (i.e adding new attributes). In the above definition, this is realized by assuming the existence of some mapping m between the old and new class hierarchy. The universe of objects must remain the same. A refinement can change the type of objects, but only if the new type is a valid refinement of the old one. New predicate symbols can also be introduced, but it is not possible to remove existing ones or change their definition. The reason (and consequence) of these restrictions is that MSRTL formulae shall be interpretable also over refinements of their original world model.

Theorem 3 (World model refinement)**.** *For any two world models* $\mathsf{WM}_1 \precsim \mathsf{WM}_2$*, and any MSRTL formula* φ *over some signature* Σ *hold the following properties:*

1. *If* WM_2 *is a* Σ*-world model then* WM_1 *is also a* Σ*-world model.*
2. *If the refinement is conservative and both* WM_1 *and* WM_2 *are* Σ*-world models, then* $\mathit{SAT}_{\mathsf{WM}_1}(\varphi) \Leftrightarrow \mathit{SAT}_{\mathsf{WM}_2}(\varphi)$*.*
3. *If the refinement is monotone, and both* WM_1 *and* WM_2 *are* Σ*-world models, then* $\mathit{SAT}_{\mathsf{WM}_1}(\varphi) \Rightarrow \mathit{SAT}_{\mathsf{WM}_2}(\varphi)$*.*

Proof. The first bullet point follows directly from the definition of $\precsim$: If WM_1 is a Σ-world model, then by definition $\Sigma \subseteq \Sigma_1 \subseteq \Sigma_2$, so WM_2 is a Σ-world model.

For the second and third bullet point we show that $\pi_1, \mu \models_{\mathsf{WM}_1} \varphi$ iff $\pi_2, \mu \models_{\mathsf{WM}_2} \varphi$ for all $\pi_1 \in \mathcal{T}(\mathsf{WM}_1)$ and $\pi_2 \in \mathcal{T}(\mathsf{WM}_2)$ with $\pi_1\!\downarrow_{\mathcal{A}_1(Obj_1)} = \pi_2\!\downarrow_{\mathcal{A}_2(Obj_2)}$. Note that—by definition of monotone refinements—such a π_2 exists for all π_1; in case of conservative refinements also vice versa. As an induction base, the above is true for formulae of the form $p(o_1.a_1, \ldots, o_n.a_n)$ because the interpretation of p is the same in both WM_1 and WM_2. It is also trivially true for terms of the form $t_1 - t_2 \sim X$. The induction step is trivial for Boolean and temporal operators, as well as time quantification. The only crucial case is object quantification. Assume that $\pi_1, \mu \models_{\mathsf{WM}_1} \exists o \in \mathtt{O} : \varphi'$. So, there exists $obj \in Obj_1$ such that $\pi_1, \mu \cup \{o \mapsto obj\} \models_{\mathsf{WM}_1} \varphi'$ and $type_1(obj) \precsim_1 \mathcal{I}_1(\mathtt{O})$. Recall that $\mathtt{O}$ is interpreted

by both WM_1 and WM_2, so $\mathtt{0} \in \mathrm{Dom}(\mathcal{I}_2) \cap \mathrm{Dom}(\mathcal{I}_2)$. By the definition of world model refinement, there exists some injective mapping $m : \mathcal{C}_2 \to \mathcal{C}_1$ such that $m(\mathcal{I}_2(\mathtt{0})) = \mathcal{I}_1(\mathtt{0})$ and $\mathit{type}_1(o) \precsim_1 \mathcal{I}_1(\mathtt{0}) \Leftrightarrow \mathit{type}_2(o) \precsim_2 \mathcal{I}_2(\mathtt{0})$. As a consequence, $\mathit{type}_2(\mathit{obj}) \precsim_2 \mathcal{I}_2(\mathtt{0})$. So, $\pi_2, \mu \cup \{o \mapsto \mathit{obj}\} \models_{\mathsf{WM}_2} \varphi'$ and therefore $\pi_2, \mu \models_{\mathsf{WM}_2} \exists o \in \mathtt{0} : \varphi'$. The other direction is analogous. □

6 Applied Model Checking Techniques & Encoding of TSCs

This section describes the approach for checking partial consistency by means of SMT solving. First, we describe how world models are structured for the consistency analysis. This also includes a description of the assumed vehicle dynamics. Then, the overall decision algorithm is presented which involves checking a necessary and a sufficient condition for satisfaction of an existential TSC. These checks are delegated to an SMT solver. Finally, in Section 6.3, the method for encoding the chart structure of a TSC into an SMT/BMC problem is presented, followed by the basic encoding of spatial views.

6.1 World Models and Symbol Dictionaries for the Consistency Analysis

Section 2.3.2 gives a formal definition of world models. Because consistency of TSCs is defined with respect to world models, the user has to provide at least a minimal definition of the world model that declares the used object types and attributes. This also allows to check if the TSCs are correctly typed.

As detailed later in Section 6.2, the partial consistency analysis implementation constructs and solves two types of SMT/BMC problems. One type encodes a necessary condition, and one type encodes a sufficient condition for the satisfiability of an existential TSC. Because both types have to take the world model dynamics into account, we need at least a two-sided approximation of the possible behavior $\mathcal{T}(\mathsf{WM})$. The exact behavior $\mathcal{T}(\mathsf{WM})$ may be defined in different ways, e.g. as a system of differential equations, or some kind of hybrid automata. However, for assessing consistency of specifications it is impractical (if not impossible) to give a concise specification of $\mathcal{T}(\mathsf{WM})$ for basically two reasons.

- In most cases, evolutions of differential equations can only be calculated numerically.
- A model that covers all possible real-world behavior is complex.

Therefore, the consistency analysis developed in this thesis does not support user-defined hybrid automata or ODEs. Instead, we assume that vehicles follow a simple single track model [110] introduced in the next section. It can be parameterized as part of the world model or the TSC. The dynamics of other objects are characterized with invariants.

6.1.1 Simple Single Track Model

The simple single track model assumes that the longitudinal axis of the vehicle is tangential to the vehicles trajectory, and touches the trajectory at the center of the (stable) rear axis, as depicted in Figure 6.1. This tangent point is used as a reference point for the vehicle position. The direction of the longitudinal axis is called the *heading angle* θ and is at the

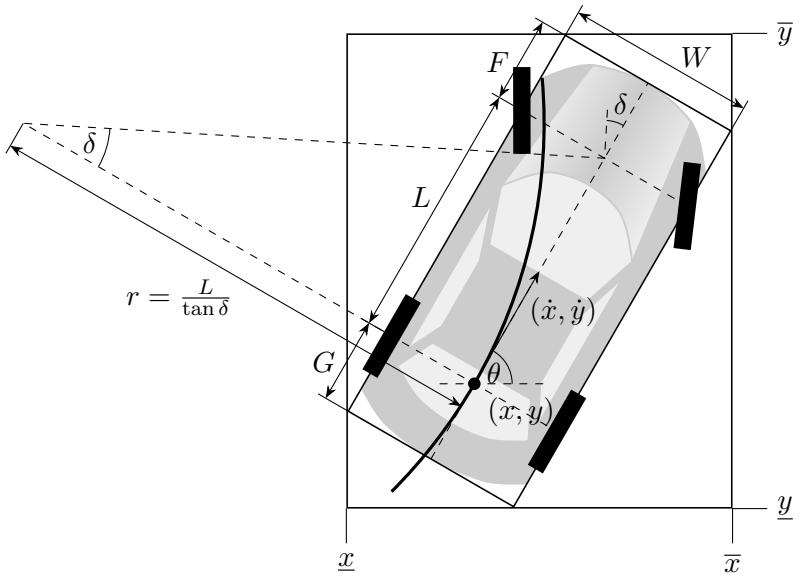


Figure 6.1: The simple single track model

same time the direction of movement. In steady state cornering, the vehicle follows a circle with radius

$$r = \frac{L}{\tan\delta}$$

depending on wheel base L and steering angle δ.

Definition 25 (Single track model ODEs)**.** *The simple single track model equations of movement can be described as*

$$\dot{x} = v\cos\theta \qquad \dot{y} = v\sin\theta \qquad \dot{\theta} = v\frac{\tan\delta}{L}$$

and are subject to the constraints

$$|\delta| \leq \delta_{\max} \qquad |\dot{\theta}v| \leq a_{lat,\max} = 0.4\text{g}$$

The hard bound for the lateral acceleration (0.4g, where $\text{g} \approx 9.8\,\text{ms}^{-2}$ is the standard gravitational acceleration) is required by Schramm, Hiller and Bardini [110] as a validity constraint for the single track model. Although this model neglects any wheel slip, it is accurate enough to describe vehicle maneuvering under moderate driving conditions [110]. Recall that its purpose is to model normal driving paths and that the consistency analysis is not supposed to be a physics simulation (cf. Section 4.2). Therefore, a more sophistic model is not required.

The axis-aligned bounding box of the vehicle depends on the size and shape, location of the reference point, and current heading of the vehicle. The single track model encoding presented in Chapter 7 assumes rectangular shaped vehicles with a fixed width W, front F, and rear overhang G as shown in Figure 6.1.

6.1.2 User-provided Domain Knowledge

Figure 6.2 shows the conceptual meta-model for world models and symbol dictionaries, and how they relate to symbol occurrences in spatial views and the bulletin board. The user

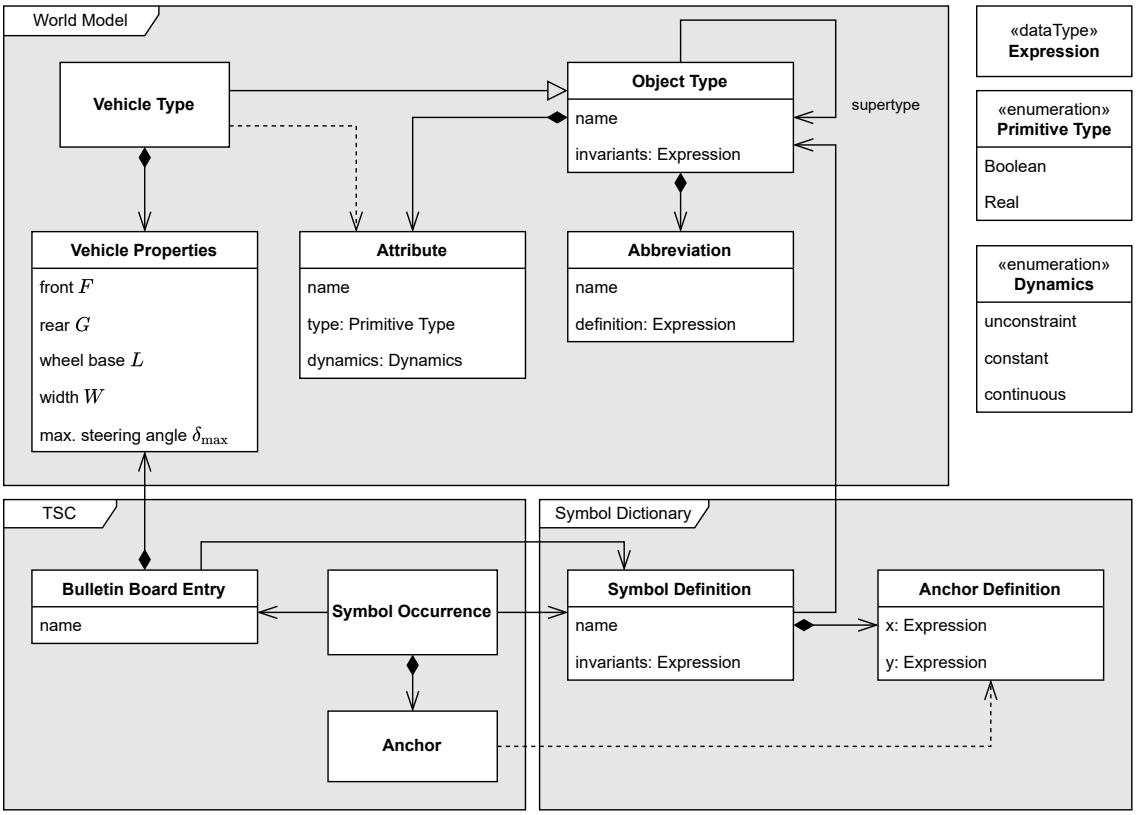


Figure 6.2: Conceptual meta-model for world models and symbol dictionaries

can define a type hierarchy and a set of attributes for each object type. In addition to normal attributes, the user can define abbreviations which are attributes whose value is defined in form of an expression. As defined in Section 2.3.2, sub-types inherit all attributes, abbreviations, and dynamics information from their super-types.

Because the consistency analysis applies special dynamics for vehicles, we conceptually introduce a dedicated meta class for vehicle types. The encoding of the single track model into SMT (Chapter 7) requires that the vehicle dimensions are a-priori known. This information can be attached to the vehicle type itself or the bulletin board entry for the vehicle.

Additional information about dynamics can be provided as invariants to each object type (vehicles and non-vehicles). Invariants can also be attached to symbol definitions and define the symbol semantics $\mathsf{O}_{s.\mathfrak{z}}(o_{s.id})$ of a symbol occurrence s in Definition 12.

In the formal descriptions in this thesis, we denote the user-provided invariants for an object type C by Φ_C. By $\Phi_C(o)$ we indicate the instantiation of these invariants for an object variable o, i.e., attribute variables a are replaced by attribute accesses $o.a$. In the remainder of this thesis, no special treatment of Boolean attributes and abbreviations is discussed. In the context of the consistency analysis, Boolean attributes can be seen as real-valued attributes restricted to the values $\{0, 1\}$. Because sub-types always restrict possible behavior of super-types, the class invariant is inherited by sub-types which, however, can add additional constraints. We assume that the behavior of any object in a trajectory lies within the possible behavior defined by its type, i.e $\pi{\downarrow_{\mathcal{A}(o)}} \in \mathcal{T}(\mathit{type}(o))$ for every $\pi \in \mathcal{T}(\mathsf{WM})$ and $o \in O_\pi$. This assumption allows the consistency analysis to identify cases

where the required or assumed behavior contradicts physical laws. On the other hand, too strong assumptions about object behavior may bias consistency results in any direction. Therefore, the two-sided approximation approach followed in this thesis uses only minimal world model constraints in one direction, while implementing the single track model in the other. As described in Section 4.2, validation of the world model requires a thorough analysis of the assumed ODD and is outside the scope of the consistency analysis.

Assumptions 1 (World model dynamics)**.** *The object types $\mathcal{C}$ in a world model WM are split into vehicle types $\mathcal{C}_{Veh} \subseteq \mathcal{C}$ and other types $\mathcal{C} \setminus \mathcal{C}_{Veh}$. For each object type $C \in \mathcal{C}$ exists an MSRTL predicate Φ_C that, together with the single track model ODEs (Definition 25) characterizes expected behavior of C. In other words, for all trajectories π over objects O_π from WM, it is $\pi \in \mathcal{T}(\mathsf{WM})$ if, and only if, for all objects $obj \in O_\pi$,*

- $\pi, \{o \mapsto obj\} \models \Box\Phi_{type(obj)}(o)$,
- *if $type(obj) \in \mathcal{C}_{Veh}$, then there is a steering input $\delta : \mathbb{R}_{\geq 0} \to \mathbb{R}$ such that $x = \pi \downarrow_{\{obj.x\}}$, $y = \pi \downarrow_{\{obj.y\}}$, $v = \pi \downarrow_{\{obj.v\}}$, $\theta = \pi \downarrow_{\{obj.\theta\}}$, δ, and their gradients fulfill the constraints in Definition 25, and*
- *π is constant on all attributes which are marked as such in the world model definition.*

These assumptions hold for the remainder of this thesis. Note, that the distinction between vehicle and non-vehicle types in the world model is a technical one. Because the simple single track model is parameterized, it can be used to model a wide range of road vehicles including passenger cars, buses, trucks, motorcycles and bicycles. Vehicles with trailers follow a more complex model which is out of scope for this work[1]. Other dynamic objects, such as pedestrians or animals, can freely move in any direction and can therefore be modeled by simpler dynamics model. In fact, the encoding provided in Section 7.4 for the simple single track model can be easily adopted for simpler dynamics by relaxing some of the constraints given there. This is briefly discussed at the end of Section 7.4.4.

6.2 The Consistency Check

This section explains the over-all process for the partial consistency check. In the following, we explain basic building blocks for the implementation of consistency analyses for TSCs. The building blocks are assembled to a (semi) decision procedure for partial consistency checking, but can be applied to similar consistency notions as well (e.g., the earlier form of partial consistency developed in [11]).

Definition 26 (TSC consistency check)**.** *A* TSC consistency check *(for some given consistency notion) is a construction that produces for every finite set $\mathbb{TSC}$ of requirement TSCs*

- *a finite indexed set $\mathbb{BC} = \{\mathsf{BC}_1, \mathsf{BC}_2, \ldots, \mathsf{BC}_{|\mathbb{BC}|}\}$ of existential TSCs, and*
- *a Boolean function $ev : \mathbb{B}^{|\mathbb{BC}|} \to \mathbb{B}$*

[1] The single track model is still trivially applicable if curved maneuvers such as turns or lane changes are excluded for vehicles with trailers.

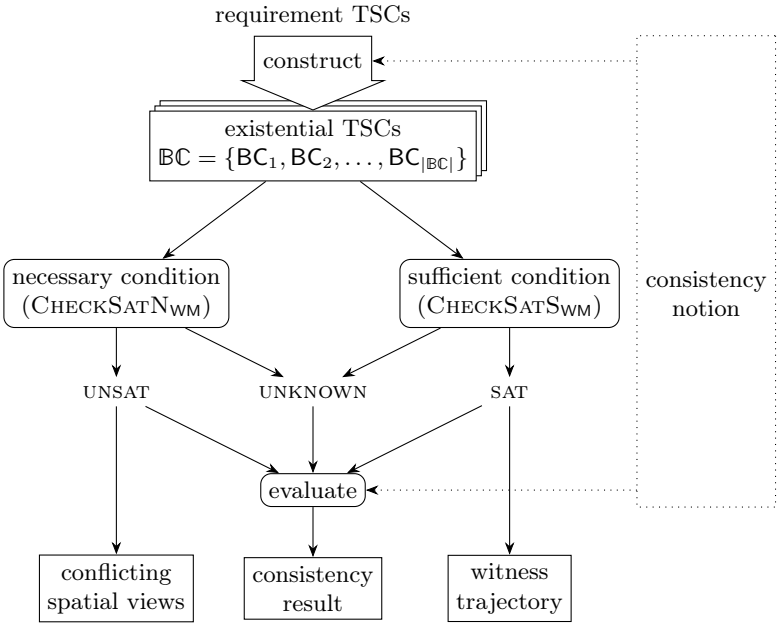


Figure 6.3: Overall strategy for consistency checking

such that $\mathbb{TSC}$ *is consistent (in the sense of the used consistency notion) with respect to some world model* WM *if, and only if,*

$$ev\big(Sat_{\mathsf{WM}}(\mathsf{BC}_1), Sat_{\mathsf{WM}}(\mathsf{BC}_2), \ldots, Sat_{\mathsf{WM}}(\mathsf{BC}_{|\mathbb{BC}|})\big) = true \ .$$

Intuitively, this definition applies to all consistency notion that reduce consistency to a finite number of cases which each rely on the satisfiability of basic charts. The function ev takes the satisfiability results for the basic charts and computes the overall consistency result. For partial consistency (Definition 21) this is a conjunction of implications

$$\bigwedge_{\mathsf{TSC},T} \textsc{Sat}_{\mathsf{WM}}\left(\mathsf{BC}_1^{\mathsf{TSC},T}\right) \Rightarrow \textsc{Sat}_{\mathsf{WM}}\left(\mathsf{BC}_2^{\mathsf{TSC},T}\right)$$

ranging over all $\mathsf{TSC} \in \mathbb{TSC}$ and $T \subseteq \mathbb{TSC} \setminus \{\mathsf{TSC}\}$.

Instead of trying to find a complete decision procedure, we approach satisfiability of basic charts BC by two semi-decision procedures CheckSatN$_{\mathsf{WM}}(\mathsf{BC})$ and CheckSatS$_{\mathsf{WM}}(\mathsf{BC})$ giving a necessary resp. sufficient condition for satisfiability wrt. some world model WM. In case CheckSatN$_{\mathsf{WM}}(\mathsf{BC})$ finds that BC is unsatisfiable, we can calculate a minimal set of spatial views from BC that cause this unsatisfiability. If, on the other hand, CheckSatS$_{\mathsf{WM}}(\mathsf{BC})$ finds BC is satisfiable, we can construct a witness trajectory showing that.

6.2.1 Two-sided Approximations for Sat$_{\mathsf{WM}}(\mathsf{BC})$

Recall that by Definition 21 a set $\mathbb{TSC}$ of TSCs is *partially inconsistent* if it is weakly existentially inconsistent or there is a $\mathsf{TSC} \in \mathbb{TSC}$ and a set $T \subseteq \mathbb{TSC} \setminus \{\mathsf{TSC}\}$ such that

$$\textsc{Sat}_{\mathsf{WM}}\left(\mathsf{BC}_1^{\mathsf{TSC},T}\right) \wedge \overline{\textsc{Sat}}_{\mathsf{WM}}\left(\mathsf{BC}_2^{\mathsf{TSC},T}\right) \ .$$

For world models that involve ODEs, it is typically not possible to decide $\text{SAT}_{\mathsf{WM}}(\mathsf{BC}_i)$ $(i = 1, 2)$. So we will approach $\text{SAT}_{\mathsf{WM}}(\mathsf{BC}_i)$ from two sides by providing two checks CheckSatS (a sufficient condition) and CheckSatN (a necessary condition) such that

$$\text{CheckSatS}_{\mathsf{WM}}(\mathsf{BC}) = \text{SAT} \implies \text{SAT}_{\mathsf{WM}}(\mathsf{BC})$$

and

$$\text{CheckSatN}_{\mathsf{WM}}(\mathsf{BC}) = \text{UNSAT} \implies \overline{\text{SAT}}_{\mathsf{WM}}(\mathsf{BC})$$

for any world model WM, and existential TSC BC that can be handled by the implementation.

Both CheckSatN and CheckSatS work by creating a BMC problem according to the process depicted in Figure 6.4. The temporal structure and the spatial views in the basic chart are translated separately from each other and in the end merged together with some glue logic. The translation of the temporal structure is the same for CheckSatN and CheckSatS, and given in Section 6.3.1. Each spatial view is first translated to a linear SMT formula using Construction 1 presented in Section 6.4. This formula may contain bound variables which need to be removed. Therefore it is handed over to an SMT solver for quantifier elimination[2]. The resulting BMC problem is then unrolled and fed into an SMT solver using the unrolling procedure in Algorithm 1. Because the algorithms rely on several building blocks which have not been introduced yet, their formal definition is postponed to later sections. The procedure CheckSatN is presented as Algorithm 7 in Section 6.5, and CheckSatS is given as Algorithm 9 at the end of Section 7.4.4.

Corollary 1 (Correctness of CheckSatS and CheckSatN)**.** *For Algorithms 7 and 9 holds*

$$\mathit{CheckSatS}_{\mathsf{WM}}(\mathsf{BC}) = \mathit{SAT} \implies \mathit{SAT}_{\mathsf{WM}}(\mathsf{BC})$$

and

$$\mathit{CheckSatN}_{\mathsf{WM}}(\mathsf{BC}) = \mathit{UNSAT} \implies \overline{\mathit{SAT}}_{\mathsf{WM}}(\mathsf{BC})$$

Proof. Correctness of CheckSatN follows from Theorem 8. Correctness of CheckSatS follows from Theorem 12. □

The referenced theorems state the correctness of different parts of the translation process from existential TSCs to SMT formulae. They are introduced and proven later in this thesis. Theorems 5 and 6 refer to the chart structure encoding and can be found in Section 6.3.1. Theorems 7 (Section 6.4) and 11 (Section 7.5) state correctness of the necessary, respectively sufficient, constraints for spatial views. Figure 6.5 shows how the theorems and lemmas from this thesis contribute to Corollary 1.

6.2.2 The Algorithm for Consistency Checking

Using CheckSatN and CheckSatS as necessary, respectively sufficient, constraints for satisfiability of existential TSCs leads to a simple semi-decision procedure for partial consistency listed in Algorithm 3. However, there is room for improvement.

[2] Z3 allows to execute simplification steps such as quantifier elimination standalone.

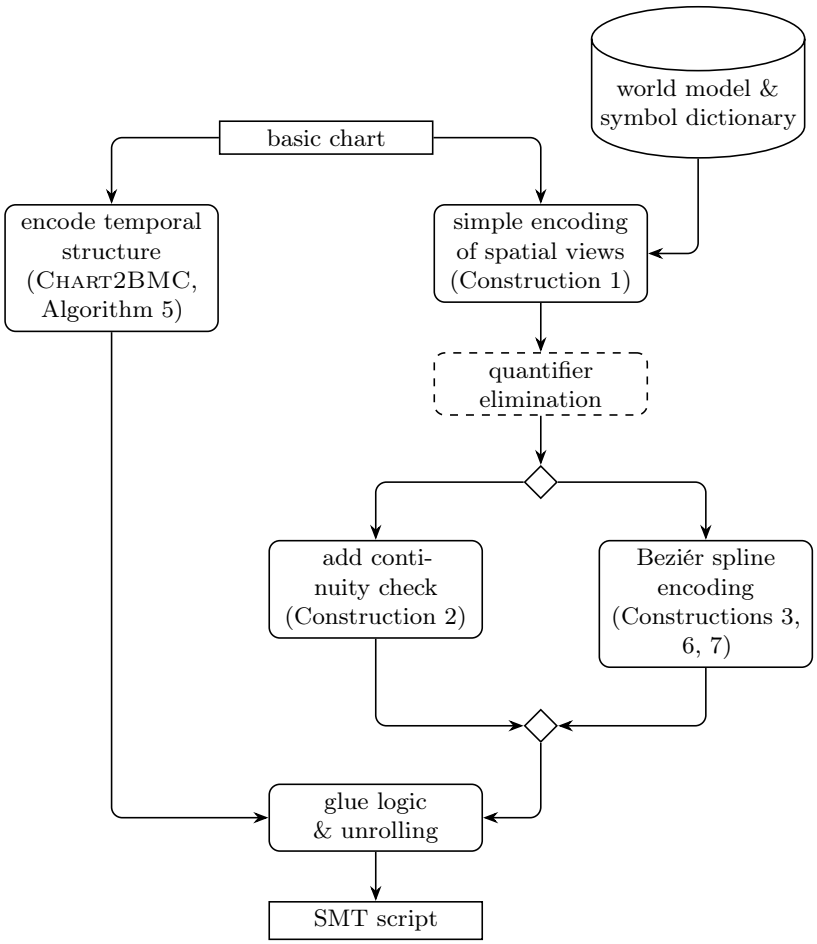


Figure 6.4: Structure of the encoding process

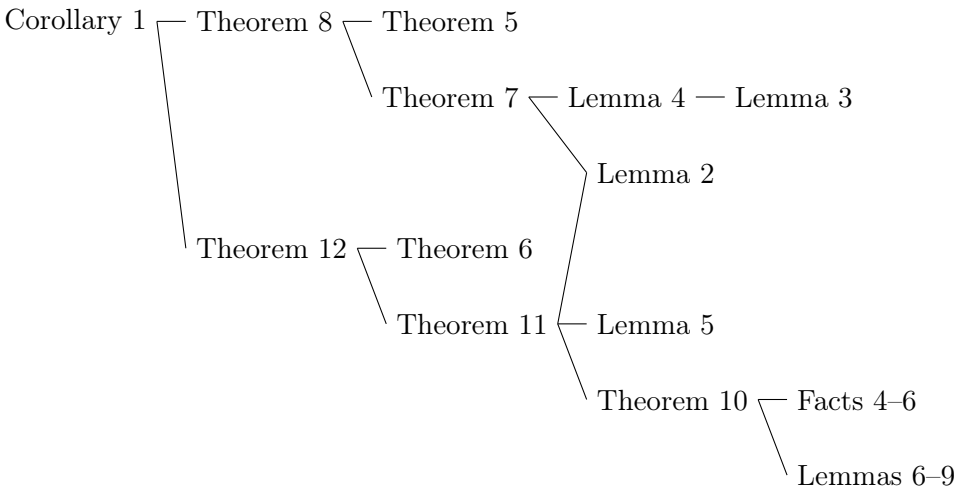


Figure 6.5: Framework of theorems that are used to prove correctness of CheckSatN and CheckSatS (Corollary 1). Theorems on the right side of an edge are used to prove the theorems on the left

Algorithm 3: Basic algorithm for partial consistency checking. $\mathsf{BC}_1^{\mathsf{TSC},T}$ and $\mathsf{BC}_2^{\mathsf{TSC},T}$ are the same as in Definition 21.

```
1 for each TSC ∈ 𝕋𝕊ℂ do
2 |   for each T ⊆ 𝕋𝕊ℂ \ {TSC} do
3 |   |   if T = ∅ or CheckSatS_WM(BC_1^{TSC,T}) = SAT then
4 |   |   |   if CheckSatN_WM(BC_2^{TSC,T}) = UNSAT then
5 |   |   |   |   Report inconsistency of (TSC, T);
6 |   |   |   end
7 |   |   end
8 |   end
9 end
```

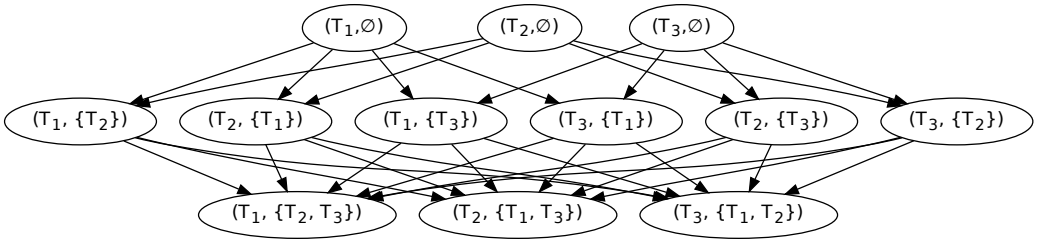


Figure 6.6: Consistency cases to check for $\mathbb{TSC} = \{\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3\}$. Each case is a pair (TSC, T) of the primary TSC TSC and the set T of other TSCs. The consistency case on the source end of an arrow is to be checked before the target end.

- Typically, one is interested in the *minimal inconsistent sets*. So, there is no need to check some pair (TSC, T) if already some inconsistency in a subset of $T \cup \{\mathsf{TSC}\}$ has been found. Smaller subsets are faster to check, so one shall check the smallest subsets first. Figure 6.6 illustrates this hierarchy for a set of three TSCs $\{\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3\}$.

- When comparing the spatial view encoding for necessary versus sufficient conditions, it is clear that necessary conditions are faster to solve due to the piecewise approximations necessary for the sufficient condition (see Chapter 7). So, it is worthwhile to check both satisfiability of BC_1 and BC_2 first with CheckSatN before sorting out false-positives to BC_1 using CheckSatS.

These considerations lead to the improved consistency check in Algorithm 4. As another performance booster on multi-core machines, all the nested loops in Algorithm 4 can be parallelized (lines 6–18 shall be executed sequentially because of above considerations), provided that some pair (TSC, T) is checked (lines 6–18) before any pair (TSC', T') with $T' \cup \{\mathsf{TSC}'\} \supset T \cup \{\mathsf{TSC}\}$. Figure 6.6 shows the resulting partial order in that the consistency cases for a set of three TSCs shall be checked.

Theorem 4 (Correctness of Algorithms 3 and 4)**.** *Each pair* (TSC, T) *reported by Algorithm 3 corresponds to a partially inconsistent subset* $T \cup \{\mathsf{TSC}\} \subseteq \mathbb{TSC}$*. Among the*

Algorithm 4: Improved algorithm for partial consistency checking. $\mathsf{BC}_1^{\mathsf{TSC},T}$ and $\mathsf{BC}_2^{\mathsf{TSC},T}$ are the same as in Definition 21.

```
1  MIS := ∅;
2  Skip := ∅;
3  for k = 0 to |𝕋𝕊ℂ| − 1 do
4  |  for each TSC ∈ 𝕋𝕊ℂ do
5  |  |  for each T ⊆ 𝕋𝕊ℂ \ {TSC} with |T| = k do
6  |  |  |  if T = ∅ then
7  |  |  |  |  if CheckSatN_WM(BC_2^{TSC,T}) = UNSAT then
8  |  |  |  |  |  add {TSC} to MIS;
9  |  |  |  |  end
10 |  |  |  else if not ∃S ⊂ T ∪ {TSC} : (S ∈ MIS ∨ (TSC, S) ∈ Skip) then
11 |  |  |  |  if CheckSatN_WM(BC_1^{TSC,T}) = UNSAT then
12 |  |  |  |  |  add (TSC, T) to Skip;
13 |  |  |  |  else if CheckSatN_WM(BC_2^{TSC,T}) ≠ UNSAT then
14 |  |  |  |  |  continue;
15 |  |  |  |  else if CheckSatS_WM(BC_1^{TSC,T}) = SAT then
16 |  |  |  |  |  add T ∪ {TSC} to MIS;
17 |  |  |  |  else add (TSC, T) to Skip;
18 |  |  |  end
19 |  |  end
20 |  end
21 end
22 return MIS;
```

partially inconsistent subsets $T \cup \{\mathsf{TSC}\}$ reported by Algorithm 3, Algorithm 4 returns the minimal sets (wrt. set inclusion).

Proof. Algorithms 3 and 4 use the following three conditions:

(A) $\textsc{CheckSatS}_{\mathsf{WM}}\left(\mathsf{BC}_1^{\mathsf{TSC},T}\right) = \textsc{sat}$

(B) $\textsc{CheckSatN}_{\mathsf{WM}}\left(\mathsf{BC}_2^{\mathsf{TSC},T}\right) = \textsc{unsat}$

(C) $\textsc{CheckSatN}_{\mathsf{WM}}\left(\mathsf{BC}_1^{\mathsf{TSC},T}\right) = \textsc{unsat}$

By Corollary 1,

- condition (A) implies $\textsc{Sat}_{\mathsf{WM}}(\mathsf{BC}_1^{\mathsf{TSC},T})$,
- condition (B) implies $\overline{\textsc{Sat}}_{\mathsf{WM}}(\mathsf{BC}_2^{\mathsf{TSC},T})$,
- conditions (A) and (C) are mutually exclusive

By the control flow in Algorithm 3, a pair (TSC, T) is reported as inconsistent only if either (B) holds and $T = \emptyset$, or both (A) and (B) hold. Note that, for $T = \emptyset$, the existential TSC $\mathsf{BC}_1^{\mathsf{TSC},\emptyset}$ is equivalent to the TSC $\mathsf{HFC}_{\mathsf{TSC}}$ from Definition 19. So, each pair (TSC, T) reported by Algorithm 3 corresponds to a partially inconsistent subset $T \cup \{\mathsf{TSC}\}$ of $\mathbb{TSC}$.

In order to prove the same for Algorithm 4, we inspect again the control flow. We show by induction over $k = |T|$ that:

- a set $T \cup \{\mathsf{TSC}\}$ is added to `MIS` if, and only if, (TSC, T) is reported by Algorithm 3, but no (TSC', T') with $T' \cup \{\mathsf{TSC}'\} \subset T \cup \{\mathsf{TSC}\}$ is reported; and
- a pair (TSC, T) is added to `Skip` only if $\mathsf{BC}_1^{\mathsf{TSC},T}$ is unsatisfiable.

The outer loop iterates over k, so, in each iteration of the innermost loop, the above holds for all $T \subseteq \mathbb{TSC}$ with $|T| < k$. For $k = 0$ only condition (B) is checked, analogous to Algorithm 3. So, the induction hypothesis holds for $k = 0$. In the following, assume we have shown it for all TSC, T with $|T| < k$, and chose some concrete pair (TSC, T) with $|T| = k > 0$ (line 5 of Algorithm 4). We distinguish two cases:

For the first case, assume (TSC, T) is reported by Algorithm 3. Then, by the control flow of that algorithm, conditions (A) and (B) hold. Recall that condition (A) implies $\textsc{Sat}_{\mathsf{WM}}(\mathsf{BC}_1^{\mathsf{TSC},T})$. By construction, $\mathsf{BC}_2^{\mathsf{TSC},T} \Rrightarrow \mathsf{BC}_2^{\mathsf{TSC},S}$ for all $S \subseteq T$, so, by induction hypothesis, there is no $S \subseteq T$ such that $(\mathsf{TSC}, S) \in$ `Skip`. If Algorithm 3 reports some (TSC', T') with $S = T' \cup \{\mathsf{TSC}'\} \subset T \cup \{\mathsf{TSC}\}$, then, by induction hypothesis, $S \in$ `MIC`. As a consequence, we reach the if-else clause in line 11 if, and only if, no such pair has been reported. Because condition (A) and (C) are exclusive, the conditions (C) and (not B) checked in lines 11 and 14 evaluate to *false* and the control flow ends in line 16, adding $T \cup \{\mathsf{TSC}\}$ to `MIC` (and not adding (TSC, T) to `Skip`). So, the induction hypothesis holds for the case that (TSC, T) is reported by Algorithm 3.

For the other case, assume that (TSC, T) is *not* reported by Algorithm 3. Assume the if-condition in line 10 evaluates to *true* and we proceed to the inner if-else (otherwise, the algorithm proceeds with the next loop iteration, and the induction hypothesis is satisfied). By the control flow of Algorithm 3, we know that at least one of conditions (A) and be is violated. We have the following possible cases:

- If condition (C) is satisfied, then the control flow reaches line 12, adding (TSC, T) to `Skip`. Because conditions (A) and (C) are contradicting, (A) must be violated, so the induction hypothesis holds.
- Otherwise, if condition (B) is violated, the control flow continues with the next loop iteration without changing `Skip` or `MIC`.
- Otherwise, condition (A) is violated instead, and the control flow reaches line 17.

In all cases, the induction hypothesis holds. □

6.2.3 A Note on Completeness

Although the two algorithms are correct in the sense that all returned subsets are partially inconsistent, they are not guaranteed to be complete, i.e., there may still be inconsistent subsets that are not found by the algorithms. This is due to the additional over and under approximations introduced by CheckSatN and CheckSatS. Therefore, there is also no

guarantee that the sets reported by Algorithm 4 are minimal inconsistent subsets of the input set $\mathbb{TSC}$ – for any set T in the result of Algorithm 4 can still exist another partially inconsistent set $S \subset T$ that is not found by either algorithm.

Having a closer look at this incompleteness, we shall distinguish two cases:

1. Incompleteness that is caused by CheckSatN$_{\mathsf{WM}}\left(\mathsf{BC}_2^{\mathsf{TSC},T}\right)$ checking only a necessary condition for satisfiability of $\mathsf{BC}_2^{\mathsf{TSC},T}$, and
2. Incompleteness that is caused by CheckSatS$_{\mathsf{WM}}\left(\mathsf{BC}_1^{\mathsf{TSC},T}\right)$ not finding a satisfying trajectory for $\mathsf{BC}_2^{\mathsf{TSC},T}$, and therefore spuriously assuming that a set of TSCs cannot be "active" together.

The first case mostly misses those inconsistencies that arise because the combination of the TSCs requires impossible maneuvers, although there are no parallel or sequencial spatial views that contradict on a logical level[3] – the evaluation in Section 8.3.1 shows that CheckSatN is quite good in detecting logical contradictions. This may be addressed by approximating partial consistency also in the opposite direction. The only thing we need to do is replacing CheckSatN by CheckSatS and vice versa in lines 13 and 15 of Algorithm 4. This, however, would likely introduce a lot of spurious inconsistency findings and thereby contradict Thesis Goal 1.

The second case may easily occur from a miss-configuration of the analysis. The encoding of the single track model (see Chapter 7) requires that the user provides a suitable discretization of vehicle heading angles, fixed step size, and unrolling depth for the BMC problem. This may be addressed providing guidance for the user when configuring the analysis (which is future work) and by reporting consistency cases where the control flow of Algorithm 4 reaches line 17 of the algorithm. This does not cause additional computational cost but allows the user to inspect the precarious cases manually and maybe adjust the configuration.

6.3 Encoding of the Chart Structure

In the following, we describe the translation of the chart structure of basic charts to BMC problems. The content of spatial views is not considered here – the translation of spatial views is abstracted by a translation function $tr()$ and described in detail later on in this thesis. For practical reasons, we restrict ourselves to positive existential TSCs, i.e., we exclude negation of charts. A similar restriction is made for the related approaches for model checking of duration calculus, e.g., by Gonnord et al. [65], Sharma et al. [113], or Fränzle [57].

Due to the structural similarities to interval logics (see Section 3.3), the translation algorithm has been loosely inspired by the BMC-encoding for integer time duration calculus developed by Gonnord, Halbwachs, and Raymond [65]. The key idea that has been adopted from this work is the use of Boolean oracle variables for switching points between sequential sub-charts. The oracles manifest as Boolean variables past_τ. These correspond to time variables τ in the MSRTL formula and encode whether the current time is before ($\mathsf{past}_\tau = \mathit{false}$) or after ($\mathsf{past}_\tau = \mathit{true}$) the switching point. This use of oracle variables

[3] By logical level we mean a contradiction that can be deduced from the MSRTL semantics without considering driving dynamics.

makes the size of the BMC problem (in terms of variables and expressions) linear to the size of the basic chart (in terms of timing annotations, operations, and invariant nodes). Furthermore, care has been taken to encode certain timing constraints with Boolean variables only. As a consequence, charts without hour glasses (but that may contain time pins) become pure Boolean SAT problems and can be handled more effectively by the SMT solver.

6.3.1 Encoding

Translation positive existential TSCs results in a restricted form of MSRTL formulae. The translation φ_{BC} of a (negation-free) chart BC is an MSRTL formula $\varphi(b, e)$ with free time variables b, e that is of the form

$$\begin{aligned}\varphi(b,e) ::= {} & \Box_{[b,e)}\phi \wedge b < e \mid \tau_1 - \tau_2 \bowtie X \\ & \mid \exists \tau \bullet \varphi_1(b,\tau) \wedge \varphi_2(\tau, e) \mid \bigwedge_{j\in J} \varphi_j(b,e) \mid \bigvee_{j\in J} \varphi_j(b,e)\end{aligned} \tag{6.1}$$

where ϕ is an MSRTL predicate, τ_1, τ_2 time variables (both τ_1 and τ_2 may be b, e, or another free or bound time variable), $\bowtie \in \{\leq, <\}$, $X \in \mathbb{R}$ a constant, and J a finite index set. Recall, that we restrict ourselves to positive existential TSCs, i.e., time variables do not occur under negations. So, further time variables introduced by timing annotations (see Definition 8) can be replaced by free variables, yielding a formula of the above form.

6.3.1.1 Needed Variables

The construction needs the following variables:

1. A global time variable t tracks the global time progress. Initially, $t = 0$, and in each step t strictly increases.

2. For each sub-formula $\varphi_i(b, e)$, introduce a Boolean variable $\mathsf{complete}_i$ that indicates whether φ_i is fulfilled between the time b and the current step (not until time e).

3. For each sub-formula φ_i of the form $\Box_{[b,e)}\phi_i \wedge b < e$, additionally introduce a Boolean variable ok_i.

4. For time variables that occur as beginning variable b of some formula $\varphi(b, e)$, or as part of a constraint $\tau_1 \bowtie \tau_2$ where both τ_1 and τ_2 are time variables, introduce a Boolean variable past_τ.

5. For time variables that occur in some time constraint that is not of the above form, introduce a real-valued variable v_τ.

Note, that items 4 and 5 can hold simultaneously for some time variable. In this case, both variables past_τ and v_τ are introduced. Both encode the value $\mu(\tau)$ of τ, but in different ways: In the BMC problem, v_τ directly holds the value of τ, whereas past_τ models that the value of τ lies in the past, i.e., $\mathsf{past}_{\tau,i}$ is *true* if $\tau \geq t_i$. In order to maintain consistency between those encodings, we add the following constraints. Recall that primed variables refer to the next step, and unprimed variables to the current step.

- If v_τ has been introduced, add a step constraint $v'_\tau = v_\tau$.

- If only past_τ has been introduced, add a transition constraint $\mathsf{past}_\tau \Rightarrow \mathsf{past}'_\tau$.
- If both v_τ and past_τ have been introduced, add an initialization constraint $\mathsf{past}_\tau \Leftrightarrow v_\tau = 0$ and a step constraint

$$(\mathsf{past}'_\tau \Rightarrow v_\tau \geq t') \wedge (\neg \mathsf{past}_\tau \Rightarrow v_\tau \leq t') \ .$$

6.3.1.2 Encoding Rules

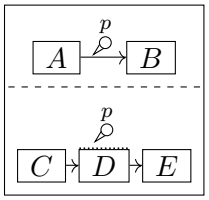
In the following, the encoding rules are introduced. Encoding happens recursively. Table 6.1 shows the translation rules for MSRTL formulae of the identified form. It can be understood as a look-up table for the translation. Any sub-formula $\varphi_i(b, e)$ matches one of the patterns in the first column, and the other columns show the respective initialization (I_i) and transition (T_i) constraints that are to be introduced for the sub-formula. Timing constraints of the form $\tau_1 - \tau_2 \bowtie 0$ (that can be written as $\tau_1 \bowtie \tau_2$) are encoded with Boolean variables only. This enables the SMT solver to tackle them in the Boolean fragment of the theory, which is more efficient. General timing constraints (comparing to a constant $X \neq 0$, last row in the table), however, require linear arithmetics and are translated literally to SMT formulae. Here, time variables τ are encoded by real variables v_τ, except the end time variable e that is replaced by the current time t. For example, a hour glass constraint that translates to $e_{\mathsf{BC}} - \tau_{hg} \bowtie X$ (see Definition 8) is encoded as $t - v_{\tau_{hg}} \bowtie X$.

As can be seen in the translation rules, the end time e is not directly considered when encoding a sub-formula $\varphi_i(b, e)$. The reason is that during translation of a choice $\bigvee_{j \in J} \varphi_j(b, e)$ it is not clear which of the branches $\varphi_j(b, e)$ ($j \in J$) evaluates to *true* in a solution of the MSRTL formula and thereby imposes a constraint on e. Therefore, the Boolean variables $\mathsf{complete}_i$ only mark candidate time points for e. When encoding sequences (i.e., sub-formulae of the form $\exists \tau \bullet \varphi_j(b, \tau) \wedge \varphi_k(\tau, e)$) in BMC, constraints (C5) and (C12) ensure that τ can only bind to time points where the first part of the sequence can end. This exploits the inductive structure of the MSRTL formulae that ensures that every begin or end time variable, except the top level ones, are bound in some sequence of that form.

For the top-level formula $\varphi(b, e)$, we add $\mathsf{complete}_\varphi$ to the target F of the BMC problem.

The described construction is presented in condensed form in Algorithm 5.

