FAKULTÄT FÜR INFORMATIK

LEHRSTUHL FÜR ROBOTIK UND ECHTZEITSYSTEME

Dissertation

Formal specification, monitoring, and verification of autonomous vehicles in Isabelle/HOL

Albert Rizaldi

FAKULTÄT FÜR INFORMATIK

LEHRSTUHL FÜR ROBOTIK UND ECHTZEITSYSTEME

Formal specification, monitoring, and verification of autonomous vehicles in Isabelle/HOL

Albert Rizaldi

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. Björn Menze
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. Matthias Althoff
	2.	Prof. Tobias Nipkow, Ph.D.

Die Dissertation wurde am 02.08.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 20.11.2019 angenommen.

Acknowledgement

First and foremost I would like to thank my supervisor Matthias Althoff without whose support this PhD thesis would not be possible. I am grateful that he accepted me as his PhD student, guided and showed me how to become a good academic, supported me during the difficult times of research, and always delivered accurate criticisms about my writings. Thanks are in order to Prof. Bjoern Menze for chairing my PhD viva and Prof. Tobias Nipkow for his time to review my thesis in great detail and his leadership of ISABELLE development upon which this thesis depends heavily; I should heed to his advice that "theorem proving can be very addictive".

I am heavily indebted to Fabian Immler who is a great friend, mentor, and collaborator. I learnt everything about the theorem prover ISABELLE from him and I thank him to make the research with ISABELLE such a challenging, interesting, and enjoyable experience. I apologise to always distract him doing works which seem tangential to his PhD thesis and I appreciate his patience of guiding and showing me the real way of using ISABELLE. It is not an overstatement to say that he is a kind of second supervisor for my PhD. I deeply appreciate the help from Jochen Feldle for his legal advice on making sense *Strassenverkehrsordnung*, Jonas Keinholz and Monika Huber, both for helping me with the proofs and formalisation; I apologise if I asked too much as a supervisor.

Special thanks to my lab mates: *1.)* Silvia Breşug (née Magdici) who thought I was an Italian by surname, to whom I apologise if I appeared to be patronising and condescending in every conversation, whose invitation to her wedding to Iaşi was such an eye-opener how of Romanians do weddings and eat, whose Vişinată I always treasure, whose refuge during the Munich shooting I am always indebted, who shared all the goods and perils of doing a PhD, whose help, moral support, encouragement, and kindness make this PhD possible; *2.)* Esra Içer to whom I am indebted in introducing me to Turkish cuisine, to whom I cannot repay back since all my favourite dishes are spicy, who taught me how to say *Teşekkürler* with such an accurate pronounciation that people think I am a genuine Turkish, whose friendship I will always cherish; *3.)* Ahmed

Elguindy whose firstname to this day I cannot say correctly — I tried Akhmed, Ahkmed, Achmed to no avail — whose daily banter in my broken Arabic I always fond of, whose life philosophy of simplicity is summarised by his favourite expression *'do X, Y, and Z, and that's it!'*, whom I always welcome to have a gym session together; *4.)* Dongkun Han who always brought a positive vibe to the group with his studied naïveté, who helped me to find an apartment in Garching, who introduced me to Honghong restaurant which serves the best Szechuan cuisine in the world, whose insistence of making dumpling by hands instead of buying ready-made ones during the Chinese New Year's Eve always irks me, who is definitely one of my best friend in this life; *5.)* Aaron Pereira who suggested me to read Ishiguro's 'Never Let Me Go' which has become my favourite novel ever since, who realised my 'malapropism' when I say 'hit the head on the nail' instead of 'hit the nail on the head', whose quirkiness of writing 'focussing' instead of 'focusing' I always use now, whose energy to work always puts my slacking off to shame; *6.)* Andrea Giusti who is my Italian 'sibling', who introduced me of how to make the authentic *Spaghetti alla Carbonara*, who suggested me his regional wine *Amarone* even though deep down I preferred *Barbera*; *7.)* Bastian Schürmann who was my office mate, who initially always said *Gesundheit* everytime I sneezed but gave up once he realised that I did it too often, who probably thought I was such a brutish since I sneezed loudly to which I deeply apologise, whose willingness to share his work I deeply appreciated; *8.)* Carmella Schürmann who realised that I am a charlatan in using British English when I uttered 'flashlight' instead of 'torch', who does not believe in Oxford comma, whose services in proofreading are always impreccable, professional, and appreciated; *9.)* Markus Koschi who brought yet another positive vibe to this lab after Dongkun left, who mistook lard for butter during our CPS outing in Malta, who looks like Charlie Cox in Daredevil which made me wondering whether he was wandering during the nights fighting crimes in Munich instead of Hell's Kitchen, whose help and warmth during my PhD training I am thankful of; *10.)* Stephanie Manzinger a fellow Garchinger whom I misrecognised at first as Melanie (Ahmed's master student), a proud German who used (uses?) the expression 'How late is it?' (a *mot à mot* translation from *Wie spät ist es?*) to ask for time, who read a poignant *laudatio* after my viva, whom I had the honour to know of; *11.)* Christian Pek who is an amazing speaker with his ability to speak in James Joyce's stream-of-consciousness style, who has kindly arranged and invited me to his CPS international dinner, who helped me a lot during my viva which I gratefully acknowledge; and *12.)* Sebastian Söntges whose praised me to be a good presenter which makes me wondering whether I should be a salesperson.

I would like to extend my sincere thanks to Ute Lomp, who always thinks that I am Andrea and Andrea me, and Amy Bücherl for helping with administrative matters.

Finally I would like to thank three most important women in my life my grandmother Tjoe Tjhiu Kit who sadly passed away during my PhD training, my mother Tjoe Soean Kie, and my wife Nora Asteria Rizaldi (née Ramli). The latter is my source of energy and motivation for completing this PhD thesis during the time when I was having writer's block. I realised that living with a person with high degree of perfectionism, low level of capability of sustaining attention, high degree of neuroticism, low level of orderliness, high degree of procrastination like me could be very tough and challenging; I deeply appreciate her patience, kindness, and understanding during the writing of this thesis.

Abstract

This thesis combines three formal verification techniques — theorem proving, satisfiability checking (runtime monitoring), and reachability analysis — for formally analysing autonomous vehicles.

First, we formalise the required elements which will be the foundation for formally analysing autonomous vehicles: the notion of safe distance between an ego vehicle and its front vehicle, the framework for predicting the spatial occupancies of other traffic participants, the road networks in which autonomous vehicles shall operate, and relevant primitives required to detect lanes in which an autonomous vehicle currently occupies. Each of these elements are proved to be correct with respect to their own specification in the theorem prover ISABELLE, and refined to functions which are guaranteed to be correct numerically.

Next, we formalise a collection of selected traffic rules from the German traffic code (*Straßenverkhersordnung*) and propose a method to monitor them formally. We translate the natural language requirements to formulas in linear temporal logic (LTL) without considering what each atomic proposition means initially. Then, we define the meaning of those atomic propositions precisely by using the previously formalised elements for formal analyses of autonomous vehicles. We monitor the satisfaction of a trace with respect to these formulas by executing the semantics of (finite-time) LTL implemented in the theorem prover ISABELLE.

Lastly, we demonstrate how to use reachability analysis to *formally* construct a manoeuvre automaton which can be used for motion planning of autonomous vehicles; this construction involves a careful interaction with *uncertified* tools other than ISABELLE. Then, we propose a variant of LTL which is interpreted over sets instead of single trajectories because traces of manoeuvre automata will be sets (due to reachability analysis). We formalise a plan with this new specification language and then use satisfiability checking in conjunction with the previously formalised monitoring framework to search for a sequence of manoeuvres that is guaranteed to satisfy the plan.

Zusammenfassung

Diese Dissertation kombiniert drei Techniken der formale Verifikation - Theorembeweisen, Erfüllbarkeitsanalyse der Aussagenlogik und Erreichbarkeitsanalyse - um autonome Fahrzeuge formal zu analysieren.

Zuerst werden die notwendigen Elemente formalisiert, die als Grundlage zur formalen Analyse autonomer Fahrzeuge dienen: der Begriff des Sicherheitsabstands zwischen einem Ego-Fahrzeug und dem vorderen Fahrzeug, die Vorhersage der Straßenbelegung anderer Verkehrsteilnehmer, das Straßennetzwerk in dem autonome Fahrzeuge fahren sollen und relevante Primitive um Fahrspuren zu erkennen, die ein autonomes Fahrzeug momentan besetzt. Jedes dieser Elemente ist mit dem Theorembeweiser Isabelle als korrekt in Bezug zu deren Spezifikation bewiesen worden mit deren Hilfe Funktionen abgeleitet wurden, die numerische korrekt sind.

Weiterhin wurde eine Menge an ausgewählten Verkehrsregeln der Straßenverkehrsordnung formalisiert und eine Methode vorgeschlagen, die deren Einhaltung formal überwacht. Die natürlichsprachigen Anforderungen wurden zu Formeln in linearer temporaler Logik (LTL) übersetzt ohne die ursprüngliche Bedeutung der Aussagenvariablen zu berücksichtigen. Als nächstes wurde die Bedeutung dieser Aussagenvariablen präzisiert, indem die zuvor formalisierten Elemente formaler Analyse autonomer Fahrzeuge herangezogen wurden. Die Einhaltung von Verhalten bezüglich dieser Formeln wird überwacht indem die im Theorembeweiser Isabelle implementierte Semantik von (zeitbeschränktem) LTL ausgeführt wird.

Zuletzt wird die Verwendung von Erreichbarkeitsanalyse zur formalen Konstruktion eines Manöver-Automaten demonstriert, der zur Bewegungsplanung autonomer Fahrzeuge verwendet werden kann; dieser Ansatz erfordert ein sorgfältiges Zusammenspiel mit nicht-zertifizierten Werkzeugen neben Isabelle. Danach schlagen wir eine Variante von LTL vor, die über Mengen von Trajektorien interpretiert wird, anstatt über einzelne Trajektorien, da Verhalten von Manöver-Automaten sich nur durch Mengen beschränken lassen (aufgrund der Erreichbarkeitsanalyse). Ein Plan wird mit dieser neuen Spezifikationssprache formalisiert und anschließend wird eine Erfüllbarkeitsanalyse der Aussagenlogik zusammen mit dem vorher formalisierten Überwachungsframework ausgeführt, um eine Folge von Manövern zu finden, die den Plan garantiert erfüllt.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 4

1.3 Publications 5

1.4 Preliminaries 6

2 Concretising safe distance **11**

2.1 Modelling the safe distance problem 12

2.2 Logical analysis of the safe distance problem 16

2.3 Designing a sound-and-complete checker 22

2.4 Sound and executable checker using interval arithmetic 29

2.5 Related work 32

3 Predicting the occupancies of other vehicles **35**

3.1 Modelling acceleration-based occupancy 36

3.2 Constructing boundaries for occupancy circles 38

3.3 Approximating boundaries with secant lines 45

3.4 Occupancies over a time interval 49

3.5 Sound and executable occupancy prediction 53

3.5.1 Enlarging rectangles due to uncertainties 53

3.5.2 Interval arithmetic-based occupancy prediction 58

3.6 Related work 63

4 Monitoring overtaking traffic rules with LTL **65**

4.1 Codifying overtaking traffic rules in LTL 66

4.2 Modelling lanelets and lanes for lane detection 68
4.2.1 Generalised lanes 69
4.2.2 Lanelets 71
4.3 Designing and verifying a lane detection function 74
4.3.1 Segments intersection test 75
4.3.2 Point-in-lanelet test 81
4.3.3 Lane detection function 84
4.4 Interpretation with lane detection and safe distance checker 87
4.5 Sound and executable LTL monitoring of traffic rules 92
4.6 Related Work 95