Example 5. *The existential TSC given below translates to the MSRTL formula on the right:*



$$\varphi :\equiv \exists p \bullet \bigwedge \left\{ \begin{array}{c} \exists \tau_1 \bullet \bigwedge \left\{ \begin{array}{c} 0 < \tau_1 \wedge \Box_{[0,\tau_1)} \phi_A \\ \tau_1 < e \wedge \Box_{[\tau_1,e)} \phi_B \\ \tau_1 =_5 p \end{array} \right\} \\ \exists \tau_2 \bullet \bigwedge \left\{ \begin{array}{c} 0 < \tau_2 \wedge \Box_{[0,\tau_1)} \phi_C \\ \exists \tau_3 \bullet \bigwedge \left\{ \begin{array}{c} \tau_2 < \tau_3 \wedge \Box_{[\tau_3,\tau_4)} \phi_D \\ \tau_2 \leq p \leq \tau_3 \\ \tau_3 < e \wedge \Box_{[\tau_4,e)} \phi_E \end{array} \right\} \end{array} \right\} \end{array} \right\}$$

We drop the outermost existential quantifier and decompose the formula as follows:

$$\begin{aligned} \varphi_0(b, e) &:\equiv \varphi_1(b, e) \wedge \varphi_2(b, e) \\ \varphi_1(b, e) &:\equiv \exists \tau_1 \bullet \varphi_3(b, \tau_1) \wedge \varphi_4(\tau_1, e) \\ \varphi_2(b, e) &:\equiv \exists \tau_2 \bullet \varphi_5(b, \tau_2) \wedge \varphi_6(\tau_2, e) \end{aligned}$$

Algorithm 5: The function CHART2BMC

Input: A basic chart BC, and a translation function $tr(\cdot)$ for spatial views
Output: A BMC problem (I, T, F)

begin
  declare a real variable t;
  $I :\equiv t = 0$; $T :\equiv t' > t$;
  let V_T the set of free and bound time variables in φ_{BC};
  decompose φ_{BC} into sub-formulae $\varphi_i(b, e)$ according to Equation (6.1);
  for each *sub-formula* $\phi_i(b, e)$ *of* φ_{BC}*, including* φ_{BC} *itself,* **do**
    introduce a Boolean variable $\mathsf{complete}_i$;
    ensure that there is a Boolean variable past_b for b;
    / switch case over the form of* φ_i*, where* b *and* e *bind to the respective begin and end variables for* φ_i*, and* j, k *are indices of sub-formulae of* φ_{BC}*: */*
    if $\varphi_i(b, e) \equiv \Box_{[b,e)}\phi_{sv} \wedge b < e$ **then** *// sv is a spatial view*
      introduce Boolean variables ok_i and b_{sv};
      $I_i :\equiv \mathsf{ok}_i \wedge \neg\mathsf{complete}_i$;
      $T_i :\equiv (\mathsf{ok}'_i \Leftrightarrow (\mathsf{past}_b \Rightarrow \mathsf{ok}_i \wedge tr(\phi))) \wedge (\mathsf{complete}'_i \Leftrightarrow (\mathsf{past}_b \wedge \mathsf{ok}'_i))$;
    else if $\varphi_i(b, e) \equiv \bigwedge_{j\in J} \varphi_j(b, e)$ *for some index set* J **then**
      $I_i :\equiv \mathsf{complete}_i \Leftrightarrow \bigwedge_{j\in J} \mathsf{complete}_j$;
      $T_i :\equiv \mathsf{complete}'_i \Leftrightarrow \bigwedge_{j\in J} \mathsf{complete}'_j$;
    else if $\varphi_i(b, e) \equiv \bigvee_{j\in J} \varphi_j(b, e)$ *for some index set* J **then**
      $I_i :\equiv \mathsf{complete}_i \Leftrightarrow \bigvee_{j\in J} \mathsf{complete}_j$;
      $T - I :\equiv \mathsf{complete}'_i \Leftrightarrow \bigvee_{j\in J} \mathsf{complete}'_j$;
    else if $\varphi_i(b, e) \equiv \exists\tau \bullet \varphi_j(b, \tau) \wedge \varphi_k(\tau, e)$ **then**
      introduce a fresh Boolean variable past_τ for τ;
      $I_i :\equiv (\mathsf{complete}_i \Leftrightarrow \mathsf{complete}_k) \wedge (\mathsf{past}_\tau \Rightarrow \mathsf{complete}_j)$;
      $T_i :\equiv (\mathsf{complete}'_i \Leftrightarrow \mathsf{complete}'_k) \wedge (\mathsf{past}'_\tau \Rightarrow (\mathsf{past}_\tau \vee \mathsf{complete}'_j)$;
    else if $\varphi_i(b, e) \equiv \tau_1 \leq \tau_2$ *with* $\tau_1, \tau_2 \in V_T \setminus \{e\}$ **then**
      $I_i :\equiv \mathsf{complete}_i \Rightarrow (\mathsf{past}_{\tau_2} \Rightarrow \mathsf{past}_{\tau_1})$;
      $T_i :\equiv (\mathsf{complete}'_i \Rightarrow (\mathsf{past}'_{\tau_2} \Rightarrow \mathsf{past}'_{\tau_1})) \wedge (\mathsf{complete}'_i \Leftrightarrow \mathsf{complete}_i)$;
    else if $\varphi_i(b, e) \equiv \tau_1 < \tau_2$ *with* $\tau_1, \tau_2 \in V_T \setminus \{e\}$ **then**
      $I_i :\equiv \mathsf{complete}_i \Rightarrow \neg\mathsf{past}_{\tau_1}$;
      $T_i :\equiv (\mathsf{complete}'_i \Rightarrow (\mathsf{past}'_{\tau_2} \Rightarrow \mathsf{past}_{\tau_1})) \wedge (\mathsf{complete}'_i \Leftrightarrow \mathsf{complete}_i)$;
    else if $\varphi_i(b, e) \equiv \tau \leq e$ *with* $\tau \in V_T \setminus \{e\}$ **then**
      $I_i :\equiv \mathsf{complete}_i \Rightarrow \mathsf{past}_\tau$;
      $T_i :\equiv \mathsf{complete}'_i \Rightarrow \mathsf{past}'_\tau$;
    else if $\varphi_i(b, e) \equiv \tau < e$ *with* $\tau \in V_T \setminus \{e\}$ **then**
      $I :\equiv \neg\mathsf{complete}_i$;
      $T :\equiv \mathsf{complete}'_i \Rightarrow \mathsf{past}_\tau$;
    else if $\varphi_i(b, e) \equiv \tau \geq e$ *with* $\tau \in V_T \setminus \{e\}$ **then**
      $I_i :\equiv$ *true* ;
      $T_i :\equiv \mathsf{complete}'_i \Rightarrow \neg\mathsf{past}_\tau$;
    else if $\varphi_i(b, e) \equiv \tau > e$ *with* $\tau \in V_T \setminus \{e\}$ **then**
      $I_i :\equiv \mathsf{complete}_i \Rightarrow \neg\mathsf{past}_\tau$;
      $T_i :\equiv \mathsf{complete}'_i \Rightarrow \neg\mathsf{past}'_\tau$;

Table 6.1: Translation rules for MSRTL formulae (tags in brackets are used to refer to the constraints in the proofs)

$\varphi_i(b,e)$	I_i	T_i
$\Box_{[b,e)}\phi_{sv} \wedge b < e$ (invariant nodes)	(C1) $\mathsf{ok}_i \wedge \neg\mathsf{complete}_i$	(C6) $\mathsf{ok}'_i \Leftrightarrow (\mathsf{past}_b \Rightarrow \mathsf{ok}_i \wedge b_{sv})$ (C7) $\mathsf{complete}'_i \Leftrightarrow (\mathsf{past}_b \wedge \mathsf{ok}'_i)$
$\varphi_j(b,e) \wedge \varphi_k(b,e)$ (concurrency)	(C2) $\mathsf{complete}_i \Leftrightarrow (\mathsf{complete}_j \wedge \mathsf{complete}_k)$	(C8) $\mathsf{complete}'_i \Leftrightarrow (\mathsf{complete}'_j \wedge \mathsf{complete}'_k)$
$\varphi_j(b,e) \vee \varphi_k(b,e)$ (choice)	(C3) $\mathsf{complete}_i \Leftrightarrow (\mathsf{complete}_j \vee \mathsf{complete}_k)$	(C9) $\mathsf{complete}'_i \Leftrightarrow (\mathsf{complete}'_j \vee \mathsf{complete}'_k)$
$\exists\tau \bullet \varphi_j(b,\tau) \wedge \varphi_k(\tau,e)$ (sequence)	(C4) $\mathsf{complete}_i \Leftrightarrow \mathsf{complete}_k$ (C5) $\mathsf{past}_\tau \Rightarrow \mathsf{complete}_j$	(C10) $\mathsf{complete}'_i \Leftrightarrow \mathsf{complete}'_k$ (C11) $\mathsf{past}_\tau \Rightarrow \mathsf{past}'_\tau$ (C12) $\mathsf{past}'_\tau \Rightarrow (\mathsf{past}_\tau \vee \mathsf{complete}'_j)$
$\tau_1 \leq \tau_2$ with $\tau_1, \tau_2 \in V_T \setminus \{e\}$	(C13) $\mathsf{complete}_i \Rightarrow (\mathsf{past}_{\tau_2} \Rightarrow \mathsf{past}_{\tau_1})$	(C19) $\mathsf{complete}'_i \Rightarrow (\mathsf{past}'_{\tau_2} \Rightarrow \mathsf{past}'_{\tau_1})$ (C20) $\mathsf{complete}'_i \Leftrightarrow \mathsf{complete}_i$
$\tau_1 < \tau_2$ with $\tau_1, \tau_2 \in V_T \setminus \{e\}$	(C14) $\mathsf{complete}_i \Rightarrow \neg\mathsf{past}_{\tau_2}$	(C21) $\mathsf{complete}'_i \Rightarrow (\mathsf{past}'_{\tau_2} \Rightarrow \mathsf{past}_{\tau_1})$ (C20) $\mathsf{complete}'_i \Leftrightarrow \mathsf{complete}_i$
$\tau \leq e, \tau \in V_T \setminus \{e\}$	(C15) $\mathsf{complete}_i \Rightarrow \mathsf{past}_\tau$	(C22) $\mathsf{complete}'_i \Rightarrow \mathsf{past}'_\tau$
$\tau < e, \tau \in V_T \setminus \{e\}$	(C16) $\neg\mathsf{complete}_i$	(C23) $\mathsf{complete}'_i \Rightarrow \mathsf{past}_\tau$
$\tau \geq e, \tau \in V_T \setminus \{e\}$	—	(C24) $\mathsf{complete}'_i \Rightarrow \neg\mathsf{past}_\tau$
$\tau > e, \tau \in V_T \setminus \{e\}$	(C17) $\mathsf{complete}_i \Rightarrow \neg\mathsf{past}_\tau$	(C25) $\mathsf{complete}'_i \Rightarrow \neg\mathsf{past}'_\tau$
any other timing constraint $\tau_1 - \tau_2 \bowtie X$ with $X \in \mathbb{R} \setminus \{0\}$	(C18) $\mathsf{complete}_i \Rightarrow \begin{Bmatrix} t & \text{if } \tau_1 = e \\ v_{\tau_1} & \text{else} \end{Bmatrix} - \begin{Bmatrix} t & \text{if } \tau_2 = e \\ v_{\tau_2} & \text{else} \end{Bmatrix} \bowtie X$	(C26) $\mathsf{complete}'_i \Rightarrow \begin{Bmatrix} t' & \text{if } \tau_1 = e \\ v'_{\tau_1} & \text{else} \end{Bmatrix} - \begin{Bmatrix} t' & \text{if } \tau_2 = e \\ v'_{\tau_2} & \text{else} \end{Bmatrix} \bowtie X$

$$
\begin{aligned}
\varphi_3(b,\tau_1) &:\equiv b < \tau_1 \wedge \Box_{[b,\tau_1)}\phi_A \wedge \tau_1 = p \\
\varphi_4(\tau_1,e) &:\equiv \tau_1 < e \wedge \Box_{[\tau_1,e)}\phi_B \wedge \tau_1 = p \\
\varphi_5(b,\tau_2) &:\equiv b < \tau_2 \wedge \Box_{[b,\tau_2)}\phi_C \\
\varphi_6(\tau_2,\tau_3) &:\equiv \exists\tau_3 \bullet \varphi_{10}(\tau_2,\tau_3) \wedge \varphi_{11}(\tau_3,e) \\
\varphi_7(b,\tau_1) &:\equiv b < \tau_1 \wedge \Box_{[b,\tau_1)}\phi_A \\
\varphi_8(b,\tau_1) &:\equiv p \leq \tau_1 \\
\varphi_9(b,\tau_1) &:\equiv p \geq \tau_1 \\
\varphi_{10}(\tau_2,\tau_3) &:\equiv \tau_2 < \tau_3 \wedge \Box_{[\tau_2,\tau_3)}\phi_D \wedge \tau_2 \leq p \leq \tau_3 \\
\varphi_{11}(\tau_3,e) &:\equiv \tau_3 < e \wedge \Box_{[\tau_3,e)}\phi_E \\
\varphi_{12}(\tau_2,\tau_3) &:\equiv \tau_2 < \tau_3 \wedge \Box_{[\tau_2,\tau_3)}\phi_D \\
\varphi_{13}(\tau_2,\tau_3) &:\equiv \tau_2 \leq p \\
\varphi_{14}(\tau_2,\tau_3) &:\equiv p \leq \tau_3
\end{aligned}
$$

For each subformula φ_i ($i = 0, 1, \ldots, 14$) we introduce and initialization constraint I_i and a step constraint T_i:

$$I_0 :\equiv \mathsf{complete}_0 \Leftrightarrow (\mathsf{complete}_1 \wedge \mathsf{complete}_5)$$

Algorithm 5: The function CHART2BMC (continued)

else if $\varphi_i(b,e) \equiv \tau_1 - \tau_2 \bowtie X$ *with* $X \in \mathbb{R} \setminus \{0\}$ **then**
  ensure that there are real variables v_{τ_1} and v_{τ_2} for τ_1 and τ_2;
  $I_i :\equiv \mathsf{complete}_i \Rightarrow \begin{Bmatrix} t & \text{if } \tau_1 = e \\ v_{\tau_1} & \text{else} \end{Bmatrix} - \begin{Bmatrix} t & \text{if } \tau_2 = e \\ v_{\tau_2} & \text{else} \end{Bmatrix} \bowtie X$;
  $T_i :\equiv \mathsf{complete}'_i \Rightarrow \begin{Bmatrix} t' & \text{if } \tau_1 = e \\ v_{\tau_1} & \text{else} \end{Bmatrix} - \begin{Bmatrix} t' & \text{if } \tau_2 = e \\ v_{\tau_2} & \text{else} \end{Bmatrix} \bowtie X$;
end
$I :\equiv I \wedge I_i$; $T :\equiv T \wedge T_i$;
end
for each *time variable* $\tau \in V_T$ **do**
if v_τ *is used in* I *or* T **then**
  T : $equiv T \wedge v'_\tau = v_\tau$;
  if past_τ *is used in* I *or* T **then**
    $I :\equiv I \wedge (\mathsf{past}_\tau \Leftrightarrow v_\tau = 0)$;
    $T :\equiv T \wedge (\mathsf{past}'_\tau \Rightarrow v_\tau \geq t') \wedge (\neg\mathsf{past}_\tau \Rightarrow v_\tau \leq t')$;
  end
else if past_τ *is used in* I *or* T **then**
  $T :\equiv T \wedge (\mathsf{past}_\tau \Rightarrow \mathsf{past}'_\tau)$;
end
end
return $(I, T, \mathsf{complete}_0)$;
end

$$T_0 :\equiv \mathsf{complete}'_0 \Leftrightarrow (\mathsf{complete}'_1 \wedge \mathsf{complete}'_5)$$
$$I_1 :\equiv (\mathsf{complete}_1 \Leftrightarrow \mathsf{complete}_3) \wedge (\mathsf{past}_{\tau_1} \Rightarrow \mathsf{complete}_4)$$
$$T_1 :\equiv (\mathsf{complete}'_1 \Leftrightarrow \mathsf{complete}'_3) \wedge (\mathsf{past}'_{\tau_1} \Rightarrow (\mathsf{past}_{\tau_1} \vee \mathsf{complete}'_4))$$
$$I_2 :\equiv (\mathsf{complete}_2 \Leftrightarrow \mathsf{complete}_6) \wedge (\mathsf{past}_{\tau_2} \Rightarrow \mathsf{complete}_5)$$
$$T_2 :\equiv (\mathsf{complete}'_2 \Leftrightarrow \mathsf{complete}'_6) \wedge (\mathsf{past}'_{\tau_2} \Rightarrow (\mathsf{past}_{\tau_2} \vee \mathsf{complete}'_5))$$
$$I_3 :\equiv \mathsf{complete}_3 \Leftrightarrow (\mathsf{complete}_7 \wedge \mathsf{complete}_8 \wedge \mathsf{complete}_9)$$
$$T_3 :\equiv \mathsf{complete}'_3 \Leftrightarrow (\mathsf{complete}'_7 \wedge \mathsf{complete}'_8 \wedge \mathsf{complete}'_9)$$
$$I_4 :\equiv \mathsf{ok}_4 \wedge \neg\mathsf{complete}_4$$
$$T_4 :\equiv (\mathsf{ok}'_4 \Leftrightarrow (\mathsf{past}_{\tau_1} \Rightarrow (\mathsf{ok}_4 \wedge tr(\phi_B)))) \wedge (\mathsf{complete}'_4 \Leftrightarrow (\mathsf{past}_{\tau_1} \wedge \mathsf{ok}'_4))$$
$$I_5 :\equiv \mathsf{ok}_5 \wedge \neg\mathsf{complete}_5$$
$$T_5 :\equiv (\mathsf{ok}'_5 \Leftrightarrow (\mathsf{past}_b \Rightarrow (\mathsf{ok}_5 \wedge tr(\phi_C)))) \wedge (\mathsf{complete}'_5 \Leftrightarrow (\mathsf{past}_b \wedge \mathsf{ok}'_5))$$
$$I_6 :\equiv (\mathsf{complete}_6 \Leftrightarrow \mathsf{complete}_{11}) \wedge (\mathsf{past}_{\tau_3} \Rightarrow \mathsf{complete}_{10})$$
$$T_6 :\equiv (\mathsf{complete}'_6 \Leftrightarrow \mathsf{complete}'_{11}) \wedge (\mathsf{past}'_{\tau_3} \Rightarrow (\mathsf{past}_{\tau_3} \vee \mathsf{complete}'_{10}))$$
$$I_7 :\equiv \mathsf{ok}_7 \wedge \neg\mathsf{complete}_7$$
$$T_7 :\equiv (\mathsf{ok}'_7 \Leftrightarrow (\mathsf{past}_b \Rightarrow (\mathsf{ok}_7 \wedge tr(\phi_A)))) \wedge (\mathsf{complete}'_7 \Leftrightarrow (\mathsf{past}_b \wedge \mathsf{ok}'_7))$$
$$I_8 :\equiv \mathsf{complete}_8 \Rightarrow \mathsf{past}_{\tau_1}$$

$$
\begin{aligned}
T_8 &:\equiv \mathsf{complete}'_8 \Rightarrow \mathsf{past}'_{\tau_1} \\
I_9 &:\equiv \mathit{true} \\
T_9 &:\equiv \mathsf{complete}'_8 \Rightarrow \neg\mathsf{past}_{\tau_1} \\
I_{10} &:\equiv \mathsf{complete}_{10} \Leftrightarrow (\mathsf{complete}_{12} \wedge \mathsf{complete}_{13} \wedge \mathsf{complete}_{14}) \\
T_{10} &:\equiv \mathsf{complete}'_{10} \Leftrightarrow (\mathsf{complete}'_{12} \wedge \mathsf{complete}'_{13} \wedge \mathsf{complete}'_{14}) \\
I_{12} &:\equiv \mathsf{ok}_{12} \wedge \neg\mathsf{complete}_{12} \\
T_{12} &:\equiv (\mathsf{ok}'_{12} \Leftrightarrow (\mathsf{past}_{\tau_2} \Rightarrow (\mathsf{ok}_{12} \wedge tr(\phi_C)))) \wedge (\mathsf{complete}'_{12} \Leftrightarrow (\mathsf{past}_{\tau_2} \wedge \mathsf{ok}'_{12})) \\
I_{13} &:\equiv \mathsf{complete}_{13} \Rightarrow (\mathsf{past}_p \Rightarrow \mathsf{past}_{\tau_2}) \\
T_{13} &:\equiv (\mathsf{complete}'_{13} \Rightarrow (\mathsf{past}'_p \Rightarrow \mathsf{past}'_{\tau_2})) \wedge (\mathsf{complete}'_{13} \Leftrightarrow \mathsf{complete}_{13}) \\
I_{14} &:\equiv \mathsf{complete}_{14} \Rightarrow \mathsf{past}p \\
T_{14} &:\equiv \mathsf{complete}'_{14} \Rightarrow \mathsf{past}'_p
\end{aligned}
$$

For the time variables b, p, τ_1, and τ_2 we furthermore have to create transition constraints

$$T_{timevars} :\equiv (\mathsf{past}_b \Rightarrow \mathsf{past}'_b) \wedge (\mathsf{past}_{\tau_1} \Rightarrow \mathsf{past}'_{\tau_1}) \wedge (\mathsf{past}_{\tau_2} \Rightarrow \mathsf{past}'_{\tau_2}) \wedge (\mathsf{past}_{\tau_3} \Rightarrow \mathsf{past}'_{\tau_3})$$

The BMC problem consists of the following initialization and transition constraints:

$$
\begin{aligned}
I &:\equiv I_0 \wedge I_1 \wedge \cdots \wedge I_{14} \\
T &:\equiv T_0 \wedge T_1 \wedge \cdots \wedge T_{14} \wedge T_{timevars}
\end{aligned}
$$

6.3.2 Correctness

This section examines the correctness of the construction, respectively Algorithm 5. We formulate and prove correctness in two directions.

6.3.2.1 From trajectory to model

In order to avoid cluttered subscripts, we denote in the following variables belonging to a specific step i of an unrolled BMC script by a superscript instead, i.e., we denote the copy of some variable x_j that that belongs to step i by x_j^i.

The first theorem states that the BMC problem has a solution whenever the basic chart is satisfiable. We bind the Boolean variables b_{sv} to necessary conditions for satisfaction of the spatial views. The algorithm has to work with any weak approximation of spatial views, while the time points at which the spatial views are evaluated depend on the trajectory and the temporal structure of the basic chart. This leads to a somewhat nested quantification of time points, the Boolean variables b_{sv}^i, and the remaining variables in the unrolled BMC problem.

Theorem 5 (Correctness of Chart Encoding – existence of satisfying model)**.** *Assume some world model* WM*, a basic chart* BC*, a trajectory* $\pi \in \mathcal{T}(\mathsf{WM})$ *over some set* O_π *of objects, and a valuation* μ *such that* $\pi, \mu \models_{\mathsf{WM}} \mathsf{BC}$*. Let* φ_{BC} *be the MSRTL formula for* BC*, and* $\mathsf{BMC} := \textsc{Chart2BMC}(\mathsf{BC})$ *be the BMC problem constructed for* BC *according to Algorithm 5. Let* $n \in \mathbb{N}$ *such that* φ_{BC} *has at most* $n+1$ *free and bound time variables, and denote by* BMC_n *the unrolling of* BMC *for* n *steps. Let* Σ *be the signature of* BMC_n*.*

Then, there exists a sequence $t_0 < t_1 < \cdots < t_n$ of time points such that, for all partial Σ-models $\mathcal{M}'$ that assign values only to copies $b^0_{sv}, b^1_{sv}, b^{n-1}_{sv}$ of the Boolean variable b_{sv} for each spatial view sv in BC, holds: If $\mathcal{M}'(b^i_{sv}) = true$ for all spatial views sv in BC and all $i = 0, 1, \ldots, n-1$ with $\pi, \mu \oplus \{b \mapsto t_i, e \mapsto t_{i+1}\} \Box_{[b,e)} \phi_{sv}$, then there exists a Σ-model $\mathcal{M} \supseteq \mathcal{M}'$ such that $\mathcal{M} \models \mathsf{BMC}_n$.

Proof. We first show how the time points $t_0, t_1, \ldots, t_n$ are chosen, and how a partial model $\mathcal{M}'$ that fulfills the assumptions stated in Theorem 5 above can be completed to a model $\mathcal{M} \supseteq \mathcal{M}'$ such that $\mathcal{M} \models \mathsf{BMC}_n$. Recall that φ_{BC} is structured in a way that the existentially bound time variables do not occur under negations. So, we can turn φ_{BC} into an equisatisfiable formula $\tilde{\varphi}_{\mathsf{BC}}$ without bound time variables. It has the same structure as φ_{BC}, except that sub-formulae of the form $\exists \tau \bullet \varphi_1(b, \tau) \wedge \varphi_2(\tau, e)$ are now of the form $\varphi_1(b, \tau) \wedge \varphi_2(\tau, e)$ (we simply drop the quantifier, assuming w.l.o.g. no name clashes occur). Assume there exists a trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ and a valuation μ such that $\pi, \mu \models \tilde{\varphi}_{\mathsf{BC}}$. By assumptions, $\tilde{\varphi}_{\mathsf{BC}}$ has at most $n+1$ time variables, so we can construct a partial model $\mathcal{M}'$ such that $\mu(\tau) \in \{\mathcal{M}'(t_0), \ldots, \mathcal{M}'(t_n)\}$ for all time variables τ in $\tilde{\varphi}_{\mathsf{BC}}$. Furthermore, assume $\pi, \mu \models \Box_{[\mathcal{M}'(t_i), \mathcal{M}'(t_{i+1}))} \phi_{sv} \Rightarrow \mathcal{M}'(b_s v^i) = true$ for spatial views sv in BC and all $i = 0, \ldots, n-1$.

We extend $\mathcal{M}'$ to a complete model $\mathcal{M}$ that also satisfies the constraints we introduce during the translation of each formula $\varphi_k(b, e)$. For each time variable τ, we set

$$\mathcal{M}(\mathsf{past}^i_\tau) \Leftrightarrow \mathcal{M}(t_i) \geq \mu(\tau)$$

and

$$\mathcal{M}(v^0_\tau) = \mathcal{M}(v^1_\tau) = \cdots = \mathcal{M}(v^n_\tau) = \mu(\tau) \ .$$

Now complete $\mathcal{M}$ in a way that constraint (C1) as well as all the equivalences (C2 – C10) in the translation table are fulfilled. Note that this is well-defined and unique because the table contains exactly two equivalences for each variable: One in the initialization column (I) that determines its initial value, and one in the transition column (T) that is used to inductively calculate values for the following unroll steps. Furthermore, there are no cyclic dependencies. We start with (C1) which gives unique values for ok^0_k and $\mathsf{complete}^0_k$ for subformulae $\varphi_k(b, e)$ of the form $\Box_{[b,e)} \phi \wedge b < e$. Then we use (C6, C7) to successively derive values for each ok^i_k and $\mathsf{complete}^i_k$ for $i = 1, \ldots, n$ for subformulae of the same type. For timing constraints $\psi_k(e)$, we set $\mathcal{M}(\mathsf{complete}^i_k) = true$ iff[4] $\pi, \mu[e = \mathcal{M}(t_i)] \models \psi_k$. Now we walk down the syntax tree and use (C2–C4, C8–C10) to derive values for each $\mathsf{complete}^i_k$ for subformulae of the other forms.

Induction Hypothesis Now, we have to show by induction that for each sub-formula $\varphi_k(b, e)$ and step $i \in \{0, \ldots, n\}$ holds

$$\mathcal{M}(t_i) = \mu(e) \wedge \pi, \mu \models \varphi_k(b, e) \Longrightarrow \mathcal{M}(\mathsf{complete}^i_k) = true \tag{6.2}$$

and that constraints (C5, C11–C24) are fulfilled.

Induction Start The induction start are formulae of the form $\Box_{[b,e)} \phi \wedge b < e$ and timing constraints $\psi(e)$. We start with timing constraints $\psi_k(e)$. For these, Equation (6.2) is satisfied by the above construction (note that by definition $\mu[e = \mathcal{M}(t_i)] = \mu$ if

[4] $\mu[e = t_i]$ denotes valuation μ with the modification $\mu(e) = t_i$.

$\mu(e) = \mathcal{M}(t_i)$). For showing that (C13–C24) hold, assume that we set $\mathcal{M}(\mathsf{complete}_k) = \mathit{true}$, so $\pi, \mu[e = \mathcal{M}(t_i)] \models \psi_k$ holds. Recall that we assume $\mu(\tau) \in \{\mathcal{M}(t_0), \ldots, \mathcal{M}(t_n)\}$ and $\mathcal{M}(\mathsf{past}^i_\tau) \Leftrightarrow \mu(\tau) \leq \mathcal{M}(t_i)$. So, we can conclude the following implications:

- C13, C19: $\mu(\tau_1) \leq \mu(\tau_2) \wedge \mathcal{M}(\mathsf{past}^i_{\tau_2}) \Rightarrow \mu(\tau_1) \leq \mu(\tau_2) \leq \mathcal{M}(t_i) \Rightarrow \mathcal{M}(\mathsf{past}^i_{\tau_1})$
- C14: $\mathcal{M}(t_0) \leq \mu(\tau_1) < \mu(\tau_2) \Rightarrow \mathcal{M}(\neg\mathsf{past}^0_{\tau_2})$
- C21: $\mu(\tau_1) < \mu(tau_2) \wedge \mathcal{M}(\mathsf{past}^{i+1}_{\tau_2}) \Rightarrow \mu(\tau_1) < \mu(tau_2) \leq \mathcal{M}(t_{i+1}) \Rightarrow \mu(\tau_1) \leq \mathcal{M}(t_i) \Rightarrow \mathcal{M}(\mathsf{past}^i_{\tau_2})$
- C15, C22: $\mu(\tau) \leq \mu(e) = \mathcal{M}(t_i) \Rightarrow \mathcal{M}(\mathsf{past}^i_\tau)$
- C16, $\mathcal{M}(t_0) \leq \mu(\tau) < \mu(e) \Rightarrow \mu[e = \mathcal{M}(t_0)](e) = \mathcal{M}(t_0) \leq \mu(\tau) \Rightarrow \pi, \mu[e = \mathcal{M}(t_0)] \not\models \tau < e$
- C23: $\mu(\tau) < \mu(e) = \mathcal{M}(t_{i+1}) \Rightarrow \mu(\tau) \leq \mathcal{M}(t_i) \Rightarrow \mathcal{M}(\mathsf{past}^i_\tau)$
- C24: $\mu(\tau) \geq \mu(e) = \mathcal{M}(t_{i+1}) > \mathcal{M}(t_i) \Rightarrow \mu(\tau) > \mathcal{M}(t_i) \Rightarrow \mathcal{M}(\neg\mathsf{past}^i_\tau)$
- C17, C25: $\mu(\tau) > \mu(e) = \mathcal{M}(t_{i+1}) \Rightarrow \mu(\tau) > \mathcal{M}(t_i) \Rightarrow \mathcal{M}(\neg\mathsf{past}^i_\tau)$

Furthermore, recall that for time variables $\mathcal{M}(v^i_\tau) = \mu(\tau)$, so $\mathcal{M}(\psi[e\backslash t_i, \tau\backslash v^i_\tau \mid \tau \in V_T \setminus \{e\}]) \Leftrightarrow \pi, \mu[e = \mathcal{M}(t_i)] \models \psi \Leftrightarrow \pi, \mu \models \psi$ by definition.

For formulae of the form $\Box_{[b,e)}\phi_{sv} \wedge b < e$, we show that inductively $\mathcal{M}(\mathsf{ok}^{i+1}) = \mathit{true}$ if $\pi, \mu \models \varphi(b, e)$ and $\mathcal{M}(t_i) < \mu(e)$: By construction $\mathcal{M}(\mathsf{ok}^0) = \mathit{true}$.

1. If $\mathcal{M}(t_i) < \mu(b)$ then by construction $\mathcal{M}(\mathsf{past}^i_b) = \mathit{false}$
2. If $\mu(b) \leq \mathcal{M}(t_i) < \mu(e)$ then the following applies: Because $\mu(e) \in \{\mathcal{M}(t_0), \ldots \mathcal{M}(t_n)\}$ and $\mathcal{M}(t_0) < \cdots < \mathcal{M}(t_n)$, it is $\mathcal{M}(t_{i+1}) \leq \mu(e)$. So, $\pi, \mu \models \Box_{[b,e)}\phi_{sv}$ implies $\pi, \mu \models \Box_{[\mathcal{M}(t_i),\mathcal{M}(t_{i+1}))}\phi_{sv}$, and, as a consequence, by assumption $\mathcal{M}(b^i_{sv}) = \mathit{true}$.

So, by (C6), in both cases $\mathcal{M}(\mathsf{ok}^{i+1}) = \mathit{true}$. Because of $b < e$ we have $\mathcal{M}(\mathsf{past}^i_b) = \mathit{true}$ if $\mathcal{M}(t_{i+1}) = \mu(e)$, so, by (C7), $\mathcal{M}(\mathsf{complete}^{i+1}) = \mathit{true}$.

Induction Step For the induction step, the crucial case are (sub-)formulae of the form $\varphi_1(b, \tau) \wedge \varphi_2(\tau, e)$ (before dropping quantifiers, of the form $\exists\tau \bullet \varphi_1(b, \tau) \wedge \varphi_2(\tau, e)$). Note that the structure of the formulae implies $\mu(b) < \mu(\tau) < \mu(e)$. Recall that we set $\mathcal{M}(\mathsf{past}^i_\tau) = \mathit{true}$ iff $\mathcal{M}(t_i) \geq \mu(\tau)$. Because $\mathcal{M}(t_{i+1}) > \mathcal{M}(t_i)$, the step constraint $\mathsf{past}_\tau \Rightarrow \mathsf{past}'_\tau$ is satisfied. The other way round, if $\mathcal{M}(\mathsf{past}^{i+1}_\tau) = \mathit{true}$ then one of the following is true:

- $\mathcal{M}(t_{i+1}) > \mu(\tau)$ and therefore $\mathcal{M}(t_i) \geq \mu(\tau)$ and $\mathcal{M}(\mathsf{past}^i_\tau) = \mathit{true}$, or
- $\mathcal{M}(t_{i+1}) = \mu(\tau)$ and therefore by induction hypothesis $\mathcal{M}(\mathsf{complete}^{i+1}_1) = \mathit{true}$.

In both cases, $\mathsf{past}'_\tau \Rightarrow (\mathsf{past}_\tau \vee \mathsf{complete}_1)$ is satisfied. □

6.3.2.2 From model to trajectory

The second theorem states that satisfaction of the constructed BMC problem is a sufficient condition for satisfaction of the temporal structure, provided the variables b_{sv} are bound to sufficient conditions for the spatial views.

Theorem 6 (Correctness of Chart Encoding – existence of satisfying trajectory)**.** *Assume some world model* WM*, a basic chart* BC*, a trajectory* $\pi \in \mathcal{T}(\mathsf{WM})$ *over some set* O_π *of objects, and a valuation* μ *such that* $\pi, \mu \models_{\mathsf{WM}} \mathsf{BC}$*. Let* $\mathsf{BMC} := \textit{CHART2BMC}(\mathsf{BC})$ *be the BMC problem constructed for* BC *according to Algorithm 5, and denote by* BMC_n *its unrolling for some* $n \in \mathbb{N}$ *steps. Then, the following holds: If there exists a model* $\mathcal{M} \models \mathsf{BMC}_n$ *such that* $\mathcal{M}(b^i_{sv}) = \mathit{true} \Rightarrow \pi, \mu \models \Box_{[\mathcal{M}'(t_i), \mathcal{M}'(t_{i+1}))} \phi_{sv}$ *for all spatial views* sv *in* BC *and* $i = 0, \ldots, n-1$*, then there exists some valuation* μ' *such that* $\pi, \mu' \models_{\mathsf{WM}} \varphi_{\mathsf{BC}}$*.*

Proof. Assume a satisfying model $\mathcal{M} \models \mathsf{BMC}_n$ and a trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ and valuation μ' such that $\pi, \mu' \models \Box_{[\mathcal{M}(t_i), \mathcal{M}(t_{i+1}))} \phi_{sv} \Rightarrow \mathcal{M}(b^i_{sv}) = \mathit{true}$ for all ϕ and $i = 0, \ldots, n-1$.

First, we extend μ' to the valuation $\mu \supseteq \mu'$ by setting

$$\mu(\tau) := \min\{\mathcal{M}(t_i) \mid i \in \{0, \ldots, n\}, \mathcal{M}(\mathsf{past}^i_\tau) = \mathit{true}\} \cup \{\mathcal{M}(t_n) + 1\}$$

respectively $\mu(\tau) = \mathcal{M}(v^i_\tau)$. For the end time variable e_{BC}, we set $\mu(e_{\mathsf{BC}}) = \mathcal{M}(t_n)$ (note that by the special treatment of end time variables, neither $v_{e_{\mathsf{BC}}}$ nor $\mathsf{past}_{e_{\mathsf{BC}}}$ is used in the BMC problem). Note that this is consistent in the sense that $\mathcal{M}(\mathsf{past}^i_\tau) \Leftrightarrow (\mu(\tau) \leq \mathcal{M}(t_i))$ holds (this is induced by the constraints $\mathsf{past}_\tau \Rightarrow \mathsf{past}'_\tau$, $\mathsf{past}'_\tau \Rightarrow v_\tau \geq t'$, and $\neg\mathsf{past}_\tau \Rightarrow v_\tau \leq t'$).

We have to show that

$$\mathcal{M}(t_i) = \mu(e) \wedge \mathcal{M}(\mathsf{complete}^i_k) = \mathit{true} \Longrightarrow \pi, \mu \models \varphi_k(b, e)$$

for any sub-formula $\varphi_k(b, e)$ of φ_{BC}. In the following, we choose the step index i such that $\mathcal{M}(t_i) = \mu(e)$ and assume $\mathcal{M}(\mathsf{complete}^i_k) = \mathit{true}$.

The induction start are again sub-formulae of the form $\varphi_k(b, e) \equiv \Box_{[b,e)} \phi \wedge b < e$ (or a timing constraint which is considered below). Assume that $\mathcal{M}(t_i) = \mu(e)$ and $\mathcal{M}(\mathsf{complete}^i_k) = \mathit{true}$ for some i. This implies by construction $i > 0$, $\mathcal{M}(\mathsf{ok}^i_k) = \mathit{true}$, and $\mathcal{M}(\mathsf{past}^{i+1}_b) = \mathit{true}$. The latter implies that $\mu(b) < \mu(e)$. Look at the constraint $\mathsf{ok}' \Leftrightarrow (\mathsf{past}_b \Rightarrow (\mathsf{ok}_k \wedge b_{sv}))$ (C6). Because $\mathcal{M}(\mathsf{past}^j_b) \Leftrightarrow \mathcal{M}(t_j) \geq \mu(b)$, we know from backward induction that $\mathcal{M}(\mathsf{ok}^i_k) = \mathit{true}$ implies $\mathcal{M}(b^j_{sv}) = \mathit{true}$ for all $j < i$ with $\mathcal{M}(t_j) \geq \mu(b)$. This implies by assumption that $\pi, \mu \models \Box_{[\mathcal{M}(t_j), \mathcal{M}(t_{j+1}))}$ for all j with $\mu(b) \leq \mathcal{M}(t_j) < \mathcal{M}(t_{j+1}) \leq \mu(e)$ and therefore $\pi, \mu \models \Box_{[b,e)} \phi_{sv} \wedge b < e$.

For $\varphi_k(b, e)$ that are timing constraints $\tau_1 - \tau_2 \bowtie X$, the induction hypothesis holds for the different forms of timing constraints as follows:

- $\tau \leq e$: By (C15, C22) holds $\mathcal{M}(\mathsf{past}^i_\tau) = \mathit{true}$ which implies $\mu(\tau) \leq \mathcal{M}(t_i) = \mu(e)$
- $\tau < e$: Constraint (C16) implies $i > 0$ and by (C23) holds $\mathcal{M}(\mathsf{past}^{i-1}_\tau) = \mathit{true}$ which implies $\mu(\tau) \leq \mathcal{M}(t_{i-1}) < \mu(e)$
- $\tau \geq e$: If $i = 0$ then trivially $\mu(e) = \mathcal{M}(t_0) \leq \mu(\tau)$ by definition. Otherwise, by (C24) holds $\mathcal{M}(\mathsf{past}^{i-1}_\tau) = \mathit{false}$ which implies $\mu(\tau) > \mathcal{M}(t_{i-1})$. Because $\mu(\tau) \in \{\mathcal{M}(t_0), \mathcal{M}(t_1), \ldots, \mathcal{M}(t_n), \mathcal{M}(t_n) + 1\}$ with $\mathcal{M}(t_0) < \mathcal{M}(t_1) < \ldots \mathcal{M}(t_n)$, it is $\mu(\tau) \geq \mathcal{M}(t_i) = \mu(e)$.

- $\tau > e$: By (C17, C25) holds $\mathcal{M}(\mathsf{past}^i_\tau) = \mathit{false}$ which implies $\mu(\tau) > \mathcal{M}(t_i) = \mu(e)$.
- $\tau_1 \leq \tau_2$ or $\tau_1 < \tau_2$ (with $\tau_1, \tau_2 \in V_T \setminus \{e\}$): By the structure of TSCs, constraints of this form occur in our MSRTL formula only as part of a larger constraint $b \leq \tau_1 < \cdots < \tau_m \leq e$. We show inductively that $\mu(\tau_\ell) \leq \mathcal{M}(t_i)$ for $\ell = 1, \ldots, m$. For $\ell = m$ this holds because $\pi, \mu \models \tau_m \leq e$ as shown above. Assuming we have shown $\mu(\tau_{\ell+1}) \leq \mathcal{M}(t_i)$ for some arbitrary but fixed $\ell < m$, there is some index j such that $\mu(\tau_{\ell+1}) = \mathcal{M}(t_j)$ and $\mathcal{M}(\mathsf{past}^j_{\tau_{\ell+1}}) = \mathit{true}$. Note that (C20) inductively implies $\mathcal{M}(\mathsf{complete}^j_k) = \mathcal{M}(\mathsf{complete}^i_k) = \mathit{true}$. By (C19, C21) it is $\mathcal{M}(\mathsf{past}^j_{\tau_\ell}) = \mathit{true}$ (respectively $\mathcal{M}(\mathsf{past}^{j-1}_{\tau_\ell}) = \mathit{true}$ and, by (C14), $j > 0$). As a consequence, $\mu(\tau_\ell) \leq \mathcal{M}(t_j) = \mu(\tau_{\ell+1}) \leq \mathcal{M}(t_i)$, respectively $\mu(\tau_\ell) \leq \mathcal{M}(t_{j-1}) < \mathcal{M}(t_j) = \mu(\tau_{\ell+1}) \leq \mathcal{M}(t_i)$.
- For other timing constraints, by (C18, C26), holds

$$\mathcal{M} \models \begin{Bmatrix} t_i & \text{if } \tau_1 = e \\ v^i_{\tau_1} & \text{else} \end{Bmatrix} - \begin{Bmatrix} t_i & \text{if } \tau_2 = e \\ v^i_{\tau_2} & \text{else} \end{Bmatrix} \bowtie X \ .$$

  Because $\mathcal{M}(v^i_\tau) = \mu(\tau)$ (for any time variable τ where v_τ is used in the BMC problem) and $\mu(e) = \mathcal{M}(t_i)$, it holds $\pi, \mu \models \psi$.

The crucial case for the induction step are again sub-formulae of the form $\varphi_k(b, e) = \exists \tau \bullet \varphi_\ell(b, \tau) \wedge \varphi_m(\tau, e)$. Choose the index j with $\mathcal{M}(t_j) = \mu(\tau)$. Note that $\mathcal{M}(\mathsf{past}^j_\tau) = \mathit{true}$. If $j = 0$ then the initialization constraint $\mathsf{past}_\tau \Rightarrow \mathsf{complete}_1$ implies $\mathcal{M}(\mathsf{complete}^0_1) = \mathit{true}$, otherwise $\mathcal{M}(\mathsf{past}^{j-1}_\tau) = \mathit{false}$ and the step constraint $\mathsf{past}'_\tau \Rightarrow (\mathsf{past}_\tau \vee \mathsf{complete}_\ell)$ (C12) implies $\mathcal{M}(\mathsf{complete}^j_\ell) = \mathit{true}$. With the induction hypothesis, in both cases $\pi, \mu \models \varphi_\ell(b, \tau)$. Because of the constraint $\mathsf{complete}^i_k \Leftrightarrow \mathsf{complete}^i_m$ (C4, C10) we know by induction hypothesis that $\pi, \mu \models \varphi_m(\tau, e)$. As a consequence, $\pi, \mu \models \varphi(b, e)$. □

6.4 Encoding of Spatial Views

This section describes the encoding of spatial views as SMT formulae. In the first subsection, the original semantics of spatial views given in [37, 38] is revisited and the technical realization described. The other subsections describe how the semantics is translated to SMT formulae. This is presented first in a general fashion that leads to a necessary condition for fulfillment of invariant nodes. Finally, the general form is refined to give sufficient conditions for invariant nodes considering a simple form of a single track dynamics model.

6.4.1 Simple encoding

The semantics of spatial views given algorithmicly in Section 2.4.2 is more or less used directly to derive an SMT formula that encodes a spatial view. In order to keep the SMT formula as simple as possible, an optimized version of `topologicalRelations` is used, given in Algorithm 6 and denoted by `topologicalRelations'`. In the presentation of the encoding here, the the resulting MSRTL formula is mapped to an SMT formula as given later on in Construction 1 (the prototype implementation, however, omits constructing the MSRTL formula and generates the SMTLib code directly).

The topological relations constructed by Algorithm 6 are equivalent to the ones given by the original definition of `topologicalRelation` (Definition 17).

Algorithm 6: $\texttt{topologicalRelations}'(f)$

Input: A frame f
Output: A constraint $\mathtt{T}_f$

$\mathtt{T}_f := \mathit{true}$;
Sort the set $\mathscr{A}(f) := \bigcup_{s \in \mathcal{S}_f \cup \mathcal{S}_{spans(f)}} \mathscr{A}(s)$ by the x positions of anchors, i.e., enumerate $\mathscr{A}(f)$ as $(s_0.\alpha_0, s_1.\alpha_1, \ldots, s_{|\mathscr{A}(f)|-1}.\alpha_{|\mathscr{A}(f)|-1})$ such that $s_0.\alpha_0.x \leq s_1.\alpha_1.x \leq \cdots \leq s_{|\mathscr{A}(f)|-1}.\alpha_{|\mathscr{A}(f)|-1}.x$;
for each $i = 1, \ldots, |\mathscr{A}(f)| - 1$ **do**
  if $s_{i-1} \neq s_i \vee s_i \notin \mathcal{S}_{spans(f)}$ **then**
    $\mathtt{T}_f := \mathtt{T}_f \wedge \begin{cases} \mathtt{A}_x(s_{i-1}.\alpha_{i-1}) < \mathtt{A}_x(s_i.\alpha_i) & \text{if } s_{i-1}.\alpha_{i-1}.x < s_i.\alpha_i.x \\ \mathtt{A}_x(s_{i-1}.\alpha_{i-1}) = \mathtt{A}_x(s_i.\alpha_i) & \text{else} \end{cases}$
  end
end
Sort the set $\mathscr{A}(f)$ by the y positions of anchors, i.e., enumerate $\mathscr{A}(f)$ as $(s_0.\alpha_0, s_1.\alpha_1, \ldots, s_{|\mathscr{A}(f)|-1}.\alpha_{|\mathscr{A}(f)|-1})$ such that $s_0.\alpha_0.y \leq s_1.\alpha_1.y \leq \cdots \leq s_{|A_y(f)|-1}.\alpha_{|A_y(f)|-1}.y$;
for each $i = 1, \ldots, |\mathscr{A}(f)| - 1$ **do**
  if $s_{i-1} \neq s_i \vee s_i \notin \mathcal{S}_{spans(f)}$ **then**
    $\mathtt{T}_f := \mathtt{T}_f \wedge \begin{cases} \mathtt{A}_y(s_{i-1}.\alpha_{i-1}) < \mathtt{A}_y(s_i.\alpha_i) & \text{if } s_{i-1}.\alpha_{i-1}.y < s_i.\alpha_i.y \\ \mathtt{A}_y(s_{i-1}.\alpha_{i-1}) = \mathtt{A}_y(s_i.\alpha_i) & \text{else} \end{cases}$
  end
end
return $\mathtt{T}_f$

Lemma 1 (Correctness of Algorithm 6)**.** *For any frame f holds*

$$\texttt{topologicalRelations}(f) \equiv \texttt{topologicalRelations}'(f)$$

provided that, if f is spanned by some box s, for the box bounds holds $\underline{x}_{s.id} \leq \overline{x}_{s.id} \wedge \underline{y}_{s.id} \leq \overline{y}_{s.id}$.

Note, that $\underline{x}_{s.id} \leq \overline{x}_{s.id} \wedge \underline{y}_{s.id} \leq \overline{y}_{s.id}$ for the spanning box is guaranteed by the topological relations in the containing frame.

Proof. All atoms in the conjunction on the right-hand side of the equivalence also occur on the left-hand side, either exactly or in an equivalent form, e.g., $\mathtt{A}_x(s.\alpha) > \mathtt{A}_x(s'.\alpha')$ instead of $\mathtt{A}_x(s'.\alpha') < \mathtt{A}_x(s.\alpha)$, or two atoms $\mathtt{A}_x(s.\alpha) \leq \mathtt{A}_x(s'.\alpha')$ and $\mathtt{A}_x(s.\alpha) \geq \mathtt{A}_x(s'.\alpha')$ instead of $\mathtt{A}_x(s.\alpha) = \mathtt{A}_x(s'.\alpha')$. So, the left-hand side implies the right-hand side.

Now we proof the opposite direction. Index the symbols and anchors in ascending x order, i.e., $s_i.\alpha_i.x \leq s_j.\alpha_j.x$ for $i < j$. Assume that the left-hand side contains some atom $\mathtt{A}_x(s_i.\alpha_i) \bowtie \mathtt{A}_x(s_j.\alpha_j)$ with $\bowtie \in \{<, \leq, \geq, >\}$. Note, that this implies $s_i.\alpha_i.x \bowtie s_j, \alpha_j.x$. If $s_i = s_j \in \mathcal{S}_{spans(f)}$ is the spanning box, then the atom is satisfied by assumption. Otherwise, if $\bowtie \in \{<, \leq\}$, then $i < j$ and Algorithm 6 has produced constraints $\mathtt{A}_x(s_i.\alpha_i) \lhd_1 \mathtt{A}_x(s_{i+1}.\alpha_{i+1}) \lhd_2 \cdots \lhd_{j-i} \mathtt{A}_x(s_j.\alpha_j)$ with $\lhd_1, \ldots, \lhd_{j-i} \in \{=, <\}$. If $\bowtie \in \{\geq, >\}$, then $j < i$ and Algorithm 6 has produced a sequence $\mathtt{A}_x(s_j.\alpha_j) \lhd_1 \mathtt{A}_x(s_{j+1}.\alpha_{j+1}) \lhd_2 \cdots \lhd_{i-j} \mathtt{A}_x(s_i.\alpha_i)$ in inverse direction. If $\bowtie \in \{<, >\}$, then at least one of the $\lhd_1, \ldots, \lhd_{|j-i|}$ is $<$. In any

case, satisfaction of the constraint sequence implies satisfaction of $\mathtt{A}_x(s_i.\alpha_i) \bowtie \mathtt{A}_x(s_j.\alpha_j)$ by transitivity of $\leq$ and $<$. This is analogous for the y constraints. □

The remaining part of the encoding technique is quite simple. One takes the result ϕ of building the semantics of the spatial view according to Definition 12 and translates it recursively into a constraint $tr(\phi)$ by replacing Boolean, arithmetic, and comparison operators to their counterparts in the signature for the SMT solver. By "counterpart" we mean the function whose interpretation, in the theory of real arithmetics, matches the semantics of the operator in MSRTL (for simplicity, we use the same function symbol f for arithmetic and comparison operators in both the solver theory as well as the MSRTL formula). We will define three versions of tr, being a necessary (i.e. weak approximation) condition tr_n, a sufficient (i.e. strong approximation) condition tr_s, and a special version tr_c. The three differ in the handling of existentially quantified object variables only. The first version, tr_n, is used in this chapter to produce the necessary conditions, but needs to be replaced by its counterpart tr_s under an odd number of negations. The latter, tr_c, is similar to tr_s except that it remains the same also under negations. The subscript c in tr_c stands for *closed world* and refers to the special case where we know about all existing objects at translation time (we will denote the set of known objects by V_O and add it as a second parameter to tr). It is used to produce sufficient constraints in Chapter 7.

The structure of the produced SMT formula is analogous to the MSRTL expression that serves as input to the construction. Quantified variables of type $\mathbb{R}$ can be translated straight forward into the SMT formula, and are to be removed by quantifier elimination in a post-processing step. The crucial part is then the translation of object variables and attribute access. The idea, however, is quite simple: For each attribute $o.a$ of each object variable introduce a variable $\mathtt{var}_{o.a}$ and use this to encode variable access; for object variables from the bulletin board, these are free variables; for attributes of existentially quantified object variables, these are existentially bound variables.

Construction 1 (Simple encoding of invariants (tr_n, tr_c, tr_s))**.** *Given some world model* WM *and a set* V_O *of object variables, define the set*

$$X := \{\mathtt{var}_{o.a} \mid o \in V_O, a \in \mathcal{A}(\sigma(o))\}$$

of free variables. The family of constructions $tr_\square$*, for* $\square \in \{n, s, c\}$*, translate an MSRTL predicate* ϕ *with free object variables* $Free(\phi) \subseteq V_O$ *into an SMT formula* $tr_\square(\phi, V_O)$ *over* X *inductively as follows (we leave out the parameter* V_O *for readability):*

$$tr_\square(o.a) := \mathtt{var}_{o.a} \quad \textit{for attribute access} \tag{6.3a}$$
$$tr_\square(x) := x \quad \textit{for real-valued variables } x \tag{6.3b}$$
$$tr_\square(c) := c \quad \textit{for constants } c \in \mathbb{R} \cup \{\textit{true}, \textit{false}\} \tag{6.3c}$$
$$tr_\square(f(\psi_1, \ldots, \psi_n)) := f(tr_\square(\psi_1), \ldots, tr_\square(\psi_n)) \quad \textit{for function symbols } f \tag{6.3d}$$
$$tr_\square(p(\psi_1, \ldots, \psi_n)) := p(tr_\square(\psi_1), \ldots, tr_\square(\psi_n)) \quad \textit{for predicate symbols } p \tag{6.3e}$$
$$tr_\square(\phi_1 \wedge \phi_2) := tr_\square(\phi_1) \wedge tr_\square(\phi_2) \tag{6.3f}$$
$$tr_\square(\phi_1 \vee \phi_2) := tr_\square(\phi_1) \vee tr_\square(\phi_2) \tag{6.3g}$$
$$tr_\square(\exists x \in \mathbb{R} : \phi) := \exists \mathtt{var}_x \in \mathbb{R} \bullet tr_\square(\phi) \tag{6.3h}$$

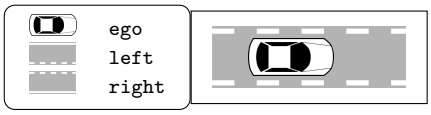


Figure 6.7: Spatial view and bulletin board for Example 6

$$tr_n(\neg\phi) := \neg tr_s(\phi) \tag{6.3i}$$
$$tr_s(\neg\phi) := \neg tr_n(\phi) \tag{6.3j}$$
$$tr_c(\neg\phi) := \neg tr_c(\phi) \tag{6.3k}$$
$$\begin{aligned} tr_\square(\exists o \in \mathtt{O} : \phi) := {} & \exists \mathtt{var}_{o.a_1} \in \mathbb{R} \cdots \exists \mathtt{var}_{o.a_{|\mathcal{A}(\mathcal{I}(\mathtt{O}))|}} \in \mathbb{R} \bullet tr_\square(\phi) \\ & \wedge \begin{cases} tr_n(\Phi_{\mathcal{I}(\mathtt{O})}(o)) & \text{for } \square = n \\ \bigvee_{o' \in V_O \mid type(o') \preceq \mathcal{I}(\mathtt{O})} \mathtt{var}_{o.a_1} = \mathtt{var}_{o'.a_1} & \text{else} \\ \quad \wedge \cdots \wedge \mathtt{var}_{o.a_{|\mathcal{A}(\mathcal{I}(\mathtt{O}))|}} = \mathtt{var}_{o'.a_{|\mathcal{A}(\mathcal{I}(\mathtt{O}))|}} & \end{cases} \\ & \text{for classes } \mathcal{I}(\mathtt{O}) \in \mathcal{C}, \text{ where } a_1, \ldots, a_{|\mathcal{A}(\mathcal{I}(\mathtt{O}))|} \in \mathcal{A}(\mathcal{I}(\mathtt{O})) \\ & \text{range over all attributes of } \mathcal{I}(\mathtt{O}) \end{aligned} \tag{6.3l}$$

Here, $\Phi_{\mathcal{I}(\mathtt{O})}(o)$ is the instantiation of the invariant $\Phi_{\mathcal{I}(\mathtt{O})}$ of class $\mathcal{I}(\mathtt{O}) \in \mathcal{C}$ (see Section 5.5) for object variable o.