5 Formal verification of motion planners based on manœuvre automata 99
5.1 Affine forms and zonotopes for representing sets 100
5.2 Constructing Manoeuvre Automata 102
5.3 Motion Planning with Manoeuvre Automata 110
5.3.1 Interpreting LTL over manoeuvre automata 111
5.3.2 Satisfiability checking of LTL over manoeuvre automata 115
5.4 Related work 117

6 Conclusion 121

Bibliography 125

1 Introduction

1.1 Motivation

Formally proving the correctness of autonomous vehicles is very challenging. Not only do we have to formally specify the requirements, which are nearly always written in natural language, but we also need to find a computational structure which is expressive enough for modelling the discrete and continuous behaviours of autonomous vehicles, yet simple enough for formal analyses. The challenges do not stop there: how can we formally prove that a fixed model satisfies a specification? Should we opt for model-theoretic or proof-theoretic approaches to show this satisfaction? How legible is the formal language used for specifying the requirements? Would an engineer need retraining for understanding the semantics?

Which techniques in the literature can we use to formally verify autonomous vehicles? If we view autonomous vehicles as hybrid systems, there are four categories of technique we can potentially choose from: model checking [1, 2], reachability analysis [3, 4, 5, 6, 7, 8, 9, 10, 11], theorem proving [12, 13, 14], and satisfiability modulo theories [15]. Each of these techniques differs in terms of the formal language used to specify properties, the structure used to model (hybrid) systems, and the method employed to prove that the formal structure satisfies a property (see Tab. 1.1). Among these dimensions of comparison, we argue in this thesis that the formal language stands out as the determining factor for choosing a method for formally verifying autonomous vehicles.

Despite the successes achieved by these techniques, we argue that their formal languages are not expressive enough for formalising system-level requirements. Take as an example that a vehicle must maintain a safe distance with the vehicle in front to avoid rear-end collisions. The Vienna Convention on Road Traffic §13(5) [16] specifies this requirement descriptively as follows:

Table 1.1: Comparison of techniques for formal verification of hybrid systems

Technique	Language	Structure	Method
Theorem Proving	Dynamic Logic	Hybrid Program	Proof-theoretic
Model Checking	Temporal Logic	Hybrid Automata	Model-theoretic
Reachability Analysis	Set theory	Hybrid Automata	Model-theoretic
Satisfiability Modulo Theory	First-Order Logic	Hybrid Automata	Model-theoretic

> The driver of a vehicle moving behind another vehicle shall keep at a sufficient distance from that other vehicle to avoid a collision if the vehicle in front should suddenly slow down or stop.

But how large is this "sufficient distance" concretely? Without a concrete number, we cannot use the specification language associated with each technique in Tab. 1.1 for formally verifying autonomous vehicles — except for that of theorem proving technique. Loos et al. [17] could specify an expression for safe distance in KeYmaera and prove that the cruise controller satisfies this property formally. As another example, take the overtaking traffic rule from *Straßenverkehrsordnung* (StVO) §5(4) as follows:

> When changing the lane to the left lane during overtaking, no following road users shall be endangered. During overtaking, a sufficient side clearance must be provided to other road users, especially pedestrians and cyclist. The driver who overtakes has to change from the fast lane to the right lane as soon as possible. The road user being overtaken shall not be obstructed.

The notion of *overtaking* cannot be generalised by an expression (as in safe distance expression in KeYmaera) any more; we need to reason geometrically about the occupancies, lane dividers, and road networks in order to define what overtaking means. In other words, we need a more expressive logic and tools to formalise these system-level requirements, or we need tools that translate from high-level specifications to simpler logics.

This thesis opts for the ISABELLE theorem prover [18] with higher-order logic (HOL) — hence the term ISABELLE/HOL — for formalising system level requirements. Being a generic theorem prover, ISABELLE[1] allows us to specify a safe distance expression symbolically and, instead of taking this safe distance expression for granted, to prove that this expression is sound.[2] Proving the soundness of a safe distance expression

[1] From now on, we shall abbreviate ISABELLE/HOL with ISABELLE only.

[2] Larsen et al. [19] for example assume that this safe distance is 5 m in their work to formally verify a controller with UPPAAL [20] model checker. However, there is no guarantee that 5 m is a safe distance — it is merely a heuristics.

requires concepts from real analysis such as continuity, differentiability, and integrability. ISABELLE provides a rich library of theories about these concepts with which we can immediately use to prove the soundness. Additionally, ISABELLE also contains formalised theories on geometry for modelling the environment in which an autonomous vehicle operates such as occupancies, lane, lane boundaries, lane dividers, etc. Such concepts are required for formalising the notion of *overtaking*.

ISABELLE serves as a platform for formal development of theories only; we still have to figure out how to prove the correct implementation of autonomous vehicles with respect to formalised properties in higher-order logic. To avoid reinventing the wheel, we combine model checking — more precisely satisfiability checking — and reachability analysis (see Tab. 1.1) on top of ISABELLE. This choice is due to two reasons: *1) automation*, since these two techniques are designed with this in mind, and *2) formalisation* of these two techniques are available in ISABELLE [14]. In this setting, properties are still formalised in higher-order logic but are now codified in linear temporal logic (LTL)[3]. Reachability analysis is used to construct hybrid automata representing autonomous vehicles which are amenable to standard model checking verification techniques.

Autonomous vehicles are complex systems which are composed of sensors, actuators, processors, operating systems, and controllers among many other things (see Fig. 1.1). Full formal verification of autonomous vehicles thus entails each of these components is also formally verified — a tall order to accomplish. Instead, this thesis focusses on formally verifying the motion planner of autonomous vehicles. The motion planner considered in this thesis does not include the mission planning subsystem which plans on the map level (routeing), but operates more on a fine-grained abstraction level which must consider static obstacles, other traffic participants, road surfaces, and traffic rules. Sensor readings are assumed to be within certain bounds.

Motion planning based on LTL specifications [22] is not the only way to do motion planning; other techniques are broadly categorised as [23]: *1) variational methods* which employ nonlinear optimisation methods, *2) graph search methods* which use search techniques over a discrete abstraction of a vehicle's environment, and *3) incremental search techniques* which use random sampling to build a search tree incrementally. There is also a growing interest in using machine learning-based techniques for motion planning. To make the formalisation in this thesis more applicable to other motion planning techniques, we also present a framework for *monitoring* plans — checking formally whether a trace satisfies LTL formulæ. To ensure safety and compliance with traffic rules, plans from these motion planners are executed only if they are certified by the monitors.

Many incidents in safety-critical systems such as the PATRIOT MISSILE failure [24] and the explosion of the ARIANE 5[25] are due to numerical errors[4]. Autonomous

[3] think of LTL as a Domain Specific Language (DSL).

[4] Vuik maintains this type of incident in `http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.`

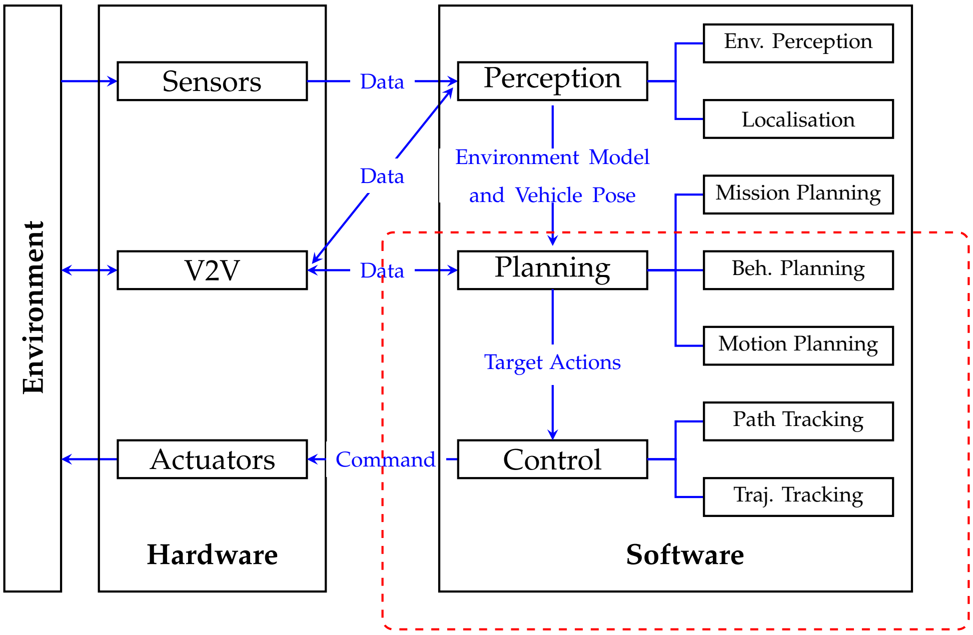


Figure 1.1: Typical architecture for autonomous vehicles (taken, reproduced, and modified from Pendleton et al. [21]). We focussed mainly on the motion planning aspect which considers the effects of control for motion planning.

vehicles are also similar to these two systems in the sense that numerical faults could result in an erroneous system state and lead to a system failure (known as the *fault-error-failure* model or chain [26]) eventually. To avoid similar incidents, this thesis considers numerical correctness seriously too. One advantage of using a generic theorem prover such as ISABELLE is that it allows us to refine theories involving real numbers into those with floating-point numbers soundly. That is, *1)* each real number is first approximated either by an interval [27] or by its affine form [28]; *2)* arithmetic operations (e.g. division, multiplication) and transcendental functions (e.g. sine, cosine, logarithm) are over-approximated safely; and then *3)* boolean expressions are evaluated soundly, e.g. inequality $[l_1; u_1] < [l_2; u_2]$ is true if and only if $u_1 < l_2$.

1.2 Contributions

The contributions of this thesis are:

- *A sound-and-complete concrete safe distance expression* (Ch. 2).
  Safe distance is ubiquitous in formalised traffic rules since many other predicates are defined in terms of it. However, most definitions of safe distance in the

`html`.

literature are at best descriptive and hence are very difficult to monitor. Even when the safe distance is given a concrete number for a specific situation, it is rarely proved that it is indeed sound. This chapter presents a verified, concrete safe distance expression. When a vehicle complies with this safe distance, it is guaranteed to avoid rear-end collisions.

- *A formally verified procedure for predicting the occupancies of other vehicles* (Ch. 3).
An autonomous vehicle must predict the physical space occupied by other traffic participants in order to plan a motion which avoids occupying the same space. The safety of the plan relies on the correctness of this prediction. If the actual space which can be occupied is under-represented, our autonomous vehicles might occupy the space which they thought to be safe but actually is not. This chapter presents a formalised procedure for computing the future occupancies of other traffic participants; other traffic participants are ensured to be enclosed by these predicted spaces.

- *Formally specified traffic rules and their formally verified monitors* (Ch. 4).
Monitoring traffic rules formally is challenging due to two reasons: concepts in legal texts are often ambiguous, and the trace from planners are generally time-sampled. By using the concretisation results from the previous chapters, this chapter formalises a subset of traffic rules in LTL to avoid these ambiguities. To ensure the faithfulness of our monitoring, time-sampled traces are enclosed safely by their continuous sets by using reachability analysis techniques. By using this framework, we can formally guarantee that a trace complies with traffic rules.