Most operators and functions propagate naturally through the above construction, except that negations toggle between tr_n and tr_s. The difference between tr_n and tr_s (respectively tr_c) is the handling of quantified object variables. In all three cases, they are handled by introducing existentially bound variables for the object attributes. For tr_n, it is only asserted that they fulfill the world model knowledge $\Phi_{\mathcal{I}(\mathtt{O})}$ that we have for objects of the encoded type. Intuitively, this can be read as "the existence of an object o of type $\mathcal{I}(\mathtt{O})$ with the constraints given in the MSRTL formula is not impossible". For tr_s and tr_c instead, it is asserted that the object o equals one of the objects denoted by V_O, which are known to exist in the scenario. This is done by setting the variables encoding attributes of o equal the attributes of one of these objects.

Example 6. *Assume the spatial view in Figure 6.7. It expresses that the ego vehicle is somewhere between the boundaries of some lane and translates to the MSRTL formula*

$$\phi \equiv \exists o \in \mathtt{Lane} : o.\underline{x} < \mathsf{ego}.\underline{x} < \mathsf{ego}.\overline{x} < o.\overline{x} \wedge o.\underline{y} < \mathsf{ego}.\underline{y} < \mathsf{ego}.\overline{y} < o.\overline{y}$$

We use the set $V_O = \{\mathsf{ego}, \mathsf{right}, \mathsf{left}\}$ of global object variables originating from the bulletin board. Furthermore, we assume that the Lane *type has an invariant $\Phi_{\mathcal{I}(\mathtt{Lane})} \equiv \overline{y} - \underline{y} = 3\,\mathrm{m}$ stating that all lanes are 3 m wide. Then, the necessary constraint for the spatial view is*

$$\begin{aligned} tr_n(\phi, V_O) \equiv \exists \mathtt{var}_{o.\underline{x}}, \mathtt{var}_{o.\overline{x}}, \mathtt{var}_{o.\underline{y}}, \mathtt{var}_{o.\overline{y}} \in \mathbb{R} : {} & \mathtt{var}_{o.\overline{y}} - \mathtt{var}_{o.\underline{y}} = 3 \\ & \wedge \mathtt{var}_{o.\underline{x}} < \mathtt{var}_{\mathsf{ego}.\underline{x}} < \mathtt{var}_{\mathsf{ego}.\overline{x}} < \mathtt{var}_{o.\overline{x}} \\ & \wedge \mathtt{var}_{o.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\overline{y}} < \mathtt{var}_{o.\overline{y}} \end{aligned}$$

which can be simplified to

$$\mathtt{var}_{\mathsf{ego}.\underline{x}} < \mathtt{var}_{\mathsf{ego}.\overline{x}} \wedge 0 < \mathtt{var}_{\mathsf{ego}.\overline{y}} - \mathtt{var}_{\mathsf{ego}.\underline{y}} < 3 \ .$$

With other words, the necessary constraint constructed for ego being between lane boundaries expresses that the width of ego must be smaller than the lane width.

The sufficient constraint is

$$\begin{aligned} tr_s(\phi, V_O) \equiv \exists \mathtt{var}_{o.\underline{x}}, \mathtt{var}_{o.\overline{x}}, \mathtt{var}_{o.\underline{y}}, \mathtt{var}_{o.\overline{y}} \in \mathbb{R} : \\ (\mathtt{var}_{o.\overline{x}} = \mathtt{var}_{\mathsf{left}.\overline{x}} \wedge \mathtt{var}_{o.\underline{x}} = \mathtt{var}_{\mathsf{left}.\underline{x}} \\ \wedge \mathtt{var}_{o.\overline{y}} = \mathtt{var}_{\mathsf{left}.\overline{y}} \wedge \mathtt{var}_{o.\underline{x}} = \mathtt{var}_{\mathsf{left}.\underline{y}} \\ \vee \mathtt{var}_{o.\overline{x}} = \mathtt{var}_{\mathsf{right}.\overline{x}} \wedge \mathtt{var}_{o.\underline{x}} = \mathtt{var}_{\mathsf{right}.\underline{x}} \\ \wedge \mathtt{var}_{o.\overline{y}} = \mathtt{var}_{\mathsf{right}.\overline{y}} \wedge \mathtt{var}_{o.\underline{x}} = \mathtt{var}_{\mathsf{right}.\underline{y}}) \\ \wedge \mathtt{var}_{o.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\underline{x}} < \mathtt{var}_{\mathsf{ego}.\overline{x}} < \mathtt{var}_{o.\overline{x}} \\ \wedge \mathtt{var}_{o.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\overline{y}} < \mathtt{var}_{o.\overline{y}} \end{aligned}$$

which simplifies to

$$\begin{aligned} & \mathtt{var}_{\mathsf{left}.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\underline{x}} < \mathtt{var}_{\mathsf{ego}.\overline{x}} < \mathtt{var}_{\mathsf{left}.\overline{x}} \\ \wedge\ & \mathtt{var}_{\mathsf{left}.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\overline{y}} < \mathtt{var}_{\mathsf{left}.\overline{y}} \\ \vee\ & \mathtt{var}_{\mathsf{right}.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\underline{x}} < \mathtt{var}_{\mathsf{ego}.\overline{x}} < \mathtt{var}_{\mathsf{right}.\overline{x}} \\ \wedge\ & \mathtt{var}_{\mathsf{right}.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\underline{y}} < \mathtt{var}_{\mathsf{ego}.\overline{y}} < \mathtt{var}_{\mathsf{right}.\overline{y}} \ . \end{aligned}$$

In other words, the sufficient constraint describes that ego is within the boundaries of one of the known lanes.

Now we come to the formal correctness of the encoding.

Lemma 2 (Properties of Construction 1)**.** *Assume some trajectory $\pi \in \mathit{Suffix}(\mathcal{T}(\mathsf{WM}))$ and a valuation μ with $V_O \subseteq \mathrm{Dom}(\mu)$. Build $\mathcal{M}_{\pi,\mu}$ as follows:*

- *For each object variable $o \in \mathrm{Dom}(\mu)$ and attribute access $o.a$ occurring in ϕ, set $\mathcal{M}_{\pi,\mu}(\mathtt{var}_{o.a}) := \pi(0)[\mu(o).a]$.*
- *For any other (real-valued) variable $x \in \mathrm{Dom}(\mu)$, set $\mathcal{M}_{\pi,\mu}(\mathtt{var}_x) := \mu(x)$.*

Then, for any MSRTL predicate ϕ where μ valuates all free variables, $\mathcal{M}_{\pi,\mu} \models tr_s(\phi, V_O)$ implies $\pi, \mu \models \phi$, and $\pi, \mu \models \phi$ implies $\mathcal{M}_{\pi,\mu} \models tr_n(\phi, V_O)$. Furthermore, if $\mu(V_O) = O_\pi$ then $\mathcal{M}_{\pi,\mu} \models tr_c(\phi, V_O)$ if, and only if, $\pi, \mu \models \phi$.

Proof. We proof this by structural induction over MSRTL predicates ϕ and terms. For readability, we again omit the parameter V_O.

First, we show that for MSRTL terms (attribute access, constants, and function application) ψ holds

$$[\![tr_n(\psi)]\!]\,\mathcal{M}_{\pi,\mu} = [\![tr_s(\psi)]\!]\,\mathcal{M}_{\pi,\mu} = [\![tr_c(\psi)]\!]\,\mathcal{M}_{\pi,\mu} = [\![\psi]\!]\,(\pi(0), \mu) \ .$$

The induction is anchored at constants and variables (Equations 6.3a–c). The induction hypothesis holds trivially for constants, and for variables by the Lemma's assumptions. Assume now that the induction hypothesis holds for terms $\psi_1, \ldots, \psi_n$. Then, the hypothesis also holds for function applications (Equation 6.3d) $f(\psi_1, \ldots, \psi_n)$. Recall that we assume that the used SMT theory $\mathcal{T}$ and the world model apply consistent interpretations to n-ary ($n > 0$) function and predicate symbols f, i.e., $\mathcal{T}(f) = \mathcal{I}(f)$. Because $\mathcal{T} \subseteq \mathcal{M}_{\pi,\mu}$ by definition, it is also $\mathcal{M}_{\pi,\mu}(f) = \mathcal{I}(f)$.

With an analogous argument, the induction hypothesis (i.e., the lemma) holds for predicate applications (Equation 6.3e). The induction step is also straight forward for conjunction and disjunction (Equations 6.3f and g). For tr_c, the induction hypothesis holds in both directions, so negations propagate naturally.

For the negation case of tr_s and tr_n, assume that we have shown $\mathcal{M}_{\pi,\mu} \models tr_s(\phi) \implies \pi, \mu \models \phi \implies \mathcal{M}_{\pi,\mu} \models tr_n(\phi)$. This is equivalent to $\mathcal{M}_{\pi,\mu} \not\models tr_n(\phi) \implies \pi, \mu \not\models \phi \implies \mathcal{M} \not\models tr_s(\phi)$ which is, by the semantics of MSRTL and SMT formulae, equivalent to $\mathcal{M}_{\pi,\mu} \models \neg tr_n(\phi) \implies \pi, \mu \models \neg\phi \implies \mathcal{M}_{\pi,\mu} \models \neg tr_s(\phi)$. So, by Equations (6.3i and k), $\mathcal{M}_{\pi,\mu} \models tr_s(\neg\phi) \implies \pi, \mu \models \neg\phi \implies \mathcal{M}_{\pi,\mu} \models tr_n(\neg\phi)$.

The lemma remains to be shown for existential quantification $\phi \equiv \exists o \in \mathtt{O}.\phi'$ resp. $\phi \equiv \exists x \in \mathbb{R}.\phi'$. Let's proof the latter first. Assume that $\mathcal{M}_{\pi,\mu} \models tr_s(\phi)$ (resp. $\mathcal{M}_{\pi,\mu} \models tr_c(\phi)$). Then, by Equation (6.3h) and the semantics of SMT formulae, there exists a value $v \in \mathbb{R}$ such that $\mathcal{M}_{\pi,\mu}[\mathtt{var}_x \mapsto v] \models tr_s(\phi')$, and by induction hypothesis $\pi, \mu \cup \{x \mapsto v\} \models \phi'$ (we can construct $\mathcal{M}_{\pi,\mu}[\mathtt{var}_x \mapsto v]$ from $\mu \cup \{x \mapsto v\}$ and π). So, $\pi, \mu \models \phi$. The other direction, as well as the case for tr_c, is analogous.

Now the remaining case $\phi \equiv \exists o \in \mathtt{O} : \phi'$: Assume that $\pi, \mu \models \phi$. Then, there exists some object $obj \in \mathcal{I}(\mathtt{O})$ such that $\pi, \mu \cup \{o \mapsto obj\} \models \phi'$. By Assumption 1 (and the semantics of $\Box$) holds $\pi, \mu \cup \{o \mapsto obj\} \models \Phi_{\mathcal{I}(\mathtt{O})}$. Construct $\mathcal{M}_{\pi,\mu\cup\{o\mapsto obj\}} \supseteq \mathcal{M}$ with $\mathcal{M}_{\pi,\mu\cup\{o\mapsto obj\}}(\mathtt{var}_{o.a}) = \pi(0)[obj.a]$ for each attribute $a \in \mathcal{A}(\mathcal{I}(\mathtt{O}))$. By induction hypothesis holds $\mathcal{M}_{\pi,\mu\cup\{o\mapsto obj\}} \models tr_n(\phi')$ and $\mathcal{M}_{\pi,\mu\cup\{o\mapsto obj\}} \models tr_n(\Phi_{\mathcal{I}(\mathtt{O})})$. So, $\mathcal{M}_{\pi,\mu} \models tr_n(\phi)$.

The other direction is similar. Assume $\mathcal{M}_{\pi,\mu} \models tr_s(\phi)$. By definition, there exists some extension $\mathcal{M}'$ of $\mathcal{M}_{\pi,\mu}$ such that $\mathcal{M}' \models tr_s(\phi')$. Furthermore, there exists some $o' \in V_O \subseteq \mathrm{Dom}(\mu)$ such that $type(o) \preceq \mathcal{I}(\mathtt{O})$ and $\mathcal{M}_{\pi,\mu}(\mathtt{var}_{o'.a}) = \mathcal{M}'(\mathtt{var}_{o.a})$ for all attributes $a \in \mathcal{A}(\mathcal{I}(\mathtt{O}))$. Hence, $\mathcal{M}' \subseteq \mathcal{M}_{\pi,\mu\cup\{o\mapsto\mu(o')\}}$, and therefore $\mathcal{M}_{\pi,\mu\cup\{o\mapsto\mu(o')\}} \models tr_s(\phi')$. By induction hypothesis, $\pi, \mu \cup \{o \mapsto \mu(o')\} \models \phi'$, so, by definition, $\pi, \mu \models \exists o \in \mathtt{O} : \phi'$. This is analogously for tr_c.

For tr_c, we also have to show $\pi, \mu \models \exists o \in \mathtt{O} : \phi' \Rightarrow \mathcal{M}_{\pi,\mu} \models tr_c(\exists o \in \mathtt{O} : \phi')$ assuming $O_\pi = \mu(V_O)$. This is analogous to the case for tr_n: If $\pi, \mu \models \exists o \in \mathtt{O} : \phi'$, then by definition there exists some $obj \in \mathcal{I}(\mathtt{O}) \cap O_\pi$ such that $\pi, \mu \cup \{o \mapsto obj\} \models \phi'$. By induction hypothesis, $\mathcal{M}_{\pi,\mu\cup\{o\mapsto obj\}} \models tr_c(\phi')$ with $\mathcal{M}_{\pi,\mu\cup\{o\mapsto obj\}} \supseteq \mathcal{M}$ and, for all attributes, $\mathcal{M}_{\pi,\mu\cup\{o\mapsto obj\}}(\mathtt{var}_{o.a}) = \pi(0)[obj.a]$. Because of $O_\pi = \mu(V_O)$, there exists some object variable $o' \in V_O$ such that $\mu(o) = \mu(o') = obj$. By construction, $\mathcal{M}_{\pi,\mu\cup\{o\mapsto obj\}}(\mathtt{var}_{o.a}) = \mathcal{M}_{\pi,\mu}(\mathtt{var}_{o'.a})$, so $\mathcal{M}_{\pi,\mu} \models tr_c(\exists o \in \mathtt{O} : \phi')$. No other cases need to be shown for the induction step. □

6.4.2 Checking Continuity

Experience has shown, that many internal inconsistencies arise from sequences of spatial views where no transition between the first and second invariant is possible. Most of the time this is because vehicle is required to be "teleported", whereas in all trajectories of the world model the position variables are continuous, i.e., no jumps are possible. Therefore, we construct a slightly stronger version tr'_n of the translation function tr_n that takes the continuity into account. In the following, we assume that the set $\mathcal{A}(C)$ of attributes of class C can be split into sets $\mathcal{A}_C(C)$ and $\mathcal{A}_D(C)$ of continuous attributes and those that allow discontinuities. Analogously, we can split the trajectory state variables (considering trajectories over objects $O \subseteq Obj$) into sets $\mathcal{A}_C(O)$ and $\mathcal{A}_D(O)$ such that all trajectories $\pi \in \mathcal{T}(\mathsf{WM})$ are continuous on $\mathcal{A}_C(O)$, i.e the projection $\pi{\downarrow_{\mathcal{A}_C(O)}}$ is a continuous function with respect to the following definition:

Definition 27 (continuous function (cf. [69]))**.** *A function $f : \mathbb{D} \to \mathbb{R}^m$ with domain $\mathbb{D} \subseteq \mathbb{R}^n$ is continuous in some point $\mathbf{x} \in \mathbb{D}$ if*

$$\forall \varepsilon > 0 \exists \delta > 0 \forall \mathbf{x}' \in \mathbb{D} : (\|\mathbf{x} - \mathbf{x}'\| < \delta \Rightarrow \|f(\mathbf{x}) - f(\mathbf{x}')\| < \varepsilon) \tag{6.4}$$

where $\|\cdot\|$ is the max norm, i.e., $\|(x_1, \ldots, x_n)^T\| := \max\{|x_1|, \ldots, |x_n|\}$.

Recall from the translation of the temporal chart structure to BMC (Chart2BMC, Algorithm 5) that each unrolling step describes what happens on a time slice from the time instance t of the current step till the next step at time t'. The Boolean variables b_{sv}, which stand for spatial views sv in the encoding, are bound to necessary, respectively sufficient, substitutes for $\Box_{[t,t')}]\phi_{sv}$. As shown in Lemma 2, Construction 1 creates SMT formulae that evaluate ϕ_{sv} at a single time point. When used in the BMC problem, they will be used to evaluate the MSRTL predicate at time t, i.e., the beginning of the time slice. The construction tr'_n encodes ϕ_{sv} also for the end of the time slice, i.e., for time t'. Because invariant nodes are evaluated on right-open intervals (see Definition 8), we need to modify the formula such that it includes a necessary constraint for $\pi^{t'-\varepsilon}, \mu \models \phi_{sv}$, but excludes evaluation t' itself. This works by approximating ϕ_{sv} by an approximation that only uses non-strict inequalities and exploiting the following property of continuous functions:.

Lemma 3 (Inequalities hold at the limit)**.** *Let $\mathbf{x} \in \mathbb{R}^n$ and $f\colon \mathbb{R}^n \to \mathbb{R}$ continuous in $\mathbf{x} \in \mathbb{R}^n$. If*

$$\forall \delta > 0 \exists \mathbf{x}' \in \mathbb{R}^n : \|\mathbf{x}' - \mathbf{x}\| < \delta \wedge f(\mathbf{x}') \leq 0 \tag{6.5}$$

then

$$f(\mathbf{x}) \leq 0 \ . \tag{6.6}$$

Proof. By contradiction. Let $\mathbf{x} \in \mathbb{R}^n$ and $f\colon \mathbb{R}^n \to \mathbb{R}$ be continuous in $\mathbf{x}$. Choose some $\delta > 0$ such that Equation (6.4) holds for all $\mathbf{x}' \in \mathbb{R}^n$ (we can find such a δ by definition). We assume that there exists $\mathbf{x}' \in \mathbb{R}^n$ such that Equation (6.5) holds. Now, we show by contradiction that $f(\mathbf{x}) \leq 0$: Assume $f(\mathbf{x}) > 0$ and set $\varepsilon := f(\mathbf{x})$. Then, by Definition 27,

$$\begin{aligned} & |f(\mathbf{x}) - f(\mathbf{x}')| < \varepsilon \\ \Rightarrow\ & f(\mathbf{x}) - f(\mathbf{x}') < f(\mathbf{x}) \\ \Rightarrow\ & f(\mathbf{x}') > 0 \end{aligned}$$

which contradicts Equation (6.5). □

Construction 2 (Necessary constraints with continuity check (tr'_n))**.** *Given an MSRTL predicate ϕ, construct a weaker version $\bar{\phi}$ of it by replacing strict inequalities of the form $x < y$ by their non-strict form $x \leq y$ if they occur under an even number of negations, and $x \leq y$ by $x < y$ if it occurs under an odd number of negations. For an MSRTL predicate ϕ and a set V_T of object variables (that contains all free object variables in ϕ), the SMT formula $tr'_n(\phi, V_O)$ is defined as*

$$tr'_n(\phi, V_O) :\equiv tr_n(\phi, V_O) \wedge \exists X'_D : tr_n(\bar{\phi}, V_O)[X_D \backslash X'_D, X_C \backslash X'_C]$$

where the set X_C contains all variables $\mathtt{var}_{o.a}$ introduced by $tr_n(\bar{\phi})$ for continuous attributes $o.a$, and the set X_D the remaining free variables. The set X'_C contains a fresh copy x' for each variable $x \in X_C$ and the set X'_D a fresh variable x' for each variable $x \in X_D$, respectively.

The constructed SMT constraint $tr'_n(\phi)$ (Construction 2) is strong enough that the weak existential and partial consistency checks find also non-obvious discontinuities. In particular, this includes the case mentioned at the end of Section 2.4.1. On the other hand, it preserves the core properties of tr_n (Construction 1), being the ability to construct from a satisfying trajectory for ϕ a satisfying model for $tr'_n(\phi)$.

Theorem 7 (Correctness of tr'_n (Construction 2))**.** *Assume an MSRTL property ϕ with free object variables from some set V_O, a trajectory π, time variables τ_1, τ_2, time points $t_1 < t_2$, and a valuation μ with $\mathrm{Dom}(\mu) \supseteq V_O \cup \{\tau_1, \tau_2\}, \mu(\tau_1) = t_1, \mu(\tau_2) = t_2$ such that $\pi, \mu \models \Box_{[\tau_1,\tau_2)}\phi$. Then, we can construct a satisfying model for $tr'_n(\phi)$ (Construction 2) from π and μ as follows:*

- *For each object variable $o \in \mathrm{Dom}(\mu)$ and attribute access $o.a$ occurring in ϕ, set $\mathcal{M}(\mathtt{var}_{o.a}) := \pi(t_1)[\mu(o).a]$ and, for continuous attributes only, set $\mathcal{M}(\mathtt{var}'_{o.a}) := \pi(t_2)[\mu(o).a]$*
- *For any other (real-valued) variable $x \in \mathrm{Dom}(\mu)$, set $\mathcal{M}_{\pi,\mu}(\mathtt{var}_x) := \mathcal{M}_{\pi,\mu}(\mathtt{var}'_x) := \mu(x)$.*

The proof of Theorem 7 relies on the following lemma.

Lemma 4 (Satisfiability of $\bar{\phi}$)**.** *For any MSRTL predicate ϕ, trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ over a finite set $O \subseteq Obj$, and any valuation μ, holds: if*

- *quantified variables (except objects) in ϕ occur only in linear inequalities, and*
- *$\pi, \mu \models \Box_{[t_1,t_2)}\phi$ for some time variables t_1, t_2 with $\mu(t_1) < \mu(t_2)$*

then there exists a state vector $\mathbf{x} \in \mathbb{R}^{\mathcal{A}(O)}$ such that $\mathbf{x}, \mu \models \bar{\phi}$ and $\mathbf{x}{\downarrow_{\mathcal{A}_C(O)}} = \pi(\mu(t_2)){\downarrow_{\mathcal{A}_C(O)}}$, where $\bar{\phi}$ is constructed from ϕ as described in Construction 2 above.

Proof. Assume that the number of objects is finite. Given some valuation μ, we can transform $\bar{\phi}$ into a function $F_\mu : \mathbb{R}^{\mathcal{A}(O)} \to \mathbb{R}$ such that for any time point t and trajectory π holds

$$\pi^t, \mu \models_{\mathsf{WM}} \bar{\phi} \Leftrightarrow F_\mu(\pi(t)) \leq 0 \ .$$

First, we transform $\bar{\phi}$ into an equivalent monotone boolean combination of linear inequalities. By construction of $\bar{\phi}$, all inequalities in this formula are non-strict. Furthermore, we can eliminate quantified variables of type $\mathbb{R}$ using, e.g., the method by Ferrante and Rackoff [55]. Inspection of this method shows that it does not re-introduce any strict inequalities. Then, we construct F_μ inductively as follows (F'_μ and F''_μ denote functions constructed for sub-formulas $\bar{\phi}'$ and $\bar{\phi}''$):

- If $\bar{\phi}$ is a linear inequality over object attributes $o_1.a_1, \ldots, o_n.a_n$ then, w.o.l.g, we can write it as $f(o_1.a_1, \ldots, o_n.a_n) \leq 0$ where f is a continuous function, and let $F_\mu(\mathbf{x}) := f(\mathbf{x}[\mu(o_1).a_1], \ldots, \mathbf{x}[\mu(o_n).a_n])$.
- If $\bar{\phi}$ is of the form $\exists o \in \mathtt{O} : \bar{\phi}'$, then let $F_\mu(\mathbf{x}) := \min_{obj \in \mathcal{I}(\mathtt{O}) \cap O} F'_{\mu \cup \{o \mapsto obj\}}(\mathbf{x})$
- For $\bar{\phi}$ of the form $\forall o \in \mathtt{O} : \bar{\phi}'$, define F_μ analogous with max instead of min.
- If $\bar{\phi}$ is of the form $\bar{\phi}' \vee \bar{\phi}''$, then let $F_\mu(\mathbf{x}) := \min\{F'_\mu(\mathbf{x}), F''_\mu(\mathbf{x})\}$.

- If $\bar{\phi}$ is of the form $\bar{\phi}' \wedge \bar{\phi}''$, then let $F_\mu(\mathbf{x}) := \max\{F'_\mu(\mathbf{x}), F''_\mu(\mathbf{x})\}$.

Note that min and max are continuous functions. Correctness follows from Definition 5 together with the following equivalences:

$$F_1(\mathbf{x}) \leq 0 \vee F_2(\mathbf{x}) \leq 0 \Leftrightarrow \min\{F_1(\mathbf{x}), F_2(\mathbf{x})\} \leq 0$$
$$F_1(\mathbf{x}) \leq 0 \wedge F_2(\mathbf{x}) \leq 0 \Leftrightarrow \max\{F_1(\mathbf{x}), F_2(\mathbf{x})\} \leq 0$$

Assume there exists a trajectory π and valuation μ such that $\pi, \mu \models \Box_{[t_1,t_2)}\phi$. Clearly, $\bar{\phi}$ is weaker than ϕ, so $\pi, \mu \models \Box_{[t_1,t_2)}\bar{\phi}$. By Definition 5, this implies

$$\forall t \in [t_1, t_2) : \pi^t, \mu \models \bar{\phi} \ .$$

Because π is piecewise continuous, there exist tie points $\tau < \tau'$ around $\mu(t_2)$ (i.e $\tau < \mu(t_2) \leq \tau_2$) and a continuous function $\tilde{\pi} : [\tau_1, \tau_2] \to \mathbb{R}^{\mathcal{A}(O)}$ such that $\pi(t') = \tilde{\pi}(t')$ for all $t' \in [\tau, \tau')$. Now, pick some $\delta < 0$. By continuity of $\tilde{\pi}$, there exists some $\varepsilon > 0$ such that for all $t \in [\tau, \tau']$ with $\mu(t_2) - t < \varepsilon$ holds $\|\tilde{\pi}(\mu(t_2)) - \tilde{\pi}(t)\| < \delta$. Because $\tau < \mu(t_2) \leq \tau'$, $\mu(t_1) < \mu(t_2)$, and $\varepsilon < 0$, there exists at least one such t with $\mu(t_1) < t < \mu(t_2)$. We construct a function F_μ from $\bar{\phi}$ and μ as above. Because of $\pi^t, \mu \models \bar{\phi}$, it is $F_\mu(\pi(t)) = F_\mu(\tilde{\pi}(t)) \leq 0$. By Lemma 3, $F_\mu(\tilde{\pi}(\mu(t_2)) \leq 0$, and, with the construction above, $\tilde{\pi}^{\mu(t_2)}, \mu \models \bar{\phi}$.

Finally, let $\mathbf{x} = \tilde{\pi}(\mu(t_2))$. Because π is continuous on $\mathcal{A}^C(O)$, it is $\mathbf{x}{\downarrow_{\mathcal{A}^C(O)}} = \pi(\mu(t_2)){\downarrow_{\mathcal{A}^C(O)}}$. Hence, the lemma holds. □

Now we come to the main proof of Theorem 7.

Proof of Theorem 7. Assume an MSRTL property ϕ with free object variables from some set V_O, a trajectory π over objects O_π, time variables τ_1, τ_2, time points $t_1 < t_2$, and a valuation μ with $\mathrm{Dom}(\mu) \supseteq V_O \cup \{\tau_1, \tau_2\}, \mu(\tau_1) = t_1, \mu(\tau_2) = t_2$ such that $\pi, \mu \models \Box_{[\tau_1,\tau_2)}\phi$. By the semantics of time-bounded globally, $\pi^{t_1}, \mu \models \phi$. So, by Lemma 2, we can construct some model $\mathcal{M}_1 \models tr_n(\phi)$. In addition, we have to show that we can construct a model $\mathcal{M} \supseteq \mathcal{M}_1$ such that $\mathcal{M} \models \exists X'_D : tr_n(\bar{\varphi}, V_O)[X_D \backslash X'_D, X_C \backslash X'_C]$. By Lemma 4, there exists a state vector $\mathbf{x} \in \mathbb{R}^{\mathcal{A}(O)}$ such that $\mathbf{x}, \mu \models \bar{\phi}$ and $\mathbf{x}{\downarrow_{\mathcal{A}_C(O_\pi)}} = \pi(t_2){\downarrow_{\mathcal{A}_C(O_\pi)}}$. From $\mathbf{x}$, we can construct a model $\mathcal{M}_2$ such that $\mathcal{M}_2 \models tr_n(\bar{\varphi}, V_O)[X_D \backslash X'_D, X_C \backslash X'_C]$ (Lemma 2 applies). Now, construct the model $\mathcal{M}$ as described in Theorem 7. Recall, that $\mathbf{x}{\downarrow_{\mathcal{A}_C(O_\pi)}} = \pi(t_2){\downarrow_{\mathcal{A}_C(O_\pi)}}$. So, by construction, $\mathcal{M}(X) = \mathcal{M}_1(X)$ and $\mathcal{M}(X'_C) = \mathcal{M}_2(X'_C)$. Therefore, $\mathcal{M} \models tr_n(\phi, V_O) \wedge \exists X'_D : tr_n(\bar{\phi}, V_O)[X_D \backslash X'_D, X_C \backslash X'_C] \equiv tr'_n(\phi)$. □

6.5 The Semi-decision Procedure CheckSatN$_{\mathsf{WM}}$

Now we are ready to present the semi-decision procedure CheckSatN$_{\mathsf{WM}}$ in Algorithm 7.

Theorem 8 (Correctness of CheckSatN (Algorithm 7)**.** *) If a basic chart* WM *is satisfiable in some world model* WM*, then* CheckSatN$_{\mathsf{WM}}$(BC) $\neq$ *unsat.*

Proof. Assume some world model WM, a basic chart BC with free object variables in V_O, a trajectory $\pi \in \mathcal{T}(\mathsf{WM})$, and a valuation μ such that $\pi, \mu \models_{\mathsf{WM}} \varphi_{\mathsf{BC}}$. Assume that φ_{BC} has $n+1$ free and bound time variables, construct the BMC problem $\mathsf{BMC} = (I, T, F)$ as described in Algorithm 7, and unroll it for n steps, yielding an SMT formula BMC_n.

Recall that Construction 2 applied in line 7 of Algorithm 7 ($b_{sv} \Leftrightarrow tr'_n(\phi_{sv}, V_O)$ for each spatial view sv in BC) introduces a variable $\mathtt{var}_{o.a}$ for each attribute $o.a$ of each object

Algorithm 7: The semi-decision procedure CHECKSATN$_{\mathsf{WM}}$(BC)

Input: A world model WM, and a basic chart BC
Output: A tuple (SAT, $\mathcal{M}$) containing a model, UNSAT, or UNKNOWN

begin
  let V_T be the set of free and bound time variables in φ_{BC};
  let V_O be the set of free object variables in φ_{BC};
  let $(I, T, F) :=$ CHART2BMC(BC) ; *// see Algorithm 5*
  for each *spatial view sv in* BC **do**
    $T :\equiv T \wedge (b_{sv} \Rightarrow tr'_n(\phi_{sv}, V_O))$; *// see Construction 2; b_{sv} is introduced by* CHART2BMC(BC)
  end
  for each $o \in V_O$ **do**
    $T :\equiv T \wedge tr'_n(\Phi_{type(o)}(o), V_O)$;
    for each *attribute a of o with dynamics "constant"* **do**
      $T :\equiv T \wedge \mathtt{var}'_{o.a} = \mathtt{var}_{o.a}$; *// $\mathtt{var}_{o.a}$ is introduced in Construction 1*
    end
  end
  return BOUNDEDMODELCHECKING($(I, T, F), [|V_T| - 1, |V_T| - 1]$); *// see Algorithm 1*
end

variable $o \in V_O$. After unrolling of the BMC problem, each $\mathtt{var}_{o.a}$ manifests as a variable $\mathtt{var}^i_{o.a}$ (for $i = 0, 1, \ldots, n$) in the SMT formula BMC_n. In the following, X, X' and X_i are sets of variables such that, for each attribute $o.a$ of $o \in V_O$, it is $\mathtt{var}_{o.a} \in X$, $\mathtt{var}'_{o.a} \in X'$, and $\mathtt{var}^i_{o.a} \in X_i$.

By construction, BMC_n is a conjunction of the following parts:

- Ψ_n which is the BMC problem $\Psi :\equiv$ CHART2BMC(BC) unrolled for n steps
- for $i = 0, 1, \ldots, n-1$:
  - $b^i_{sv} \Rightarrow tr'_n(\phi_{sv}, V_O)[X \backslash X_i, X' \backslash X_{i+1}]$ for each spatial view sv in BC
  - $tr'_n(\Phi_{type(o)}(o), V_O)[X \backslash X_i, X' \backslash X_{i+1}]$ for each $o \in V_O$
  - $\mathtt{var}^{i+1}_{o.a} = \mathtt{var}^i_{o.a}$ for each constant attribute $o.a$ of each $o \in V_O$

Now, we construct a model $\mathcal{M}$ by applying the constructions given in Theorem 7 and the proof of Theorem 5. Assume time points $t_0 < t_1 < \ldots t_n$ as in Theorem 5. Construct a model $\mathcal{M}_1$ where $\mathcal{M}_1(\mathtt{var}^i_{o.a}) = \pi(t_i)[\mu(o).a]$ for each $\mathtt{var}^i_{o.a}$. By construction, $\mathcal{M}_1 \models (\mathtt{var}^{i+1}_{o.a} = \mathtt{var}^i_{o.a})$ for all constant attributes $a.o$ and $i = 0, 1, \ldots, n-1$.

Construct another model $\mathcal{M}'_2$ (a partial model for Ψ_n) with only $\mathcal{M}(b^i_{sv}) :\Leftrightarrow \mathcal{M}_1 \models tr'_n(\phi_{sv}, V_O)[X \backslash X_i, X' \backslash X_{i+1}]$ for $i = 0, 1, \ldots, n-1$. By Theorem 4 holds

$$\pi, \mu \models \Box_{[t_i, t_{i+1})} \phi_{sv} \Longrightarrow \mathcal{M}_1 \models tr'_n(\phi_{sv}, V_O)[X \backslash X_i, X' \backslash X_{i+1}]$$

for all $i = 0, 1, \ldots, n-1$. Hence, by Theorem 5, we can construct a complete model $\mathcal{M}_2 \supseteq \mathcal{M}'_2$ such that $\mathcal{M}_2 \models \Psi_n$. Because $\mathcal{M}_1$ and $\mathcal{M}_2$ do not have variables in common, we can construct a model $\mathcal{M} := (\mathcal{M}_1 \cup \mathcal{M}_2) \models \mathsf{BMC}_n$. □

6.6 Pointing to the Cause of the Inconsistency

Section 6.2 shows how partial inconsistencies in a larger set of TSCs can be narrowed down to the smallest subsets for which a partial inconsistency can be deduced. This already helps the engineer to identify problematic TSCs in a specification. This section shows how SMT solving can be utilized to narrow down the source of a partial inconsistency to a set of spatial views in the TSC. In the following, we call such a set of spatial views *conflicting*. Intuitively, a (minimal) conflicting set is a (minimal) subset of spatial views that suffice to make a basic chart – for partial inconsistencies this is the chart $\mathsf{BC}_2^{\mathsf{TSC},T}$ given in Definition 21 – unsatisfiable, i.e., the chart is unsatisfiable even if all other spatial views are replaced by an equivalent of *true* (i.e., the empty spatial view).

In the following, we assume that our SMT solver implements a complete decision procedure for linear arithmetics, i.e., CheckSatN never returns UNKNOWN.

Definition 28 (Conflicting Spatial Views)**.** *Assume a positive basic chart (or positive existential TSC)* BC *that is unsatisfiable in some world model* WM *and let* SV *be the set of spatial views in* BC*. Then, a set* $\mathsf{SV}' \subseteq \mathsf{SV}$ *of spatial views is called* conflicting in BC wrt. WM *if the basic chart* $\mathsf{BC} \downarrow \mathsf{SV}'$*, which is derived from* BC *by replacing all spatial views in* $\mathsf{SV} \setminus \mathsf{SV}'$ *by the empty spatial view, is unsatisfiable. Analogously, it is called* weakly conflicting *if* $\mathit{CheckSatN}_{\mathsf{WM}}(\mathsf{BC} \downarrow \mathsf{SV}') = \mathit{UNSAT}$*. It is called a* minimal (weakly) conflicting set *if no proper subset of* SV' *is a (weakly) conflicting set.*

Fact 3 (Minimal conflicting sets)**.** *Let* SV *be the set of spatial views in a positive basic chart* BC*. If* $\mathsf{SV}' \subseteq \mathsf{SV}$ *is conflicting in* BC*, then every set* $\mathsf{SV}' \subseteq \mathsf{SV}'' \subseteq \mathsf{SV}$ *is conflicting. This holds analogously for weak conflicting sets.*

Proof. Recall that for the empty spatial view ε it is $\phi_\varepsilon = \mathit{true}$. In the semantics (MSRTL formula) φ_{BC} of positive basic charts BC, the formulae ϕ_{sv} for spatial views $sv \in \mathsf{SV}$ do not occur under negations, so $\mathsf{BC} \Rrightarrow \mathsf{BC} \downarrow \mathsf{SV}'$ for any $\mathsf{SV}' \subseteq \mathsf{SV}$ (this can be easily shown by structural induction over positive basic charts). Note, that $\mathsf{BC} \downarrow \mathsf{SV}' = (\mathsf{BC} \downarrow \mathsf{SV}'') \downarrow \mathsf{SV}'$ for all $\mathsf{SV}' \subseteq \mathsf{SV}'' \subseteq \mathsf{SV}$. As a consequence, unsatisfiability of $\mathsf{BC} \downarrow \mathsf{SV}'$ implies unsatisfiability of $\mathsf{BC} \downarrow \mathsf{SV}''$.

For weak conflicting sets, this can be proven by inspection of Algorithm 7 (CheckSatN). Let (I, T, F) be the BMC problem constructed for BC, and (I', T', F') the BMC problem for $\mathsf{BC} \downarrow \mathsf{SV}'$, for some $\mathsf{SV}' \subset \mathsf{SV}$. By construction, $I = I'$, $F = F'$, and $T \Rightarrow T'$ (note that T and T' differ only in the constraints $b_{sv} \Rightarrow tr'_n(\mathit{true}, V_O)$ added in line 6 of the algorithm, which becomes $b_{sv} \Rightarrow \mathit{true}$ in T' for $sv \notin \mathsf{SV}'$). So, $\mathrm{CheckSatN}_{\mathsf{WM}}(\mathsf{BC} \downarrow \mathsf{SV}') = \mathrm{UNSAT}$ implies $\mathrm{CheckSatN}_{\mathsf{WM}}(\mathsf{BC}) = \mathrm{UNSAT}$. □

Recall that the BMC problem (that encodes a basic chart in CheckSatN) contains a constraint $b_{sv} \Rightarrow tr'_n(\phi_{sv}, V_O)$ for each spatial view sv. We can remove some spatial view sv from the chart, i.e., replacing it by an empty spatial view, by simply removing the corresponding constraint from the BMC problem. We can construct a (minimal) weakly conflicting set from a so-called (minimal) *high-level unsatisfiable core* of the unrolled BMC problem constructed in Algorithm 7.

Definition 29 (Minimal unsatisfiable core [92])**.** *Given a set* $\Psi = \{R_1, R_2, \ldots, R_m\}$ *of constraints, and a constraint set* Ω *such that* $\Psi \cup \Omega$ *is unsatisfiable, a set* $\Psi' \subseteq \Psi$ *is a* high-level unsatisfiable core *if* $\Psi' \cup \Omega$ *is also unsatisfiable. It is* minimal *if removing any constraint from* Ψ' *makes* $\Psi' \cup \Omega$ *satisfiable.*

In order to identify a minimal conflicting set of spatial views in an unsatisfiable basic chart, we exploit that in CHART2BMC (Algorithm 5) the spatial views are represented by Boolean placeholder variables b_{sv}, which are replaced during unrolling by n instances $b_{sv,i}$, for $i = 0, 1, \ldots, n$ (where n is the unrolling depth). We separate the glue logic (line 6 of Algorithm 7), which binds the spatial view encodings to the placeholder variables, from the rest of the unrolled BMC problem. Removing part of the glue logic from the SMT problem is then equivalent to replacing the affected spatial views by empty ones. Hence, we can utilize the high-level unsatisfiable core to identify those spatial views that make the SMT problem unsatisfiable. This is described in detail with the following theorem.

Theorem 9 (Conflicting set extraction)**.** *Given a basic chart* BC *over spatial views* SV *with free object variables* V_O *and free and bound timing variables* V_T *(as in Algorithm 7), construct the BMC problem* (I, T, F) *as described in Algorithm 7, and unroll it for* $|V_T| - 1$ *steps, yielding the SMT problem* $(I, T, F)_{|V_T|-1}$*. Let* X *be the set of free variables in* (I, T, F)*. Split the constraints in* $(I, T, F)_{|V_T|-1}$ *into the sets* $\Psi = \{R_{sv} \mid sv \in \mathsf{SV}\}$ *and* Ω *such that*

$$R_{sv} :\equiv \bigwedge_{i=0}^{|V_T-1|} (b_{sv,i} \Rightarrow tr'_n(\phi_{sv}, V_O)[X \backslash X_i, X' \backslash X_{i+1}])$$

for all $sv \in \mathsf{SV}$*,* $(\Psi \cup \Omega) \equiv (I, T, F)_{|V_T|-1}$ *and none of the constraints in* Ψ *occurs in* Ω*.*

Then, $\mathsf{SV}' \subseteq \mathsf{SV}$ *is a (minimal) weakly conflicting set in* BC *if, and only if,* $\Psi' := \{R_{sv} \mid sv \in \mathsf{SV}'\}$ *is a (minimal) high-level unsatisfiable core of* Ψ *and* Ω*.*

Proof. This follows directly from the construction described in Theorem 9. For some basic chart BC, construct Ψ and Ω as described in the theorem. Given a set $\mathsf{SV}' \subseteq \mathsf{SV}$, let $\Psi' :\equiv \{R_{sv} \mid sv \in \mathsf{SV}'\}$ with R_{sv} defined as in the theorem. Furthermore, construct the unrolled BMC problem $(I', T', F')_{|V_T|-1}$ from $\mathsf{BC} \downarrow \mathsf{SV}'$ as in Algorithm 7, and constraints Ψ'' and Ω'' as described in the theorem such that $(\Psi'' \cup \Omega'') \equiv (I', T', F')_{|V_T|-1}$. Note, that BC and $\mathsf{BC} \downarrow \mathsf{SV}$ have the same temporal structure and we use the same set V_O of object variables, so $\Omega'' = \Omega$ by construction. Furthermore, it is $\Psi'' \equiv \Psi' \cup \{R'_{sv} \mid sv \in \mathsf{SV} \setminus \mathsf{SV}'\}$ with

$$R'_{sv} :\equiv \bigwedge_{i=0}^{|V_T|-1} (b_{sv,i} \Rightarrow \mathit{true}) \equiv \mathit{true} \ ,$$

so $\Psi'' \equiv \Psi'$. By definition, SV' is a weakly conflicting set in BC if, and only if, $\mathsf{BC} \downarrow \mathsf{SV}$ is unsatisfiable and so is $\Psi' \cup \Omega \equiv (I', T', F')_{|V_T|-1}$. As a consequence, $\Psi' \subseteq \Psi$ is a (minimal) high-level unsatisfiable core of Ψ and Ω if, and only if, SV' is a (minimal) weakly conflicting set in BC. □

Nadel [92] describes an approach to utilize features of SMT solvers, in particular a feature called "solving under assumptions", in order to generate minimal high-level unsatisfiable cores. SMTLib-compliant solvers (such as Z3 used in this thesis) that implement this feature, provide a variant `(check-sat-assuming (`a_1 a_2 `...` a_m`))` of the `(check-sat)` command which takes a list $a_1, a_2, \ldots, a_m$ of literals[5] as additional assumptions. It checks the conjunction of the current assertions and the provided literals for satisfiability. In case `check-sat-assuming` returns UNSAT, the function `get-unsat-assumptions` can be used to retrieve those literals that together with the assertions caused a conflict. Because the

[5]A literal is a Boolean constraint of the form p or $\neg p$ where p is a Boolean variable or constant.

solver usually does not guarantee that the returned set of literals is minimal, a dedicated algorithm needs to be implemented on top of the solver logic in order to minimize the unsatisfiable core. One possible approach, which is based on Nadel [92, Algorithm 3], is shown in Algorithm 8. The core idea is to introduce selector variables s_i for the constraints R_i in Ψ. In the beginning, all members of Ψ are candidates for the minimal unsatisfiable core, stored (in form of the selector variables) in the set *muc_cands*. In each iteration, at least one candidate is removed from *muc_cands* and either noted as member of the minimal unsatisfiable core, or withdrawn.

Algorithm 8: Construction of a minimal high-level unsatisfiable core (based on [92, Algorithm 3])

Input: Sets $\Psi = \{R_1, R_2, \ldots, R_m\}$ and Ω of constraints
Output: A minimal high-level unsatisfiable core

```
1  (assert Ω);
2  for each R_i ∈ Ψ do
3  |  (declare-fun s_i () Bool);
4  |  (assert s_i ⇒ R_i);
5  end
6  muc_cands := {s_i | R_i ∈ Ψ};
7  muc := ∅;
8  while muc_cands ≠ ∅ do
9  |  remove some s_k from muc_cands;
10 |  if (check-sat-assuming muc_cands) = SAT then
11 |  |  add s_k to muc;
12 |  |  (assert s_k);
13 |  else
14 |  |  muc_cands := (get-unsat-assumptions);
15 |  |  (assert ¬s_k);
16 |  end
17 end
18 return {R_i | s_i ∈ muc}
```

7 Trajectories as Splines

The previous chapter presented the overall approach for encoding a TSC into an SMT problem. In particular, the combination of Construction 2 and Algorithm 5 into Algorithm 7 lead to a necessary condition for the satisfiability of an existential TSC: If a satisfying trajectory for the TSC exists, then the SMT problem is also satisfiable. However, the opposite does not hold. If we find a satisfying model for the SMT problem, this does not yet imply that there is a satisfying trajectory. In this chapter, we extend the encoding of spatial views in order to derive a sufficient condition for satisfiability of the TSC. Being a sufficient condition means that we can use a satisfying model for the SMT problem to generate a trajectory that satisfies the TSC by construction.

This chapter incorporates parts of the author's published work [12]. In particular, Lemmas 7, 8, and 9 (including their proofs) from Section 7.4, and the proof of Theorem 10, item 1, have been taken from the published paper [12], in slightly modified form.

7.1 Encoding Strategy

The method for producing sufficient conditions from spatial views presented in this thesis is restricted to linear longitudinal and lateral constraints. We consider spatial views that translate to a first order constraints with predicates of the form

$$\sum_{i \in I} b_i o_i.a_i \bowtie b \ ,$$

where I is a finite index set, $\bowtie \in \{<, \leq\}$, $b \in \mathbb{R}$ and $b_i \in \mathbb{R}$ (for $i \in I$) are constants, and $o_i.a_i$ (for $i \in I$) denotes attribute access. Technically, we create these constraints by constructing the SMT formula $tr_c(\phi, V_O)$ (Construction 1, where ϕ is the MSRTL semantics of the spatial view), and let the SMT solver do the simplification/quantifier elimination for us (see Chapter 6)[1]. Recall that tr_c replaces bound object variables by free object variables from the set V_O. Hence, the result is a quantifier-free, monotone boolean combination of inequalities of the above form, which is equivalent to or stronger than the original invariant ϕ. Since we want to construct a sufficient constraint, this approximation is fine.