- *A formally verified motion planner based on manoeuvre automata* (Ch. 5).
Monitoring only ensures that a single trace satisfies a formalised property. To formally verify a motion planner, we have to ensure that all traces from a plan produced by the motion planner also satisfy the formalised property. We formally construct manœuvre automata-based motion planners in which each motion primitive is equipped with its reachable sets. With this formally constructed manœuvre automata, a verified plan can be found by performing satisfiability checking — a search over the states of manœuvre automata — and using the monitoring framework from Ch. 4.

Related work can be found at the end of each chapter.

1.3 Publications

This thesis is based on the following publications:

[1] A. Rizaldi, S. Söntges, and M. Althoff. "On time-memory trade-off for collision detection". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. June 2015, pp. 1173–1180.

[2] A. Rizaldi and M. Althoff. "Formalising Traffic Rules for Accountability of Autonomous Vehicles". In: *Proc. of the IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 1658–1665.

[3] A. Rizaldi, F. Immler, and M. Althoff. "A Formally Verified Checker of the Safe Distance Traffic Rules for Autonomous Vehicles". In: *NASA Formal Methods*. Ed. by S. Rayadurgam and O. Tkachuk. Cham: Springer International Publishing, 2016, pp. 175–190.

[4] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow. "Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL". In: *Integrated Formal Methods*. Ed. by N. Polikarpova and S. Schneider. Cham: Springer International Publishing, 2017, pp. 50–66.

[5] A. Rizaldi, F. Immler, B. Schürmann, and M. Althoff. "A Formally Verified Motion Planner for Autonomous Vehicles". In: *Automated Technology for Verification and Analysis*. Ed. by S. K. Lahiri and C. Wang. Cham: Springer International Publishing, 2018, pp. 75–90.

1.4 Preliminaries

Types. ISABELLE is a typed specification language. Although a specification language does not have to be typed (cf. the debate in [29]), a typed language is useful for eliminating ill-defined expressions. Types in ISABELLE could be a base type such as boolean $\mathbb{B}$, natural numbers $\mathbb{N}$, and real numbers $\mathbb{R}$, or a type variable α, β, γ and so on. Classically trained mathematicians and engineers (in set theory) are used to writing $t \in \mathbb{R}$ for denoting a real variable t. In ISABELLE, this is more or less translated into $t :: \mathbb{R}$ (which reads "t is of type $\mathbb{R}$"). However, *types* and *sets*, strictly speaking, are not the same. Types are constructed via function type constructor $\alpha \Rightarrow \beta$, product type constructor $\alpha \times \beta$, list type constructor $(\alpha)\mathit{list}$, or set type constructor $(\alpha)\mathit{set}$ — the last two are written in postfix notation.

Given a term $t :: \alpha \times \beta$, we write $\mathit{fst}(t) :: \alpha$ and $\mathit{snd}(t) :: \beta$ to refer to the first and second element of t, respectively. For $n \geq 2$, referring to a certain element in the tuple could involve complex combination of *fst* and *snd*, which makes the expression less readable. In such cases, we can use the keyword **record** with a proper function to access each element (field). For example, we can declare the following *rectangle* type with **record** as follows:

$$\textbf{record}\ \mathit{rectangle} = \mathit{centre} :: \mathbb{R}^2 + \mathit{width} :: \mathbb{R} + \mathit{length} :: \mathbb{R} + \mathit{ori} :: \mathbb{R}\ ,$$

If we use a 4-tuple $\mathbb{R}^2 \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ instead of **record**, we must use the expression $\mathit{fst} \circ \mathit{snd} \circ \mathit{snd}$ to access the third element in the tuple. With **record** as in the type definition above, we can simply use *length* to obtain the third element of a rectangle.

For a term xs :: $(\alpha)\mathit{list}$, the expression $\mathit{hd}(xs)$ :: α refers to the first element of the list and $\mathit{tail}(xs)$ :: $(\alpha)\mathit{list}$ is the list xs without the first element. Several important operators related to lists are: *1.) concatenation* of two lists xs and ys is denoted by $xs\,@\,ys$; *2.) length* of a list xs is denoted by $|xs|$; *3.) repeating* an element x for n times is denoted by $\mathit{replicate}(n, x)$; and *4.) selecting* the n-th element of a list xs is denoted by $xs\,!\,n$, provided that $n < |xs|$. Sometimes, it is also useful to represent lists as sets. This is achieved via the function $\mathit{set}(xs)$, and it is not the same with the expression on the type level, i.e., $(\alpha)\mathit{set}$; the previous is a function while the latter type constructor. Lastly, we use the notation $\{exp \mid x\,y\,\ldots z.\ P\}$ as a set builder. Here exp is the expression, variables $x, y, \ldots z$ are bound variables, and P is a predicate. For example, the Minkowski addition of two sets A and B is written as $\{a + b \mid a\,b.\ a \in A\ \wedge\ b \in B\}$.

Apart from the predefined type constructors above, users can also define their own types by using **datatype** keyword. For instance, in traffic scenarios, we can define the collection of all obstacles as follows:

$$\textbf{datatype}\ \mathit{obstacle}\ =\ \mathit{Left\text{-}bound} \mid \mathit{Right\text{-}bound} \mid \mathit{Car}(i\ ::\ \mathbb{N})$$

This new type *obstacle* consists of the left and right boundary of the road alongside with other cars whose identifiers are labelled with natural numbers. Note that the usage of keyword **datatype** for constructing a new type is similar to the notation used in Backus–Naur Form (BNF) to describe the grammar of a syntax. One of most frequently used, predefined type constructed in this fashion is the $(\alpha)\mathit{option}$ type defined as follows:

$$\textbf{datatype}\ (\alpha)\mathit{option}\ =\ \mathit{None} \mid \mathit{Some}(x\ ::\ \alpha)\ .$$

This type is useful to handle *undefinedness* in computation, and we show how we use this type in the next section.