Figure 7.1 shows the overall flow of the encoding strategy that is explained in the following. The encoding scheme has already been presented in the author's published work [12], in a condensed form. Recall that Algorithm 5 (CHART2BMC, Section 6.3.1) encodes the temporal structure of basic charts into BMC problems. Hereby, each unrolling step i of the BMC problem corresponds to a time interval $[t_i, t_{i+1})$. The goal of this chapter is to provide techniques to

- encode vehicle dynamics in a BMC/SMT problem such that all solutions correspond to physically sound trajectories (i.e., they adhere to vehicle dynamics and global invariants stated in the world model), and

[1] The prototype implementation executes all constructions presented in this chapter on the abstract syntax tree of the SMTLib formula generated by construction tr_c and simplified by Z3

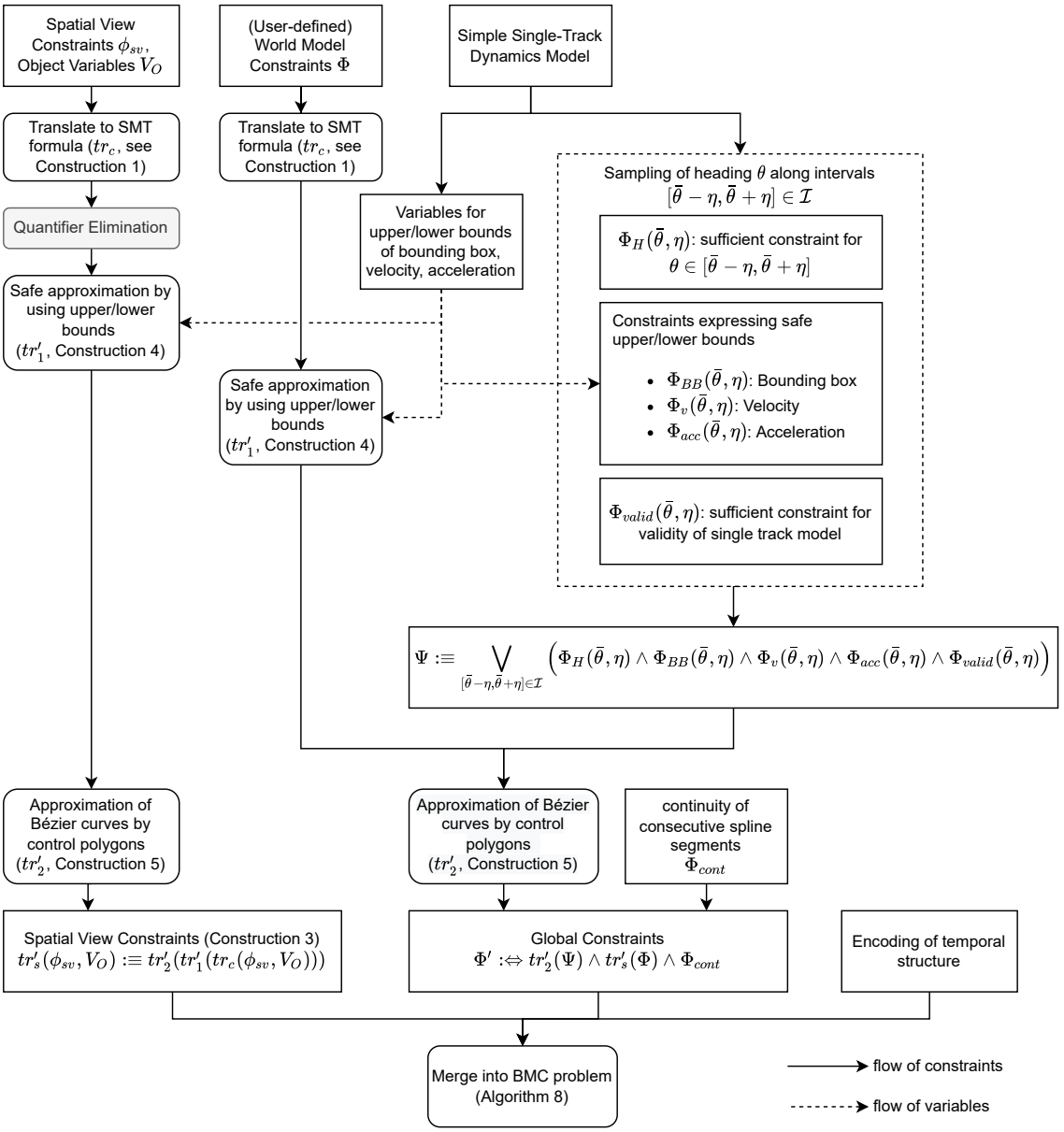


Figure 7.1: Overall encoding data flow

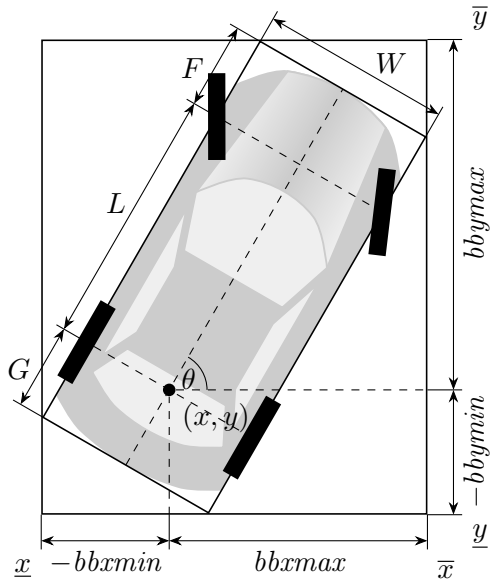


Figure 7.2: Bounding box dimensions of rotated rectangular objects

- for each spatial view sv occuring in the translated chart, bind the Boolean variable b_{sv} in the BMC problem to a sufficient constraint for satisfaction of sv on the whole time interval.

To achieve this, we describe the motion of vehicles as *Bézier splines* [99], where each spline segment (a *Bézier curve*) describes the trajectory (both position and velocity) on one time interval $[t_i, t_{i+1})$. Bézier curves are described in detail in Section 7.2. A Bézier curve of degree n is uniquely defined by $n+1$ *control points*. The state vector is hence encoded with free variables that describe the control points of the Bézier curves. More precisely, if Bézier curves of degree n are used, the state vector will contain variables $x_{o,0}, y_{o,0}, x_{o,1}, y_{o,1}, \ldots, x_{o,n}, y_{o,n}$ to describe the position of the object bound to variable o (every pair $(x_{o,i}, y_{o,i})$ forms one control point)[2]. Note that the x and y components of the position vector can be considered being separate (one dimensional) Bézier curves $x_o(s)$ and $y_o(s)$ with control points $x_{o,0}, \ldots, x_{o,n}$ and $y_{o,0}, \ldots, y_{o,n}$, respectively. In the following, it is described which attributes of a vehicle are considered. We drop the index o.

- The position $\mathbf{p} = (x, y)^T$ defines the center of the vehicle's rear axis; other anchors that are used in spatial constraints are given by the extrema $\underline{x} = x + bbxmin$, $\overline{x} = x + bbxmax$, $\underline{y} = y + bbymin$, and $\overline{y} = y + bbymax$ of the object's bounding box. The bounding box is aligned to the global x and y axes, see Figure 7.2.

- The velocity is measured in heading direction[3] as speed $v = \sqrt{\dot{x}^2 + \dot{y}^2}$.

- The longitudinal acceleration, $acc = \dot{v}$.

[2] In compliance with Section 6.4 we should better write $\texttt{var}_{o.x,i}$, $\texttt{var}_{o.y,i}$ instead of $x_{o.i}$, $y_{o.i}$. For this chapter the latter naming scheme has been chosen because it allows us later on to drop the subscripts and use simple variable names such as x and y.

[3] The used vehicle model assumes that the reference point always moves in heading direction.

Because the vehicles move non-linearly, the bounding box dimensions change depending on the current heading angle $\theta(t)$ at time t. Similarly, v and acc cannot be expressed as a linear equation over x and y or their derivatives without fixing θ. Therefore, we approximate these attributes with safe lower and upper bounds (this is the first part of the translation, denoted tr'_1). Furthermore, in the following sections linear constraints $\Phi_{BB}(\bar{\theta}, \eta)$, $\Phi_v(\bar{\theta}, \eta)$, and $\Phi_{acc}(\bar{\theta}, \eta)$ are developed that characterize these safe bounds depending on a primary heading angle $\bar{\theta}$ and a heading angle deviation η. These bounds are valid as long as $\theta(t) \in [\bar{\theta} - \eta, \bar{\theta} + \eta]$ during the whole time slice. A constraint $\Phi_H(\bar{\theta} + \eta)$ is defined that ensures this property. Another constraint $\Phi_{valid}(\bar{\theta}, \eta)$ encodes sufficient conditions for the validity of the single track model on the time slice. Finally, we sample $\bar{\theta}$ and η from a finite set $\mathcal{I}$ of closed intervals $I = [\bar{\theta} - \eta, \bar{\theta} + \eta]$ and combine the constraints to

$$\Psi :\equiv \bigvee_{[\bar{\theta}-\eta,\bar{\theta}+\eta]\in\mathcal{I}} \Big(\Phi_H(\bar{\theta}, \eta) \wedge \Phi_{BB}(\bar{\theta}, \eta) \wedge \Phi_v(\bar{\theta}, \eta) \wedge \Phi_{acc}(\bar{\theta}, \eta) \wedge \Phi_{valid}(\bar{\theta}, \eta)\Big) \ . \tag{7.1}$$

The formula is explained in detail later on.

Now, we have three sets of monotone boolean formulae:

- those from the spatial views
- user-defined additional constraints from the world model (e.g. speed and acceleration limits)
- the validity and bounding constraints from the single track model described above
- the continuity constraints Φ_{cont} from Section 7.2

All of these contain linear inequalities over Bézier curves and their derivatives. As a last step in the translation $tr'(\phi)$ and towards the global constraints Φ', these inequalities are safely approximated as linear constraints over the Bézier curve control points using the properties of Bézier curves described in Section 7.2 (this is part 2 of the constraint translation, denoted tr'_2).

Note, that the granularity of the approximation – meaning the satisfying trajectories that can be found for some TSC and world model – depends on the chosen time slice size Δ, and the chosen heading angle sampling intervals $\mathcal{I}$ for each vehicle. Some possible configurations with which solutions for basic driving maneuvers can be found are presented as part of the evaluation.

In the following, Bézier curves and the single track model are introduced in Sections 7.2 and 6.1.1. Then, in Section 7.4, the bounding and validity constraints are explained, followed by the translation functions tr'_1 and tr'_2 in Section 7.3. The chapter concludes with a correctness proof in Section 7.5.

7.2 Encoding Trajectories as Splines

Bézier curves [99, 121] of degree n are defined by interpolation between $n + 1$ points $\mathbf{p}_0, \dots \mathbf{p}_n$, where $\mathbf{p}_0$ and $\mathbf{p}_n$ are endpoints of the curve, and $\mathbf{p}_1, \dots, \mathbf{p}_{n-1}$ are bend-points.

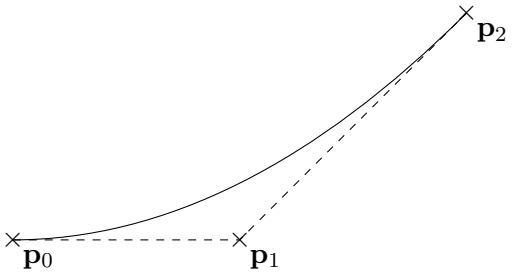


Figure 7.3: A quadratic Bezier curve

Definition 30 (Bézier curves and splines)**.** *Having points* $\mathbf{p}_0, \ldots, \mathbf{p}_n \in \mathbb{R}^m$ *fixed, the* Bézier curve $\mathbf{p} : [0,1] \to \mathbb{R}^m$ *of degree* n *is defined by the function*

$$\mathbf{p}(s) := \sum_{k=0}^{n} \binom{n}{k} \mathbf{p}_k (1-s)^k s^{n-k} \quad .$$

A Bézier spline *is a continuous parametric function that is piecewise defined by Bézier curves.*

One-dimensional Bézier curves (i.e., $\mathbf{p}_0, \ldots, \mathbf{p}_n \in \mathbb{R}$) are also called *Bernstein polynomials.*

Bézier curves have the following properties [99, 121]:

(P1) Because $\mathbf{p}(s)$ is a weighted sum where the weights sum up to 1, the curve is always located within the polygon spanned by its control points.

(P2) A weighted sum of Bézier curves is again a Bézier curve where the control points are weighted sums of the individual control points:

$$D(s) = \sum_i a_i \mathbf{p}_i(s) = \sum_{k=0}^{n} \binom{n}{k} D_k (1-s)^k s^{n-k}$$

with $D_k = \sum_i a_i \mathbf{p}_{i,k}$.

(P3) The first derivative is again Bézier curve of degree $n-1$:

$$\dot{\mathbf{p}}(s) = \sum_{k=0}^{n-1} \binom{n-1}{k} \dot{\mathbf{p}}_k (1-s)^k s^{n-1-k} \text{ with } \dot{\mathbf{p}}_k = n(\mathbf{p}_{k+1} - \mathbf{p}_k)$$

(P4) A Bézier curve of degree n is also a Bézier curve of degree $n+1$ with new control points, i.e.,

$$\mathbf{p}(s) = \sum_{k=0}^{n+1} \binom{n+1}{k} \mathbf{p}'_k (1-s)^k s^{n+1-k}$$

with $\mathbf{p}'_0 = \mathbf{p}_0$, $\mathbf{p}'_{n+1} = \mathbf{p}_n$ and $\mathbf{p}'_k = \frac{k}{n+1}\mathbf{p}_{k-1} + \frac{n+1-k}{n+1}\mathbf{p}_k$ for $k = 1, \ldots, n$.

(P5) A Bézier curve can be split into a sequence of Bézier curves of same degree, i.e., we can choose arbitrary $0 = s_0 < s_1 < \cdots < s_m = 1$ and find curves $\mathbf{p}_1, \ldots, \mathbf{p}_m$ such that $\mathbf{p}(s) = \mathbf{p}_i(\frac{s-s_{i-1}}{s_i - s_{i-1}})$ for $s \in [s_{i-1}, s_i]$.

We define the trajectories of moving objects (e.g., vehicles) as Bézier splines. More precisely, we cut the time domain into slices $[t_i, t_{i+1})$ ($i \in \mathbb{N}$, $t_i \in \mathbb{R}$). For the length of each slice we write $\Delta_i = t_{i+1} - t_i$. On each slice, we describe the evolution of an object's position by a Bézier curve, i.e., on the ith slice by

$$\mathbf{p}_i(s) = \sum_{k=0}^{n} \binom{n}{k} \mathbf{p}_{i,k} (1-s)^k s^{n-k} \quad .$$

Hence, the position at time $t \in [t_i, t_{i+1})$ is

$$\mathbf{p}(t) = \mathbf{p}_i \left(\frac{t - t_i}{t_{i+1} - t_i} \right) \quad .$$

Note, that the time on slice i is uniformly mapped to $s := \frac{t-t_i}{t_{i+1}-t_i}$. This implies that when forming derivatives of the position vector, they must be scaled by powers of the time slice length $\Delta_i = t_{i+1} - t_i$, respectively. This means for the velocity vector

$$\mathbf{v}(t) = \dot{\mathbf{p}}(t) = \frac{\mathrm{d}}{\mathrm{d}t} \mathbf{p}_i \left(\frac{t - t_i}{t_{i+1} - t_i} \right) = \frac{1}{\Delta_i} \dot{\mathbf{p}}_i \left(\frac{t - t_i}{t_{i+1} - t_i} \right) \tag{7.2}$$

and for the acceleration

$$\mathbf{a}(t) = \ddot{\mathbf{p}}(t) = \frac{\mathrm{d}^2}{\mathrm{d}t^2} \mathbf{p}_i \left(\frac{t - t_i}{t_{i+1} - t_i} \right) = \frac{1}{\Delta_i^2} \ddot{\mathbf{p}}_i \left(\frac{t - t_i}{t_{i+1} - t_i} \right) \quad . \tag{7.3}$$

Because movement shall be continuous on the whole trajectory, we enforce continuity between time slices with additional constraints. This is visualized in Figure 7.4.

$$\Phi_{\textit{cont-1}} :\equiv \mathbf{p}_{i,n} = \mathbf{p}_{i+1,0} \wedge \underbrace{\frac{n}{\Delta_i}(\mathbf{p}_{i,n} - \mathbf{p}_{i,n-1})}_{\mathbf{v}_{i,n-1}} = \underbrace{\frac{n}{\Delta_{i+1}}(\mathbf{p}_{i+1,1} - \mathbf{p}_{i+1,0})}_{\mathbf{v}_{i+1,0}} \tag{7.4}$$

Note, that the left part can be skipped if the same variables are used to encode both $\mathbf{p}_{i,n}$ and $\mathbf{p}_{i+1,0}$. Furthermore, the right part is a linear constraint if the step size Δ is constant (the term $\frac{n}{\Delta_i}$ resp. $\frac{n}{\Delta_{i+1}}$ may be dropped).

Fact 4 ($\Phi_{\textit{cont-1}}$ ensures continuously differentiable trajectories)**.** *If Equation* (7.4) *holds for all control points* $\mathbf{p}_{i,j}$*, then* $\mathbf{p}(t)$ *is continuously differentiable.*

Proof. By definition, $\mathbf{p}$ is continuously differentiable on each interval $[t_i, t_{i+1})$. Furthermore, by Equation (7.6), it is (for $i > 0$)

$$\lim_{t \nearrow t_i} \mathbf{p}(t) = \mathbf{p}_{i-1}(1) = \mathbf{p}_{i-1,n} = \mathbf{p}_{i,0} = \mathbf{p}_i(0) = \mathbf{p}(t_i)$$

and

$$\lim_{t \nearrow t_i} \dot{\mathbf{p}}(t) = \mathbf{v}_{i-1}(1) = \mathbf{v}_{i-1,n-1} = \mathbf{v}_{i,0} = \mathbf{v}_i(0) = \lim_{t \searrow t_i} \dot{\mathbf{p}}(t) \quad .$$

So, $\mathbf{p}(t)$ is continuous in t_i. □

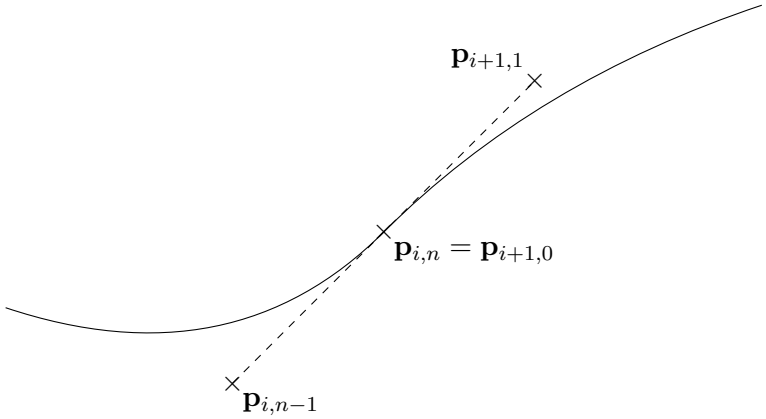


Figure 7.4: Visualization of Equation (7.6): Placing $\mathbf{p}_{i,n-1}$, $\mathbf{p}_{i,n} = \mathbf{p}_{i+1,0}$, and $\mathbf{p}_{i+1,1}$ on a row avoids sharp bends in the trajectory.

As long as the speed is non-zero, Equation (7.4) ensures that the heading does not jump. However, as soon as the speed vanishes, the movement direction (which is used as heading) becomes ambiguous. Therefore, we keep track of the heading angle in stand-still points explicitly by tightening the lower and upper bounds θ^l and θ^u.

$$\Phi_{\textit{cont-2}} :\equiv \mathbf{p}_{i,n} - \mathbf{p}_{i,n-1} = \mathbf{0} \Longrightarrow \theta^l_{i+1} = \theta^u_{i+1} = \theta^l_i = \theta^u_i \tag{7.5}$$

Finally, we combine the above to

$$\boxed{\Phi_{\textit{cont}} :\equiv \Phi_{\textit{cont-1}} \wedge \Phi_{\textit{cont-2}}} \ . \tag{7.6}$$

For the reminder of this chapter, we consider only single time slices, so we can drop some of the subscripts. Furthermore, we use the Bézier spline domain $s \in [0, 1)$ instead of the time domain.

Fact 5 (Properties of the single track model)**.** *The differential equations of the single track model imply*

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} \qquad \dot{\theta} = \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{v^2} \qquad \kappa = \frac{\dot{\theta}}{v} = \frac{\tan\delta}{L}$$

Proof. The facts follow from basic trigonometry. Note that we mirror the equations for the proof.

$$\begin{aligned}\sqrt{\dot{x}^2 + \dot{y}^2} &= \sqrt{v^2\cos^2(\theta) + v^2\sin^2(\theta)} \\ &= v\sqrt{\sin^2(\theta) + \cos^2(\theta)} \\ &= v\sqrt{1} = v\end{aligned}$$

$$\begin{aligned}\frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{v^2} &= \frac{v\cos(\theta)(\dot{v}\sin(\theta) + v\cos(\theta)\dot{\theta}) - (\dot{v}\cos(\theta) - v\sin(\theta)\dot{\theta})v\sin(\theta)}{v^2} \\ &= \frac{v^2\cos^2(\theta)\dot{\theta} + v\dot{\sin}(\theta)v\cos(\theta) - v\dot{\cos}(\theta)v\sin(\theta) + v^2\sin^2(\theta)\dot{\theta}}{v^2} \\ &= \frac{v^2\dot{\theta}(\cos^2(\theta) + \sin^2(\theta))}{v^2} = \dot{\theta}\end{aligned}$$

$$\begin{aligned}\kappa &= \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{(\dot{x}^2 + \dot{y}^2)^{3/2}} \\ &= \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{v^3} = \frac{\dot{\theta}}{v} = \frac{\tan\delta}{L}\end{aligned}$$

□

7.3 Approximation with Control Points

Recall, that by using Construction 1 ($tr_c(\phi, V_O)$), followed by quantifier elimination, a spatial view ϕ is translated to a monotone Boolean combination of linear inequalities. Based on the previous sections, we group the coefficients into three groups, i.e., we write the inequalities as

$$\sum_{i\in I_1} a_i A_i + \sum_{i\in I_2} b_i B_i + \sum_{i\in I_3} c_i C_i \bowtie d$$

with finite index sets I_1, I_2, I_3, constants $a_i, b_i, c_i \in \mathbb{R}$, and

$$\begin{aligned}A_i &\in \mathfrak{A} := \{x_o, y_o \mid o \in V_{veh}\} \\ B_i &\in \mathfrak{B} := \{bbxmin_o, bbxmax_o, bbymin_o, bbymax_o, \theta_o \mid o \in V_{veh}\} \\ C_i &\in \mathfrak{C} := \{v_o, acc_o \mid o \in V_{veh}\}\end{aligned}$$

and $\bowtie \in \{<, \leq\}$. In order to reduce the use of subscripts during this chapter, we denote the attribute variables $\mathtt{var}_{o.a}$ introduced by $tr_c(\phi, V_O)$ by the shorter a_o. The set V_{veh} refers to object variables of our vehicle type and term d may be an arbitrary expression that does not contain terms from $\mathfrak{A} \cup \mathfrak{B} \cup \mathfrak{C}$. For example, d may be a constant or refer to attributes of (non-moving) obstacles or lanes. Note, that we assume the value of d to be constant on each time slice (when considering the whole trajectory, it is piecewise constant).

The two groups introduced above indicate that the attributes are handled differently: Attributes from the set $\mathfrak{A}$ have an exact representation as Bézier splines. We encode $A_i \in \mathfrak{A}$ as a Bézier curve $A_i(s)$. Attributes from the set $\mathfrak{B}$ are approximated by a global lower bound B_i^l and an upper bound B_i^u for each time slice, and variables from the set $\mathfrak{C}$ will be approximated by an upper and lower Bernstein polynomial $C_i^l(s)$ resp. $C_i^u(s)$. Here, the Bernstein polynomials v_o^l and v_o^u for speed of vehicle $o \in V_{veh}$ are of degree $n-1$, and acc_o^l and acc_o^u are of degree $n-2$.

Definition 31 (Free variables for bounding vehicle attributes)**.** *Given a fixed integer $n \geq 2$, the SMT formulae constructed in this chapter use the following free variables for each vehicle o:*

1. *$x_{o,k}, y_{o,k}$ for each $k = 0, 1, \ldots, n$, describing polynomials*

$$x_o(s) := \sum_{k=0}^{n-1} x_{o,k} \binom{n}{k} (1-s)^k s^{n-k} \text{ and } y_o(s) := \sum_{k=0}^{n-1} y_{o,k} \binom{n}{k} (1-s)^k s^{n-k}$$

2. *$bound_o^l$, $bound_o^u$ for each $bound \in \{bbxmin, bbxmax, bbymin, bbymax, \theta\}$*

3. $v^l_{o,k}, v^u_{o,k}$ *for each* $k = 0, 1, \ldots, n-1$ *describing polynomials*

$$v^l_o(s) := \sum_{k=0}^{n-1} v^l_{o,k} \binom{n-1}{k} (1-s)^k s^{n-1-k}$$

and

$$v^u_o(s) := \sum_{k=0}^{n-1} v^u_{o,k} \binom{n-1}{k} (1-s)^k s^{n-1-k}$$

4. $acc^l_{o,k}, acc^u_{o,k}$ *for each* $k = 0, 1, \ldots, n-2$ *describing polynomials*

$$acc^l_o(s) := \sum_{k=0}^{n-2} acc^l_{o,k} \binom{n-2}{k} (1-s)^k s^{n-2-k}$$

and

$$acc^u_o(s) := \sum_{k=0}^{n-1} acc^u_{o,k} \binom{n-2}{k} (1-s)^k s^{n-2-k}$$

These variables are introduced for every time slice.

Using these variables and Bernstein polynomials, we translate the linear inequalities of the form above into parameterized inequalities $\phi(s)$ of the form

$$\sum_{i \in I_1} a_i A_i(s) + \sum_{i \in I_2} b_i \bar{B}_i \bowtie d$$

where $A_i(s)$ is one of the polynomials introduced above (i.e., $A_i(s)$ is one of $x_o(s)$, $y_o(s)$, $v^l_o(s)$, $v^u_o(s)$, $acc^l_o(s)$, or $acc^u_o(s)$ subsuming attributes from the sets $\mathfrak{A}$ and $\mathfrak{C}$), and the $\bar{B}_i$ are one of the variables $bound^l_o, bound^u_o$ (for some $bound \in \{bbxmin, bbxmax, bbymin, bbymax, \theta\}$, i.e., approximating an attribute from the set $\mathfrak{B}$). We can evaluate such a constraint at fixed points $s \in [0, 1]$. We translate such a constraint with the following construction.

Construction 3 (Sufficient constraint (tr'_s))**.** *Given a set* V_O *of object variables, a sufficient constraint for a spatial view constraint* ϕ *is constructed as*

$$tr'_s(\phi, V_O) :\equiv tr'_2(tr'_1(\mathsf{QE}(tr_c(\phi, V_O))))$$

split into steps tr_c *(Construction 1), quantifier elimination (*QE*),* tr'_1 *(Construction 4), and* tr'_2 *(Construction 5).*

The reason for splitting is that we will use the construction tr'_2 later on in Section 7.4.4.

Construction 4 (Safe approximation by lower and upper bounds (tr'_1))**.** *For invariants* ϕ *as defined above, define* tr'_1 *inductively as follows:*

- *For linear inequalities* $\phi' \equiv \sum_{i\in I_1} a_i A_i + \sum_{i\in I_2} b_i B_i + \sum_{i\in I_3} c_i C_i \bowtie d$ *(with finite index sets* I_1, I_2, I_3*, constants* a_i, b_i, c_i*, vehicle attributes* $A_i \in \mathfrak{A}, B_i \in \mathfrak{B}, C_i \in \mathfrak{C}$*,* $\bowtie \in \{<, \leq\}$*, and expression* d *not containing attributes from* $\mathfrak{A} \cup \mathfrak{B} \cup \mathfrak{D}$*) it is*

$$tr'_1(\phi') :\equiv \forall s \in [0,1) : \sum_{i\in I_1} a_i A_i(s) + \sum_{i\in I_2} b_i \bar{B}_i + \sum_{i\in I_3} c_i \bar{C}_i(s) \bowtie d$$

with

$$\bar{B}_i = \begin{cases} B_i^l & \text{if } b_i \leq 0 \\ B_i^u & \text{else} \end{cases} \qquad \text{and} \qquad \bar{C}_i = \begin{cases} C_i^l & \text{if } c_i \leq 0 \\ C_i^u & \text{else} \end{cases} .$$

- *Boolean connectives propagate naturally, i.e.,* $tr'_1(\phi_1 \wedge \phi_2) :\equiv tr'_1(\phi_1) \wedge tr'_1(\phi_2)$ *and* $tr'_1(\phi_1 \vee \phi_2) :\equiv tr'_1(\phi_1) \vee tr'_1(\phi_2)$*.*

W.l.o.g we can assume that all A_i and $\bar{C}_i$ are Bernstein polynomials of same degree n with control points $A_{i,j}$ resp. $\bar{C}_{i,j}$ (for $i = 0, 1, \ldots, n$). This can be ensured by increasing the degree if needed (we denote the control points that are result of degree elevation to n by $A_{i,j}^{(n)}$ resp. $C_{i,j}^{(n)}$). Then, from the output of construction tr'_1, construction tr'_2 creates safe approximations using the control points.

Construction 5 (Safe approximation by control polygon (tr'_2))**.** *Define* tr'_2 *inductively as follows:*

- *For linear inequalities, parameterized by* s*, of the form* $\phi'(s) \equiv \sum_{i\in I_1} a_i A_i(s) + \sum_{i\in I_2} b_i \bar{B}_i \bowtie d$ *where* I_1, I_2 *are finite index sets,* $A_i(s) := \sum_{k=0}^{m_i} \binom{m_i}{k} A_{i,k}(1-s)^k s^{m_i-k}$ *are Bernstein polynomials of (possibly different) degree* $n - 2 \leq m_i \leq n$ *(declared in Definition 31, items 1, 3, and 4),* $\bar{B}_i$ *are free variables (declared in Definition 31, item 2),* $\bowtie \in \{<, \leq\}$*, and* $d \in \mathbb{R}$ *a constant, construct*

$$tr'_2(\forall s \in [0,1) : \phi'(s)) :\equiv \bigwedge_{k=0}^{n} \sum_{i\in I_1} a_i A_{i,k}^{(n)} + \sum_{i\in I_2} b_i \bar{B}_i \bowtie_k d$$

with $\bowtie_0 = \bowtie$ *and* $\bowtie_k = \leq$ *for* $k > 0$*. The* $A_{i,k}^{(n)}$ *are the control points of* $A_i(s)$ *after degree elevation to* n*, i.e.,*

  - $A_{i,k}^{(n)} = x_{i,k}$ *for* $k = 0, 1, \ldots, n$ *if* $A_i = x_o$ *for some vehicle* o*.*
  - *Analogously,* $A_{i,k}^{(n)} = y_{i,k}$ *for* $k = 0, 1, \ldots, n$ *if* $A_i = y_o$ *for some vehicle* o*.*
  - $A_{i,0}^{(n)} = v_{i,0}^l$*,* $A_{i,n}^{(n)} = v_{i,n-1}^l$*, and* $A_{i,k}^{(n)} = \frac{n-k}{n} v_{i,k-1}^l + \frac{k}{n} v_{i,k}^l$ *for* $k = 1, 2 \ldots, n-1$ *if* $A_i = v_o^l$ *for some vehicle* o*.*
  - *Analogously for* $A_i \in \{v_o^u, acc_o^l, acc_o^u \mid o \in V_{Veh}\}$*, except that for* acc_o^l *and* acc_o^u *degree elevation is applied twice.*

- $tr'_2(\forall s \in [0,1) : \bigwedge_j \phi'_j) :\equiv \bigwedge_j tr'_2(\forall s \in [0,1) : \phi'_j)$

- *For monotone boolean formulae* ϕ_1*,* ϕ_2 *over formulae of the form* $\forall s \in [0,1) : \phi'(s)$ *, it is* $tr'_2(\phi_1 \wedge \phi_2) :\equiv tr'_2(\phi_1) \wedge tr'_2(\phi_2)$ *and* $tr'_2(\phi_1 \vee \phi_2) :\equiv tr'_2(\phi_1) \vee tr'_2(\phi_2)$*.*

The second bullet point has been introduced for translating the constraints $\Phi_H(\bar{\theta},\eta)$, $\Phi_{BB}(\bar{\theta},\eta)$, $\Phi_v(\bar{\theta},\eta)$, $\Phi_{acc}(\bar{\theta},\eta)$, and $\Phi_{valid}(\bar{\theta},\eta)$. Note, that the Bernstein polynomials denoted by A_i in the above construction correspond to vehicle attributes from the sets $\mathfrak{A} \cup \mathfrak{C}$.

Example 7. *Figure 7.5 shows a spatial view that specifies a time-to-collision $TTC(a,b) \geq 2\,\mathrm{s}$ between two vehicles a and b. It translates to*

$$\begin{aligned}\phi \equiv\ & (b.\underline{x} - a.\overline{x}) \geq (a.v - b.v) \cdot 2\,\mathrm{s} \\ & \wedge a.\underline{x} < a.\overline{x} < b.\underline{x} < b.\overline{x} \wedge a.\underline{y} < a.\overline{y} \wedge b.\underline{y} < b.\overline{y}\ .\end{aligned}$$

After substituting the bounding box bounds $\underline{x}, \overline{x}, \underline{y}, \overline{y}$, the construction yields:

$$\begin{aligned}\phi_1 \equiv tr_c(\phi,\{a,b\}) \equiv\ & (x_b + bbxmin_b) - (x_a + bbxmax_a) \geq 2(v_a - v_b) \\ & \wedge x_a + bbxmax_a < x_b + bbxmin_b \\ & \wedge bbxmin_a < bbxmax_a \wedge bbymin_a < bbymax_a \\ & \wedge bbxmin_b < bbxmax_b \wedge bbymin_b < bbymax_b \\ \phi_2 \equiv tr_1'(\phi_1) \equiv\ & \forall s \in [0,1): \\ & (x_b(s) + \mathtt{bbxmin}_b^l) - (x_a(s) + \mathtt{bbxmax}_a^u) \\ & \qquad \geq 2\,\mathrm{s} \cdot (v_a^u(s) - v_b^l(s)) \\ & \wedge x_a(s) + \mathtt{bbxmax}_a^u < x_b(s) + \mathtt{bbxmin}_b^l \\ & \wedge \mathtt{bbxmin}_a^u < \mathtt{bbxmax}_a^l \wedge \mathtt{bbymin}_a^u < \mathtt{bbymax}_a^l \\ & \wedge \mathtt{bbxmin}_b^u < \mathtt{bbxmax}_b^l \wedge \mathtt{bbymin}_b^u < \mathtt{bbymax}_b^l \\ tr_s'(\phi) \equiv tr_2'(\phi_2) \equiv\ & \bigwedge_{k=0}^{n} \big((x_{k,b} + \mathtt{bbxmin}_b^l) - (x_{k,a} + \mathtt{bbxmax}_a^u) \\ & \qquad \geq 2\,\mathrm{s} \cdot (v_{k,a}^{u,(n)} - v_{k,b}^{l,(n)})\big) \\ & \wedge x_{0,a} + \mathtt{bbxmax}_a^u < x_{0,b} + \mathtt{bbxmin}_b^l \\ & \wedge \bigwedge_{k=1}^{n} x_{k,a} + \mathtt{bbxmax}_a^u \leq x_{k,b} + \mathtt{bbxmin}_b^l \\ & \wedge \mathtt{bbxmin}_a^u < \mathtt{bbxmax}_a^l \wedge \mathtt{bbymin}_a^u < \mathtt{bbymax}_a^l \\ & \wedge \mathtt{bbxmin}_b^u < \mathtt{bbxmax}_b^l \wedge \mathtt{bbymin}_b^u < \mathtt{bbymax}_b^l\end{aligned}$$

Here, $v_{k,a}^{u,(n)}$ and $v_{k,b}^{l,(n)}$ are the control points of v_a^u respectively v_b^l after degree elevation from $n-1$ to n, i.e., $v_{0,a}^{u,(n)} = v_{0,a}^u$, $v_{n,a}^{u,(n)} = v_{n-1,a}^u$, and $v_{k,a}^{u,(n)} = \frac{k}{n} v_{k-1,a}^u + \frac{n-k}{n} v_{k,a}^u$ for $k = 1,\ldots,n-1$ (analogous for $v_{k,b}^{l,(n)}$).

Lemma 5 (Approximation by control polygon)**.** *For constraints ϕ handled by tr_2' holds $tr_2'(\phi) \Rrightarrow \phi$.*

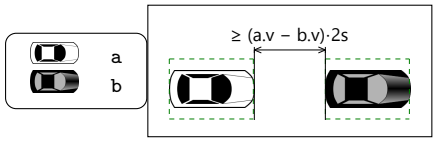


Figure 7.5: Spatial view for specifying a time-to-collision $TTC(a, b) \geq 2\,\mathrm{s}$

Proof. We show this by structural induction. As an induction start, we consider formulae formulae of the form $\phi'(s)$ as in the construction. First of all, it is

$$\sum_{i\in I} a_i A_i(s) + \sum_{i\in I_2} b_i \bar{B}_i \bowtie d \Leftrightarrow \underbrace{-d - \sum_{i\in I_2} b_i \bar{B}_i + \sum_{i\in I} a_i A_i(s)}_{=:D(s)} \bowtie 0 \ .$$

By property (P2) of Bézier curves (see Section 7.2), D is a Bézier curve with control points

$$D_k := -d - \sum_{i\in I_2} b_i \bar{B}_i + \sum_{i\in I_1} a_i A_{i,k} \ ,$$

$k = 0, 1, \ldots, n$. By construction,

$$tr_2'(\forall s \in [0,1) : \phi'(s)) \equiv D_0 \bowtie 0 \wedge \bigwedge_{k=1}^{n} D_k \leq 0$$

with $\bowtie \in \{<, \leq\}$. Recall that $D(s) = \sum_{k=0}^{n} D_k \mathcal{B}_k^n(s)$ with basis polynomials $\mathcal{B}_k^n(s) := \binom{n}{k}(1-s)^k s^{n-k}$. Fix some $s \in [0,1)$ and assume $tr_2'(\forall s \in [0,1) : \phi'(s))$ evaluates to *true*. Because $\mathcal{B}_0^n(s) > 0$ and $\mathcal{B}_k^n(s) \geq 0$ for $k \neq 0$, the constraints $D_0 \bowtie 0$ and $D_k \leq 0$ (for $k \neq 0$) imply $D_0 \mathcal{B}_0^n(s) \bowtie 0$ and $D_k \mathcal{B}_k^n(s) \leq 0$. For both cases ($\bowtie = <$ and $\bowtie = \leq$) this results in $D(s) \bowtie 0$.This holds for all $s \in [0,1)$, so the lemma holds for the induction start. Because $\wedge$ and $\vee$ propagate naturally through tr_2', the induction step is straight forward. □

7.4 Bounding Constraints

Recall that we consider invariants ϕ that are monotone Boolean combinations of linear inequalities of the form

$$\sum_{i\in I_1} a_i A_i + \sum_{i\in I_2} b_i B_i + \sum_{i\in I_3} c_i C_i \bowtie d$$

with finite index sets I_1, I_2, I_3, $\bowtie \in \{<, \leq\}$, and object attributes $A_i \in \mathfrak{A}$, $B_i \in \mathfrak{B}$, $C_i \in \mathfrak{C}$ as defined in Section 7.3.

As already briefly explained in the previous section, we choose safe piecewise linear approximations for the upper and lower bounds. For sampling the approximations, we choose some *primary direction of movement* $\bar{\theta}$ for the vehicle in form of a yaw angle with respect to the positive X axis. The actual heading θ may diverge from $\bar{\theta}$ by at most $\pm\eta$ with $\eta < \frac{\pi}{2}$. Equation (7.1) combines the different approximation to a big disjunction. The term Φ_H ensures that the correct approximation matching the actual heading is chosen. In the following subsections, we describe the different approximations.

In order to ease the following calculations, we fix some $\bar{\theta}$ and rotate coordinate system by $-\bar{\theta}$. We define accordingly rotated versions (X_k, Y_k) of the trajectory control points (x_k, y_k):

$$\begin{pmatrix} X_k \\ Y_k \end{pmatrix} = R(-\bar{\theta}) \begin{pmatrix} x_k \\ y_k \end{pmatrix} = \begin{pmatrix} x_k \cos\bar{\theta} + y_k \sin\bar{\theta} \\ -x_k \sin\bar{\theta} + y_k \cos\bar{\theta} \end{pmatrix}$$

The functions $X(s)$ and $Y(s)$ denote the rotated versions of $x(s)$ and $y(s)$.

In order to avoid writing of many similar formulas, the operator $\pm$ abbreviates that both terms with $+$ and $-$ shall be included in a set. For example, the set $\{\pm a \pm b\}$ abbreviates $\{a+b, a-b, -a+b, -a-b\}$.

7.4.1 The Heading Constraint

According to the single track model, it is $\dot{Y}(s) = \dot{X}(s)\tan(\theta(s) - \bar{\theta})$, provided $\dot{X}(s) > 0$. Because the tangents is monotone on the interval $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, the assumption $\forall s \in [0,1) : \theta(s) \in [\bar{\theta} - \eta, \bar{\theta} + \eta]$ can be expressed equivalently as

$$\boxed{\Phi_H(\bar{\theta}, \eta) :\equiv \forall s \in [0,1) : \dot{X}(s) \geq 0 \wedge |\dot{Y}(s)| \leq \dot{X}(s)\tan\eta \wedge \theta^l \leq \bar{\theta} - \eta \wedge \theta^u \geq \bar{\theta} + \eta}\ .$$

Lemma 6 (Heading bounds)**.** *Whenever, for some $s \in [0,1)$ and function $\theta : [0,1) \to \mathbb{R}$,*

$$\exists u > 0 : \dot{x}(s) = u\cos(\theta(s)) \wedge \dot{y}(s) = u\sin(\theta(s))\ ,$$

then, for arbitrary $\bar{\theta} \in \mathbb{R}$, $\eta \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, the constraint $\Phi_H(\bar{\theta}, \eta)$ implies

$$\exists k \in \mathbb{Z} : \bar{\theta} - \eta \leq \theta(s) + 2k\pi \leq \bar{\theta} + \eta\ .$$

Proof. Rotating the Beziér control points results in a rotation of the curve. So, the vector $(\dot{X}(s), \dot{Y}(s))^T$ can also be expressed as a rotation of the vector $(\dot{x}(s), \dot{y}(s))^T$ by $-\bar{\theta}$. We drop the function argument s and get:

$$\begin{aligned}
\dot{X} &= \dot{x}\cos\bar{\theta} + \dot{y}\sin\bar{\theta} \\
&= u\cos\theta\cos\bar{\theta} + u\sin\theta\sin\bar{\theta} \\
&= u\cos(\theta - \bar{\theta}) = u\cos(\theta + 2k\pi - \bar{\theta}) \\
\dot{Y} &= \dot{y}\cos\bar{\theta} - \dot{x}\sin\bar{\theta} \\
&= u\sin\theta\cos\bar{\theta} - u\cos\theta\sin\bar{\theta} \\
&= u\sin(\theta + 2k\pi - \bar{\theta})
\end{aligned}$$

Now we prove the lemma by contra position. Assume there is no integer k such that $\bar{\theta} - \eta \leq \theta + 2k\pi \leq \bar{\theta} + \eta$. Hence, $|\theta + 2k\pi - \theta| \geq \eta$. W.l.o.g. we assume $|\theta + 2k\pi - \theta| \leq \pi$. In case $|\theta + 2k\pi - \theta| \in (\pi/2, \pi]$, the above equation (with $u > 0$) implies $\dot{X} < 0$. This violates $\Phi_H(\bar{\theta}, \eta)$. In the other case $|\theta + 2k\pi - \theta| < \pi$, it is $|\dot{Y}| = |u\sin(\theta + 2k\pi - \bar{\theta})| = |u\cos(\theta + 2k\pi - \bar{\theta})\tan(\theta + 2k\pi - \bar{\theta})| = |\dot{X}\tan(\theta + 2k\pi - \bar{\theta})| > |\dot{X}\tan\eta|$ (because of monotony of tan) which violates the other part of $\Phi_H(\bar{\theta}, \eta)$. In the remaining third case, $|\theta + 2k\pi - \theta| = \pi$, it is violated because of $\dot{Y} = 1$ and $\dot{X} = 0$. □

The opposite direction holds if the said $u > 0$ exists for all $s \in [0,1)$, i.e., the vehicle has non-zero speed everywhere on the spline segment.

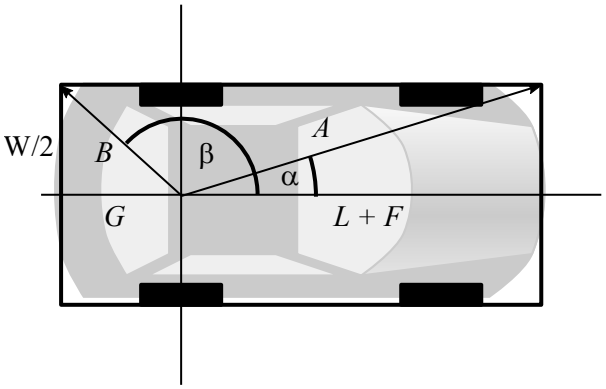


Figure 7.6: Polar coordinates of bounding box corners

7.4.2 Approximating the Bounding Box

For an easy calculation of the rotation-dependent bounding box bounds, we translate the position of the (unrotated) bounding box corners to polar coordinates (Figure 7.6)

$$A = \sqrt{\frac{W^2}{4} + (L+F)^2} \qquad B = \sqrt{\frac{W^2}{4} + G^2}$$
$$\alpha = \arccos\frac{L+F}{A} \qquad \beta = \arccos\frac{-G}{B}$$

The bounding box corners can be calculated as

$$bbxmin(\theta) = \min\{A\cos(\theta \pm \alpha), B\cos(\theta \pm \beta)\} \tag{7.7a}$$
$$bbxmax(\theta) = \max\{A\cos(\theta \pm \alpha), B\cos(\theta \pm \beta)\} \tag{7.7b}$$
$$bbymin(\theta) = \min\{A\sin(\theta \pm \alpha), B\sin(\theta \pm \beta)\} \tag{7.7c}$$
$$bbymax(\theta) = \max\{A\sin(\theta \pm \alpha), B\sin(\theta \pm \beta)\} \tag{7.7d}$$

and thus the lower and upper bounds as

$$bound^l(\bar{\theta}, \eta) = \min_{\theta \in [\bar{\theta}-\eta, \bar{\theta}+\eta]} bound(\theta)$$
$$bound^u(\bar{\theta}, \eta) = \max_{\theta \in [\bar{\theta}-\eta, \bar{\theta}+\eta]} bound(\theta)$$

for $bound \in BB := \{bbxmin, bbxmax, bbymin, bbymax\}$. It is not necessary to check the entire range $\bar{\theta} - \eta \leq \theta \leq \bar{\theta} + \eta$. Because sin and cos have their extrema at multiples of π, all bounding box bounds have their extrema at θ in the finite set

$$\mathcal{E}(\bar{\theta}, \eta) := \{\bar{\theta} - \eta, \bar{\theta} + \eta\} \cup \left(\left\{k\frac{\pi}{2} \pm \xi \;\middle|\; k \in \mathbb{Z} \wedge \xi \in \{0, \alpha, \beta\}\right\} \cap [\bar{\theta} - \eta, \bar{\theta} + \eta]\right)$$

which simplifies the bound calculation to

$$bound^l(\bar{\theta}, \eta) = \min_{\theta \in \mathcal{E}(\bar{\theta}, \eta)} bound(\theta)$$
$$bound^u(\bar{\theta}, \eta) = \max_{\theta \in \mathcal{E}(\bar{\theta}, \eta)} bound(\theta) \;\; .$$

So, the constraint Φ_{BB} for the bounding box bounds is

$$\boxed{\Phi_{BB}(\bar{\theta},\eta) :\equiv \bigwedge_{bound\in BB} \left\{ \begin{matrix} bound^l \leq bound^l(\bar{\theta},\eta) \\ bound^u \geq bound^u(\bar{\theta},\eta) \end{matrix} \right\}}$$

with BB, $bound^l(\bar{\theta},\eta)$, and $bound^u(\bar{\theta},\eta)$ defined as above.

Fact 6 (Properties of Φ_{BB})**.** *If $\Phi_{BB}(\bar{\theta},\eta)$ with $-\pi/2 < \eta < \pi/2$ holds, then $bound^l, bound^u$ (for $bound \in \{bbxmin, bbxmax, bbymin, bbymax\}$) are lower/upper bounds for the bounding box dimensions according to Figure 7.2 of a rectangular object rotated by $\theta \in [\bar{\theta}-\eta, \bar{\theta}+\eta]$.*

The proof is omitted here because it is a replication of the explanations above in this section.

7.4.3 Bounding Curvature, Acceleration, and Speed

In the previous section, the Beziér curve trajectory model has been derived from the simple single track model without an argument about the validity of the model. The trajectories generated are valid if, for some single track model of the vehicle class,

- the trajectory can be generated within that model, and
- the single track model is valid for the trajectory.

The first criterion is true whenever we can find time-dependent velocity and steering values $v(t)$ and $\delta(t)$ such that the single track model generates $\mathbf{p}(t)$. Furthermore, $v(t)$, $\dot{v}(t)$, and $\delta(t)$ must lay within the speed, acceleration, and steering bounds of the vehicle class. The second criterion is met if the lateral acceleration along the trajectory is below $a_{lat,\max} \approx 4\text{ms}^{-2}$ [110]. In the following, the Beziér curve trajectory model is evaluated with respect to these criteria. Furthermore, parameter bounds are determined that meet the above criteria.

In the following, we focus on the steering angle δ and the lateral acceleration a_{lat}. Both depend on the curvature κ:

$$\delta(s) = \arctan(\kappa(s)L) \qquad a_{lat}(s) = |\kappa(t)|v^2(s)$$

Validity constraints are expressed as:

$$\boxed{\Phi_{valid}(\bar{\theta},\eta) :\equiv \forall s \in [0,1) : \bigwedge \left\{ \begin{matrix} \ddot{X}(s) = 0 \quad (\text{for } \eta > 0 \text{ only}) \\ \ddot{Y}(s) \leq a_{lat,\max}\Delta^2 \\ \ddot{Y}(s) \geq -a_{lat,\max}\Delta^2 \\ \dot{X}(s) \geq \dfrac{2(n-1)L\tan(\eta)}{\tan(\delta_{\max})} \end{matrix} \right\}}$$

Lemma 7 (Correctness of validity constraints[12, Lemma 2])**.** *If, for $\bar{\theta} \in \mathbb{R}, \eta \in [0, \pi/2)$, both the heading constraint $tr_2'(\Phi_H(\bar{\theta},\eta))$ (on the control polygon) and the above validity constraint $\Phi_{valid}(\bar{\theta},\eta)$ hold on a time slice, then also $|\kappa(s)| \leq \frac{\tan(\delta_{\max})}{L}$ and $|a_{lat}(s)| \leq a_{lat,\max}$ for all $s \in [0,1)$.*

Note, that the lemma requires that the heading constraint is applied to the control polygon of $\dot{X}$, respectively $\dot{Y}$, as done by the construction tr_2'.

Proof. Recall that the curvature of a trajectory is given by

$$\kappa(s) = \frac{\dot{X}(s)\ddot{Y}(s) - \ddot{X}(s)\dot{Y}(s)}{(\dot{X}^2(s) + \dot{Y}^2(s))^{3/2}} \ .$$

By the inequalities $\sqrt{\dot{X}^2(s) + \dot{Y}^2(s)} \geq \dot{X}(s) > 0$ and $|\dot{Y}(s)| \leq \dot{X}(s)\tan\eta$ (by $\Phi_H(\bar{\theta}, \eta)$), the curvature is bounded by

$$|\kappa(s)| \leq \frac{|\ddot{Y}(s)| + |\ddot{X}(s)\tan\eta|}{\dot{X}^2(s) + \dot{Y}^2(s)} \ .$$

As described in Section 7.4, it is $v(s) = \frac{1}{\Delta}\sqrt{\dot{X}^2(s) + \dot{Y}^2(s)}$, so the lateral acceleration $a_{lat}(s) = \kappa(s)v^2(s)$ is bounded by

$$a_{lat}(s) \leq \frac{|\ddot{Y}(s)| + |\ddot{X}(s)\tan\eta|}{\Delta^2}$$

and the curvature by

$$|\kappa(s)| \leq \frac{|\ddot{Y}(s)| + |\ddot{X}(s)\tan\eta|}{\Delta^2 v^2(s)} \ .$$

In the case $\eta = 0$, the bound $|\dot{Y}(s)| \leq \dot{X}(s)\tan\eta$ implies $\dot{Y}(s) = 0$ for all s, so $\ddot{Y}(s) = 0$. As a consequence of the above equations, both $a_{lat}(s) = 0$ and $\kappa(s) = 0$.

For $\eta > 0$, the constraint $\Phi_{valid}(\bar{\theta}, \eta)$ forces $\ddot{X}(s) = 0$. Hence, we can drop the coefficients containing $|\ddot{X}(s)|$ from the above equations. This simplifies the approximation for a_{lat} to

$$a_{lat}(s) \leq \frac{|\ddot{Y}(s)|}{\Delta^2} \ .$$

Furthermore, it means that $\dot{X}(s)$ is constant. Recall that $\dot{X}$ and $\dot{Y}$ are Bernstein polynomials of degree $n-1$ with $\dot{X}_0 = \dot{X}_1 = \cdots = \dot{X}_{n-1} = \dot{X}(s) > 0$. The constraint $tr_2'(\Phi_H(\bar{\theta}, \eta))$ implies $\bigwedge_{i=0}^{n-1} |\dot{Y}_i| \leq \dot{X}_i$, so control points of $\dot{Y}$ can be bounded by $|\dot{Y}_i| \leq \dot{X}(s)\tan(\eta)$ for $i = 0, 1, \ldots, n-1$. This implies that $\ddot{Y}$ is a Bernstein polynomial of degree $n-2$ with

$$|\ddot{Y}_i| = (n-1)|\dot{Y}_{i+1} - \dot{Y}_i| \leq (n-1)(|\dot{Y}_{i+1}| + |\dot{Y}_i|) \leq 2(n-1)\dot{X}(s)\tan(\eta) \ .$$

Using $\dot{X}(s)$ as a lower bound for $\Delta \cdot v(s)$, it is

$$|\kappa(s)| \leq \frac{2(n-1)\dot{X}(s)\tan(\eta)}{\dot{X}^2(s)} = \frac{2(n-1)\tan(\eta)}{\dot{X}(s)} \ .$$

Inserting the bound

$$\dot{X}(s) \geq \frac{2(n-1)L\tan(\eta)}{\tan(\delta_{\max})}$$

implied by $\Phi_{valid}(\bar{\theta}, \eta)$ for $\dot{X}(s)$, this leads to

$$|\kappa(s)| \leq \frac{\tan(\delta_{\max})}{L} \ .$$

Hence, the above validity constraint $\Phi_{valid}(\bar{\theta}, \eta)$ implies $|a_{lat}(s)| \leq a_{lat,\max}$ and $|\kappa(s)| \leq \frac{\tan(\delta_{\max})}{L}$. □

Similar approximations are used to encode bounds for velocity v and longitudinal acceleration *acc*. It is

$$\Phi_v(\bar{\theta},\eta) :\equiv \forall s \in [0,1) : \begin{cases} v^l(s) = v^u(s) = \Delta^{-1}\dot{X}(s) & \text{for } \eta = 0 \\ \bigwedge \begin{cases} v^l(s) \le \Delta^{-1}\dot{X}(s) \\ v^u(s) \ge \Delta^{-1}\dot{X}(s) + \Delta^{-1}\tan\left(\frac{\eta}{2}\right)|\dot{Y}(s)| \end{cases} & \text{else} \end{cases}$$

and

$$\Phi_{acc}(\bar{\theta},\eta) :\equiv \forall s \in [0,1) \begin{cases} acc^l(s) = acc^u(s) = \Delta^{-2}\ddot{X}(s) & \text{for } \eta = 0 \\ acc^l(s) \le -|\Delta^{-2}\ddot{Y}(s)| \wedge acc^u(s) \ge |\Delta^{-2}\ddot{Y}(s)| & \text{else} \end{cases}.$$

Lemma 8 (Correctness of velocity bounds [12, Lemma 3])**.** *If, for some $\bar{\theta} \in \mathbb{R}, \eta \in [0, \pi/2)$, both the heading constraint $\Phi_H(\bar{\theta},\eta)$ and the above constraint $\Phi_v(\bar{\theta},\eta)$ hold, then $v^l(s) \le v(s) \le v^u(s)$ for all $s \in [0,1]$.*

Proof. Fix some relative time point $s \in [0,1]$ in the time frame. Define $v_X := \Delta^{-1}\dot{X}(s)$, $v_Y = \Delta^{-1}\dot{Y}(s)$, and recall that $v(s) = \sqrt{v_x^2 + v_y^2}$.

In case $\eta = 0$, the constraint $|\dot{Y}| \le \dot{X}\tan\eta$ contained in $\Phi_H(\bar{\theta},\eta)$ enforces $v_Y = 0$. Hence, $v(s) = v_X$ and by definition $v^l(s) = v^u(s) = v_X = v(s)$.

The remaining case is $\eta > 0$. If $v_Y = 0$ then $\tan\left(\frac{\eta}{2}\right)|\dot{Y}(s)| = 0$, so, by definition of $\Phi_v(\bar{\theta},\eta)$, we have $v^l(s) \le= v_X = s$ and $v^u(s) \ge v_X = s$. In case $v_X = 0$, the constraint $|\dot{Y}| \le \dot{X}\tan\eta$ leads to $v_Y = 0$, too. So, we have to show the case $v_X > 0$ and $v_Y \neq 0$. We choose some angle $\eta' \in [-\pi, +\pi]$ such that $v_X = v\cos\eta'$ and $v_Y = v\sin\eta'$. Note, that by $v_X > 0$ and $0 < |v_Y| \le v_X\tan\eta$ we know $0 < |\eta'| \le \eta$. In any case, $v^l(s) \le v_X \le v(s)$, so we only have to show $v(s) \le v^u(s)$:

$$\begin{aligned} v(s) &= v(s)\cos(\eta') + v(s)(1-\cos(\eta')) \\ &= v(s)\cos(\eta') + \frac{1-\cos(\eta')}{\sin(\eta')}v(s)\sin(\eta') \\ &= v_X + \frac{1-\cos(\eta')}{\sin(\eta')}v_Y \\ &= v_X + \tan\left(\frac{\eta'}{2}\right)v_Y \end{aligned}$$

and by monotony of tan on the interval $(-\pi, +\pi)$

$$\le v_X + \tan\left(\frac{\eta}{2}\right)|v_Y| \qquad = \Delta^{-1}\dot{X}(s) + \Delta^{-1}\tan\left(\frac{\eta}{2}\right)|\dot{Y}(s)| \le v^u(s)$$

by definition of $\Phi_v(\bar{\theta},\eta)$. As a consequence, the lemma holds. □

Lemma 9 (Correctness of acceleration bounds[12, Lemma 4])**.** *If, for some $\bar{\theta} \in \mathbb{R}, \eta \in [0, \pi/2)$, both the heading constraint $\Phi_H(\bar{\theta},\eta)$, the constraint $\Phi_{valid}(\bar{\theta},\eta)$, and the above constraint $\Phi_v(\bar{\theta},\eta)$ hold, then $acc^l(s) \le acc(s) \le acc^u(s)$.*

Proof. The acceleration $acc(s)$ is measured in movement direction, i.e,

$$acc(s) = \frac{\ddot{X}(s)\dot{X}(s) + \ddot{Y}(s)\dot{Y}(s)}{\Delta^2\sqrt{\dot{X}(s)^2 + \dot{Y}(s)^2}} \ .$$

If $\eta = 0$, then, by constraint $\Phi_H(\bar{\theta}, \eta)$, it is $\dot{Y}(s) = \ddot{Y}(s) = 0$ and therefore $acc(s) = \Delta^2\ddot{X}(s)$. If $\eta > 0$, then $\Phi_{valid}(\bar{\theta}, \eta)$ enforces $\ddot{X}(s) = 0$. With $\sqrt{\dot{X}(s)^2 + \dot{Y}(s)^2} \leq |\dot{Y}(s)|$ we have $|acc(s)| \leq \Delta^{-2}|\ddot{Y}(s)|$. In both cases, the lemma follows immediately from the definition of Φ_{acc}. □

7.4.4 Assembling the Constraints Together

With the following modifications, the presented encoding of the single track model also considers vehicles driving in reverse:

- Introduce an additional Boolean variable rws for each vehicle that indicates reverse driving.
- Replace Ψ (Equation (7.1)) by

  $$(\neg\mathsf{rws} \rightarrow \Psi_{forward}) \wedge (\mathsf{rws} \rightarrow \Psi_{reverse})$$

  where $\Psi_{forward}$ is the original Ψ and $\Psi_{reverse}$ the original Ψ with the following replacements: $\dot{X}_o$ by $-\dot{X}_o$, v_o^l by $-v_o^u$, v_o^u by $-v_o^l$, acc_o^l by $-acc_o^u$, and acc_o^u by $-acc_o^l$.
- Extend Φ_{cont} (Equation (7.6)) by an additional constraint

  $$\dot{\mathbf{p}}_0' \neq \mathbf{0} \rightarrow (\mathsf{rws} \leftrightarrow \mathsf{rws}')$$

  allowing switching between forward and reverse movements only in standstill.

The correctness proofs for the reverse case are analogous to the forward case.

The following construction assembles the constraints defined in the above subsections together.

Construction 6 (Encoding of the single track model [12])**.** *Given a set $\mathcal{I}$ of heading intervals, the simple single track model is approximated by a constraint*

$$\Phi :\equiv \Phi_{cont} \wedge \bigvee_{I\in\mathcal{I}} ((\neg\mathsf{rws} \rightarrow \Phi_{fwd}(I)) \wedge (\mathsf{rws} \rightarrow \Phi_{rws}(I)))$$

with $\Phi_{fwd}(I) :\equiv \Phi'_H(I) \wedge \Phi'_{BB}(I) \wedge \Phi'_{valid}(I) \wedge \Phi'_v(I) \wedge \Phi'_{acc}(I)$ *where*

$$\begin{aligned}
\Phi_{cont} &:\equiv \mathbf{p}_n{=}\mathbf{p}_0' \wedge \dot{\mathbf{p}}_n{=}\dot{\mathbf{p}}_0' \wedge (\dot{\mathbf{p}}_n{=}\mathbf{0} \rightarrow \theta^l{=}\theta^u{=}\theta^{l\prime}{=}\theta^{u\prime}) \wedge (\dot{\mathbf{p}}_n{\neq}\mathbf{0} \rightarrow (\mathsf{rws}{\leftrightarrow}\mathsf{rws}')) \\
\Phi'_{BB}(I) &:\equiv \bigwedge_{bound\in BB} \left(bound^l \leq \min_{\theta\in I} bound(\theta) \wedge bound^u \geq \max_{\theta\in I} bound(\theta)\right) \\
\Phi'_H(I) &:\equiv [\theta^l, \theta^u] \supseteq I \wedge \bigwedge_{k=0}^{n-1} \left(\dot{X}(I)_k \geq 0 \wedge |\dot{Y}(I)_k| \leq \dot{X}(I)_k \tan\eta\right) \\
\Phi'_{valid}(I) &:\equiv \bigwedge_{k=0}^{n-2} \left(\underbrace{\ddot{X}(I)_k = 0}_{\eta>0 \ only} \wedge \left|\frac{\ddot{Y}(I)_k}{\Delta^2}\right| \leq a_{lat,\max}\right)
\end{aligned}$$

$$\wedge \bigwedge_{k=0}^{n-1} \left(\dot{X}(I)_k \geq \frac{2(n-1)L\tan(\eta)}{\tan(\delta_{\max})} \right)$$

$$\Phi'_v(I) :\equiv \bigwedge_{k=0}^{n-1} \begin{cases} v_k^l = v_k^u = \Delta^{-1}\dot{X}(I)_k & \text{for } \eta = 0 \\ v_k^l \leq \Delta^{-1}\dot{X}(I)_k \wedge v_k^u \geq \frac{\dot{X}(I)_k + \tan\left(\frac{\eta}{2}\right)|\dot{Y}(I)_k|}{\Delta} & \text{for } \eta > 0 \end{cases}$$

$$\Phi'_{acc}(I) :\equiv \bigwedge_{k=0}^{n-2} \begin{cases} acc_k^l = acc_k^u = \Delta^{-2}\ddot{X}(I)_k & \text{for } \eta = 0 \\ acc_k^l \leq -|\Delta^{-2}\ddot{Y}(I)_k| \wedge acc_k^u \geq |\Delta^{-2}\ddot{Y}(I)_k| & \text{for } \eta > 0 \end{cases}$$

with $I = [\bar{\theta} - \eta, \bar{\theta} + \eta]$,

$$X(I)_k := x_k \cos\bar{\theta} + y_k \sin\bar{\theta}, \text{ and} Y(I)_k := -x_k \sin\bar{\theta} + y_k \cos\bar{\theta}$$

for all $k = 0, \ldots, n$*), and where* $(\dot{X}(I)_0, \dot{Y}(I)_0)^T$, ..., $(\dot{X}(I)_{n-1}, \dot{Y}(I)_{n-1})^T$ *(respectively* $(\ddot{X}(I)_0, \ddot{Y}(I)_0)^T$, ..., $(\ddot{X}(I)_{n-1}, \ddot{Y}(I)_{n-1})^T$*) are control points of the first (respectively second) derivative of the Bézier curve spanned by* $(X(I)_0, Y(I)_0)^T, \ldots, (X(I)_n, Y(I)_n)^T$.

The constraint $\Phi_{rws}(I)$ *is the same as* $\Phi_{fwd}(I)$ *except for the following replacements:* $\dot{X}(I)_k$ *by* $-\dot{X}(I)_k$, v_k^l *by* $-v_k^u$, *and* v_k^u *by* $-v_k^l$.

Note, that the constraints $\Phi'_\square(I)$ (for $\square \in \{BB, H, v, acc, valid\}$) are sufficient conditions for the respective constraints $\Phi_\square(\bar{\theta}, \eta)$ from Section 7.4 in the sense that $\Phi'_\square(I) \equiv tr'_2(\Phi'_\square(\bar{\theta}, \eta))$.

We encode user-provided object type invariants Φ_C from the world model using the same technique as we use to encode spatial views. Combining the encoding of the single track model with the user provided invariants results in the following encoding of the world model into BMC:

Construction 7 (Sufficient world model constraints)**.** *For some set of object variables* V_O*, and a particular variable* $o \in V_O$ *of type* $type(o)$*, the sufficient world model constraints are*

$$\Phi'_{type(o)}(o) :\equiv \Phi(o) \wedge tr'_s(\Phi_{type(o)}(o), V_O) \wedge \bigwedge_{a\ const} a'_o = a_o$$

where $\Phi_{type(o)}(o)$ *and* $\Phi(o)$ *are the instantiations of* $\Phi_{type(o)}$ *and* Φ *for object variable* o *(i.e, replacement of all references to* $a \in \mathcal{A}(type(o))$ *by* $o.a$*), and* a *ranges over all the constant attributes of* o*. If* $type(o)$ *is a vehicle type, then* Ψ *is given in Construction 6, otherwise* $\Phi \equiv true$*. For all objects together, it is*

$$\Phi'_{\mathsf{WM}}(V_O) :\equiv \bigwedge_{o \in V_O} \Phi'_{type(o)}(o) \ .$$

Construction 6 may be adapted for dynamic objects that follow simpler dynamics and are not bound to trajectories. For example, it is possible to let the wheel base approach $L \to 0$ and omit the constraint $\Phi_{cont\text{-}2}$ (Equation 7.5) for objects that can turn on the spot, such as pedestrians or animals.

Now we finally have all parts at hand to define the semi-decision procedure CheckSatN$_{\mathsf{WM}}$ as Algorithm 9. The correctness proof is given at the end of Section 7.5

Algorithm 9: The semi-decision procedure CHECKSATS$_{\mathsf{WM}}$(BC)

Input: A world model WM, and an existential TSC BC
Output: A tuple (SAT, $\mathcal{M}$) containing a model, UNSAT, or UNKNOWN

begin
  translate BC to an MSRTL φ_{BC};
  let V_T be the set of bound time variables in φ_{BC};
  let V_O be the set of free object variables in φ_{BC};
  let $(I, T, F) :=$ CHART2BMC(BC) ; *// see Algorithm 5*
  for each *spatial view sv in* BC **do**
    $T := T \wedge (b_{sv} \Leftrightarrow tr'_s(\phi_{sv}, V_O))$; *// see Construction 2; b_{sv} is introduced by* CHART2BMC(BC)
  end
  for each $o \in V_O$ **do**
    $T := T \wedge \Phi'_{type(o)}(o)$; *// see Construction 7*
  end
  Choose user-defined $l, u \in \mathbb{N}$ such that $|V_T| < l \leq u$;
  return BOUNDEDMODELCHECKING($(I, T, F), [l, u]$); *// see Algorithm 1*
end

7.5 Correctness Proof: From Models to Trajectories

This section presents the correctness proof for the encoding presented in this chapter. First, we construct a trajectory π from a model $\mathcal{M}$ in three simple steps.

Construction 8 (From Model to Trajectory)**.** *Assume a set V_O of object variables, some signature Σ, and Σ-model $\mathcal{M}$ that, for some $m, n \in \mathbb{N}$, defines the following variables:*

- *real-valued variables $t_0, t_1, \ldots, t_n$ with $\mathcal{M}(t_0) < \mathcal{M}(t_1) < \ldots, \mathcal{M}(t_n)$*
- *for all object variables o of a vehicle type, real-valued variables $x_{o,i,k}$, $y_{o,i,k}$ for $i = 0, 1, \ldots, n-1$ and $k = 0, 1, \ldots, m$*
- *a variable $a_{o,i}$ for each attribute a of each object variable o that is not described by the simple single track model*

Then, construct a trajectory π and valuation μ piecewise (i.e., for all i and $t \in [\mathcal{M}(t_i), \mathcal{M}(t_{i+1}))$) as follows:

1. *For object variables o of a vehicle type, let $\pi(t) \begin{bmatrix} \mu(o).x \\ \mu(o).y \end{bmatrix} := \mathbf{p}_i \left(\frac{t - \mathcal{M}(t_i)}{\mathcal{M}(t_{i+1}) - \mathcal{M}(t_i)} \right)$ where $\mathbf{p}_i$ is the Bézier spline segment with control points $\mathbf{p}_{i,k} := (\mathcal{M}(x_{o,i,k}), \mathcal{M}(y_{o,i,k}))^T$.*
2. *Construct the remaining attribute values (such as velocity, acceleration and bounding box) from the x and y attributes with help of the equations of motion and bounding boxes (Equations 7.7).*
3. *For all other attributes a of any object o, let $\pi(t)[\mu(o).a] = \mathcal{M}(a_{o,i})$.*

It remains to show that, if the model is a solution for the BMC problem constructed in CheckSatSWM, the encoded basic chart (especially, the spatial views) is satisfied, and that the derived trajectory is contained within the world model. First, we focus on the global constraints Ψ and Φ_{cont} which encode the single track model for each time slice and vehicle. We still consider a single vehicle.

Theorem 10 (Global Constraints)**.** *Assume some trajectory π split into time slices $[t_i, t_{i+1})$ such that Ψ (defined in Equation 7.1) holds for all time slices, and Φ_{cont} holds for all consecutive pairs of slices. Then,*

1. *the differential equations of the single track model (given in Definition 25) hold,*
2. *$\bar{\theta}$ and η bound the heading θ (modulo 2π) on that time slice (i.e, $\bar{\theta}-\eta \leq \theta+2k\pi \leq \bar{\theta}+\eta$ for some $k \in \mathbb{Z}$), and*
3. *B^l, B^u (resp. C^l, C^u) are valid bounds for attributes $B \in \mathfrak{B}$, $C \in \mathfrak{C}$.*

Proof. The proof for item 1 follows the already published work [12, Lemma 1]: We derive the inputs for the single track model from the trajectory – more precisely from the velocity vector and the curvature – as

$$\begin{aligned} v(t) &= \|\mathbf{v}(t)\|_2 \\ \delta(t) &= \arctan(\kappa(t)L) \ , \end{aligned}$$

and we initialize x and y with the first control point from the first time slice. We have to show that, when using the derived $v(t)$ and $\delta(t)$ as input, the vehicle position according to the single track model ODEs is the same as in the generated trajectory. Note that the ODE system for the single track model can also be formulated as linear ODEs

$$\dot{\mathbf{h}}(t) = \begin{bmatrix} 0 & -v(t)\kappa(t) \\ v(t)\kappa(t) & 0 \end{bmatrix} \mathbf{h}(t) \qquad\qquad \dot{\mathbf{p}}(t) = v(t)\mathbf{h}(t) \tag{7.8}$$

with piecewise continuous inputs $v(t)$ and $\kappa(t)$, and the heading $\mathbf{h}(t)$ being unit vector (i.e., $|\mathbf{h}(t)| = 1$). We can easily verify that the above linear ODEs (7.8) imply the single track model ODEs if $\mathbf{h}(t) = (\cos\theta(t), \sin\theta(t))^T$ and $\kappa(t) = \frac{\tan\delta(t)}{L}$:

$$\begin{aligned} \dot{\mathbf{h}}(t) &= \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\cos\theta(t)\\ \sin\theta(t)\end{bmatrix} = \begin{bmatrix}-\dot{\theta}(t)\sin\theta(t)\\ \dot{\theta}(t)\cos\theta(t)\end{bmatrix} = \begin{bmatrix}-v(t)\frac{\tan\delta(t)}{L}\sin\theta(t)\\ v(t)\frac{\tan\delta(t)}{L}\cos\theta(t)\end{bmatrix} \\ &= \begin{bmatrix} 0 & -v(t)\kappa(t) \\ v(t)\kappa(t) & 0 \end{bmatrix} \mathbf{h}(t) \\ \dot{\mathbf{p}}(t) &= \begin{bmatrix}v(t)\cos\theta(t)\\ v(t)\sin\theta(t)\end{bmatrix} = v(t)\mathbf{h}(t) \end{aligned}$$

We fix some time point t and call the ith spline segment, with $t_{i-1} \leq t \leq t_i$, the *current segment*. We derive $v(t)$ and $\kappa(t)$ from $\mathbf{p}(t) = (x(t), y(t))^T$ using the equations given in Fact 5 (in case of $\dot{\mathbf{p}}(t) = \mathbf{0}$ we set $\kappa(t) = 0$) and set

$$\mathbf{h}(t) = \begin{cases} (\cos\bar{\theta}, \sin\bar{\theta})^T & \text{if } v(t) = 0 \\ \frac{\dot{\mathbf{p}}(t)}{v(t)} & \text{else.} \end{cases}$$

By definition of Φ, there exists some heading interval $I = [\bar{\theta} - \eta, \bar{\theta} + \eta] \in \mathcal{I}$ such that $\Phi_H(I)$ holds. Furthermore, Φ_{cont} holds. In the following, we show that the linear ODEs (7.8) hold and that $\mathbf{h}(t)$ is continuous on the whole trajectory (the latter implies that we can construct continuous $\theta(t)$ such that $\mathbf{h}(t) = (\cos\theta(t), \sin\theta(t))^T$).

In case $\dot{\mathbf{p}}(t) = \mathbf{0}$, the constraint $\Phi_H(I)$ implies that we are on a straight segment with $\dot{x}(t') = v(t')\cos\bar{\theta}$ and $\dot{y}(t') = v(t')\sin\bar{\theta}$ for all $t' \in [t_{i-1}, t_i)$. This holds both for $v(t') = 0$ and $v(t') \neq 0$. In case $t = t_{i-1}$, the interplay of Φ_{cont} and $\Phi_H(\bar{\theta}, \eta)$ implies that also the previous segment is straight, and has the same heading. In any case, $\mathbf{h}$ is constant in some non-empty environment of t, so $\dot{\mathbf{h}} = \mathbf{0}$ and, because $v(t) = 0$, the single track model ODEs hold.

In case $\dot{\mathbf{p}}(t) \neq \mathbf{0}$, we have

$$\begin{aligned}
\dot{\mathbf{h}}(t) &= \frac{d}{dt}\frac{\dot{\mathbf{p}}(t)}{v(t)} \\
&= \frac{d}{dt}\begin{bmatrix} \frac{\dot{x}(t)}{\sqrt{\dot{x}(t)^2+\dot{y}(t)^2}} \\ \frac{\dot{y}(t)}{\sqrt{\dot{x}(t)^2+\dot{y}(t)^2}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\ddot{x}(t)\dot{y}(t)^2 - \ddot{y}(t)\dot{x}(t)\dot{y}(t)}{(\dot{x}(t)^2+\dot{y}(t)^2)^{3/2}} \\ \frac{\ddot{y}(t)\dot{x}(t)^2 - \ddot{x}(t)\dot{x}(t)\dot{y}(t)}{(\dot{x}(t)^2+\dot{y}(t)^2)^{3/2}} \end{bmatrix} \\
&= \begin{bmatrix} -\kappa(t)\dot{y}(t) \\ \kappa(t)\dot{x}(t) \end{bmatrix} \\
&= \begin{bmatrix} 0 & -v(t)\kappa(t) \\ v(t)\kappa(t) & 0 \end{bmatrix} \mathbf{h}(t)
\end{aligned}$$

Because Φ_{cont} implies $\frac{\dot{\mathbf{p}}^i(1)}{\Delta_i} = \frac{\dot{\mathbf{p}}^{i+1}(0)}{\Delta_{i+1}}$, it is $\lim_{t\to t_i}\mathbf{h}(t) = \lim_{t\to t_i}\frac{\dot{\mathbf{p}}(t)}{\|\dot{\mathbf{p}}(t)\|_2} = \frac{\dot{\mathbf{p}}^i(1)}{\Delta_i\|\dot{\mathbf{p}}^i(1)\|_2} = \frac{\dot{\mathbf{p}}^{i+1}(0)}{\Delta_{i+1}\|\dot{\mathbf{p}}^{i+1}(0)\|_2} = \mathbf{h}(t_i)$. Hence, no discontinuities for $\mathbf{h}$ occur and again the ODEs hold.

Furthermore, by Lemmas 6 and 7, we can find a continuous $\delta(t)$ with $\kappa(t) = \frac{\tan\delta(t)}{L}$ that respects the validity bounds for $\delta(t)$ and lateral acceleration (see Definition 25).

Item 2 follows from Lemma 6. Item 3 follows from Item 2 together with Fact 6 and Lemmas 8 and 9.

□

Now, we can prove the correctness of the encoded object type invariants and spatial views.

Theorem 11 (Correctness of the Bézier spline encoding)**.** *For some world model* WM*, type-correct valuation* $\mu : V_O \to O_{\mathsf{WM}}$ *of object variables* V_O *to world model objects* O_{WM}*, and unrolling depth* $N \in setN$*, construct*[4] $\Phi_N :\equiv \bigwedge_{i=0}^{N} \Phi'_{\mathsf{WM}}(V_O)[X\backslash X_i, X'\backslash X_{i+1}]$*, where* $\Phi'_{\mathsf{WM}}(V_O)$ *is a constraint over sets* X *and* X' *of variables constructed according to Construction 7 with step size* Δ*. Furthermore, assume some model* $\mathcal{M} \models \Phi_N$*.*

Then, we can construct a trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ *such that, for all MSRTL predicates* ϕ *with free variables in* V_O *and* $i = 0, 1, \ldots, N-1$*, holds*

$$\mathcal{M} \models tr'_s(\phi)[X\backslash X_i] \Longrightarrow \pi, \mu \models \Box_{[i\Delta,(i+1)\Delta)}\phi \ .$$

[4]We use N instead on n here to denote the number of unrolling steps because throughout this chapter n refers to the Bézier curve degree.

Recall that the set X contains variables that encode object attributes. In this context it includes particularly the upper and lower bounds and Bézier curve control points defined throughout this chapter.

Proof. From some model $\mathcal{M} \models \Phi_N$ as defined above, construct a trajectory π and a valuation μ as described in Construction 8. Recall that Φ_N instances the constraint Φ' on each time slice which contains $tr'_2(\Psi)$. So, by Lemma 5, the constraint Ψ holds on every time slice. Furthermore (also by definition of Φ'), the continuity constraint Φ_{cont} holds.

First, we prove that $\mathcal{M} \models tr'_s(\phi)[X \backslash X_i] \Rightarrow \pi, \mu \models \Box_{[i\Delta,(i+1)\Delta)}\phi$ holds. We fix some $i \in \{0, 1, \ldots, N-1\}$ and assume, in addition to the above assumptions, that $\mathcal{M} \models tr'_s(\phi)[X \backslash X_i]$ holds. Let $t_i = i\Delta$ and $t_{i+1} = i\Delta$. We need to show that $\pi, \mu \models \Box_{[\mathcal{M}(t_i),\mathcal{M}(t_{i+1}))}\phi$. Recall that the latter has the semantics $\forall t \in [\mathcal{M}(t_i), \mathcal{M}(t_{i+1})) : \pi^t, \mu \models \phi$. We fix some arbitrary $t \in [\mathcal{M}(t_i), \mathcal{M}(t_{i+1}))$, construct a model $\mathcal{M}_t$ for $\pi(t)$ and μ as described in Lemma 2, and show by induction over structure of $\phi' :\equiv tr_c(\phi)$ that $\mathcal{M}_t \models \phi'$ holds. With Lemma 2 then follows $\pi^t, \mu \models \phi$. In the following induction proof, we handle the chain of tr'_1 and tr_2, followed by variable substitution for unrolling step i, as a single function $tr'(\phi') := tr'_2(tr'_1(\phi))[X \backslash X_i]$.

As an induction start, consider linear inequalities of the form

$$\phi' \equiv \sum_{i \in I_1} a_i A_i + \sum_{i \in I_2} b_i B_i + \sum_{i \in I_3} c_i C_i \bowtie d$$

as defined before in this chapter. Let $s := \frac{\mathcal{M}(t_i) - t}{\mathcal{M}(t_{i+1}) - \mathcal{M}(t_i)}$. By construction, it is $[\![A]\!](\pi^t(0), \mu) = \mathcal{M}(A) =: A(s)$ for $A \in \mathfrak{A} \cup \mathfrak{B} \cup \mathfrak{C}$ (note, that we use the letter A for both the attribute access as MSRTL formula and the corresponding variable in the SMT formula ϕ', see Construction 1). By assumption, $\mathcal{M} \models tr'(\phi')$. So, by Lemma 5, the constraint $tr'_1(\phi')$ holds which is defined as

$$tr'_1(\phi') :\equiv \forall s \in [0, 1) : \sum_{i \in I_1} a_i A_i(s) + \sum_{i \in I_2} b_i \bar{B}_i \sum_{i \in I_3} c_i \bar{C}_i(s) \bowtie d$$

with

$$\bar{B}_i = \begin{cases} B_i^l & \text{if } b_i \leq 0 \\ B_i^u & \text{else} \end{cases} \quad \text{and} \quad \bar{C}_i(s) = \begin{cases} C_i^l(s) & \text{if } c_i \leq 0 \\ C_i^u(s) & \text{else} \end{cases}.$$

By Theorem 10 holds $B_i^l \leq B_i(s) \leq B_i^u$ and $C_i^l(s) \leq C_i(s) \leq C_i^u(s)$. So, for both cases $b_i, c_i \leq 0$ and $b_i, c_i \geq 0$, it is $b_i \bar{B}_i \leq B_i(s)$ and $c_i \bar{C}_i(s) \leq C(s)$. As a consequence,

$$\sum_{i \in I_1} a_i A_i(s) + \sum_{i \in I_2} b_i \bar{B}_i \sum_{i \in I_3} c_i \bar{C}_i(s) \leq \sum_{i \in I_1} a_i A_i + \sum_{i \in I_2} b_i B_i + \sum_{i \in I_3} c_i C_i \bowtie d$$

for the fixed s. As a consequence, $\mathcal{M}_t \models \phi'$.

For the induction step, assume we have shown $\mathcal{M} \models tr'(\phi_i) \Rightarrow \mathcal{M}_t \models \phi_i$ for formulas ϕ_1 and ϕ_2. Then we have for $\boxplus \in \{\wedge, \vee\}$:

$$\begin{aligned} & \mathcal{M} \models tr'(\phi_1 \boxplus \phi_2) & \\ \Leftrightarrow\ & \mathcal{M} \models tr'_1(\phi_1) \boxplus tr'_1(\phi_2) & \text{(def. of } tr'_1 \text{ and } tr'_1\text{)} \\ \Rightarrow\ & (\mathcal{M}_t \models \phi_1) \boxplus (\mathcal{M}_t \models \phi_2) & \text{(ind. hyp.)} \\ \Leftrightarrow\ & \mathcal{M}_t \models (\phi_1 \boxplus \phi_2) & \text{(SMT formula semantics)} \end{aligned}$$

This closes the induction proof.

It remains to show that $\pi \in \mathcal{T}(\mathsf{WM})$. By Theorem 10, the vehicle trajectories are solutions of the single track model. Recall that the user-provided world model constraints Φ_C are translated using the same function tr'_s as applied for spatial views, i.e., Φ_N contains a constraint $tr'_s(\Phi_{type(o)}(o), V_O)$ for each $o \in V_O$. So, the above part of the proof applies analogously and the user-provided constraints hold in addition to the differential equations and validity constraints of the single track model. Note also, that $\mathcal{M} \models \Phi_N$ implies $\mathcal{M}(a_o^0) = \mathcal{M}(a_o^1) = \mathcal{M}(a_o^N)$ for all constant object attributes. Hence, by Construction 8, the attribute value $\pi[\mu(o).a]$ is constant for all object attributes marked so in the world model definition. So, in summary, the trajectory is element of $\mathcal{T}(\mathsf{WM})$. □

Now, we can give the correctness proof for CheckSatS (Algorithm 9).

Theorem 12 (Correctness of CheckSatS (Algorithm 9))**.** *Assume a world model* WM *and a basic chart* BC*. If* $\textsc{CheckSatS}_{\mathsf{WM}} = (\mathtt{sat}, \mathcal{M})$*, with some model* $\mathcal{M}$*, then there exists a trajectory* $\pi \in \mathcal{T}(\mathsf{WM})$ *and a valuation* μ *such that* $\pi, \mu \models_{\mathsf{WM}} \varphi_{\mathsf{BC}}$*, and* π *and* μ *can be constructed from* $\mathcal{M}$*.*

Proof. Assume some world model WM, a basic chart BC with free object variables in V_O, construct the BMC problem $\mathsf{BMC} = (I, T, F)$ as described in Algorithm 9, and unroll it for N steps, yielding an SMT formula BMC_N.

Recall that the function tr'_s (Construction 3) applied in line 7 of Algorithm 9 ($b_{sv} \Leftrightarrow tr'_s(\phi_{sv}, V_O)$ for each spatial view sv in BC) introduces a set X of variables that encode object attributes. The set X' contains primed versions of these variables, and the sets X_i (for $i = 0, 1, \ldots, N$) distinct copies of X for each unrolling step.

By construction, BMC_n is a conjunction of the following parts:

- BMC'_n that is the BMC problem $\mathsf{BMC}' :\equiv \textsc{Chart2BMC}(\mathsf{BC})$ unrolled for n steps
- $t_0 = 0$
- for $i = 0, 1, \ldots, N-1$:
  - $t_{i+1} = t_i + \Delta$
  - $b^i_{sv} \Leftrightarrow tr'_n(\phi_{sv}, V_O)[X \backslash X_i]$ for each spatial view sv in BC
  - $\Phi'_{\mathsf{WM}}(V_O)[X \backslash X_i, X' \backslash X_{i+1}]$

Therefore, by Theorem 11, we can, for any model $\mathcal{M} \models \mathsf{BMC}_n$, construct a trajectory $\pi \in \mathcal{T}(\mathsf{WM})$ such that $\mathcal{M}(b^i_{sv}) \Rightarrow \Box_{[\mathcal{M}(t_i), \mathcal{M}(t_{i+1}))} \phi_{sv}$ for all spatial views in BC (note that $\mathcal{M}(t_i) = i\Delta$). With Theorem 6 we can conclude that $\pi, \mu \models_{\mathsf{WM}} \varphi_{\mathsf{BC}}$. □

8 Implementation & Evaluation

In this chapter, the encoding techniques for existential TSCs and the partial consistency analysis prototype are evaluated in five experiments. The first two experiments evaluate scalability of the SMT-based satisfiability checking of existential TSCs described in Chapter 6 independently from a consistency analysis. The other three experiments evaluate the partial consistency analysis. The prototype implements a parallelized version of Algorithm 4. Except for the full consistency analysis of the ALKS case study (Section 8.3.3) which has been executed on server hardware with 20 AMD EPYC 7542 32-Core@3GHz processors and 128GB RAM, all experiments have been performed on a Windows notebook running an Intel® Core™ i7-1185G7 @ 3.00GHz and 16GB RAM. As an SMT solver, Z3 version 4.13.0 in its default configuration (i.e., no memory limit) has been used.

8.1 Prototype Implementation

The consistency analysis prototype has been integrated into the *TSC Editor*, which is an in-house tool for specification and evaluation of TSCs at DLR-SE. The TSC Editor builds on Eclipse Modeling Tools[1] and provides editors for world models, symbol dictionaries, and TSCs.

The core of the tooling is a model-based framework for working with TSCs, world models, and symbol dictionaries. It is based on the Eclipse Modeling Framework[2]. On top of this framework, the editors as well as the consistency analysis prototype have been implemented. For TSCs, a graphical editor (based on the Eclipse Graphical Editing Framework[3], Figure 8.2) is provided that allows editing of TSCs via Drag'n'Drop of symbols. The TSC editor has been designed with maximum flexibility in mind, making only minimal assumptions on the underlying domain model. Hence, all the world model types and

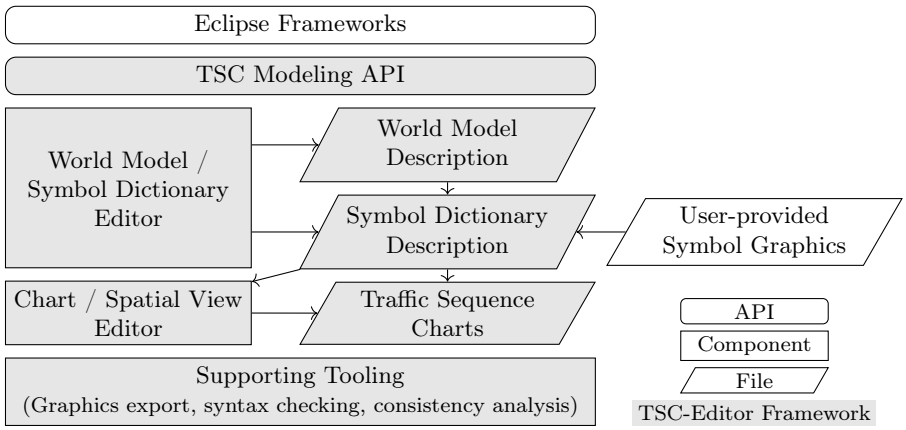


Figure 8.1: TSC Editor architecture overview (taken from [18])

[1] `https://www.eclipse.org/modeling/`, last accessed on 2023-03-06
[2] `https://eclipse.dev/modeling/emf/`, accessed on 2024-11-29
[3] `https://projects.eclipse.org/projects/tools.gef`, accessed on 2024-11-29

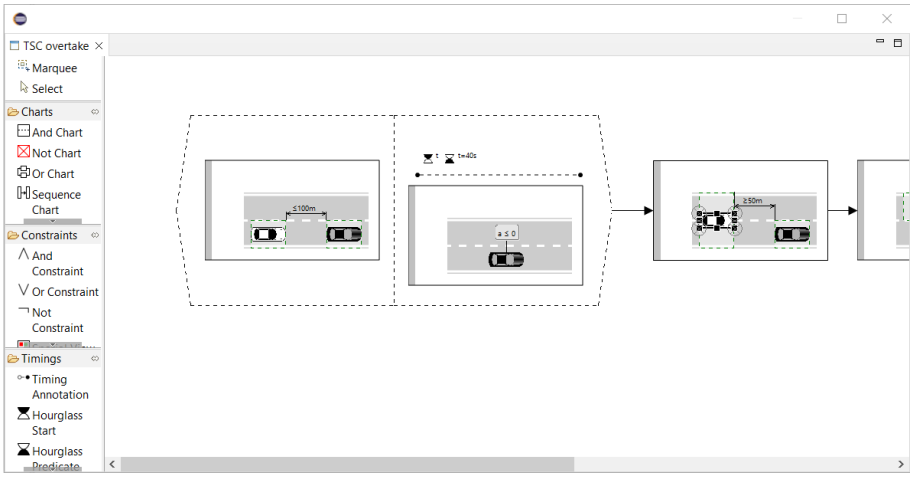


Figure 8.2: Screenshot of the graphical editor

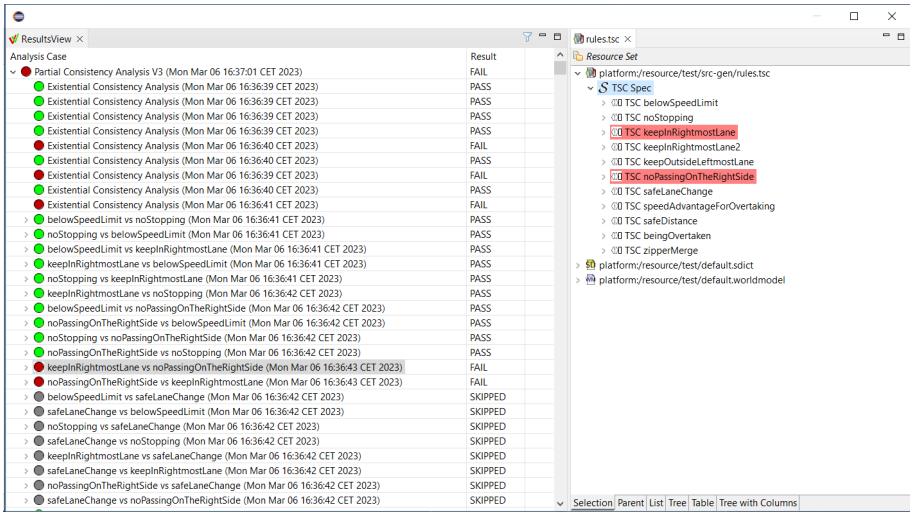


Figure 8.3: Screenshot of the results viewer. On the left-hand side, analysis cases (inclusive sub-cases) are listed. On the right-hand side the TSCs are highlighted that have been analyzed in the selected case.

symbols are user-provided[4]. On the other hand, the TSC editor focuses on syntactical correctness of the TSCs, so the user has to properly define all used symbols and types before being able to use them in a TSC. Besides editing, the TSC Editor provides export functionality for TSCs. This feature has been used to produce drawings of TSCs in this thesis.

The prototype implementation provides a graphical user interface for starting and configuring the consistency analysis. Analysis results are presented in a dedicated viewer (Figure 8.3) in the TSC Editor. The user can inspect the generated witness trajectories in the *trace editor* shown in Figure 8.4. The table in the upper part shows the attribute values for each object and step, and at the bottom an approximation of the traffic situation to some time instant is drawn.

[4]The only exception is the single track model which is a pre-requisite for the application of the SMT encoding presented in Chapter 7 and currently hard-coded.

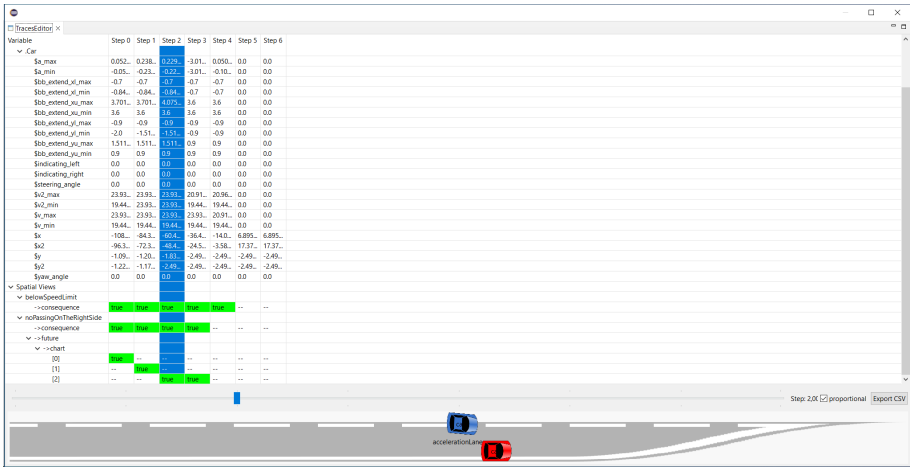


Figure 8.4: Screenshot of the trace editor. The table shows attribute values and which spatial views are "active" in each step, and below the table the traffic situation in the current step is sketched.

8.2 Performance Evaluations

This section reports about two experiments that evaluate the efficiency and scalability of the TSC encoding in terms of the time needed to decide whether some existential TSC is satisfiable. Therefore, experiments have been designed that can be instantiated with an arbitrary number of participants. Increasing the number of participants increases the size of the chart in terms of invariant nodes, but leaves the invariant nodes the same, modulo substitution of object variables.

8.2.1 Simple Encoding: Platooning Example

The first experiment is based on a case study [32] from 2019. The goal of this study was to extend TSCs with recursive and iterative constructs allowing to specify parameterized TSCs that scale to a given number of traffic participants or iterations. The case study has been applied to a platooning use case from the CrESt project [108], where automated vehicles form a platoon in order to reduce energy consumption and improve traffic throughput. For the platoon, a number of maneuvers have been defined. One of the maneuvers is *dense2open*, where the platoon switches from a so-called *dense formation* to an *open formation*. During the former, the vehicles in the platoon maintain a very short distance to each other in order to maximize efficiency, while the latter is characterized by larger distances and is needed for driving maneuvers that require more flexibility, such as circumventing an obstacle.

The platoon changes from dense to open formation by one vehicle after the other decelerating from high traveling speed (`v_hi`) to low speed (`v_lo`). The last vehicle in the platoon decelerates first, such that the distance between the vehicles grows from `d_dense` to `d_open`. A tolerance of `delta_d` resp. `delta_v` is allowed for distance and speed. The TSC in Figure B.1 shows how this is formalized for n cars (ignore the timing annotations in the TSC by now). In contrast to the original TSCs in [32] this is not formalized as a sequence of invariants which each reference the whole platoon, but as a parallel composition of short sequences specifying how each vehicle (referred to as `curr`) shall behave with respect to its predecessor (`prev`) and its successor (`next`).

The experiments have been performed with a varying number of cars and with timing annotations either being used or omitted from the TSCs. Furthermore, the encoding

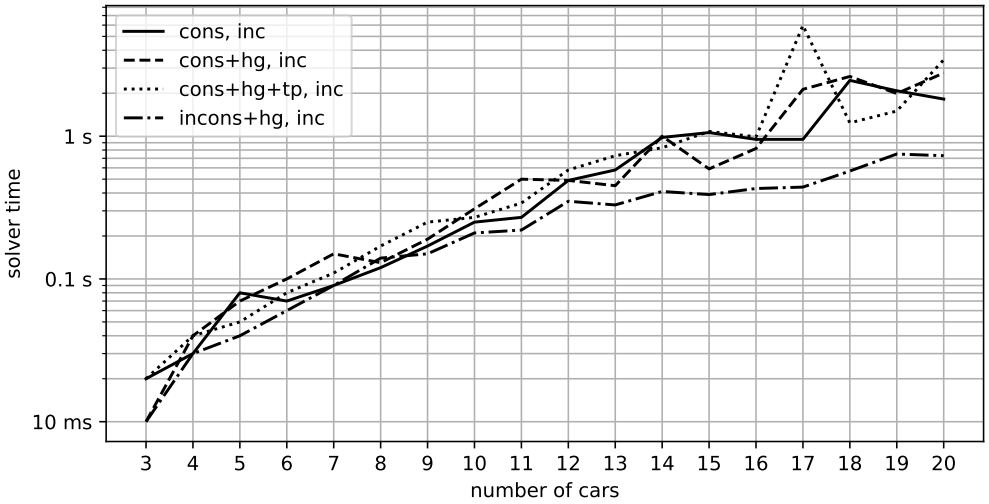


Figure 8.5: Platooning: Performance of the incremental encoding scheme

technique described in Section 6.3 has been compared to an alternative encoding (see Annex A). Figures 8.5–8.7 present the execution times for the different variants, encoding schemes, and number of cars. The figure captions use the following naming schemes to identify the used settings:

- cons: consistent case (TSC in Figure B.1) without timing annotations (unless noted by +hg or +tp, see below)
- incons: inconsistent case (TSC in Figure B.2) without timing annotations (unless noted by +hg or +tp, see below)
- +hg: with hour glasses as denoted in Figure B.1
- +tp: with time pins as denoted in Figure B.1
- inc: incremental encoding scheme described in Section 6.3
- ninc: non-incremental encoding scheme, see Annex A

The results show clearly that the computational complexity grows exponentially with the size of the problem, which is not surprising at all. The comparison shows that the incremental encoding scheme is solved faster than the non-incremental one. Surprisingly, this effect is stronger if the TSC is inconsistent (see Figure 8.7). With both encoding schemes, inconsistencies are found faster than witness solutions for consistent TSCs.

8.2.2 Spline Encoding: Nested Overtaking

For evaluating the spline encoding, a nested overtaking scenario is used. The scenario has already been used in earlier work [15] as an evaluation. However, due to a bug in the prototype implementation (generated trajectories have not been guaranteed to be correct) at time of writing the original publication [15], the experiment has been repeated with the corrected implementation.

The evaluation scenario, for $n \geq 2$ cars, is as follows:

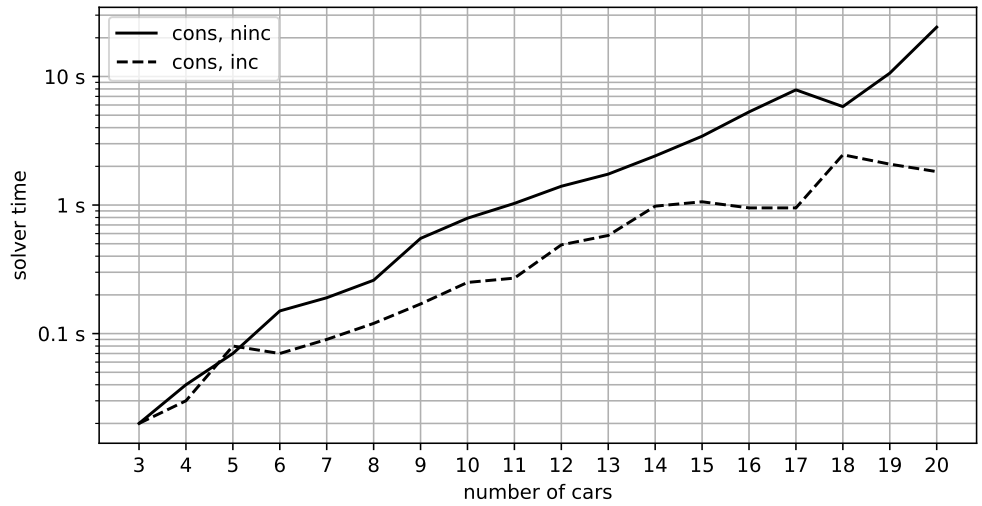


Figure 8.6: Platooning: Performance of the incremental vs. the non-incremental encoding scheme, consistent case

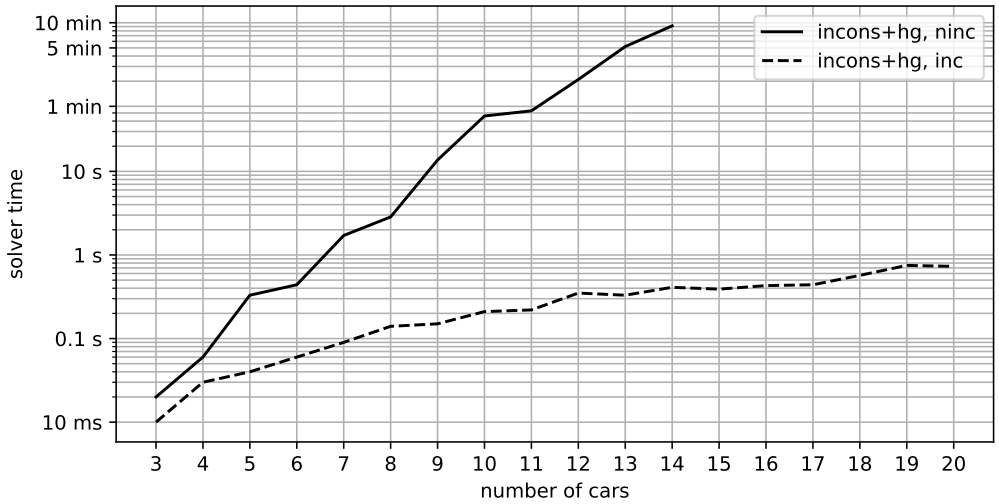


Figure 8.7: Platooning: Performance of the incremental vs. the non-incremental encoding scheme, inconsistent case

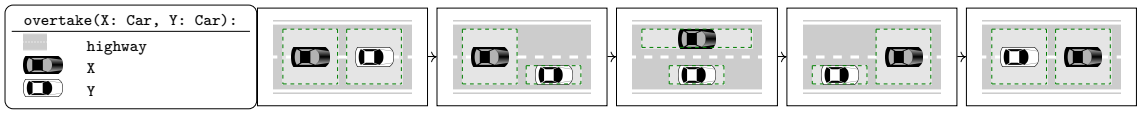


Figure 8.8: Pair-wise overtaking between two cars

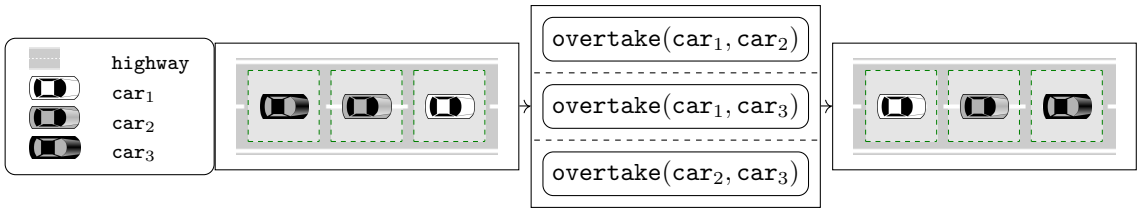


Figure 8.9: Evaluation scenario for $n = 3$

- In the beginning of the scenario, car i drives in front of car $i + 1$, so car n is the leading vehicle, and car 1 the last vehicle.
- Then, each car overtakes each other car in front of it. Pair-wise overtaking between two cars is depicted in Figure 8.8. The overtaking maneuvers can take place in any order.
- In the end of the scenario, the ordering of cars has been reversed. Car 1 is the leading vehicle, and car n the last one in the row.