Terms. ISABELLE can be viewed as a functional programming language and hence its terms are formed as in functional programming too: they could either be a constant, a variable, a function application, or a function abstraction $(\lambda x.\ t)$. As opposed to the standard functional programming convention, function application is written not by juxtaposing the function and its argument — as in $f\ x\ y$ — but by classically placing a pair of enclosing parentheses — as in $f(x, y)$. This decision primarily serves the purpose of increasing the readership of this thesis; seeing function application without clearly distinguishing its *function* and *arguments* for the uninitiated could be unnatural and challenging. Moreover, we rarely talk about *currying* in this thesis, and it does not hurt to clearly separate a set of arguments with commas and to explicitly enclose them with a pair of parentheses.

One of the basic tools for handling complexity is the abstraction through *naming* (definition). Instead of writing the expression $\lambda a\, b\, c.\ b^2 - 4 \cdot a \cdot c$ everytime we need it, we could name this expression as discr. In this thesis, definitions are written following the standard mathematical notation ':=' such as $\mathsf{discr}(a, b, c) := b^2 - 4 \cdot a \cdot c$. When we define a function in mathematics, the expression could vary depends on a certain condition. For example, determining whether the equation $a \cdot x^2 + b \cdot x + c = 0$ has roots or not can be checked via the following function:

$$\mathsf{has\text{-}roots}(a, b, c) = \begin{cases} \mathsf{True} & \text{if } \mathsf{discr}(a, b, c) \geq 0, \\ \mathsf{False} & \text{otherwise .} \end{cases}$$

In this thesis, we simply write $\mathsf{has\text{-}roots}(a, b, c) := \textbf{if } \mathsf{discr}(a, b, c) \geq 0 \textbf{ then } \mathsf{True} \textbf{ else } \mathsf{False}$ — utilising the familiar and intuitive *if-then-else* construct found in many programming languages. Another useful construct for handling complexity used in this thesis is the **let** − **in** construct. For example, measuring the distance between two points $p_1, p_2 :: \mathbb{R} \times \mathbb{R}$ is defined as follows:

$$\mathsf{dist}(p_1, p_2) \ := \ \textbf{let } dx = \mathsf{fst}(p_1) - \mathsf{fst}(p_2);\ dy = \mathsf{snd}(p_1) - \mathsf{snd}(p_2) \textbf{ in } \sqrt{dx^2 + dy^2} \ .$$

Given a type constructed via **datatype**, we can use the construct **case** to perform a pattern matching. Suppose that we want to define a function to obtain an identifier for an obstacle. This function has the type signature of $\mathsf{traffic\text{-}id} :: \mathsf{obstacle} \Rightarrow (\mathbb{N})\mathsf{option}$ and can be defined as follows:

$$\mathsf{traffic\text{-}id}(o) \ := \ \textbf{case } o \textbf{ of } \mathsf{Left\text{-}bound} \Rightarrow \mathsf{None} \mid \mathsf{Right\text{-}bound} \Rightarrow \mathsf{None} \mid \mathsf{Car}(i) \Rightarrow \mathsf{Some}(i) \,.$$

Note that the expression above has to enumerate all constructors although we are concerned only with Car; both $\mathsf{Left\text{-}bound}$ and $\mathsf{Right\text{-}bound}$ do not have identifiers. To streamline the expression above, we can use the wildcard pattern '_' to denote "don't care" as follows:

$$\mathsf{traffic\text{-}id}(o) \ := \ \textbf{case } o \textbf{ of } \mathsf{Car}(i) \Rightarrow \mathsf{Some}(i) \mid \_ \ \Rightarrow \mathsf{None} \ .$$

Theorems. We use the double arrow '$\Longrightarrow$' to denote a deduction in ISABELLE. Deductions are traditionally written in logic textbooks as a line separating the expressions above (antecedents) and below (consequents) the line such as $\dfrac{A \wedge B}{A}$; this is equivalent to $A \wedge B \Longrightarrow A$ in ISABELLE. In case there are multiple antecedents, we simply curry them so that the last expression is the consequent and everything before the last '$\Longrightarrow$' is the antecedent. For example, the rule for introducing a conjunction is typically written as $\dfrac{A \quad B}{A \wedge B}$, but we simply write $A \Longrightarrow B \Longrightarrow A \wedge B$ in ISABELLE. Note that there is another arrow $\longrightarrow$ which is used to denote an implication. The following formalisation of *modus ponens* rule in ISABELLE clarifies the difference between $\Longrightarrow$ and $\longrightarrow$:

$$P \longrightarrow Q \ \Longrightarrow P \Longrightarrow Q \ .$$

Proofs. In a system development thesis, it is rarely the case that the detail source code is provided and discussed; the description and the discussion happen on a higher level of abstraction, i.e., the pseudocode. Similarly, we refrain from providing ISABELLE's proof for each proved theorem and only provide a proof sketch instead. The presentation of the proof sketch most often follows the calculational style [30, 31, 32, 33]. The rough format of the proof is shown as follows:

$$\begin{array}{ll} & A \\ \square & \{ \text{ hint of why } A \mathrel{\square} B \ \} \\ & B \\ \square & \{ \text{ hint of why } B \mathrel{\square} C \ \} \\ & C \end{array}$$

Typically, the placeholder $\square$ could either be $\Longrightarrow$, $\Longleftarrow$, or $\equiv$ where $A \equiv B$ if and only if $A \Longrightarrow B$ and $A \Longleftarrow B$. After performing this calculation, we can deduce that $A \mathrel{\square} B$ due to the transitivity property — provided that each step is valid. When $\square$ is a series of $\Longleftarrow$ and $\equiv$ in the calculation, we are employing the technique of backward proof; it usually starts with the formula we want to prove and ends with the constant *True*. When $\square$ is a series of $\Longrightarrow$ and $\equiv$, we are utilising the technique of forward proof; it usually starts with the antecedent of the premise and end with the consequent of the deduction. If we want to prove by contradiction, we start with the negated formula and end with the constant *False*.