The evaluation scenario for $n = 3$ is depicted in Figure 8.9. The results are listed in Table 8.1.

Table 8.1: Experimental results for parallel overtaking scenario

#cars	#overtake maneuvers	#steps		execution time [s]		solver stats for last step	
		start	solution	overall	solver	time [s]	memory [MB]
2	1	5	5	0.45	0.11	0.01	21.67
3	3	5	9	0.95	0.32	0.07	30.71
4	6	9	10	3.44	2.30	1.35	44.43
5	10	10	13	38.66	36.67	22.84	88.09
6	15	13	14	243.68	240.70	186.03	152.65

8.3 Case Studies

In this section, three case studies are reported where the consistency analysis prototype is applied to two example specifications of medium size (8 respectively 9 TSCs), and one larger set of requirements (23 TSCs). Although the experiments demonstrate that the analysis is quite fast in checking sets of TSCs (the smaller experiments terminate in less than two minutes, the large case study within 24 hours using server hardware), scalability is not in the focus. Rather, the experiments evaluate whether the analysis is able to find inconsistencies in realistic use cases. The first experiment analyzes a set of TSCs that has been created by a rather inexperienced engineer (a student employee at the author's research institute), and the second is based on LTL-formalizations of different international traffic

Table 8.2: World model invariants

Car	$0 \leq v \leq 160\,\text{km/h}$ $4.1\,\text{m} \leq \overline{x} - \underline{x} \leq 4.6\,\text{m}$ $1.8\,\text{m} \leq \overline{x} - \underline{x} \leq 3.1\,\text{m}$
Abstract Lane	$\mathsf{length} := \overline{x} - \underline{x} > 0$ $\mathsf{width} := 3.5\,\text{m}$ $\underline{y} = y - \mathsf{width}/2$ $\overline{y} = y + \mathsf{width}/2$
Driving Lane	$y = 0 \vee y = \mathsf{width}$
Acceleration Lane	$y = -\mathsf{width}$

Table 8.3: Vehicle parameters for the demonstration scenarios (taken from [12])

parameter	**variable**	**value**
max. steering angle	$\delta_{\max}$	$36.0°$
front	F	$0.8\,\text{m}$
wheel base	L	$2.8\,\text{m}$
rear	G	$0.7\,\text{m}$
width	W	$1.8\,\text{m}$

rules taken from other research [49, 51]. The third one (the larger one) formalizes a subset of the requirements on advanced lane keeping systems (ALKS) given in UN regulation No. 157 [116]. The first two studies use the world model and symbol dictionary introduced in Section 2.3.2 with additional abbreviations and invariants given in Table 8.2. For the third case study, this world model has been further extended.Vehicle speed is limited to $160\,\text{km/h}$, and minimal and maximal dimensions of the bounding box are asserted. The single track model used for the sufficient constraints is parameterized as given in Table 8.3 with two heading intervals $\mathcal{I} = \{[0°, 0°], [10°, 10°]\}$, i.e., it is assumed that the heading angle diverges by at most $10°$ from the road direction. Because the case studies consider highways and rural roads only, this is a reasonable choice. Note, that the bounding box size upper and lower bounds from Table 8.2 enclose the possible bounding box dimensions assumed by the encoding of the single track model (the exact bounding box sizes can be calculated with Equation 7.7).

The lane width is fixed to $3.5\,\text{m}$. The y-positions of lanes is centered on the lane and fixed to a certain global positions. The latter is not necessary but ensures valid lane layouts.

For the sufficient condition, 10 unrolling steps with a step size of $\Delta = 1\,\text{s}$ are used.

8.3.1 Overtaking

For this evaluation, the consistency check is applied to a set of eight TSCs that formalize German traffic rules regarding overtaking (a subset of §5 StVO). The original TSCs have been created 2019 by a student employee under supervision of scientific researchers. The TSCs are not a perfect formalization of traffic rules, but rather an example for a specification done by unexperienced engineers. The goal of this experiment is to evaluate whether the consistency analysis can detect errors in such a specification.

At time of writing the specification, the syntax of TSCs was still under development (and no tool support available), so the syntax in the original documents differs from the syntax used in this thesis. For the evaluation, the TSCs have been adapted by the thesis author to the current syntax and format implemented in the analysis prototype, staying as close as possible to the original semantics.

The TSCs are depicted in Figure B.3. The The consistency analysis find that TSCs (b) – (g) are weakly existentially inconsistent. All six TSCs contain at least one discontinuity between consecutive spatial views, which is identified by the consistency analysis. After correcting the existential inconsistencies, no further partial inconsistencies are found in the set. After all, from the total of 1024 cases, 897 (87.6%) can be skipped and for none it is necessary to check the sufficient condition (line 15 of Algorithm 4). In total, the analysis took 32 seconds. In summary, the experiment shows that the consistency analysis does a good job in pointing out specification faults. In the author's experience, discontinuities between spatial views are the most common specification error in TSCs. Because the TSCs in the analyzed set belong to an established rule set that discusses only one specific driving situation, it is not surprising that no further partial inconsistencies are found.

8.3.2 Various traffic rules

The basis for this case study is a collection of German traffic rules formalizations by Klemens Esterle and his co-authors [49, 51]. In the original work, the traffic rules have been formalized in LTL. The LTL formulae used there follow certain patterns and are of the form

$$
\begin{aligned}
R &::= \Box(S \Rightarrow P) \mid \Box(P \Rightarrow \neg S) \mid S \Rightarrow \Box(P_1 \Rightarrow P_2) \mid \Box P \\
S &::= S_1 \wedge S_2 \mid P \,\mathcal{U}\, S \mid P \wedge \bigcirc(P \,\mathcal{U}\, S) \\
P &::= p \mid \neg P \mid P_1 \wedge P_2 \mid P_1 \vee P_2 \mid P_1 \Rightarrow P_2
\end{aligned}
$$

where p is a predicate literal. In the LTL traffic rules, the predicate literals describe an atomic constraint for some car with index i or a (spatial) relation between two cars with indexes i and j. The authors indicate by a superscript (i), respectively $(i \to j)$, to which cars the atom is applied. In the LTL rules selected for evaluation, the following atoms are used:

- $\mathit{succ}^{(i \to j)}$: Car i is the direct successor of car j, i.e., car j drives directly in front of car i with no other car in between.
- $\mathit{right}^{(i \to j)}$ ($\mathit{left}^{(i \to j)}$, $\mathit{in\text{-}front}^{(i \to j)}$, $\mathit{behind}^{(i \to j)}$): Car i is right (left, in front, behind) of car j.
- $\mathit{merged}^{(i)}$: Car i has passed a point where two adjacent lanes merged.
- $\mathit{sd\text{-}front}^{(i)}$ ($\mathit{sd\text{-}rear}^{(i)}$): Car i has free safe space to the front (to the rear), i.e., there is no vehicle with distance smaller than $s_{dist} = 10\,\mathrm{m}$ in front (behind) of car i.
- $\mathit{lane\text{-}change}^{(i)}$: Car i is performing a lane change.
- $\mathit{near}^{(i \to j)}$: Car j is near car i, i.e., within a distance of $d_{near} = 5\,\mathrm{m}$.
- $\mathit{near\text{-}lane\text{-}end}^{(i)}$: Car i is close to the end of its lane, i.e., closer than $d_{rem} = 55\,\mathrm{m}$.

- $\mathit{acc}^{(i)}$: Car i is accelerating (with more than $a_{lim} = 0.5\,\mathrm{ms}^{-2}$).
- $\mathit{below\text{-}speed}^{(i)}(v_{lim})$: The current speed of car i is below v_{lim}.
- $\mathit{speed\text{-}adv}^{(i \to j)}$: Car i is faster than car j with a speed difference of at least $v_{diff} = 10\,\mathrm{km/h}$.
- $\mathit{acc\text{-}lane}^{(i)}$: Car i is on an acceleration lane.
- $\mathit{rightmost\text{-}lane}^{(i)}$: Car i is on the right-most driving lane.

This predicate symbols are encoded as spatial views given in Table 8.5. Furthermore, it is $\mathit{left}^{(i \to j)} :\equiv \mathit{right}^{(j \to i)}$ and $\mathit{behind}^{(i \to j)} :\equiv \mathit{in\text{-}front}^{(j \to i)}$.

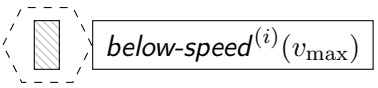
In contrast to Esterle, we only consider restricted environments. Therefore, we replace the following predicate symbols by constants:

- $\mathit{dense}^{(i)}$: This predicate indicates that car i is in dense traffic, i.e., there are at least N other cars around it. Esterle uses $N = 8$. Because we evaluate consistency only for scenarios up to three cars, we can set $\mathit{dense}^{(i)} \equiv \mathit{false}$.
- $\mathit{built\text{-}up}^{(i)}$ indicates that car i drives within a build-up area[5]. However, we exclude this case and define $\mathit{built\text{-}up}^{(i)} \equiv \mathit{false}$.
- $\mathit{div\text{-}lane}^{(i)}$ means that car i is on a diverging lane. Our world model does not include diverging lanes and therefore $\mathit{div\text{-}lane}^{(i)} \equiv \mathit{false}$.
- $\mathit{on\text{-}road}^{(i)}$: Esterle uses the LTL rules for monitoring of traffic. Here, $\mathit{on\text{-}road}^{(i)}$ tracks that car i has not left the monitored part of the road. We do not need this and can safely assume $\mathit{on\text{-}road}^{(i)} \equiv \mathit{true}$.
- $\mathit{num\text{-}lanes\text{-}ge}^{(i)}(3)$ models that car i drives on a road with at least three driving lanes. Because we consider two lane motorways only, it is $\mathit{num\text{-}lanes\text{-}ge}^{(i)}(3) \equiv \mathit{false}$.

In order to translate the LTL rules to TSCs, we apply the translation patterns listed in Table 8.4. Propositional formulae P are translated to invariant nodes. The translation patterns shall reflect the intuition behind the LTL patterns.

For the evaluation, we choose the following eight traffic rules:

- *below speed limit*: This rule asserts that vehicles shall adhere to the legal speed limit of the road. In LTL, this is formalized as $\Box\mathit{below\text{-}speed}^{(i)}(v_{\max})$. For the evaluation, we choose $v_{\max} = 120\,\mathrm{km/h}$. The corresponding TSC is:

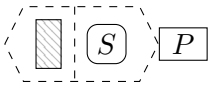
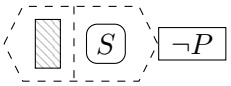
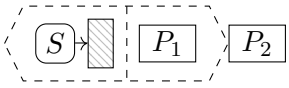
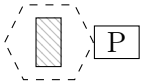
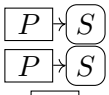


- *no stopping*: This describes that stopping is forbidden on motorways, unless enforced by the environment. In LTL, this is formalized as

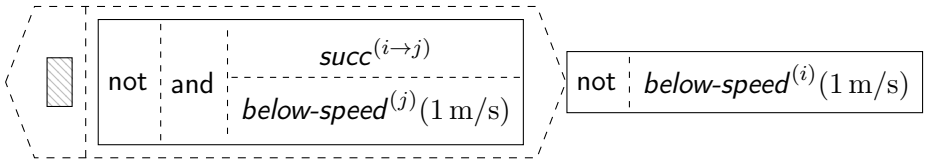
$$\Box\big(\neg(\mathit{succ}^{(i \to j)} \wedge \mathit{below\text{-}speed}^{(i)}[j](v_{stop})) \Rightarrow \neg\mathit{below\text{-}speed}^{(i)}(v_{stop})\big)$$

[5]Meant are urban environments, where different traffic rules apply compared to motorways.

Table 8.4: Translation patterns from LTL to charts

LTL	Chart
Rules R	
$\square(S \Rightarrow P)$	
$\square(P \Rightarrow \neg S)$	
$S \Rightarrow \square(P_1 \Rightarrow P_2)$	
P	
Sequential formulas S	
$P \,\mathcal{U}\, S$	
$P \wedge \bigcirc(P \,\mathcal{U}\, S)$	
P	

which allows stopping (i.e., low speed below $v_{stop} = 1\,\mathrm{m/s}$) only if the preceding vehicle stops. As a TSC, this is:

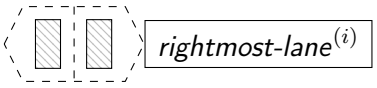


- *keep in right-most lane*: This describes the obligation to use the right lane, including some exceptions such as dense traffic, urban areas (except inner city motorways), or three lane roads. In LTL, it is written as $\square(P_1 \Rightarrow P_2)$ with:

  $P_1 \equiv \neg \mathit{dense}^{(i)} \wedge (\neg \mathit{built\text{-}up}^{(i)} \vee \mathit{motorway}^{(i)}) \wedge (\mathit{built\text{-}up}^{(i)} \vee \neg \mathit{num\text{-}lanes\text{-}ge}^{(i)}(3))$

  $P_2 \equiv \mathit{rightmost\text{-}lane}^{(i)}$

  Because we neither assume dense traffic, urban areas, nor three lane roads, P_1 collapses to *true*. As a consequence, we formalize this rule as:

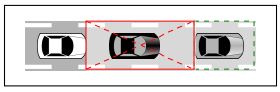
- *no passing on the right side*: This rule forbids overtaking on the right side, with some exceptions. The LTL rule is of the form $\square(\neg P \Rightarrow \neg S)$ where

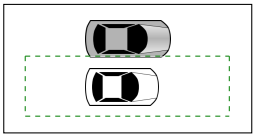
  $$S \equiv \boxed{\mathit{behind}^{(i \to j)}} \to \boxed{\mathit{right}^{(i \to j)}} \to \boxed{\mathit{in\text{-}front}^{(i \to j)}}$$

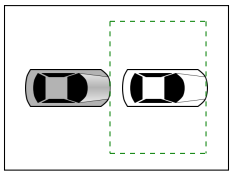
  is a right side overtaking sequence, and

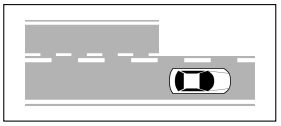
  $$P \equiv \neg \mathit{div\text{-}lane}^{(i)} \vee \mathit{acc\text{-}lane}^{(i)} \vee \mathit{div\text{-}lane}^{(i)} \vee \mathit{dense}^{(i)} \vee (\mathit{built\text{-}up}^{(i)} \wedge \neg \mathit{motorway}^{(i)})$$

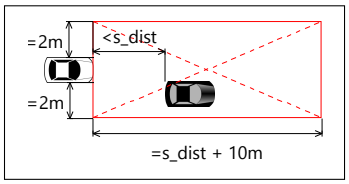
Table 8.5: Translation of predicate symbols (i: white car, j: gray car)

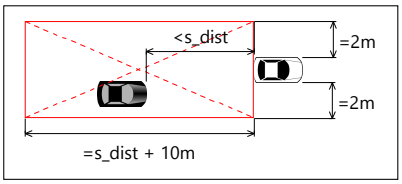
$succ^{(i \to j)} :=$ 

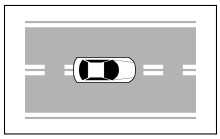
$right^{(i \to j)} :=$ 

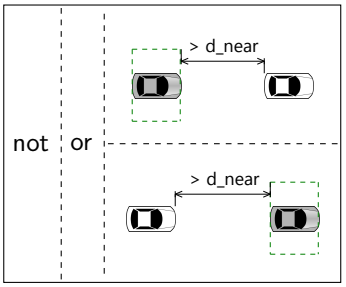
$in\text{-}front^{(i \to j)} :=$ 

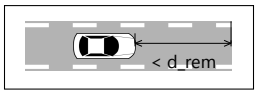
$merged^{(i)} :=$ 

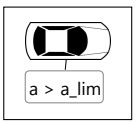
$sd\text{-}front^{(i)} :=$ 

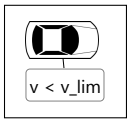
$sd\text{-}rear^{(i)} :=$ 

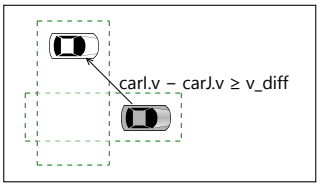
$lane\text{-}change^{(i)} :=$ 

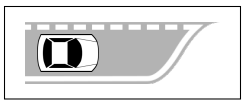
$near^{(i \to j)} :=$ 

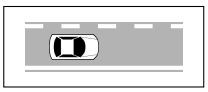
$near\text{-}lane\text{-}end^{(i)} :=$ 

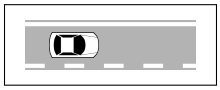
$acc^{(i)} :=$ 

$below\text{-}speed^{(i)}(v_{lim}) :=$ 

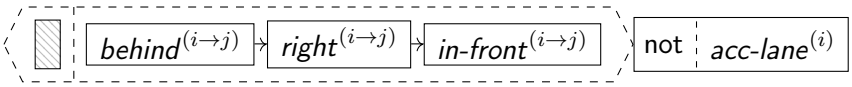
$speed\text{-}adv^{(i \to j)} :=$ 

$acc\text{-}lane^{(i)} :=$ 

$rightmost\text{-}lane^{(i)} :=$ 

$leftmost\text{-}lane^{(i)} :=$ 

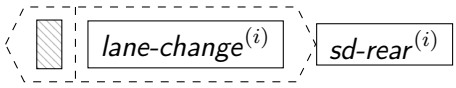
lists the exceptions. The only exception that is applicable in our evaluation are acceleration lanes, therefore P simplifies to $P \equiv \textit{acc-lane}^{(i)}$. Negated sequences are not supported by the consistency analysis. Therefore, we bring this rule into the logically equivalent form $\square(S \Rightarrow P)$ first. This is then translated to



- *safe lane change*: This rule states that lane changes are only allowed when there is safe space behind the vehicle, expressed as:

$$\square(\textit{lane-change}^{(i)} \Rightarrow \textit{sd-rear}^{(i)})$$

As a TSC, this can be formalized as:



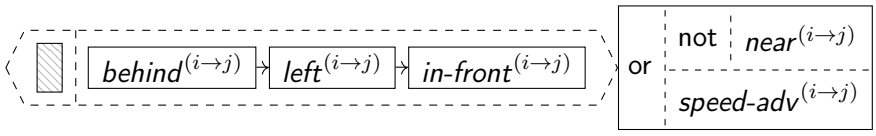
- *speed advantage for overtaking*: This rule states that overtaking is only allowed if the overtaking vehicle is notably faster than the overtaken one. In LTL, this is written as $\square(S \Rightarrow P)$ where

$$S \equiv \textit{behind}^{(i \to j)} \wedge \bigcirc(\textit{behind}^{(i \to j)}\, \mathcal{U}\, \textit{left}^{(i \to j)}\, \mathcal{U}\, \textit{in-front}^{(i \to j)})$$

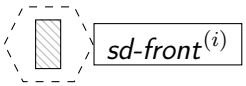
is the overtaking sequence (this time on the left side) and

$$P \equiv \textit{near}^{(i \to j)} \Rightarrow \textit{speed-adv}^{(i \to j)}$$

expresses that the overtaking car i is faster than the overtaken car j whenever they are close to each other. As a TSC, this is:



- *safe distance*: This mandates a safe distance to vehicles ahead, expressed as $\square\textit{sd-front}^{(i)}$ in LTL and

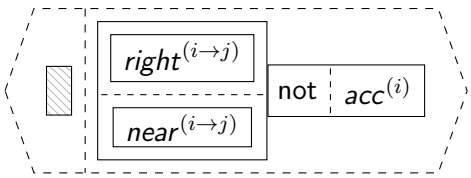


as a TSC.

- *being overtaken*: This rule forbids to accelerate when being overtaken. In LTL, this is formalized as

$$\square(\textit{right}^{(i \to j)} \wedge \textit{near}^{(i \to j)} \Rightarrow \neg \textit{acc}^{(i)})$$

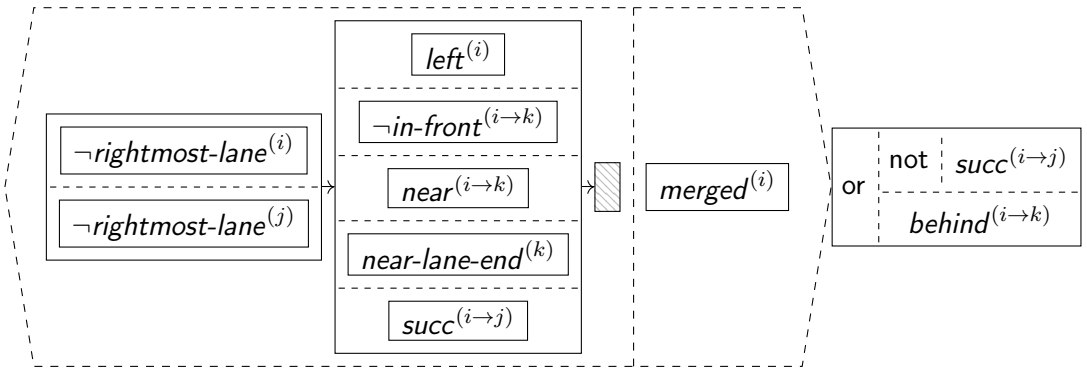
where $\textit{right}^{(i \to j)} \wedge \textit{near}^{(i \to j)}$ is an overtaking situation (ar i being right of and close to another car j). As a TSC, this is:



- *zipper merge*: A zipper merge is applied whenever two driving lanes on a motorway merge. Esterle formalizes a zipper merge of three cars i, j, and k as $S \Rightarrow \Box(P_1 \Rightarrow P_2)$ with:

$$\begin{aligned} S &\equiv (\neg \textit{rightmost-lane}^{(i)} \wedge \neg \textit{rightmost-lane}^{(j)}) \mathcal{U} \\ &\qquad (\textit{left}^{(i)} \wedge \neg \textit{in-front}^{(i \to k)} \wedge \textit{near}^{(i \to k)} \wedge \textit{near-lane-end}^{(k)}) \\ P_1 &\equiv \textit{merged}^{(i)} \wedge \textit{on-road}^{(k)} \\ &\equiv \textit{merged}^{(i)} \\ P_2 &\equiv \neg \textit{succ}^{(i \to j)} \vee \textit{behind}^{(i \to k)} \end{aligned}$$

  Here, S describes the situation where a zipper merge has to be applied, P_1 the point when the zipper merge must be finished, and P_2 the required situation at the end of the zipper merge. In a TSC, we formalize this as:



  Note, that in the TSC, the end condition is checked strictly after the situation S has been recognized.

The complete TSCs (with the resulting spatial views) are given in Figure B.4.

The TSCs are analyzed for partial inconsistencies. The analysis takes about four minutes. From the 2304 cases have 1841 (80%) been skipped. The consistency analysis finds inconsistencies between the following TSCs:

(a) keep in right-most lane

(d) no passing on the right side

(e) safe lane change

(i) zipper merge

TSCs (a), (d), and (i) are pair-wise inconsistent, furthermore the analysis finds that (d) and (i) are inconsistent. The reason for this lies in the formalization.

- (a) $\leftrightarrow$ (d): Rule (d) is formalized as an exception, i.e., right passing is only allowed on acceleration lanes. However, rule (a) forbids to use any other lane than the right one, so the exception never applies.

- (a) $\leftrightarrow$ (e): Similarly to the former, (a) forbids car i leaving the lane boundaries. So, no lane changes in the sense of rule (e) can occur without breaking rule (a).

- (d) ↔ (e): The TSC for rule (e) models lane changes between left and right lane only and neglects the acceleration lane. So, rule (e) cannot be applied while rule (d) is in force.

- (d) ↔ (i): Similar to the conflict (d) ↔ (e), the pre-chart of the zipper merge may involve an overtaking maneuver to the right, which is allowed by rule (d) only in acceleration lanes.

In summary, the inconsistencies stem from the fact that the rules apply to different situations which are not stated clearly in the pre-charts. Especially, there are certain situations that allow to leave the right lane (e.g., overtaking slower vehicles, or dense traffic) which are not specified by rule (a). Furthermore, TSC (d) uses the pre-chart to express forbidden behavior, which contradicts its intended use. The expected specification style is to write TSCs in a "proactive" form: "When you are in the situation formulated in the pre-chart, behave as specified by the consequence". TSC (d) violates this rule which is recognized as an inconsistency here.

8.3.3 Requirements for an Advanced Lane Keeping System

UN Regulation No. 157 [116] defines requirements for advanced lane keeping systems (ALKS). For the following case study, 23 requirements on "system safety and fail-safe response" from Section 5 of the regulation have been chosen for formalization as TSCs. The formalization has been carried out independently by two Bachelor students under supervision of the thesis author. In order to minimize the influence of the thesis author on the case study outcome, the students received only minimal instructions on how to formalize the requirements. Furthermore, the TSCs have not been reviewed by the thesis author before applying the consistency analysis. The following instructions have been given to the students:

- Each requirement shall be formalized as a separate TSC. If needed, a requirement can be formalized using multiple TSCs.

- The world model and symbol dictionary is provided by the thesis author. This is the same symbol dictionary as used for the other case studies, with some extensions:

  - A dedicated object type for the Ego vehicle is introduced that extends the Car type by the following Boolean attributes which describe the state of the ALKS.
    - `alks_activated`: the ALKS is activated
    - `transition_demand`: the ALKS requests the driver to take over control
    - `min_risk_maneuver`: the ALKS executes a Minimum Risk Maneuver (MRM)
    - `emergency_maneuver`: the ALKS executes an Emergency Maneuver (EM)
  - The symbol dictionary defines two symbols that indicate the state of hazard warning lights (see Figure 8.10).
  - Object types and corresponding symbols for emergency vehicles and pedestrians are added (see Figure 8.10).

- A naming/symbol convention for symbol declarations in bulletin boards shall be followed.

Figure 8.10: Additional symbols introduced for the ALKS case study

  - The ego vehicle is named `ego` and shall have the symbol type `EgoCar`. This is the vehicle that is equipped with the ALKS.
  - Follow a consistent naming scheme for lanes (i.e., the lane where the ego drives on shall have the same name in all TSCs).

- The Boolean attributes of `EgoCar` described above shall be used to model the state of the ALKS.

- When formalizing Requirement 5.4.2.2, the "unplanned event" shall be an approaching emergency vehicle.

The rationale behind the given instructions is to ensure applicability of the consistency analysis. The UN Regulation contains separate requirements on the conditions under which a MRM or EM shall be executed, and on how a MRM or EM shall be carried out. Introducing the ALKS state variables described above allows to maintain this separation (and therefore a traceability between TSCs and textual requirements) in the TSC specification without doubling information.

Constructing the charts $\mathsf{BC}_1^{\mathsf{TSC},T}$ and $\mathsf{BC}_2^{\mathsf{TSC},T}$ for a partial consistency analysis case (see Definition 21) requires to merge the bulletin boards of the individual TSCs. Note, that according to TSC semantics (Definition 9) object variables from the bulletin board are all-quantified. Therefore, different strategies for merging bulletin boards are possible. The analysis prototype does this by unifying bulletin board entries with the same name and type. So, maintaining a naming scheme increases the likelihood that existing inconsistencies are discovered by the analysis. Furthermore, a naming scheme improves readability of the specification.

The UN regulation states that the list of "unplanned events" that an ALKS must be able to recognize is the outcome of a risk analysis to be carried out as part of the system development. Therefore, the UN regulation only gives examples here. For the sake of this case study, it makes sense to consider a single well-known "unplanned event".

The TSCs have not been reviewed by the thesis author before application of the consistency analysis. In order to evaluate the applicability of the consistency analysis to a larger set of TSCs, the students have been told to specify all requirements completely before using the analysis.

The specification resulted in a set of 23 TSCs. Each selected requirement has been formalized as a single TSC. It turns out that the students formalized the rule set in different ways, which may come from their different background and experience. Student A tried to cover as many (corner) cases as possible, including cases not explicitly described in the requirements. For example, Student A's TSC for Requirement 5.2.2 considers two other vehicles merging from different lanes. Student B instead aimed at a minimalistic formalization, including only explicit described cases. Consequently, his specification is more sober and much less complex. In the following, we refer to the TSC specification

produced by Student A as Version A, and the one produced by Student B as Version B, respectively.

In the following, the consistency analysis results for Version A are examined, because this is the more complex one and therefore more interesting for the analysis. The consistency analysis results for Version B are briefly discussed afterwards.

8.3.3.1 Consistency Results for Version A

For Version A, a first run of the consistency analysis tool uncovered syntax errors in two of the TSCs (the analysis prototype emits error messages in this case), which could be fixed by the student. Appendix B.4 lists the 23 TSCs after fixing these syntax errors. A complete partial consistency analysis consists of approximately 96.5 million cases, of which 200 409 cases pass (i.e., no inconsistency found by the prototype implementation), and 59 are partially inconsistent. The remaining ca. 96.3 million cases (99.79%) can be skipped. This includes all subsets of 14 or more TSCs. The 59 inconsistent cases consist of three weakly existentially inconsistent TSCs (ALKS requirements 5.2.5.1 [Figure B.11] and 5.2.5.2 [Figure B.12]), 46 partially inconsistent pairs, and 10 partially inconsistent triples.

The weak existential inconsistencies can be explained as follows:

- The TSC for 5.2.5.1 contains a sequence of two spatial views where the first one demands $\mathsf{other.acc} \geq 0$ while the second one requires $\mathsf{other.acc} < 0$. Because spatial views are evaluated on left-closed intervals, this causes a discontinuity in the acceleration.

- The TSC for 5.3.3.2 has a similar inconsistency.

- The TSC for 5.2.5.2 contains self-contradicting timing annotations.

The pairwise partial inconsistencies have different causes which are summarized below.

a) *Incomplete case distinctions.* The requirement 5.2.3.3 defines speed-dependent minimal distances. In the TSC, this is realized using a choice chart describing the distances for the different speed intervals. This choice does not include a case for standstill of ego. This causes an inconsistency with other TSCs, for instance 5.2.4 and 5.5.2.

b) *Missing deceleration end.* The TSC for 5.5.1 requires permanent deceleration (even after standstill), while 5.5.2 requires deceleration to standstill. This is detected as an inconsistency.

c) *Incomplete pre-charts.* In some TSCs, the pre-charts do not describe the situations completely in that the requirements are to be applied. Consequently, the behaviors described in the consequences are contradictory to other TSCs in the set. This includes for instance the pairs 5.2.2 $\leftrightarrow$ 5.2.3.3 and 5.2.2 $\leftrightarrow$ 5.2.4.

d) *Incomplete mode descriptions/prioritization.* Some requirements/TSCs, for instance 5.2.2, describe normal operation, while others apply to EMs, MRMs, or transition demands. This can be seen as a special form of the case described in the previous item, but indicates especially an unclear prioritization between the different maneuvers. It applies to the pairs 5.2.2 $\leftrightarrow$ 5.3.1.1 and 5.4.3.1 $\leftrightarrow$ 5.2.3.3, for instance.

e) *Mixing up pre-chart and consequence.* Reviewing the specification shows that for some TSCs, part of the consequence describes behavior that actually belongs to the application condition of the requirement, and – being erroneously specified in the consequence instead of the pre-chart – causes conflicting behavior. This is for instance detected in the pair 5.2.2 $\leftrightarrow$ 5.2.3.3.

f) *Mode changes.* Many reported partial inconsistencies are caused by mode changes described in one of the requirements. For instance, 5.5.5 requires that the ALKS is shut off, while, e.g., 5.2.1 applies only if the system is active. A similar case is caused, for instance, by TSC 5.5.4 describing when a MRM shall end. In contrast to the cases above, the inconsistency is caused by a conflict between the consequence of one TSC and the future of an other.

Items a – e in the above list describe specification faults in the TSCs that indeed cause contradicting or impossible behavior. The weak existential inconsistencies and the partial inconsistencies under item e above are probably caused by missing experience of the student in specifying requirements – so far, the student had only experience with formalizing test cases as TSCs. Therefore, the role of the pre-chard may not have been fully understood. Items a – d instead uncover more subtle conflicts in the requirements. Especially prioritization between requirements (item d) is a known challenge for formal specification, as they are often only implicitly contained in textual specifications [120].

Item f, however, indicates a possible gap in the used formal consistency notion. Inspecting the conflicting sets of spatial views for these cases shows some pattern: In each of the pairs, one of the TSCs requires a behavior that makes the other TSC inapplicable. Both of the TSCs are applicable in general, but will not occur at the same time. The formal definitions of strong existential consistency (Definition 20) and partial consistency (Definition 21) consider this already for mutually exclusive pre-charts. The consistency notion may be extended to also exclude consistency cases where pre-chart and consequence of different TSCs exclude each other.

8.3.3.2 Consistency Results for Version B

For Version B, the consistency analysis also reported syntax errors, which have been fixed by the thesis author. In the fixed version, four TSCs are found weakly existentially inconsistent. Furthermore, 25 partially inconsistent pairs are found (sets of larger size have not been analyzed for Version B).

One of the weak existential inconsistencies is a discontinuity between consecutive spatial views that forces a vehicle to jump between lanes (i.e., an intermediate spatial view is missing). In another case, two vehicles are put in one instead of two somewhere boxes, causing unwanted and contradicting constraints on vehicle sizes. In the two remaining cases, a textual predicate annotation contradicts with the visually indicated state of the hazard warning lights.

The partial inconsistencies are all of the same form as item f identified for Version A above: A conflict between operating modes in the pre-chart of one TSC and in the consequence of another TSC. These inconsistencies have been identified above as a possible gap in the consistency notion which is to be addressed in future work.

8.3.4 Summarized Findings from the Case Studies

Overall, the case studies show scalability of the consistency analysis to medium sized specifications – using server hardware[6], a complete specification consisting of 23 TSCs can be analyzed within approximately four hours. Smaller sets ($\sim$ 8 TSCs) can be analyzed on a desktop PC within a few minutes. This is possible because in practice only a small subset of the consistency cases need to be evaluated. For the smaller case studies, only 20% (resp. 13%) of the cases need to be executed, for the big case study even only 0.7%. Furthermore, all inconsistencies found in the case studies are within single TSCs (i.e., weak existential inconsistencies), or pairs of TSCs. For the biggest case study, all subsets with more than 14 TSCs can be skipped from the analysis. This indicates that a complete consistency analysis is not necessary in practice.

When analyzing the found inconsistencies, the computation of minimal conflicting sets of spatial views (see Section 6.6) turned out to be a very helpful tool in the in-depth inspection of inconsistent subsets.

The found weak existential inconsistencies are, except for one case, all caused by discontinuities in consecutive invariant nodes. This seems to be a common pitfall when formalizing requirements (or scenarios in general) with TSCs that reliably seems to be found by the consistency analysis.

Also, most of the pairwise partial inconsistencies found in the case studies indicate problems in the specification. Beneath individual specification faults in individual TSCs, two main causes for these inconsistencies can be identified:

- In the second case study, traffic rules have been simplified during formalization. Especially, exceptions to the general traffic rules (e.g, cases in that a vehicle is allowed to leave the right-most lane) have been omitted. This leads to other traffic rules being inapplicable.

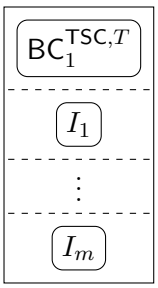
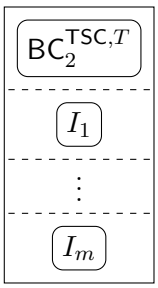
- In the third case study (Version A), the specification is quite exhaustive. However, the pre-charts (i.e., application conditions for the requirements) tend to be incomplete, which then results in conflicts between requirements.

However, some of the experimental results indicate that the definition of *partial consistency* has room for improvement. The consistency analysis is tailored to requirements following a reaction pattern, i.e., it is assumed that the consequence formulates an action to be executed by the HAV when it finds itself in the situation specified in the pre-chart. This may lead to unexpected inconsistency results when the specification refrains from this pattern. For example, the case study presented in Section 8.3.2 contains a TSC where the pre-chart describes generally forbidden behavior and the consequence states under which condition this behavior is still allowed. The consistency analysis treats this as a malformed specification which is inconsistent to other TSCs in the set.

Furthermore, an inspection of the witness trajectories (produced by the premises $\mathsf{BC}_1^{\mathsf{TSC},T}$ of the consistency check) shows that the solver may produce witness trajectories where the vehicles collide. This occurs especially in the second case study when analyzing the zipper merge. The sufficient condition (Algorithm 9) does not encode collision freedom per se. Most spatial views exclude collisions between the depicted vehicles by definition, so

[6]Version A of the ALKS study has been executed on a server with 20 AMD EPYC 7542 32-Core@3GHz processors and 128GB RAM (assigning 38GB of memory to the Java virtual machine), running up to 50 solver instances parallel.

it turns out to be no problem in most cases. In the second case study, however, spatial constraints have been modeled quite selectively, so the TSCs do not induce collision freedom in all cases. This can be addressed by either extending Algorithm 9 to add pairwise collision freedom constraints between all vehicles, or by parallel-composing $\mathsf{BC}_1^{\mathsf{TSC},T}$ with respective invariant nodes. Note that adding collision freedom as additional TSCs to the requirement specification does not exclude this case, because the consistency check still forms consistency cases without these TSCs. This may be addressed by modifying partial consistency (and analogously strong existential consistency) to handle invariant TSCs differently from the remaining TSCs in the specification. Something similar is done in related work for pattern-based requirements [10], where partial consistency is checked in the context of some set of invariants. A TSC may be considered invariant if it has an empty pre-chart, and the consequence consists of a single invariant node. When checking partial consistency, this invariant node can be parallel-composed to both $\mathsf{BC}_1^{\mathsf{TSC},T}$ and $\mathsf{BC}_2^{\mathsf{TSC},T}$. For some set $S = \{\mathsf{Inv}_1, \ldots, \mathsf{Inv}_m\}$ of invariant TSCs with consequences $\{I_1, \ldots, I_m\}$, the existential TSCs $\mathsf{BC}_1^{\mathsf{TSC},T}$ and $\mathsf{BC}_2^{\mathsf{TSC},T}$ would become

$\widehat{\mathsf{BC}}_1^{\mathsf{TSC},T,S} :\equiv$  and $\widehat{\mathsf{BC}}_2^{\mathsf{TSC},T,S} :\equiv$  .

The third case study contains TSCs where the pre-chart of one TSC is mutually exclusive to the consequence of another TSC. An example are TSCs for the ALKS requirements 5.4.3.1 and 5.4.4. Here, 5.4.4 ends a transition demand during the consequence, while 5.4.3.1 assumes (in the pre-chart) that a transition demand is active. Similar cases occur in the second case study, for example between TSC (a) "keep in right-most lane" which renders TSC (e) "safe lane change" inapplicable. However, while in the latter case TSC (e) is permanently inapplicable, which can be considered an issue in the specification, while in the former case both TSCs are still applicable – they just can't occur at the same time, which is not a specification issue. In order to resolve this, *partial consistency* may be refined, adopting another idea from the thesis author's earlier work [10]. Partial consistency for pattern-based requirements [10] evaluates multiple premises: It is checked whether each requirement can be triggered also parallel to the *actions* of the other requirements. Transferred to TSCs, it excludes the described case: A set of requirement TSCs is not considered inconsistent, if the consequence of another TSC in the set prevents occurrence of the pre-chart. The version of partial consistency presented in this thesis, however, only detects mutual exclusive pre-charts in the set.

Extending the consistency notion of *partial consistency* in the sketched ways can alter the analysis results in both directions—both excluding the unwanted inconsistency results identified above, as well as preventing the detection of currently found inconsistencies.

8.4 Basic Maneuvers and Simulation Accuracy

Section 7.5 gives formal proofs of the correctness of the vehicle dynamics encoding. All trajectories generated by the approach adhere to the vehicle dynamics assumed by a single

Table 8.6: Coverage of maneuvers in demonstration scenarios

maneuver	**demonstration scenario**		
	overtaking	**turn**	**parking**
vehicle state maneuvers			
accelerate/driveaway			X
keep velocity	X		
decelerate/halt			X
reversing			X
infrastructure maneuvers			
follow lane	X	X	X
change lane	X		X
approach/cross junction		X	
right/left turn		X	
park			X
object-related maneuvers			
follow object	X		
approach object			X
passing	X		X

Table 8.7: Vehicle parameters for the demonstration scenarios (taken from [12])

parameter	**variable**	**value**
max. steering angle	$\delta_{\max}$	45.0°
front	F	0.8 m
wheel base	L	2.8 m
rear	G	0.7 m
width	W	1.8 m

track model. However, due to the conservative step-wise linear approximations, there is no guarantee that a solution for a particular scenario is found. In order to give at least some empirical evidence that the dynamics encoding is sufficiently complete, we examine its ability to generate trajectories for common driving maneuvers. Hartjen et al. [70] provide a terminology of basic driving maneuvers in urban traffic scenarios. Similar maneuvers are also contained in the automotive traffic ontologies such as OpenXOntology and A.U.T.O, as well as OpenSCENARIO. In one of the thesis author's publications [12], it is evaluated whether it is possible to find trajectories for basic driving maneuvers using the construction of sufficient constraints described throughout this thesis. The following scenarios have been examined: an overtaking (Figure 8.11a), a right and left turning (Figures 8.11b and 8.11c), and a parking scenario (Figure 8.11d). The four scenarios have been selected in order to cover the driving maneuvers identified by Hartjen et al. [70]. The mapping from maneuvers to the demonstration scenarios is given in Table 8.6.

The scenario parameters (vehicle characteristics and inputs for Algorithm 6, dimensions of the crossing, unrolling depth and step size) can be found in Tables 8.7 and 8.8, and Figure 8.12. For all four scenarios, solutions are found by the SMT solver. The resulting trajectories are plotted in Figure 8.13.

Table 8.8: Heading intervals, number of slices (N), and slice length (Δ) for the demonstration scenarios (taken from [12])

scenario	**intervals $\mathcal{I}$**	**#slices N**	**slice length Δ**
overtaking	$[0,0]^\circ$, $[-10,10]^\circ$	20	2 s
parking	$[0,0]^\circ$, $[-15,15]^\circ$, $[15,15]^\circ$, $[0,30]^\circ$	8	3 s
left turn	$[0,0]^\circ$, $[0,45]^\circ$, $[45,90]^\circ$, $[90,90]^\circ$	8	3 s
right turn	$[0,0]^\circ$, $[0,-45]^\circ$, $[-45,-90]^\circ$, $[-90,-90]^\circ$	8	3 s

8.5 Achievement of Thesis Goals

In Section 4.3, a set of requirements on the TSC consistency analysis has been formulated. This section describes how each of them has been addressed in this thesis.

Thesis Goal 1. *The consistency analysis shall produce only valid inconsistency results, and the correctness of the identified inconsistencies shall be prioritized over completeness.*

This goal is satisfied by the design of the consistency notions and the rigorous formal approach of the work. Goal 1 is ensured by the relation between strong existential consistency and partial consistency (see Theorem 2) as well as the proven correctness of Algorithm 4 (see Theorem 4). However, the case study results presented in Section 8.3.4 indicate that the consistency notions require some fine-tuning. The consistency analysis framework presented is flexible enough to support improved future variants of weak existential and partial consistency.

Thesis Goal 2. *It shall be possible to reuse consistency analysis results about a subset of requirements to evaluate a refined set of requirements.*

The definitions of strong existential consistency (Definition 20) and partial consistency (Definition 21) ensure that inconsistency results about subsets of requirement TSCs remain valid under addition of new requirements (see also Figure 5.1 in Chapter 5). A refinement of the world model, however, can turn a consistent subset into an inconsistent one and an inconsistent subset into a consistent one. In the case of monotone refinements (Definition 24), the former can occur if the world model refinement restricts behaviors in a way that some requirement TSCs cannot be fulfilled in parallel anymore. The latter can happen if the refinement excludes the behavior that leads to the conflicting situation. According to Theorem 3, if the left or right side of the implication (CONS) in Definition 20 (strong existential consistency) was *false* before a monotone refinement, then it remains *false* after the refinement. The same holds for the implication (PCONS) in Definition 21 (partial consistency). This means that when re-evaluating a consistency check, many BMC problems can be skipped. In particular, we can initialize the `Skip` set at the beginning of Algorithm 4 with the final `Skip` set from the previous run.

Thesis Goal 3. *The consistency analysis shall be able to point out the source of an identified inconsistency, e.g., by producing minimal inconsistent subsets.*

According to Theorem 4, the inconsistent sets produced by Algorithm 4 are minimal with respect to the complete set of inconsistent subsets that can be found using the described SMT encoding of TSCs. Furthermore, the inconsistencies can be narrowed down to minimal

sets of spatial views within the TSCs. This turned out to be especially useful for explaining why a set of TSCs is inconsistent, and for resolving inconsistencies.

Thesis Goal 4. *The consistency analysis shall be able to produce trajectories as witnesses.*

For non-singleton minimal inconsistent sets, line 15 of Algorithm 4 produces a trajectory showing the conflicting application of the pre-charts. By definition, a witness trajectory does not exist for existentially inconsistent TSCs. However, the CheckSatS algorithm (when used in stead of CheckSatN) can produce witness trajectories for weak existentially consistent TSCs. Due to the approximations used in the BMC encoding, this may require manual tuning of parameters for the single track model encoding and is not complete. However, the examples given in Section 8.4 show that trajectories for a wide range of different maneuvers can be created, including those required by the following goal:

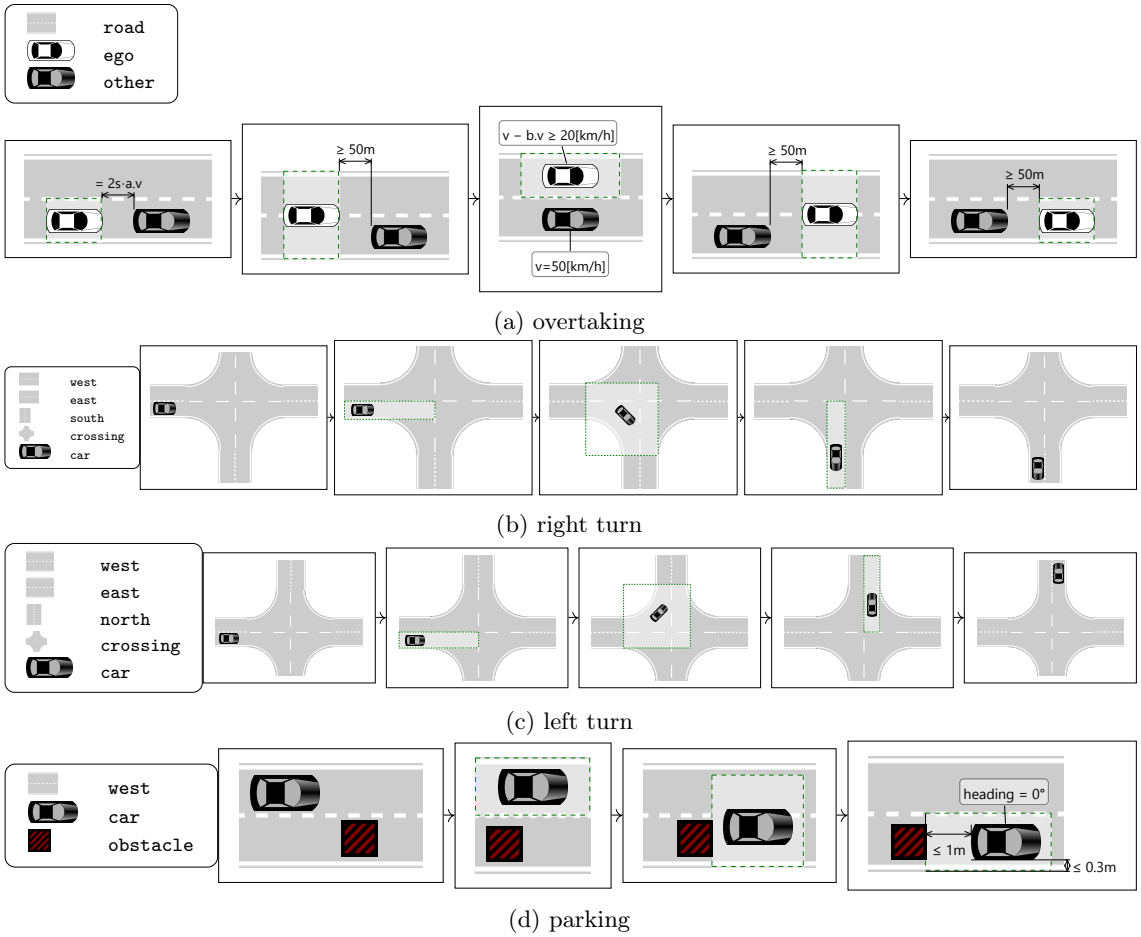
Thesis Goal 6. *The consistency analysis shall use a simple vehicle dynamics model that is capable of generating sound trajectories for abstract scenarios involving:*

- *acceleration from/to standstill*
- *lane change*
- *right/left turns*
- *velocity, acceleration, and distance constraints*

The reader may have noticed that one goal is missing from the list above:

Thesis Goal 5. *The consistency analysis shall support user-defined world models containing custom entity types and attributes.*

This goal is fulfilled by the prototype implementation shown in Section 8.1 which reads the world model and symbol dictionary definitions from XML files. Thereby, vehicles are the only objects handled specially during BMC problem construction; the specification of other object types is entirely done in the XML file, including the specification of object invariants. Extra attributes and invariants can also be added to vehicle types.



(a) overtaking

(b) right turn

(c) left turn

(d) parking

Figure 8.11: Additional demonstration scenarios (taken from [12])

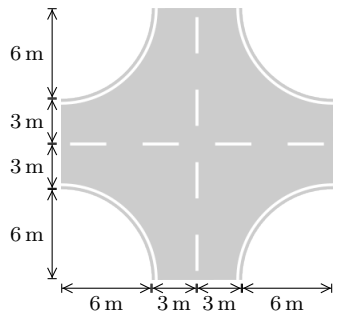
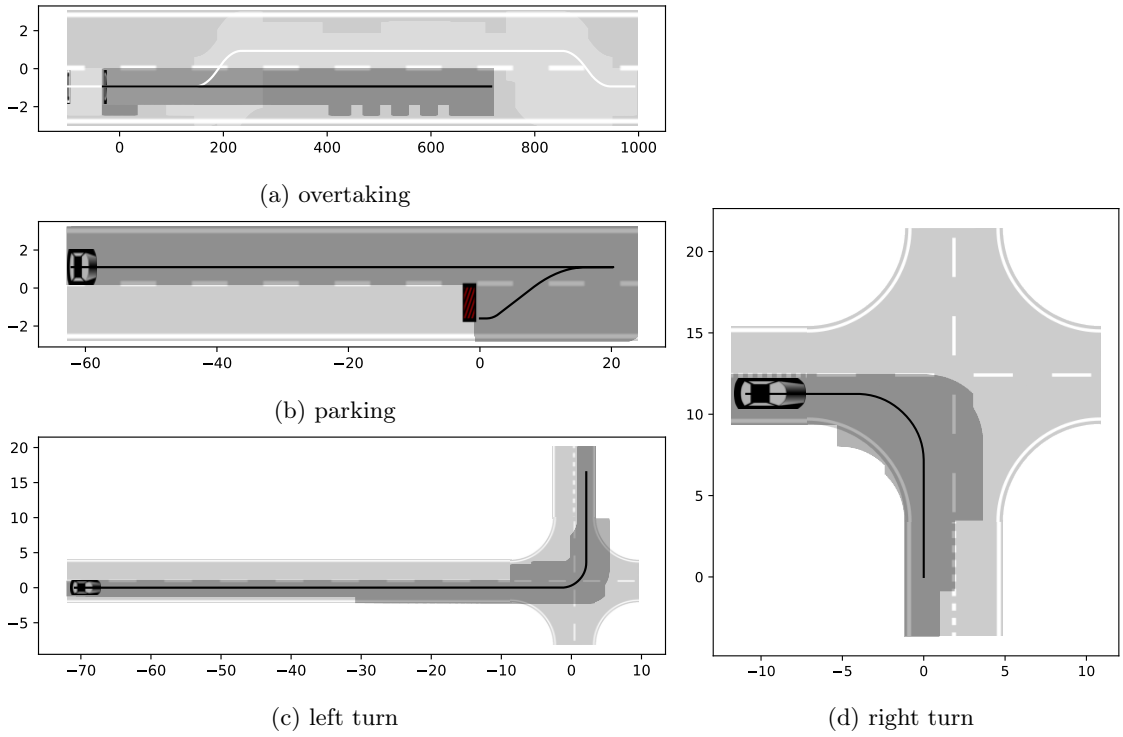


Figure 8.12: Dimensions of the crossing in the demonstration scenarios (taken from [12])



(a) overtaking

(b) parking

(c) left turn

(d) right turn

Figure 8.13: Generated trajectories (taken from [12]); the solid lines show vehicle trajectories, and the shaded areas the bounding box approximation; axes are in Meters

9 Conclusion & Outlook

In this thesis, a consistency analysis framework for scenario-based requirements is developed. TSCs are used as a specification language for those requirements. Driven by its role in a development process for HAVs, the consistency analysis is a rigorous formal framework around TSCs with the goal to find provably inconsistent subsets, as expressed in the following goal:

Thesis Goal 1. *The consistency analysis shall produce only valid inconsistency results, and the correctness of the identified inconsistencies shall be prioritized over completeness.*

As this goal clearly states, the focus is not on completeness but on correctness of the findings. Therefore, the algorithms are based on a formal notion of consistency. The correctness proofs are a major part of this thesis.

The main idea behind the consistency analysis procedure is to reduce consistency to existence of trajectories satisfying certain existential TSCs. In order to tackle the computational complexity of deciding that, a two-sided approximation with linear satisfiability checking is chosen. The TSCs are translated to BMC problems which are passed to a state-of-the-art SMT solver. One side of the approximation is based on a conservative linear encoding of a single track vehicle dynamics model. As a side effect of the approach, trajectories for the vehicles in the scenario can be produced.

The consistency analysis prototype has been implemented as part of the TSC editor, a specification tool for traffic sequence charts which is developed at the author's research institute.

Different experiments (including three use cases, one based on independent work [49, 50]) that asses the applicability and scalability of the approach have been carried out with the prototype implementation. From the experimental results it can be concluded that the analysis is applicable to small and medium-sized specifications (the case studies consist of up to 23 TSCs). The inconsistencies found stem from impossible transitions (i.e., discontinuities) between basic charts, and underspecification of application conditions (i.e., incomplete pre-charts), including missing prioritization between requirements. At the same time, the case studies uncover some limitations of the approach presented.

- Finding satisfying trajectories for TSCs (via solving sufficient constraints as described in Chapter 7) requires some manual tuning of algorithm parameters, especially step size and discretization of the vehicle heading during the scenario. If the discretization is to coarse, no trajectory may be found. On the other hand, a fine discretization increases the complexity of the SMT problem. In this thesis, standard parameters are proposed that provide a good trade-off for highway scenarios. Urban scenarios with turns have shown to be tricky. Here, future work shall improve the encoding. Also, heuristics for parameter selection would be useful. In the end of Section 6.2 a technique is proposed to identify all subsets that may contain an undiscovered inconsistency due to this problem. In the case studies, this occurs once. Besides that, the two-sided approximation seems not to be a problem for applicability or usability of the analysis method.

- The consistency analysis is tailored to requirements following a reaction pattern, i.e., it is assumed that the consequence formulates an action to be executed by the HAV when it finds itself in the situation specified in the pre-chart. This may lead to unexpected inconsistency results when the specification refrains from this pattern. For example, the case study presented in Section 8.3.2 contains a TSC where the pre-chart describes generally forbidden behavior and the consequence states under which condition this behavior is still allowed. The consistency analysis treats this as a malformed specification which is inconsistent to other TSCs in the set. In order to resolve this, other specification patterns may be incorporated into the analysis.

- Similarly, the partial consistency reports inconsistencies if the consequence of one TSC excludes the future of another one. If the first TSC is an invariant (i.e., it does not specify a pre-chart and the consequence consists of a single invariant node), this indicates a problem in the specification, as the second TSC is never applicable. If none of the TSCs is an invariant, however, this case may be tolerated. Related work [10] that specifies partial consistency for pattern-based requirements, considers this case already. Hence, ideas from the related work may be incorporated into the consistency notions.

- The consistency analysis may benefit from a special handling of invariant TSCs. Besides improving the analysis results, this will allow to incorporate additional domain knowledge formalized as TSCs.

The consistency analysis method proposed in this thesis has been designed with flexibility in mind – it can be implemented for other consistency notions as well, provided they allow to reduce consistency to satisfiability of a finite number of positive basic charts. Therefore, future work can, and shall, sharpen the formal consistency notions in order to improve their benefit for engineers.

The consistency analysis prototype is able to point out the spatial views that cause an identified inconsistency. This allows the user of the analysis tool to inspect any found inconsistency and to decide whether and how it shall be resolved. This includes dealing with the limitations of partial consistency developed in this thesis. One lesson learned from the case studies is, however, that resolving inconsistencies requires experience in requirement formalization. Furthermore, TSC specifications may suffer from other issues, such as unconsidered adverse behavior of other traffic participants, that is outside the scope of a consistency analysis (adverse environment conditions are explicitly excluded from the problem definition in this thesis, see Section 4.2). The consistency analysis can support engineers in writing high quality specifications, but cannot replace a thorough training in the use of TSCs. Furthermore, it shall be applied in combination with other methods, such as risk analysis [101] and model-based testing [18] of the specification.

Related work sees consistency as a multimodal concept, covering relations between all kinds of artifacts (specifications, models, code, ...) in the development process. While this thesis focuses on the technical realization of a consistency analysis between scenario-based requirements, future work shall integrate it into a larger conceptual framework such as the V-SUM method described by Reussner et al. [105] and Feichtinger et al. [53].

Ongoing work is about to extend and adapt the TSC formalism itself and the tooling to other transportation domains, such as maritime or railway. This also gives opportunities for combining the applied methods with other established approaches from the field of hybrid systems model checking. An open research question here is whether the use of non-linear

and numerical solvers (e.g., iSAT with ODE extensions [46]) can overcome the limitations of the current encoding of the single track vehicle model. Furthermore, guidelines need to be developed that guide engineers in the use of TSCs and the existing and future analysis methods.

Outside the scope of a consistency analysis, the generation of trajectories has been shown to be quite useful for the generation of test scenarios [18] and as a supporting tool for criticality analysis [14, 15]. Furthermore, translating spatial views to quantifier-free mathematical formulae may be an intermediate step in the automated synthesis of scenario monitors [66]. Therefore, these parts of the developed framework will be subject to future research and development work.

A Alternative Encoding

The following describes the alternative encoding scheme for the chart structure that is used in Section 8.2.1. The encoding scheme is based on [58, 113].

In contrast to the incremental encoding scheme from Section 6.3, this is a more monolithic approach, where the set of constraints Φ_n is constructed for some given unrolling depth $n \in \mathbb{N}$. The translation works as follows:

- For each time variable τ introduce some real-sorted variable v_τ
- Introduce real-valued variables $t_1, t_2, \ldots, t_n$ and a constant $t_0 := 0$
- For each sub-formula φ and indices $0 \leq i < j \leq n$ introduce a Boolean variable $\alpha_\varphi^{i,j}$
- for each state predicate ϕ and index $0 \leq i < n$ introduce a Boolean variable β_ϕ^i

The set of constraints Φ_n consists of the following constraints:

- $\alpha_{\varphi_0}^{0,n}$ for top-level formula φ_0
- A constraint for each sub-formula $\varphi(b, e)$ of φ_0 (including φ_0 itself) and indices $0 \leq i < j \leq n$ according to the following table

$\varphi(b,e)$	Constraint
$\Box_{[b,e)}\phi \wedge b{<}e \wedge \psi$	$\alpha_\varphi^{i,j} \Leftrightarrow \widetilde{\psi} \wedge \bigwedge_{k=i}^{j-1} \beta_\phi^k$
$\varphi_1(b,e) \wedge \varphi_2(b,e)$	$\alpha_\varphi^{i,j} \Leftrightarrow \alpha_{\varphi_1}^{i,j} \wedge \alpha_{\varphi_2}^{i,j}$
$\varphi_1(b,e) \vee \varphi_2(b,e)$	$\alpha_\varphi^{i,j} \Leftrightarrow \alpha_{\varphi_1}^{i,j} \vee \alpha_{\varphi_2}^{i,j}$
$\exists m \bullet \varphi_1(b,m) \wedge \varphi_2(m,e)$	$\alpha_\varphi^{i,j} \Leftrightarrow \bigvee_{k=i+1}^{j-1} \alpha_{\varphi_1}^{i,k} \wedge \alpha_{\varphi_2}^{k,j}$

  with $\widetilde{\psi} := \psi[b \backslash t_i, e \backslash t_j, \tau \backslash v_\tau \mid \tau \in V_T \setminus \{b, e\}]$.

- A constraint $\beta_\phi^i \Leftrightarrow tr(\phi)[X \backslash X_i, X' \backslash X_{i+1}]$ for each $0 \leq i < n$

B TSCs from the Case Studies and Experiments

B.1 TSCs for Section 8.2.1 – Platooning

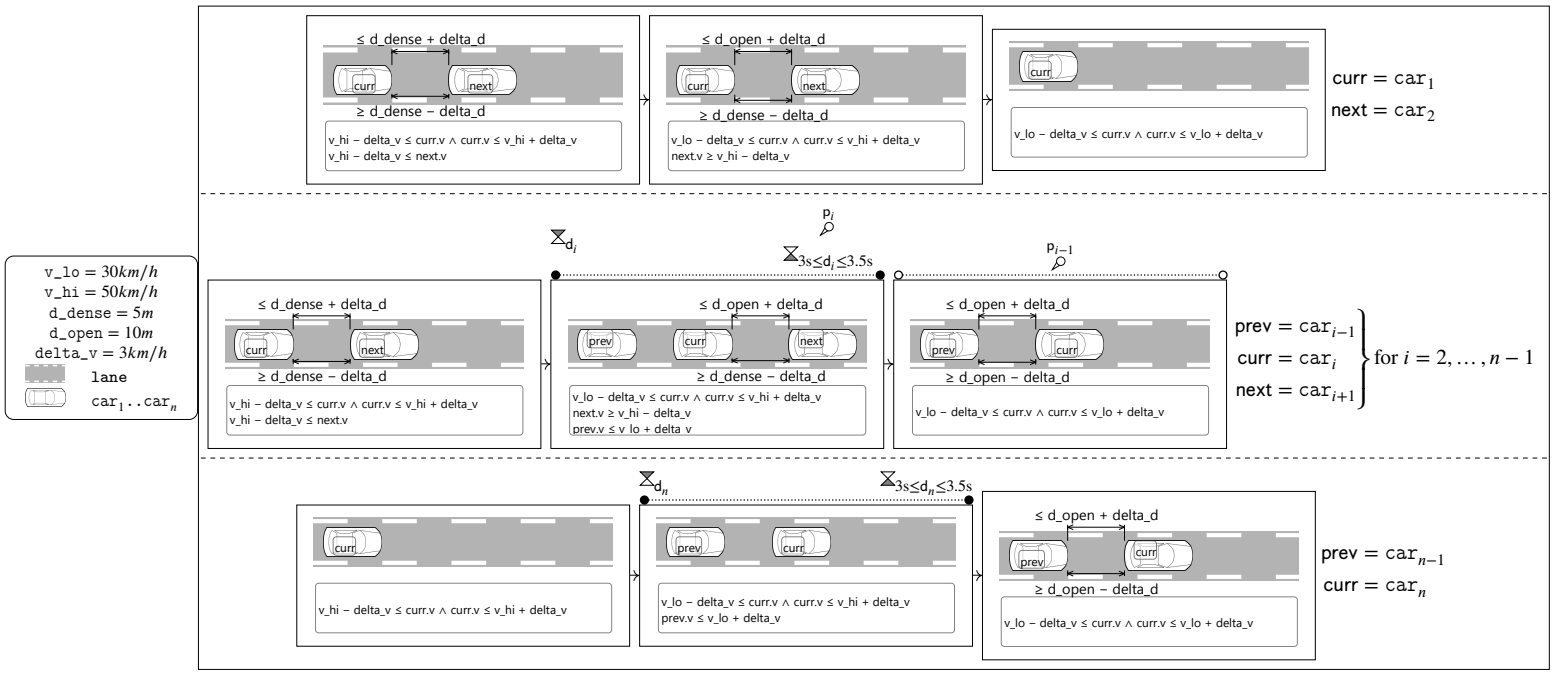


Figure B.1: From dense to open formation – consistent case

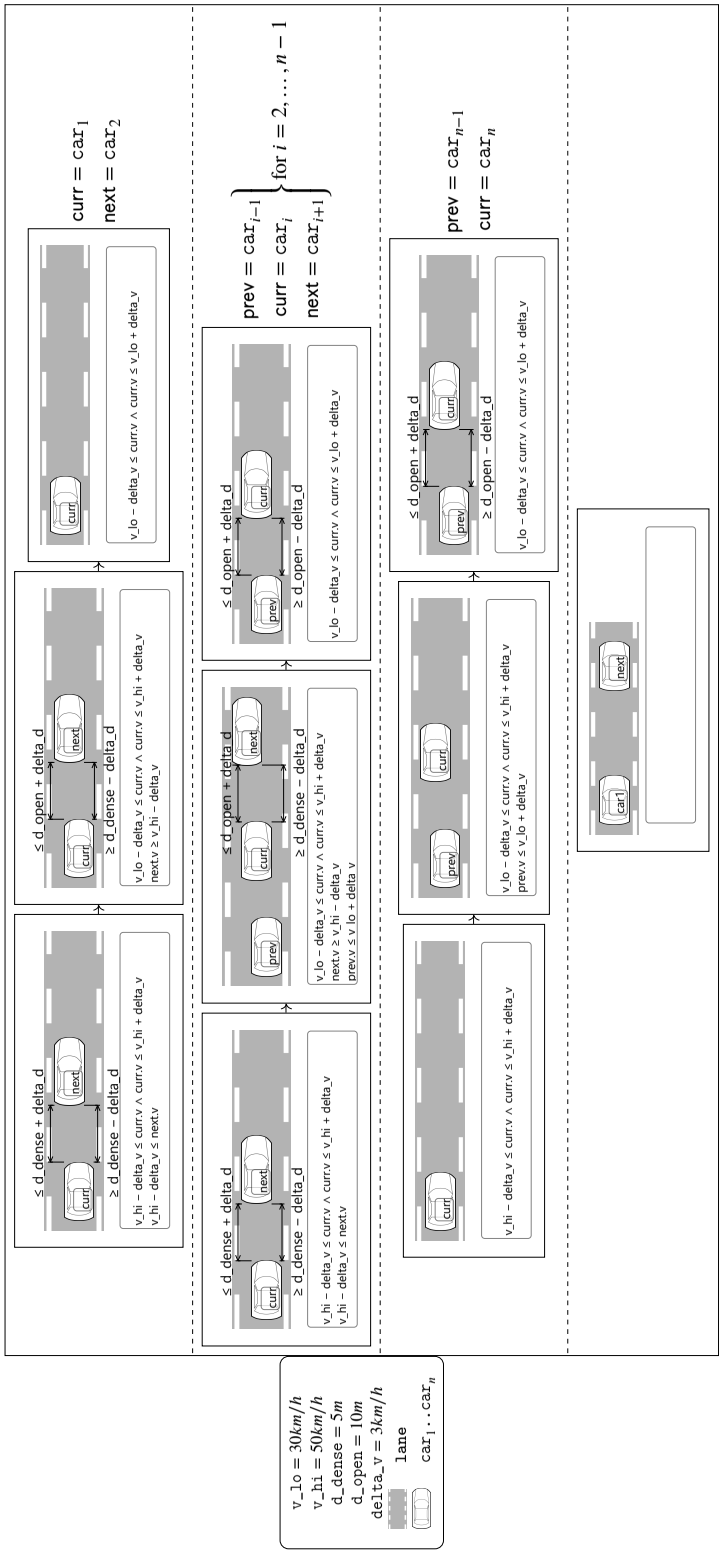


Figure B.2: From dense to open formation – inconsistent case

B.2 TSCs for Section 8.3.1 – Overtaking Rules

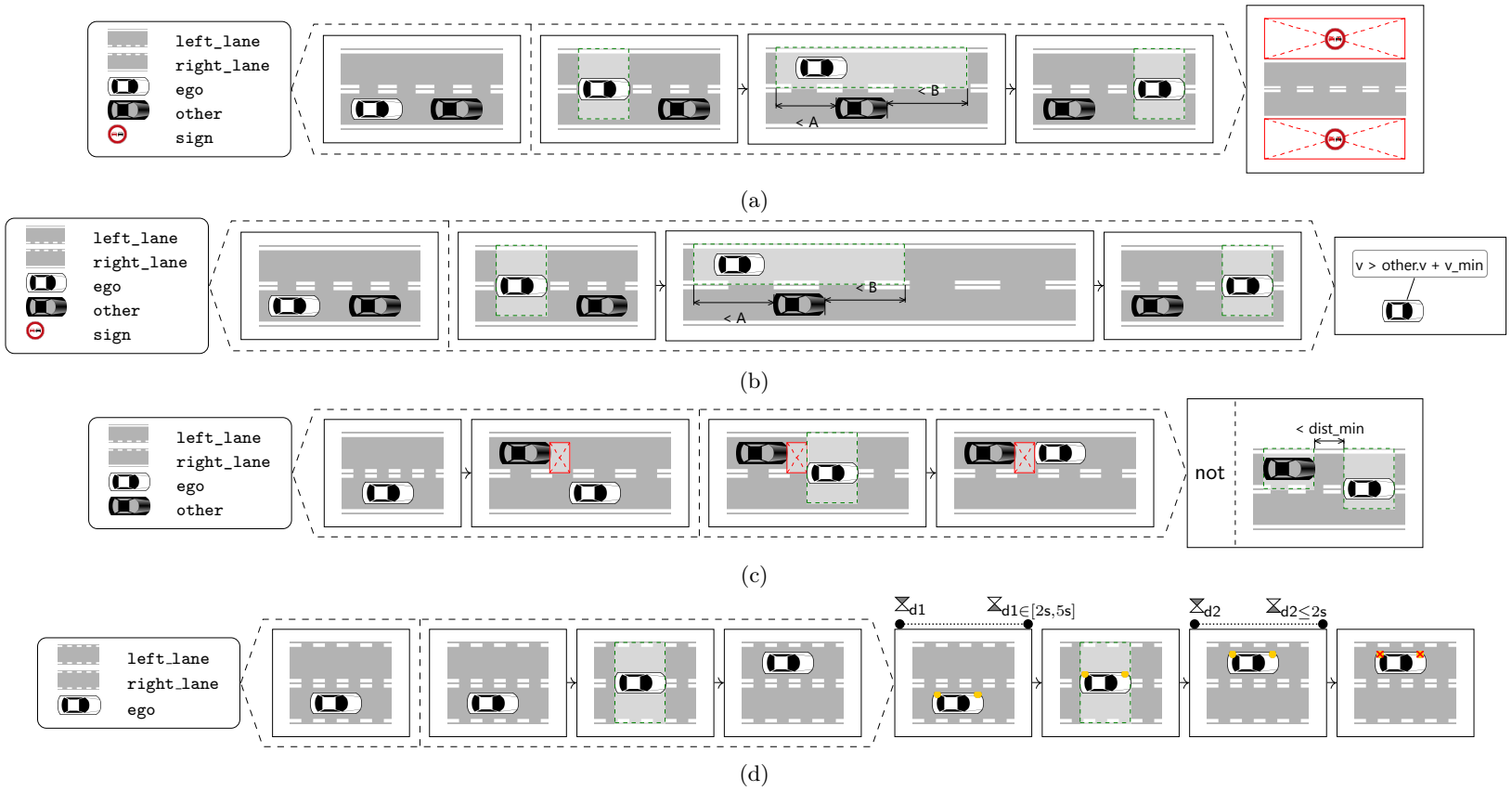


Figure B.3: TSCs for German overtaking rules ($A := ego.\overline{x} - ego.\underline{x}$, $B := A + dist\_min$)

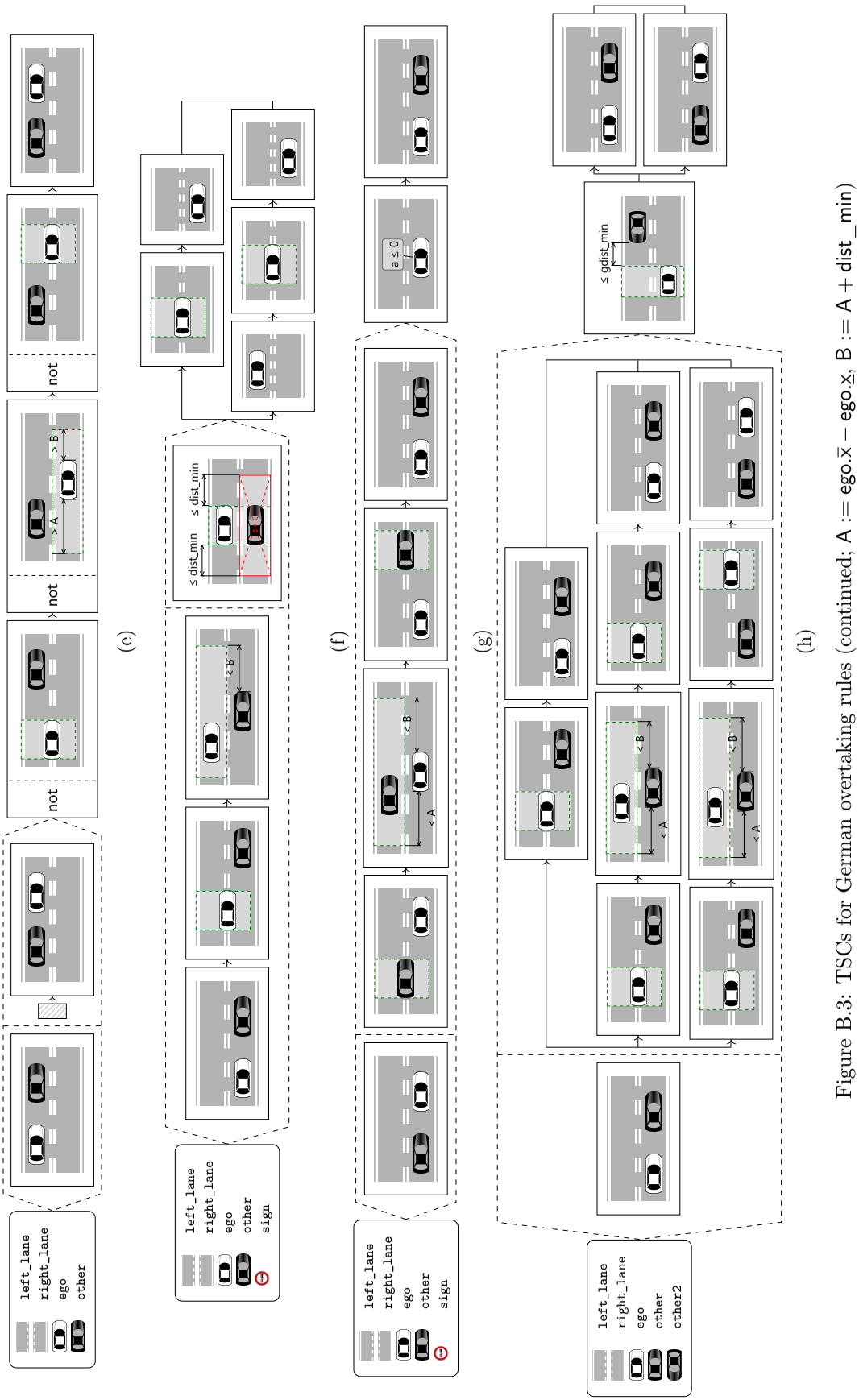
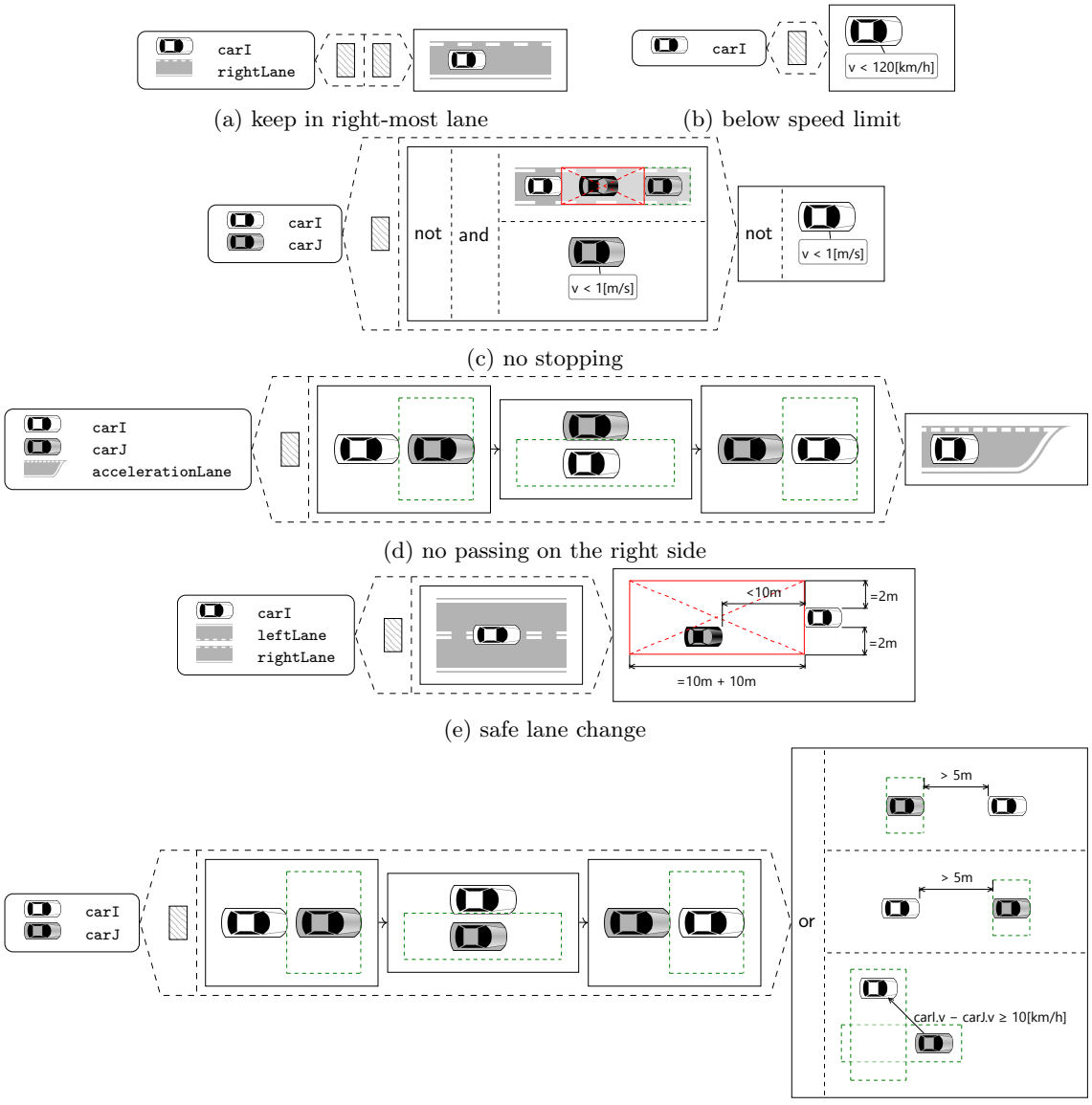


Figure B.3: TSCs for German overtaking rules (continued; $A := ego.\bar{x} - ego.\underline{x}$, $B := A + dist\_min$)

B.3 TSCs for Section 8.3.2 – Various Traffic Rules



(a) keep in right-most lane

(b) below speed limit

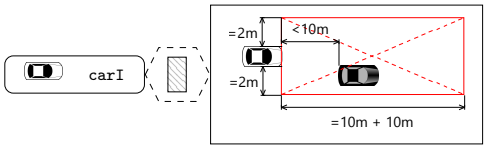
(c) no stopping

(d) no passing on the right side

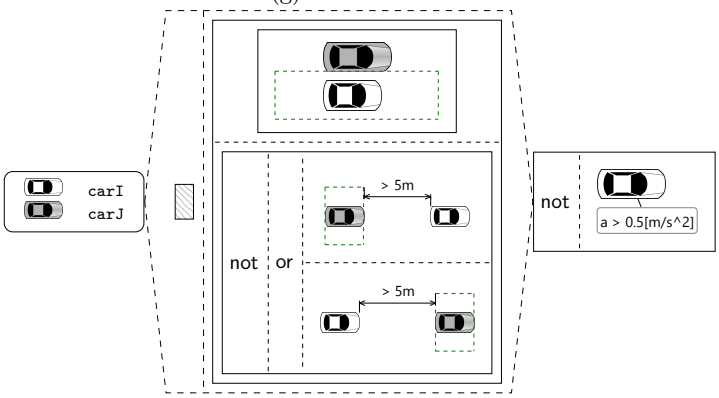
(e) safe lane change

(f) speed advantage for overtaking

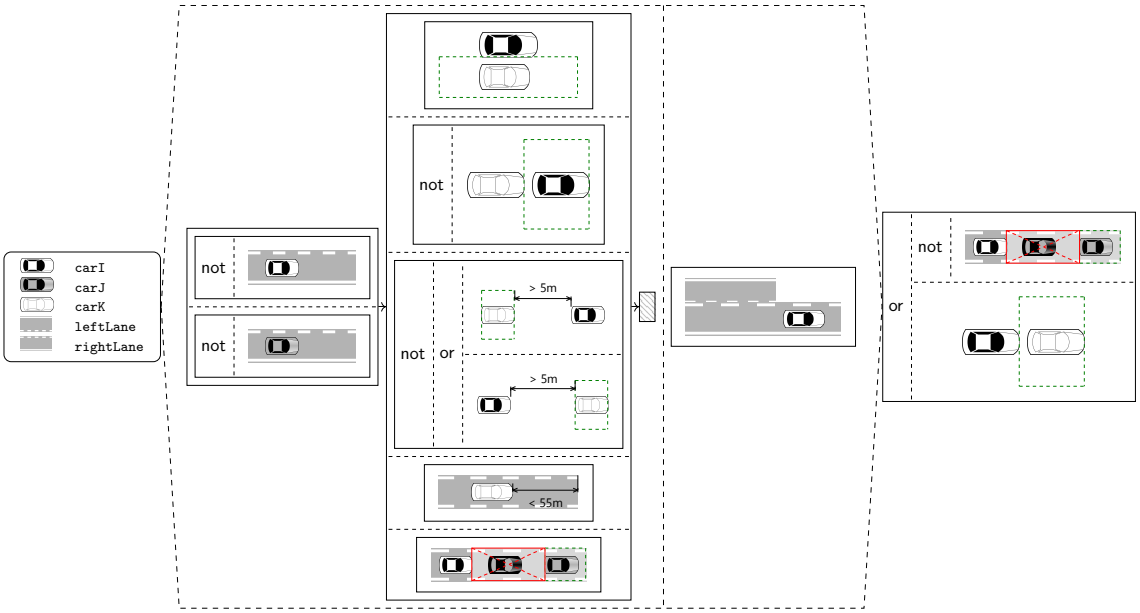
Figure B.4: Traffic rules



(g) safe distance



(h) being overtaken



(i) zipper merge

Figure B.4: Traffic rules (continued)

B.4 TSCs for Section 8.3.3 – ALKS

B.4.1 Requirement 5.2.1.

"The activated system shall keep the vehicle inside its lane of travel and ensure that the vehicle does not cross any lane marking (outer edge of the front tyre to outer edge of the lane marking). The system shall aim to keep the vehicle in a stable lateral position inside the lane of travel to avoid confusing other road users." [116]

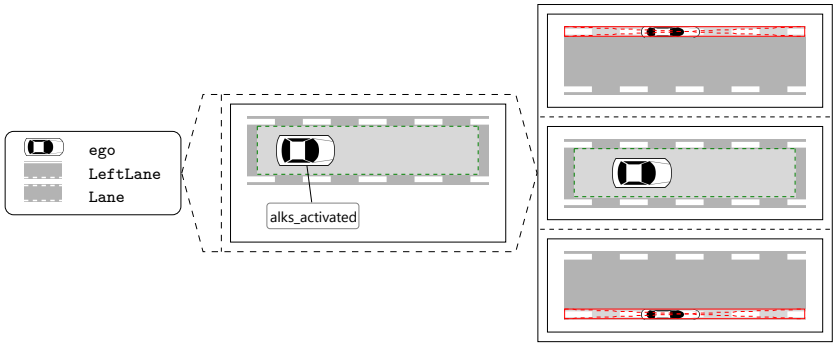


Figure B.5: Representation of the requirement 5.2.1. as TSC

B.4.2 Requirement 5.2.2.

"The activated system shall detect a vehicle driving beside as defined in paragraph 7.1.2. and, if necessary, adjust the speed and/or the lateral position of the vehicle within its lane as appropriate." [116]

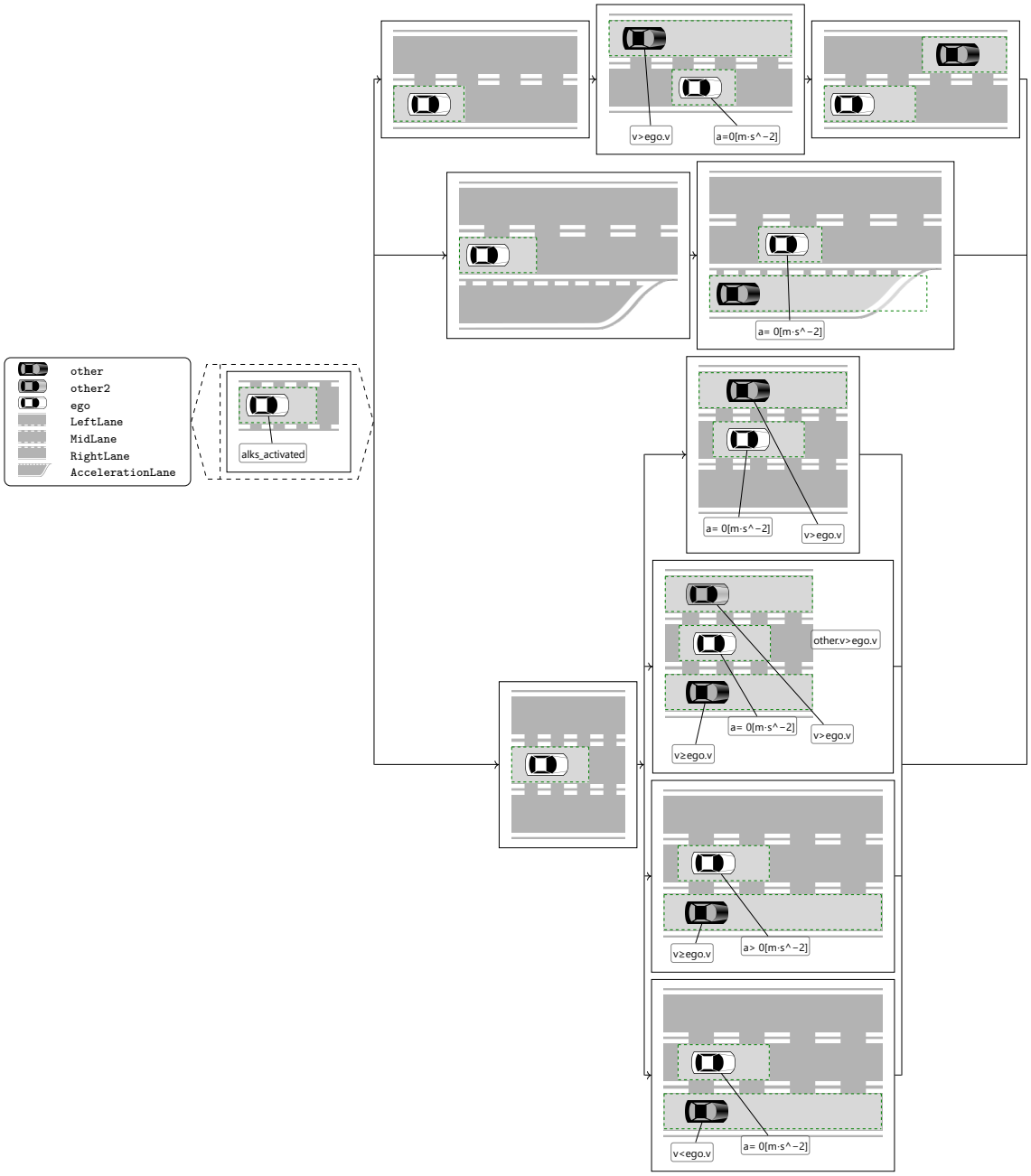


Figure B.6: Representation of the requirement 5.2.2 as TSC

B.4.3 Requirement 5.2.3.1.

"The maximum speed up to which the system is permitted to operate is 60 km/h." [116]

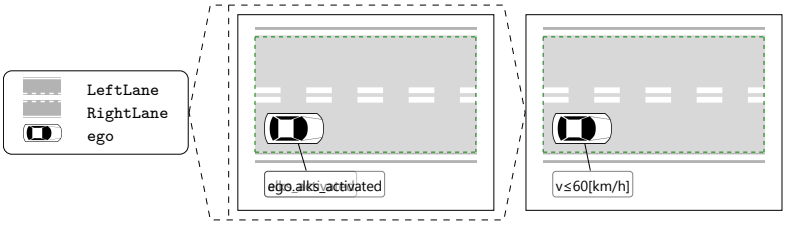


Figure B.7: Representation of the requirement 5.2.3.1. as TSC

B.4.4 Requirement 5.2.3.3.

"The activated system shall detect the distance to the next vehicle in front as defined in paragraph 7.1.1. and shall adapt the vehicle speed in order to avoid collision. [...]" [116]

The minimum following distance is speed-dependent and taken from a lookup-table provided in the regulations.

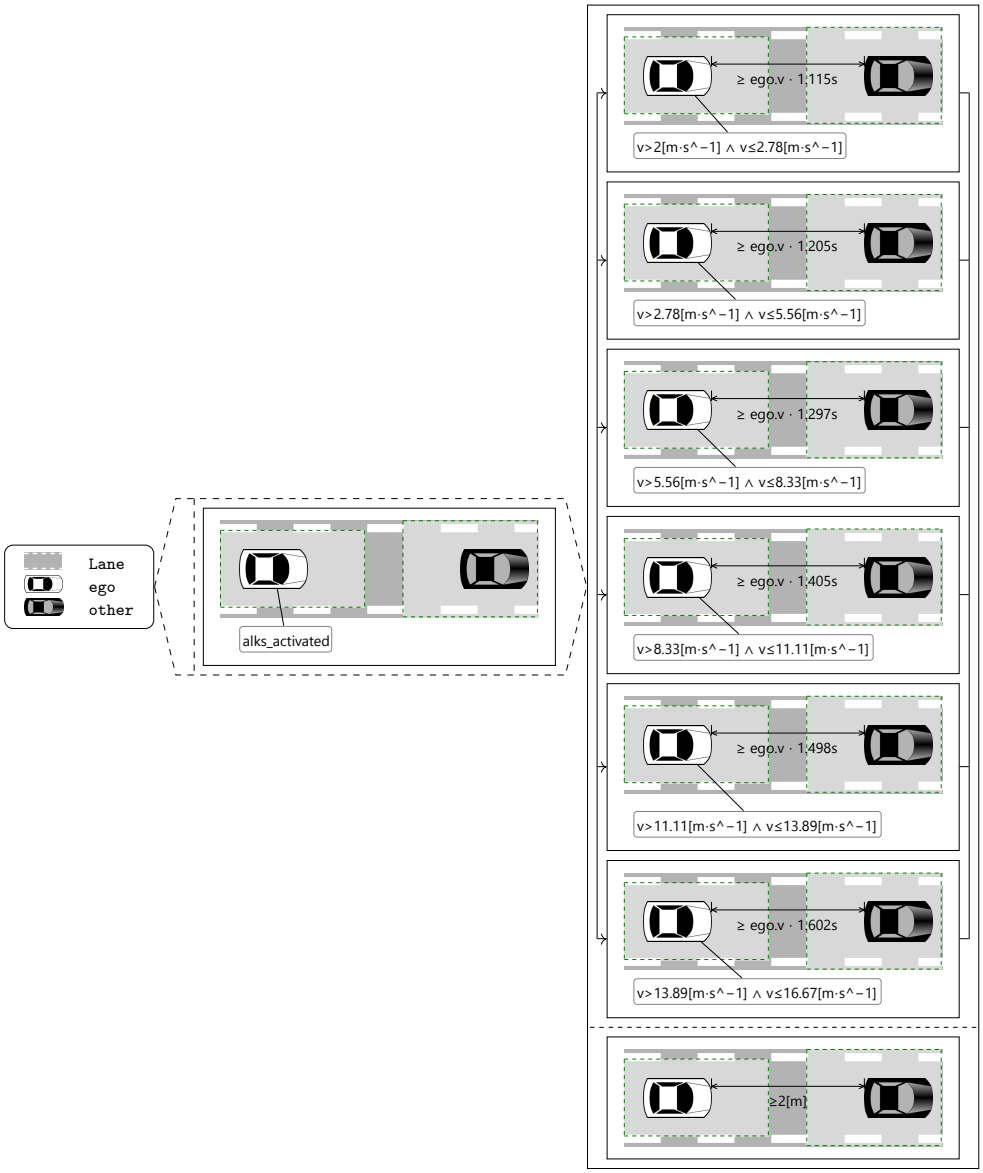


Figure B.8: Representation of the requirement 5.2.3.3. as TSC

B.4.5 Requirement 5.2.4.

"The activated system shall be able to bring the vehicle to a complete stop behind a stationary vehicle, a stationary road user or a blocked lane of travel to avoid a collision. This shall be ensured up to the maximum operational speed of the system." [116]

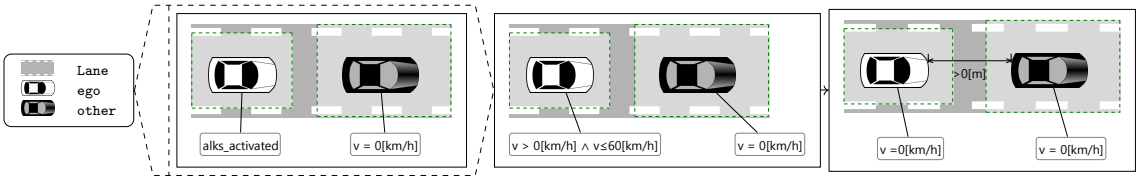


Figure B.9: Representation of the requirement 5.2.4. as TSC

B.4.6 Requirement 5.2.5.

"The activated system shall detect the risk of collision in particular with another road user ahead or beside the vehicle, due to a decelerating lead vehicle, a cutting in vehicle or a suddenly appearing obstacle and shall automatically perform appropriate manoeuvres to minimize risks to safety of the vehicle occupants and other road users." [116]

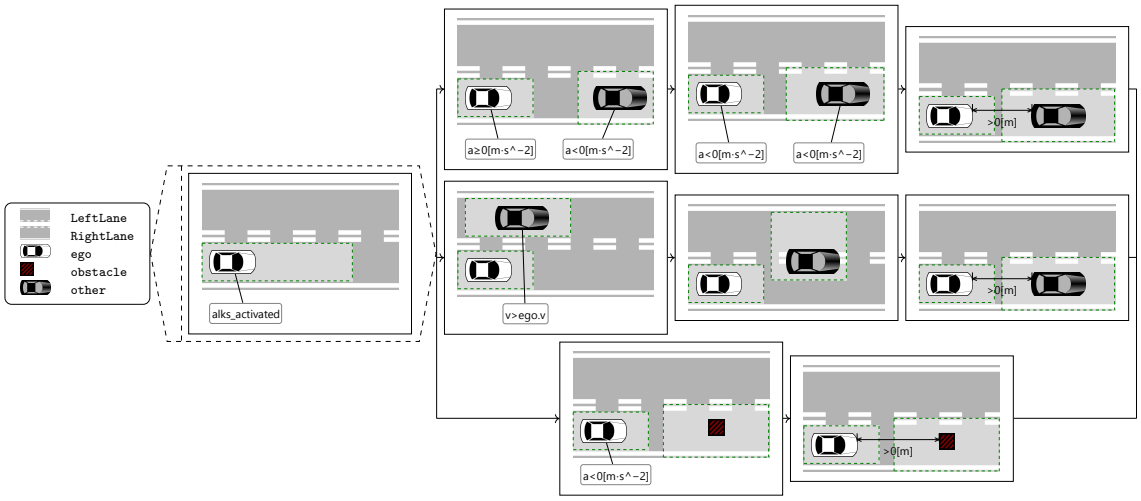


Figure B.10: Representation of the requirement 5.2.5. as TSC

B.4.7 Requirement 5.2.5.1.

"The activated system shall avoid a collision with a leading vehicle which decelerates up to its full braking performance provided that there was no undercut of the minimum following distance the ALKS vehicle would adjust to a leading vehicle at the present speed due to a cut in manoeuvre of this lead vehicle." [116]

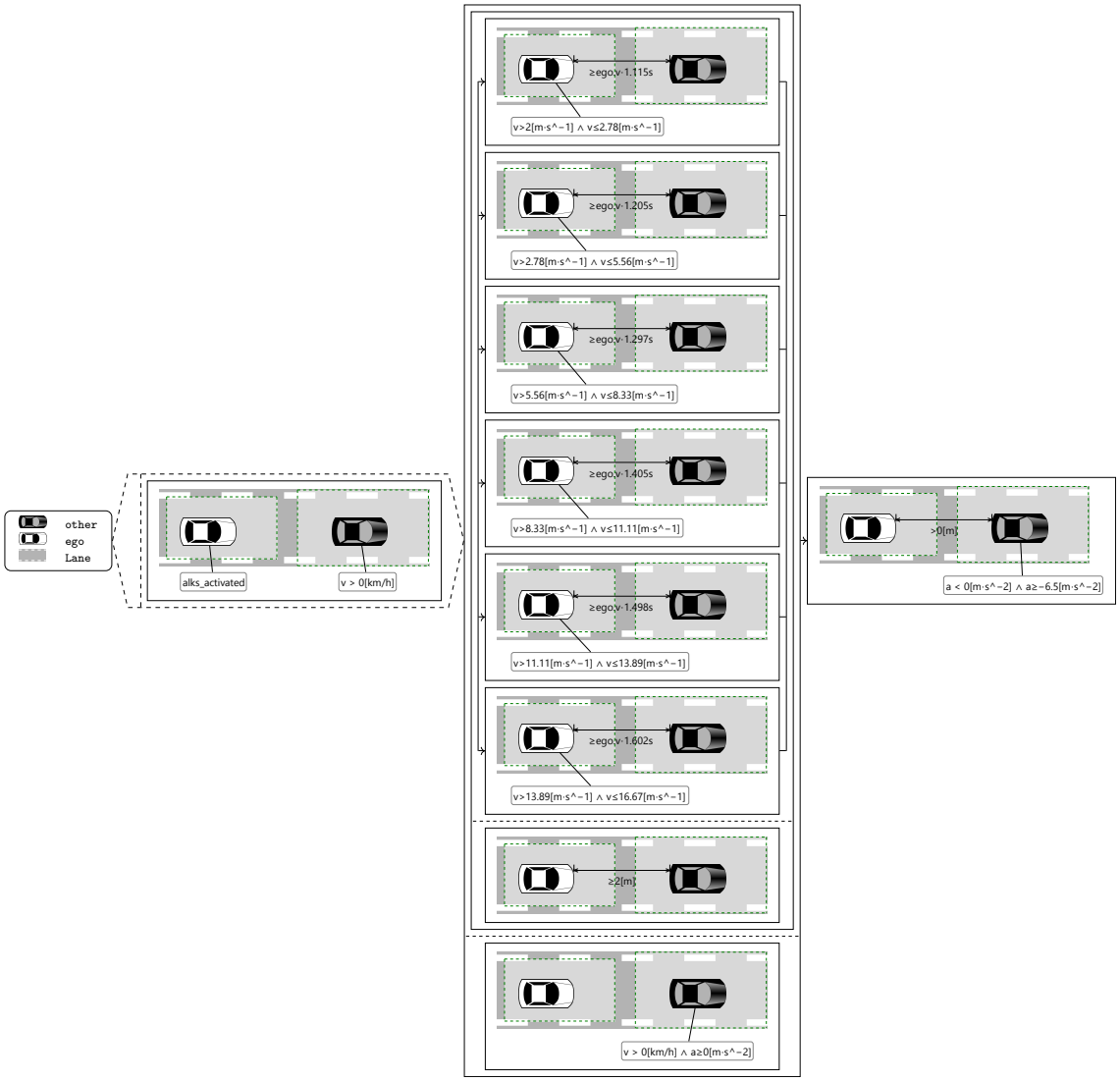


Figure B.11: Representation of the requirement 5.2.5.1. as TSC

B.4.8 Requirement 5.2.5.2.

"The activated system shall avoid a collision with a cutting in vehicle,

- *(a) Provided the cutting in vehicle maintains its longitudinal speed which is lower than the longitudinal speed of the ALKS vehicle and*
- *(b) Provided that the lateral movement of the cutting in vehicle has been visible for a time of at least 0.72 seconds before the reference point for TTCLaneIntrusion is reached,*
- *(c) When the distance between the vehicle's front and the cutting in vehicle's rear corresponds to a TTC calculated by the following equation:*

[. . .]" [116] The detailed description for calculating the distance is omitted here for the sake of brevity.