2 Concretising safe distance

The notion of safe distance is very fundamental in formalising and monitoring traffic rules. Traffic codes such as the Vienna Convention, the UK Highway Code, and the German *Straßenverkehrsordnung* explicitly mention that a vehicle has to maintain a safe distance to vehicles in front of it such that, whenever the vehicle in front brakes immediately, there will not be any rear-end collision. Additionally, many other predicates in traffic rules can be defined in terms of safe distance. Consider the first sentence in StVO §5(4) about overtaking:

> When changing the lane to the left lane during overtaking, no following road users shall be endangered.

According to the legal experts whom we consulted during a legal analysis process, the predicate 'endangered' has the interpretation that, when it starts to move to the next lane, the ego vehicle must leave a safe distance for the vehicle behind in the next lane. Consider also the last two sentences in the overtaking rules:

> The driver who overtakes has to change from the fast lane to the right lane as soon as possible. The road user being overtaken shall not be obstructed.

The predicate 'obstructed' is interpreted again — according to the legal experts — as not giving sufficient (safe) distance to the vehicle being overtaken. Even the predicate 'as soon as possible' is interpreted as the earliest time a vehicle has leave a safe distance to the vehicle being overtaken.

Admittedly, not everybody shall agree with the interpretation of 'endangered' and 'obstructed' with leaving safe distance with respective vehicle. The Vienna Convention §7(4), for example, also mentions that a vehicle shall not cause noise, raise dust or smoke which could also be the interpretation for 'endangered' and 'obstructed'. The UK

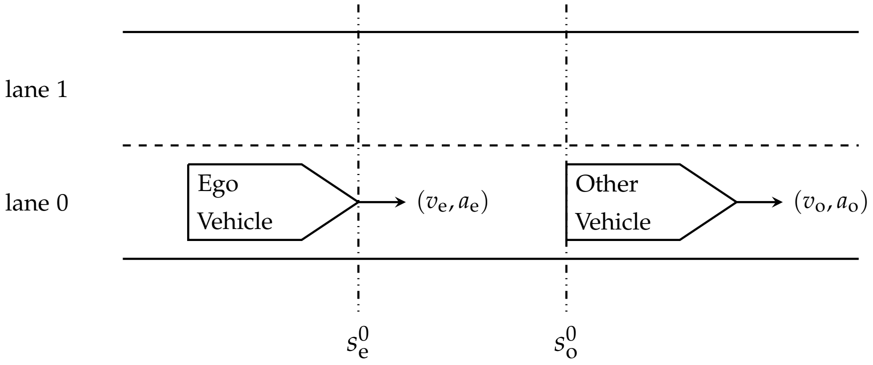


Figure 2.1: Scenario for safe distance problem.

Highway Code also strongly prohibits road users to throw anything out of a vehicle, e.g. food, food packaging, cans, papers, or carrier bags — which "could *endanger* [emphasis mine] other road users, particularly motorcyclists and cyclists"(rule 147). However, as emphasised in the introduction, this thesis considers only the motion planning aspect of autonomous vehicles, hence, events such as causing noise, raising smoke, throwing papers are abstracted away. Therefore, I argue, the only sensible interpretations of 'endangered' and 'obstructed' are those which depend on the notion of safe distance.

The main objective of this chapter is to define a sound-and-complete, concrete safe distance expression. In reality, it is impossible to guarantee with 100% certainty that a distance is always safe; certain assumptions must hold. This chapter assumes that vehicles move according to the point-mass model (Sec. 2.1) which allows us to formulate the stopping distances for the ego vehicle and the vehicle in front. These expressions are then used as bases for case analysis (Sec. 2.2) to decide whether a collision will happen in each case or not. This case analysis, in turn, serves as a guide to design a checker (Sec. 2.3) which is proved to be sound and complete. We also extend the checker safely to deal with the limited precision in floating-point numbers (Sec. 2.4).

This chapter describes joint work with Fabian Immler and Matthias Althoff [34].

2.1 Modelling the safe distance problem

Figure 2.1 illustrates the safe distance problem considered in this thesis. The scenario consists of two vehicles: the *ego* vehicle and the closest vehicle in front of it—which we term the *front* vehicle. This scenario is uniquely characterised by six constants: $s_e^0, v_e, a_e :: \mathbb{R}$ from the ego vehicle and $s_f^0, v_f, a_f :: \mathbb{R}$ from the front vehicle. Constants s^0, v, a denote the initial position, initial speed, and maximum deceleration value, respectively, of a vehicle. Note that s_e^0 denotes the *frontmost* position of the ego vehicle, while s_f^0 denotes the *rearmost* position of the front vehicle. Additionally, we also make the following assumptions:

Assumption 2.1. *The values of v_e and v_f are non-negative:* $0 \leq v_e \ \wedge \ 0 \leq v_f$.

Assumption 2.2. *The values of a_e and a_o are negative:* $a_e < 0 \ \wedge \ a_o < 0$.

Assumption 2.3. *The front vehicle is indeed located in front of the ego vehicle:* $s_e^0 < s_f^0$.

The first assumption is assumed so because we are only considering highway scenarios, in which case driving backwards is prohibited. The second assumption is only a matter of style; one could also assume that the value of maximum decelerations to be positive, but it would change the expression $\frac{1}{2} \cdot a \cdot t^2$ in (2.1) into $-\frac{1}{2} \cdot a \cdot t^2$. We assume the values to be negative in order to make the signs, i.e. addition, uniform in (2.1). The third assumption is self-explanatory; it is the mathematically precise way to define a vehicle being a *front* vehicle.