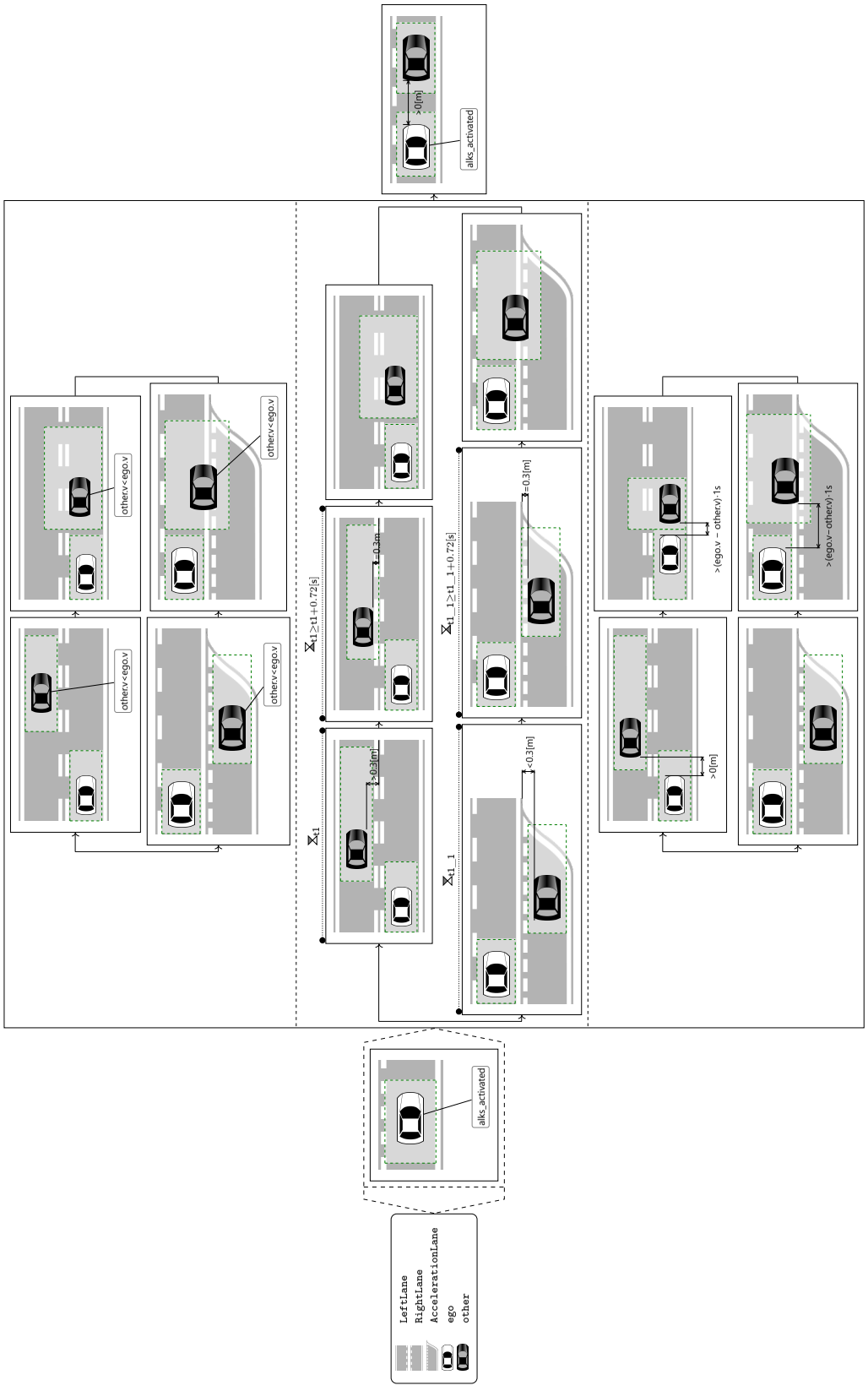


Figure B.12: Representation of the requirement 5.2.5.2. as TSC

B.4.9 Requirement 5.2.5.3.

"The activated system shall avoid a collision with an unobstructed crossing pedestrian in front of the vehicle. In a scenario with an unobstructed pedestrian crossing with a lateral speed component of not more than 5 km/h where the anticipated impact point is displaced by not more than 0.2 m compared to the vehicle longitudinal center plane, the activated ALKS shall avoid a collision up to the maximum operational speed of the system." [116]

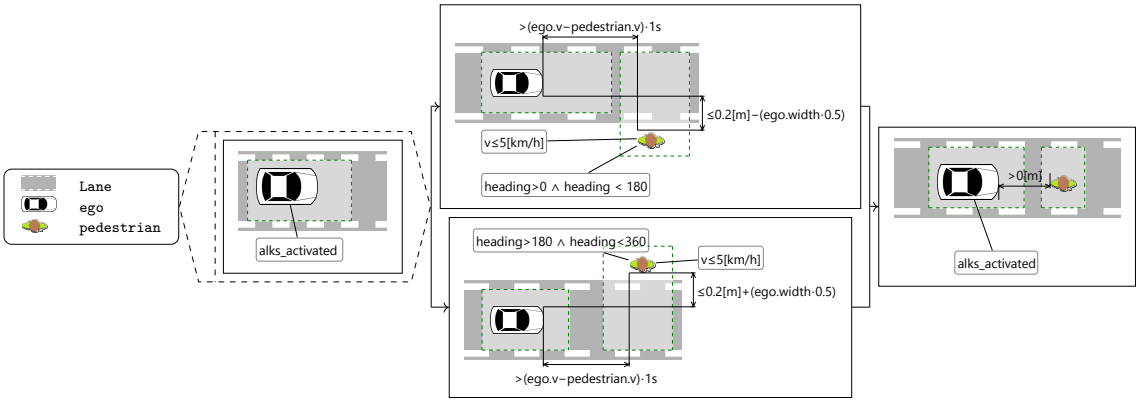


Figure B.13: Representation of the requirement 5.2.5.3. as TSC

B.4.10 Requirement 5.3.1.

"An Emergency Manoeuvre shall be carried out in case of an imminent collision risk." [116]

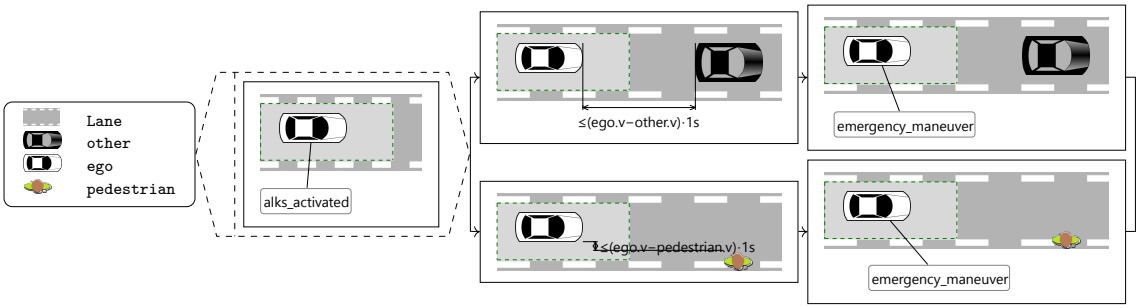


Figure B.14: Representation of the requirement 5.3.1. as TSC

B.4.11 Requirement 5.3.1.1.

"Any longitudinal deceleration demand of more than 5.0 m/s^2 of the system shall be considered to be an EM." [116]

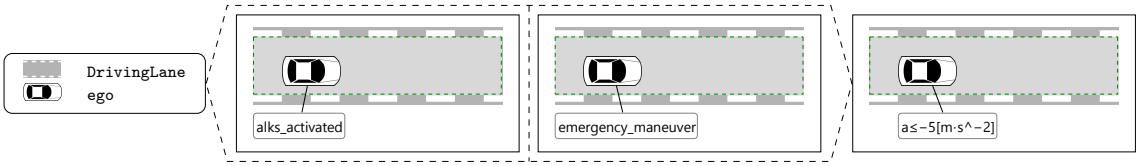


Figure B.15: Representation of the requirement 5.3.1.1. as TSC

B.4.12 Requirement 5.3.2.

"This manoeuvre shall decelerate the vehicle up to its full braking performance if necessary and/or may perform an automatic evasive manoeuvre, when appropriate. If failures are affecting the braking or steering performance of the system, the manoeuvre shall be carried out with consideration for the remaining performance. During the evasive manoeuvre the ALKS vehicle shall not cross the lane marking (outer edge of the front tyre to outer edge of the lane marking)." [116]

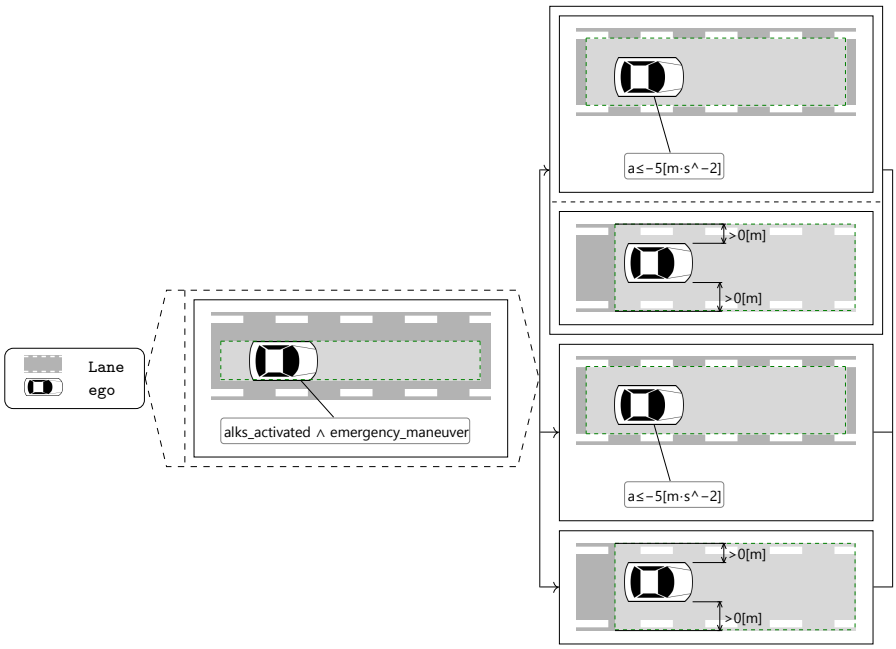


Figure B.16: Representation of the requirement 5.3.2. as TSC

B.4.13 Requirement 5.3.3.

"An emergency manoeuvre shall not be terminated, unless the imminent collision risk disappeared, or the driver deactivated the system." [116]

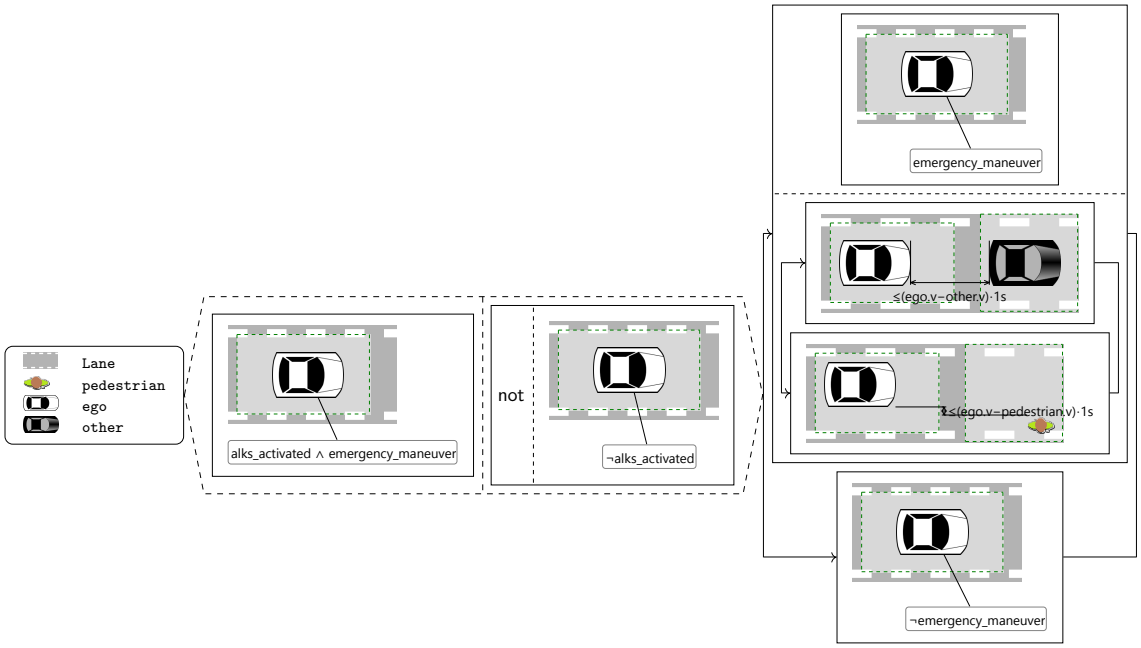


Figure B.17: Representation of the requirement 5.3.3. as TSC

B.4.14 Requirement 5.3.3.1.

"After an emergency manoeuvre is terminated the system shall continue to operate." [116]

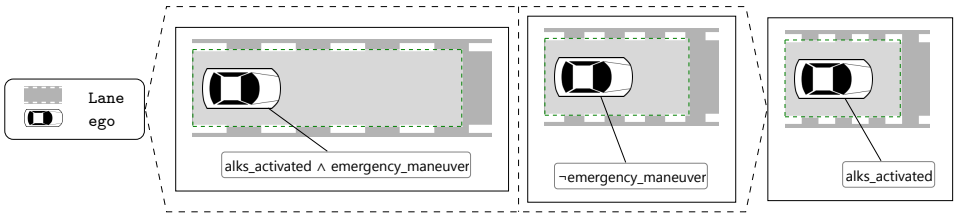


Figure B.18: Representation of the requirement 5.3.3.1. as TSC

B.4.15 Requirement 5.3.3.2.

"If the emergency manoeuvre results in the vehicle being at standstill, the signal to activate the hazard warning lights shall be generated. If the vehicle automatically drives off again, the signal to deactivate the hazard warning lights shall be generated automatically." [116]

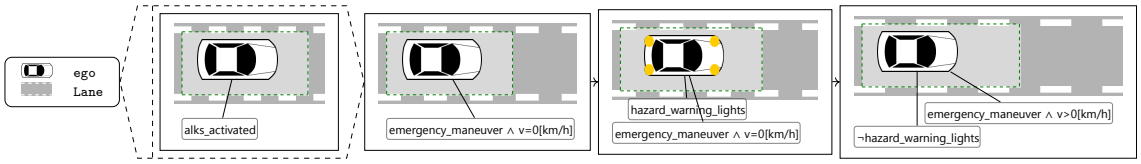


Figure B.19: Representation of the requirement 5.3.3.2. as TSC

B.4.16 Requirement 5.4.2.2.

"In case of an unplanned event, a transition demand shall be given upon detection." [116]

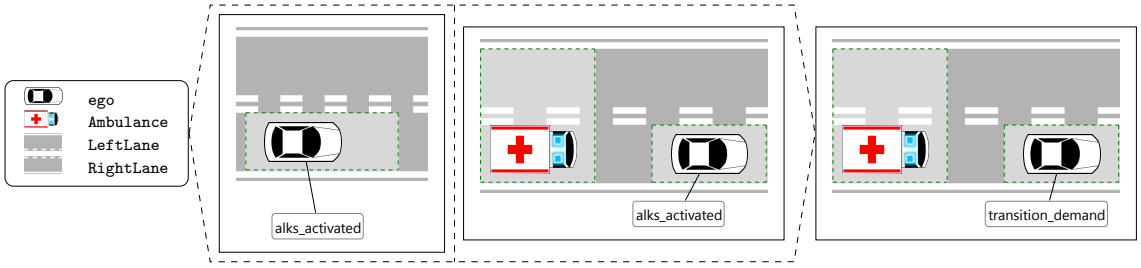


Figure B.20: Representation of the requirement 5.4.2.2. as TSC

B.4.17 Requirement 5.4.3.1.

"Once in standstill the vehicle may remain in this condition and shall generate the signal to activate the hazard warning lights within 5 s." [116]

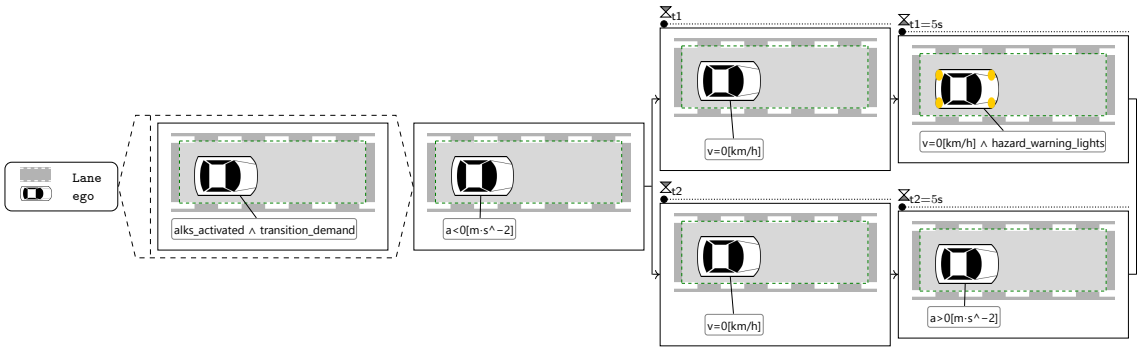


Figure B.21: Representation of the requirement 5.4.3.1. as TSC

B.4.18 Requirement 5.4.4.

"A transition demand shall only be terminated once the system is deactivated or a minimum risk manoeuvre has started." [116]

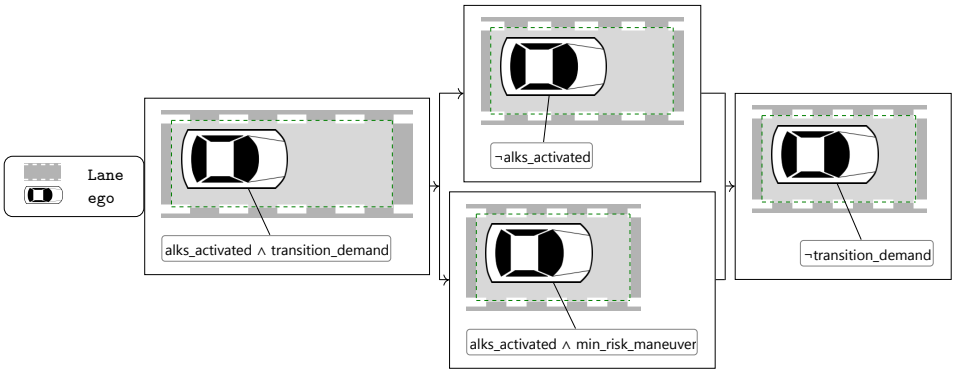


Figure B.22: Representation of the requirement 5.4.4. as TSC

B.4.19 Requirement 5.4.4.1.

"In case the driver is not responding to a transition demand by deactivating the system (either as described in paragraph 6.2.4. or 6.2.5.), a minimum risk manoeuvre shall be started, earliest 10 s after the start of the transition demand." [116]

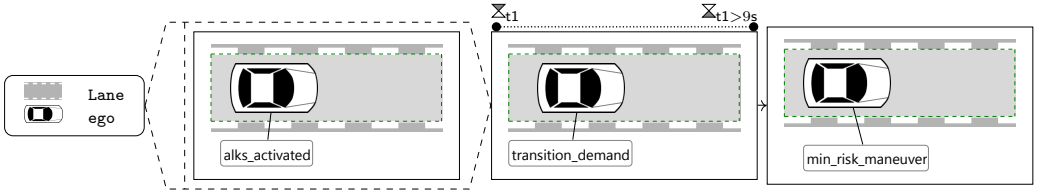


Figure B.23: Representation of the requirement 5.4.4.1. as TSC

B.4.20 Requirement 5.5.1.

"During the minimum risk manoeuvre the vehicle shall be slowed down inside the lane or, in case the lane markings are not visible, remain on an appropriate trajectory taking into account surrounding traffic and road infrastructure, with an aim of achieving a deceleration demand not greater than $4.0\mathrm{m/s}^2$*.*

Additionally, the signal to activate the hazard warning lights shall be generated with the start of the minimum risk manoeuvre." [116]

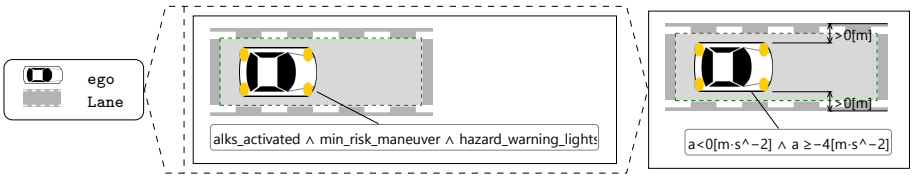


Figure B.24: Representation of the requirement 5.5.1. als TSC

B.4.21 Requirement 5.5.2.

"The minimum risk manoeuvre shall bring the vehicle to standstill unless the system is deactivated by the driver during the manoeuvre." [116]

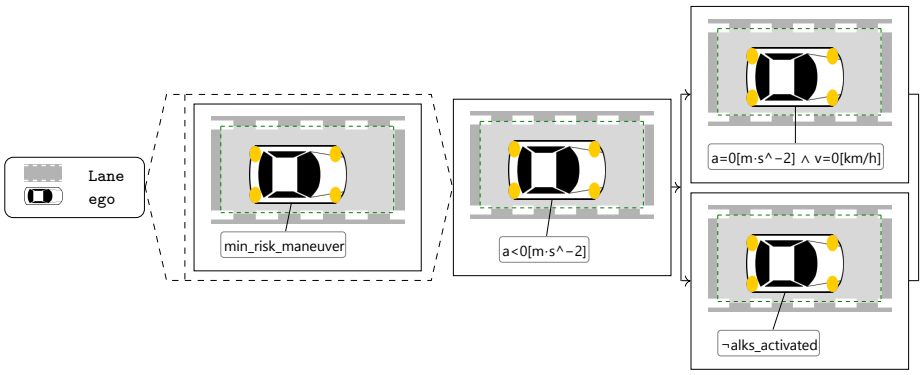


Figure B.25: Representation of the requirement 5.5.2. as TSC

B.4.22 Requirement 5.5.4.

"A minimum risk manoeuvre shall only be terminated once the system is deactivated or the system has brought the vehicle to a standstill." [116]

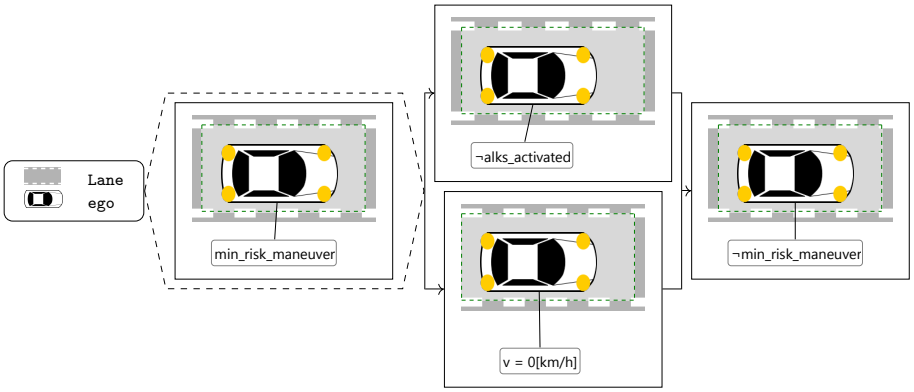


Figure B.26: Representation of the requirement 5.5.4. as TSC

B.4.23 Requirement 5.5.5.

"The system shall be deactivated at the end of any minimum risk manoeuvre. The hazard warning lights shall remain activated unless deactivated manually and the vehicle shall not move away after standstill without manual input." [116]

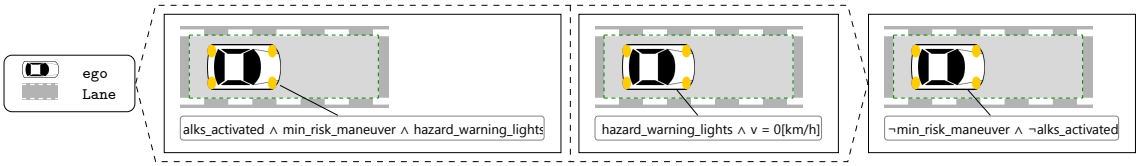


Figure B.27: Representation of the requirement 5.5.5. as TSC

C Own Publications

This is a list of the author's own publications that contributed to this thesis.

- J. S. Becker, "Analyzing consistency of formal requirements", in *18th International Workshop on Automated Verification of Critical Systems*, 2018
- J. S. Becker, "Partial consistency for requirement engineering with traffic sequence charts", in *Automotive Software Engineering (ASE2020)*, 2020
- J. S. Becker et al., "Simulation of abstract scenarios: Towards automated tooling in criticality analysis", in Feb. 2022, pp. 42–51. DOI: 10.5281/zenodo.5907154
- J. S. Becker, "A consistency analysis method for traffic sequence charts", *VEHITS'24 Doctoral Consortium*, 2024. [Online]. Available: https://arxiv.org/abs/2409.03774
- J. S. Becker, "Safe linear encoding of vehicle dynamics for the instantiation of abstract scenarios", in *Formal Methods for Industrial Critical Systems*, ser. LNCS, vol. 14952, Springer, 2024
- J. S. Becker and C. Neurohr, "Correct-by-construction instantiation of abstract scenarios", *International Journal on Software Tools for Technology Transfer*, 2025

Bibliography

[1] B. K. Aichernig, K. Hörmaier, F. Lorber, D. Ničković, and S. Tiran, "Require, test and trace IT", in *Formal Methods for Industrial Critical Systems - 20th International Workshop, FMICS 2015, Proceedings*, ser. LNCS, vol. 9128, Springer, 2015, pp. 113–127.

[2] E. Althaus et al., "Verification of linear hybrid systems with large discrete state spaces using counterexample-guided abstraction refinement", *Science of Computer Programming*, vol. 148, pp. 123–160, 2017.

[3] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems", in *Hybrid systems*, Springer, 1992, pp. 209–229.

[4] ASAM, *ASAM OpenXOntology*, online, version 1.0.0, accessed on 2024-01-25, 2022. [Online]. Available: `https://www.asam.net/standards/asam-openxontology/`

[5] S. Babisch, C. Neurohr, L. Westhofen, S. Schoenawa, H. Liers, et al., "Leveraging the GIDAS database for the criticality analysis of automated driving systems", *Journal of Advanced Transportation*, 2023.

[6] J. Bach, S. Otten, and E. Sax, "Model based scenario specification for development and test of automated driving functions", in *2016 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2016, pp. 1149–1155. DOI: `10.1109/ivs.2016.7535534`

[7] J. Barnat, P. Bauch, N. Beneš, L. Brim, J. Beran, and T. Kratochvíla, "Analysing sanity of requirements for avionics systems", *Formal Aspects of Computing*, vol. 28, no. 1, pp. 45–63, Mar. 2016, ISSN: 1433-299X. DOI: `10.1007/s00165-015-0348-9`

[8] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability modulo theories", *Handbook of satisfiability*, vol. 185, pp. 825–885, 2009.

[9] J. S. Becker, "A consistency analysis method for traffic sequence charts", *VEHITS'24 Doctoral Consortium*, 2024. [Online]. Available: `https://arxiv.org/abs/2409.03774`

[10] J. S. Becker, "Analyzing consistency of formal requirements", in *18th International Workshop on Automated Verification of Critical Systems*, 2018.

[11] J. S. Becker, "Partial consistency for requirement engineering with traffic sequence charts", in *Automotive Software Engineering (ASE2020)*, 2020.

[12] J. S. Becker, "Safe linear encoding of vehicle dynamics for the instantiation of abstract scenarios", in *Formal Methods for Industrial Critical Systems*, ser. LNCS, vol. 14952, Springer, 2024.

[13] J. S. Becker, "Virtual integration for pattern-based contracts with the Kind2 model checker", in *Formal Methods for Industrial Critical Systems*, F. Howar and J. Barnat, Eds., Cham: Springer International Publishing, 2018, pp. 131–146, ISBN: 978-3-030-00244-2.

[14] J. S. Becker and C. Neurohr, "Correct-by-construction instantiation of abstract scenarios", *International Journal on Software Tools for Technology Transfer*, 2025.

[15] J. S. Becker et al., "Simulation of abstract scenarios: Towards automated tooling in criticality analysis", in Feb. 2022, pp. 42–51. DOI: 10.5281/zenodo.5907154

[16] T. Bienmüller, T. Teige, A. Eggers, and M. Stasch, "Modeling requirements for quantitative consistency analysis and automatic test case generation", in *Workshop on Formal and Model-Driven Techniques for Developing Trustworthy Systems*, ser. Computing Science Technical Report Series, Newcastle University, vol. CS-TR-1503, 2016.

[17] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs", *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, May 2012, ISSN: 0022-0000. DOI: 10.1016/j.jcss.2011.08.007

[18] P. Borchers et al., "TSC2CARLA: An abstract scenario-based verification toolchain for automated driving systems", *Science of Computer Programming*, vol. 242, p. 103 256, 2025, ISSN: 0167-6423. DOI: https://doi.org/10.1016/j.scico.2024.103256 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642324001795

[19] A. Bouajjani, Y. Lakhnech, and R. Robbana, "From duration calculus to linear hybrid automata", in *LNCS*, vol. 939, Jul. 1995, pp. 196–210. DOI: 10.1007/3-540-60045-0_51

[20] M. Butz et al., "SOCA: Domain analysis for highly automated driving systems", in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294438

[21] C. Carlan, D. PetriSor, B. Gallina, and H. Schoenhaar, "Checkable safety cases: Enabling automated consistency checks between safety work products", in *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, Oct. 2020, pp. 295–302. DOI: 10.1109/issrew51248.2020.00088

[22] Z. Chaochen, M. R. Hansen, and P. Sestoft, "Decidability and undecidability results for duration calculus", in *STACS 93, 10th Annual Symposium on Theoretical Aspects of Computer Science, Würzburg, Germany, February 25-27, 1993, Proceedings*, 1993, pp. 58–68. DOI: 10.1007/3-540-56503-5_8 [Online]. Available: https://doi.org/10.1007/3-540-56503-5_8

[23] Z. Chaochen, C. A. R. Hoare, and A. Ravn, "A calculus of durations", *Information Processing Letters*, vol. 40, pp. 269–276, Dec. 1991.

[24] F. Chen and J. Terken, "Design process", in *Automotive Interaction Design: From Theory to Practice*. Singapore: Springer Nature Singapore, 2023, pp. 165–179, ISBN: 978-981-19-3448-3. DOI: 10.1007/978-981-19-3448-3_10 [Online]. Available: https://doi.org/10.1007/978-981-19-3448-3_10

[25] X. Chen and S. Sankaranarayanan, "Decomposed reachability analysis for nonlinear systems", in *Real-Time Systems Symposium (RTSS), 2016 IEEE*, IEEE, 2016, pp. 13–24.

[26] J.-w. Choi, R. Curry, and G. Elkaim, "Path planning based on Bézier curve for autonomous ground vehicles", in *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, IEEE, 2008, pp. 158–166.

[27] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded model checking using satisfiability solving", *Formal methods in system design*, vol. 19, pp. 7–34, 2001.

[28] D. R. Cok, *The SMT-LIBv2 language and tools: A tutorial*, 1.2.1, Nov. 2013. [Online]. Available: `http://smtlib.github.io/jSMTLIB/SMTLIBTutorial.pdf`

[29] B. Combemale and M. Wimmer, "Towards a model-based DevOps for cyber-physical systems", in *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: Second International Workshop, DEVOPS 2019, Château de Villebrumier, France, May 6–8, 2019, Revised Selected Papers 2*, Springer, 2020, pp. 84–94.

[30] K. Czarnecki, "Operational world model ontology for automated driving systems–part 1: Road structure", *Waterloo Intelligent Systems Engineering Lab (WISE) Report, University of Waterloo*, 2018.

[31] K. Czarnecki, "Operational world model ontology for automated driving systems–part 2: Road users, animals, other obstacles, and environmental conditions", *Waterloo Intelligent Systems Engineering Lab (WISE) Report, University of Waterloo*, 2018.

[32] W. Damm, P. Heidl, J. Oehlerking, A. Rakow, and B. Wirtz, "Traffic sequence charts for platooning", 2019.

[33] W. Damm, S. Kemper, E. Möhlmann, T. Peikenkamp, and A. Rakow, "Using traffic sequence charts for the development of HAVs", in *ERTS 2018*, 2018.

[34] W. Damm and R. Galbas, "Exploiting learning and scenario-based specification languages for the verification and validation of highly automated driving", in *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, 2018, pp. 39–46.

[35] W. Damm, M. Horbach, and V. Sofronie-Stokkermans, "Decidability of verification of safety properties of spatial families of linear hybrid automata", in *International Symposium on Frontiers of Combining Systems*, Springer, 2015, pp. 186–202.

[36] W. Damm, S. Kemper, E. Möhlmann, T. Peikenkamp, and A. Rakow, "Traffic sequence charts: A visual language for capturing traffic scenarios", in *Embedded Real Time Software and Systems - ERTS2018*, Feb. 2018.

[37] W. Damm, S. Kemper, E. Möhlmann, T. Peikenkamp, and A. Rakow, "Traffic sequence charts: From visualization to semantics", SFB/TR 14 AVACS, Reports of SFB/TR 14 AVACS 117, Oct. 2017.

[38] W. Damm, E. Möhlmann, T. Peikenkamp, and A. Rakow, "A formal semantics for traffic sequence charts", in *Principles of Modeling: Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday*. Cham: Springer International Publishing, 2018, pp. 182–205, ISBN: 978-3-319-95246-8. DOI: `10.1007/978-3-319-95246-8\_11`

[39] W. Damm, E. Möhlmann, and A. Rakow, "A scenario discovery process based on traffic sequence charts", in *Validation & Verification of Automated Systems – Results of the ENABLE-S3 Project*, 2019.

[40] W. Damm, E. Möhlmann, and A. Rakow, "Traffic sequence charts for the ENABLE-S_3 test architecture", in *Validation & Verification of Automated Systems – Results of the ENABLE-S3 Project*, 2019.

[41] W. Damm, G. Pinto, and S. Ratschan, "Guaranteed termination in the verification of LTL properties of non-linear robust discrete time hybrid systems", *Int. J. Found. Comput. Sci.*, vol. 18, no. 1, pp. 63–86, 2007. DOI: 10.1142/S0129054107004577 [Online]. Available: https://doi.org/10.1142/S0129054107004577

[42] L. De Moura and N. Bjørner, "Satisfiability modulo theories: An appetizer", in *Brazilian Symposium on Formal Methods*, Springer, 2009, pp. 23–36.

[43] A. Dokhanchi, B. Hoxha, and G. Fainekos, "Formal requirement debugging for testing and verification of cyber-physical systems", *ACM Transactions on Embedded Computing Systems*, vol. 17, no. 2, pp. 1–26, Dec. 2017, ISSN: 1558-3465. DOI: 10.1145/3147451

[44] L. Doyen, G. Frehse, G. J. Pappas, and A. Platzer, "Verification of hybrid systems", *Handbook of Model Checking*, pp. 1047–1110, 2018.

[45] A. Eggers, M. Stasch, T. Teige, T. Bienmüller, and U. Brockmeyer, "Constraint systems from traffic scenarios for the validation of autonomous driving", in *Third International Workshop on Satisfiability Checking and Symbolic Computation, Part of FLOC 2018*, 2018.

[46] A. Eggers, N. Ramdani, N. Nedialkov, and M. Fränzle, "Improving SAT modulo ODE for hybrid systems analysis by combining different enclosure methods", in *International Conference on Software Engineering and Formal Methods*, Springer, 2011, pp. 172–187.

[47] A. Egyed, "Automatically detecting and tracking inconsistencies in software design models", *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 188–204, Mar. 2011, ISSN: 0098-5589. DOI: 10.1109/TSE.2010.38

[48] C. Ellen, S. Sieverding, and H. Hungar, "Detecting consistencies and inconsistencies of pattern-based functional requirements", in *Formal Methods for Industrial Critical Systems: 19th International Conference, FMICS 2014*, ser. Lecture Notes in Computer Science, vol. 8718, Springer, 2014, pp. 155–169. DOI: 10.1007/978-3-319-10702-8_11

[49] K. Esterle, "Formalizing and modeling traffic rules within interactive behavior planning", Ph.D. dissertation, Universität München, 2021.

[50] K. Esterle, V. Aravantinos, and A. Knoll, "From specifications to behavior: Maneuver verification in a semantic state space", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 2140–2147.

[51] K. Esterle, L. Gressenbuch, and A. Knoll, "Formalizing traffic rules for machine interpretability", in *2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS)*, IEEE, 2020, pp. 1–7.

[52] K. Esterle, T. Kessler, and A. Knoll, "Optimal behavior planning for autonomous driving: A generic mixed-integer formulation", in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1914–1921. DOI: 10.1109/IV47402.2020.9304743

[53] K. Feichtinger, K. Kegel, R. Pascual, U. Aßmann, B. Beckert, and R. Reussner, "Towards formalizing and relating different notions of consistency in cyber-physical systems engineering", in *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, IEEE, 2024, pp. 915–919.

[54] P. Feiler, L. Wrage, and J. Hansson, "System architecture virtual integration: A case study", in *Embedded Real-time Software and Systems Conference*, 2010.

[55] J. Ferrante and C. Rackoff, "A decision procedure for the first order theory of real addition with order", *SIAM Journal on Computing*, vol. 4, no. 1, pp. 69–76, 1975.

[56] P. Filipovikj, G. Rodriguez-Navas, M. Nyberg, and C. Seceleanu, "SMT-based consistency analysis of industrial systems requirements", in *Proceedings of the Symposium on Applied Computing*, ACM, 2017, pp. 1272–1279.

[57] M. Fränzle, "Model-checking dense-time duration calculus", *Formal Asp. Comput.*, vol. 16, no. 2, pp. 121–139, 2004. DOI: 10.1007/s00165-004-0032-y

[58] M. Fränzle, "Take it NP-easy: Bounded model construction for duration calculus", in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, Springer, 2002, pp. 245–264.

[59] M. Fränzle and M. R. Hansen, "A robust interpretation of duration calculus", in *Theoretical Aspects of Computing: ICTAC 2005, Second International Colloquium*, ser. Lecture Notes in Computer Science, Springer, vol. 3722, 2005, pp. 257–271. DOI: 10.1007/11560647_17

[60] M. Fränzle and M. R. Hansen, "Deciding an interval logic with accumulated durations", in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, Springer, vol. 4424, 2007, pp. 201–215.

[61] M. Fränzle, M. R. Hansen, and H. Ody, "No need knowing numerous neighbours", in *Correct System Design*, ser. Lecture Notes in Computer Science, vol. 9360, Springer, 2015, pp. 152–171. DOI: 10.1007/978-3-319-23506-6_11

[62] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech", in *International workshop on hybrid systems: computation and control*, Springer, 2005, pp. 258–273.

[63] G. Frehse et al., "SpaceEx: Scalable verification of hybrid systems", in *International Conference on Computer Aided Verification*, Springer, 2011, pp. 379–395.

[64] D. J. Fremont et al., "Scenic: A language for scenario specification and data generation", *Machine Learning*, vol. 112, no. 10, pp. 3805–3849, 2023. DOI: 10.1007/s10994-021-06120-5

[65] L. Gonnord, N. Halbwachs, and P. Raymond, "From discrete duration calculus to symbolic automata", *Electronic Notes in Theoretical Computer Science*, vol. 153, no. 4, pp. 3–18, 2006.

[66] D. Grundt, A. Köhne, I. Saxena, R. Stemmer, B. Westphal, and E. Möhlmann, "Towards runtime monitoring of complex system requirements for autonomous driving functions", *arXiv preprint arXiv:2209.14032*, 2022.

[67] L. Han, H. Yashiro, H. T. N. Nejad, Q. H. Do, and S. Mita, "Bezier curve based path planning for autonomous vehicle in urban environment", in *2010 IEEE intelligent vehicles symposium*, IEEE, 2010, pp. 1036–1042.

[68] M. R. Hansen, "Model-checking discrete duration calculus", *Formal Aspects of Computing*, vol. 6, no. 1, pp. 826–845, 1994.

[69] J. F. Harper, "Defining continuity of real functions of real variables", *BSHM Bulletin: Journal of the British Society for the History of Mathematics*, vol. 31, no. 3, pp. 189–204, 2016.

[70] L. Hartjen, R. Philipp, F. Schuldt, B. Friedrich, and F. Howar, "Classification of driving maneuvers in urban traffic for parametrization of test scenarios", in *9. Tagung Automatisiertes Fahren*, 2019.

[71] C. L. Heitmeyer, R. D. Jeffords, and B. G. Labaw, "Automated consistency checking of requirements specifications", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 5, no. 3, pp. 231–261, 1996.

[72] T. A. Henzinger, "The theory of hybrid automata", in *Verification of Digital and Hybrid Systems*, Springer, 2000, pp. 265–292.

[73] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems", *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 110–122, 1997.

[74] T. A. Henzinger, B. Horowitz, and R. Majumdar, "Rectangular hybrid games", in *International Conference on Concurrency Theory*, Springer, 1999, pp. 320–335.

[75] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?", *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 94–124, 1998. DOI: 10.1006/jcss.1998.1581

[76] *ISO 26262:road vhicles—functional safety*, International Organization for Standardization, Nov. 2011.

[77] M. S. Jaffe, N. G. Leveson, M. P. E. Heimdahl, and B. E. Melhart, "Software requirements analysis for real-time process-control systems", *IEEE transactions on software engineering*, vol. 17, no. 3, pp. 241–258, 1991.

[78] T. Jéron, N. Markey, D. Mentré, R. Noguchi, and O. Sankur, "Incremental methods for checking real-time consistency", in *Formal Modeling and Analysis of Timed Systems*. Springer, 2020, pp. 249–264. DOI: 10.1007/978-3-030-57628-8_15

[79] M. Kamalrudin and S. Sidek, "A review on software requirements validation and consistency management", *International Journal of Software Engineering and Its Applications*, vol. 9, no. 10, pp. 39–58, Oct. 2015. DOI: 10.14257/ijseia.2015.9.10.05

[80] T. Kessler, K. Esterle, and A. Knoll, "Linear differential games for cooperative behavior planning of autonomous vehicles using mixed-integer programming", in *2020 59th IEEE Conference on Decision and Control (CDC)*, IEEE, 2020, pp. 4060–4066. DOI: 10.1109/CDC42340.2020.9304495

[81] M. Klischat and M. Althoff, "Synthesizing traffic scenarios from formal specifications for testing automated vehicles", in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2020, pp. 2065–2072.

[82] R. Könighofer, G. Hofferek, and R. Bloem, "Debugging formal specifications: A practical approach using model-based diagnosis and counterstrategies", *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 5–6, pp. 563–583, Dec. 2011. DOI: 10.1007/s10009-011-0221-y

[83] P. Koopman and F. Fratrik, "How many operational design domains, objects, and events?", in *Safeai@ aaai*, 2019.

[84] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation", *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016. [Online]. Available: http://www.jstor.org/stable/26167741

[85] I. Koren, F. Rinker, K. Meixner, J. Matevska, and J. Walter, "Challenges and opportunities of DevOps in cyber-physical production systems engineering", in *2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS)*, IEEE, 2023, pp. 1–6.

[86] B. Kramer, C. Neurohr, M. Büker, E. Böde, M. Fränzle, and W. Damm, "Identification and quantification of hazardous scenarios for automated driving", vol. 12297, pp. 163–178, 2020. DOI: 10.1007/978-3-030-58920-2_11

[87] N. Leveson, "Completeness in formal specification language design for process-control systems", in *Proceedings of the third workshop on Formal methods in software practice*, 2000, pp. 75–87.

[88] J. Ma, X. Che, Y. Li, and E. M.-K. Lai, "Traffic scenarios for automated vehicle testing: A review of description languages and systems", *Machines*, vol. 9, no. 12, p. 342, 2021.

[89] R. Mariani, "An overview of autonomous vehicles safety", in *2018 IEEE International Reliability Physics Symposium (IRPS)*, IEEE, 2018, 6A–1.

[90] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles", in *2018 IEEE Intelligent Vehicles Symposium (IV)*, (Changshu), IEEE, 2018, pp. 1821–1827. DOI: 10.1109/IVS.2018.8500406

[91] R. Meyer, J. Faber, J. Hoenicke, and A. Rybalchenko, "Model checking duration calculus: A practical approach", *Formal Aspects of Computing*, vol. 20, no. 4, pp. 481–505, 2008.

[92] A. Nadel, "Boosting minimal unsatisfiable core extraction", in *Formal methods in computer aided design*, IEEE, 2010, pp. 221–229.

[93] C. Neurohr, L. Westhofen, M. Butz, M. H. Bollmann, U. Eberle, and R. Galbas, "Criticality analysis for the verification and validation of automated vehicles", *IEEE Access*, vol. 9, pp. 18 016–18 041, 2021. DOI: 10.1109/ACCESS.2021.3053159

[94] C. Neurohr, L. Westhofen, T. Henning, T. de Graaff, E. Mohlmann, and E. Bode, "Fundamental considerations around scenario-based testing for automated driving", in *2020 IEEE Intelligent Vehicles Symposium (IV)*, (Las Vegas, NV, USA), IEEE, 2020, pp. 121–127. DOI: 10.1109/IV47402.2020.9304823

[95] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges", *AI communications*, vol. 29, no. 1, pp. 151–162, 2016.

[96] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Falsification of LTL safety properties in hybrid systems", *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 4, pp. 305–320, 2013.

[97] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Hybrid systems: From verification to falsification", in *International Conference on Computer Aided Verification*, Springer, 2007, pp. 463–476.

[98] A. Post, J. Hoenicke, and A. Podelski, "Rt-inconsistency: A new property for real-time requirements", in *Fundamental Approaches to Software Engineering - 14th International Conference, FASE 2011. Proceedings*, 2011, pp. 34–49. DOI: 10.1007/978-3-642-19811-3_4

[99] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline techniques*. Springer Science & Business Media, 2013.

[100] A. Pütz, A. Zlocki, J. Bock, and L. Eckstein, "System validation of highly automated vehicles with a database of relevant traffic scenarios", *situations*, vol. 1, E5, 2017.

[101] L. Putze and E. Böde, "Systematic identification and analysis of hazards for automated systems", *INSIGHT*, vol. 25, no. 4, pp. 100–104, 2022.

[102] X. Qian, I. Navarro, A. de La Fortelle, and F. Moutarde, "Motion planning for urban autonomous driving using Bézier curves and MPC", in *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, IEEE, 2016, pp. 826–833.

[103] J.-D. Quesel and A. Platzer, "Playing hybrid games with KeYmaera", in *International Joint Conference on Automated Reasoning*, Springer, 2012, pp. 439–453.

[104] A. Rauschert and G. Amid, *ASAM OpenSCENARIO XML 1.3.0: Release presentation*, Presentation, Last accessed on 2024-05-08, 2024. [Online]. Available: https://www.asam.net/standards/detail/openscenario-xml/

[105] R. Reussner, I. Schaefer, B. Beckert, A. Koziolek, and E. Burger, "Consistency in the view-based development of cyber-physical systems (convide)", in *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, IEEE, 2023, pp. 83–84.

[106] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles", *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020. DOI: 10.1109/ACCESS.2020.2993730

[107] K. Scheibler, A. Eggers, T. Teige, M. Walz, T. Bienmüller, and U. Brockmeyer, "Solving constraint systems from traffic scenarios for the validation of autonomous driving", in *Proc. of the 4th SC-square Workshop*, 2019, pp. 2–12.

[108] H. Schlingloff, "Crest use cases", in *Model-Based Engineering of Collaborative Embedded Systems*, Springer, 2021, pp. 1–14.

[109] M. Scholtes et al., "6-layer model for a structured description and categorization of urban traffic and environment", *IEEE Access*, vol. 9, pp. 59 131–59 147, 2021.

[110] D. Schramm, M. Hiller, and R. Bardini, "Single track models", in *Vehicle Dynamics*, Springer, 2014, pp. 223–253.

[111] M. Schwammberger, "An abstract model for proving safety of autonomous urban traffic", *Theoretical Computer Science*, vol. 744, pp. 143–169, 2018.

[112] D. Šenkýr and P. Kroha, "Problem of inconsistency in textual requirements specification", in *Proceedings of the 16th international conference on evaluation of novel approaches to software engineering–ENASE*, 2021.

[113] B. Sharma, P. K. Pandya, and S. Chakraborty, "Bounded validity checking of interval duration logic", in *TACAS*, Springer, vol. 5, 2005, pp. 301–316.

[114] E. Thorn, S. C. Kimmel, M. Chaka, B. A. Hamilton, et al., "A framework for automated driving system testable cases and scenarios", United States. Department of Transportation. National Highway Traffic Safety Agency, Tech. Rep., 2018.

[115] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving", in *2015 IEEE 18th international conference on intelligent transportation systems*, IEEE, 2015, pp. 982–988.

[116] UNECE, *Regulation no. 15: Automated lane keeping systems (ALKS)*, 2021.

[117] W. Wachenfeld and H. Winner, "The new role of road testing for the safety validation of automated vehicles", in *Automated Driving*, Springer, 2017, pp. 419–435.

[118] L. Wan, C. Wang, D. Luo, H. Liu, S. Ma, and W. Hu, "Semantic consistency and correctness verification of digital traffic rules", *Engineering*, vol. 33, 2024. DOI: 10.1016/j.eng.2023.04.016

[119] L. Westhofen, C. Neurohr, M. Butz, M. Scholtes, and M. Schuldes, "Using ontologies for the formalization and recognition of criticality for automated driving", *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 519–538, 2022. DOI: 10.1109/OJITS.2022.3187247

[120] L. Westhofen, I. Stierand, J. S. Becker, E. Möhlmann, and W. Hagemann, "Towards a congruent interpretation of traffic rules for automated driving-experiences and challenges", in *Proceedings of the International Workshop on Methodologies for Translating Legal Norms into Formal Representations (LN2FR 2022) in association with the 35th International Conference on Legal Knowledge and Information Systems (JURIX 2022)*, 2022, pp. 8–21.

[121] Wikipedia contributors, "Bézier curve", in *Wikipedia, The Free Encyclopedia*, [Online; accessed 6-March-2023], 2023.

[122] X. Zhang, S. Khastgir, and P. Jennings, "Scenario description language for automated driving systems: A two level abstraction approach", in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2020, pp. 973–980.

[123] D. Zowghi and V. Gervasi, "On the interplay between consistency, completeness, and correctness in requirements evolution", *Information & Software Technology*, vol. 45, no. 14, pp. 993–1009, 2003. DOI: 10.1016/S0950-5849(03)00100-9

Declaration/Erklärung

I hereby certify that I have written this thesis independently and that I have not used any sources other than those specified and I have not made use of any unauthorized external help. Furthermore, the work was not produced using unrecognizable generative AI. I also confirm that I have followed the general principles of scientific work and publication as laid down in the guidelines of good scientific practice of the Carl von Ossietzky University of Oldenburg. Furthermore, I declare that the thesis, in the provided or a modified form, has not yet been submitted to an examination authority.

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen verwendet, sowie keine unzulässige fremde Hilfe in Anspruch genommen habe. Zudem wurde die Arbeit nicht unter unkenntlichem Einsatz generativer KI erbracht. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlichen Arbeitens und Veröffentlichens, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Oldenburg, 3. Juli 2026 Jan Steffen Becker