The point-mass model specifies the movement of a vehicle $p \ :: \ \mathbb{R} \Rightarrow \mathbb{R}$ with a second-order Ordinary Differential Equation (ODE) $p''(t) = a$ and initial value conditions $p(0) = s^0$ and $p'(0) = v$. The closed-form solution to this ODE is as follows:

$$\begin{aligned} p(t) \ &:= \ s^0 + v \cdot t + \frac{1}{2} a \cdot t^2 \ , && (2.1) \\ p'(t) \ &:= \ v + a \cdot t \ . && (2.2) \end{aligned}$$

Since (2.1) is a quadratic equation, it has the shape of a parabola when $a \neq 0$. This fact implies that a vehicle would move backward after it stops — which is not the case in reality. Hence, (2.1) is only valid for the interval $[0; t_{\text{stop}}]$ where t_{stop} is the stopping time. The stopping time t_{stop} is the time when the first derivative of p is zero, that is, $p'(t_{\text{stop}}) = 0$. Substituting t with t_{stop} in (2.2) results into the following expression for t_{stop}:

$$t^{\text{stop}} := -\frac{v}{a} \ . \tag{2.3}$$

Thus, we can extend the movement p of (2.1) by introducing discrete jumps (the deceleration makes a jump from $a < 0$ to $a = 0$) into the overall movement s_{f} for the front vehicle as follows:

$$s_{\text{f}}(t) := \begin{cases} s_{\text{f}}^0 & \text{if } \ t \leq 0 \ , \\ p_{\text{f}}(t) & \text{if } \ 0 \leq t \leq t_{\text{f}}^{\text{stop}} \ , \\ p_{\text{f}}(t_{\text{f}}^{\text{stop}}) & \text{if } \ t_{\text{f}}^{\text{stop}} \leq t \ , \end{cases} \tag{2.4}$$

where p_{f} is p in (2.1) with variables s^0, v, and a are instantiated by $s_{\text{f}}^0, v_{\text{f}}$, and a_{f}, respectively; the term $t_{\text{f}}^{\text{stop}}$ is interpreted similarly too (see Fig. 2.2). Hence, after $t_{\text{f}}^{\text{stop}}$ the vehicle will stop at[1]

$$s_{\mathrm{f}}^{\mathrm{stop}} \;:=\; s_{\mathrm{f}}\,(t_{\mathrm{f}}^{\mathrm{stop}}) \;=\; s_{\mathrm{f}}^{0} - \frac{v_{\mathrm{f}}^{2}}{2 \cdot a_{\mathrm{f}}} \ . \tag{2.5}$$

Two important lemmas for the movement of the front vehicle is about *monotonicity* and *maximum at stopping times*.

Lemma 2.1. $t_1 \leq t_2 \implies s_f(t_1) \leq s_f(t_2)$ ***and*** $s_f(t) \leq s_f(t_f^{stop})$

Proof. There are three sets to consider: $(-\infty, 0]$, $[0, t_{\mathrm{f}}^{\mathrm{stop}}]$, and $[t_{\mathrm{f}}^{\mathrm{stop}}, \infty)$. We first assume that both t_1 and t_2 belong to one of these sets. If both t_1 and t_2 are on either the set $(-\infty; 0]$ or $[t_{\mathrm{f}}^{\mathrm{stop}}, \infty)$, it is obvious that $t_1 \leq t_2 \implies s_{\mathrm{f}}(t_1) \leq s_{\mathrm{f}}(t_2)$ because $s_{\mathrm{f}}(t_1) = s_{\mathrm{f}}(t_2)$ according to (2.4). Next, we show that p_{f} is non-decreasing in $[0, t_{\mathrm{f}}^{\mathrm{stop}}]$ as follows:

$$
\begin{array}{ll}
 & p_{\mathrm{f}}(t_1) \;\leq\; p_{\mathrm{f}}(t_2) \\
\Longleftarrow & \quad \{ \text{ unfolding } p_f \text{ according to (2.1) } \} \\
 & s_{\mathrm{f}}^{0} + v_{\mathrm{f}} \cdot t_1 + \frac{1}{2} \cdot a_{\mathrm{f}} \cdot t_1^2 \;\leq\; s_{\mathrm{f}}^{0} + v_{\mathrm{f}} \cdot t_2 + \frac{1}{2} \cdot a_{\mathrm{f}} \cdot t_2^2 \\
\Longleftarrow & \quad \{ \text{ arithmetic } \} \\
 & 0 \leq v_{\mathrm{f}} \cdot (t_2 - t_1) + \frac{1}{2} \cdot a_{\mathrm{f}} \cdot (t_2^2 - t_1^2) \\
\Longleftarrow & \quad \{ \text{ assume } 0 < t_2 - t_1 \text{; divide both sides with } t_2 - t_1 \text{;} \\
 & \quad\;\; \text{identity } a^2 - b^2 = (a + b) \cdot (a - b) \ \} \\
 & 0 \leq v_{\mathrm{f}} + \frac{1}{2} \cdot a_{\mathrm{f}} \cdot (t_2 + t_1) \\
\Longleftarrow & \quad \{ \text{ arithmetic; assumption } a_{\mathrm{f}} < 0 \ \} \\
 & t_1 + t_2 \leq -\frac{2 \cdot v_{\mathrm{f}}}{a_{\mathrm{f}}} \\
\Longleftarrow & \quad \{ \text{ arithmetic; (2.3) } \} \\
 & t_1 \leq t_{\mathrm{f}}^{\mathrm{stop}} \ \wedge\ t_2 \leq t_{\mathrm{f}}^{\mathrm{stop}} \\
\Longleftarrow & \quad \{ \text{ assumption both } t_1 \text{ and } t_2 \text{ in } [0; t_{\mathrm{f}}^{\mathrm{stop}}] \ \} \\
 & \mathsf{True}
\end{array}
$$

If both t_1 and t_2 belong to the set $[0, t_{\mathrm{f}}^{\mathrm{stop}}]$ and $t_1 \leq t_2$, we can deduce $s_{\mathrm{f}}(t_1) \leq s_{\mathrm{f}}(t_2)$ because $s_{\mathrm{f}}(t_1) = p_{\mathrm{f}}(t_1) \leq p_{\mathrm{f}}(t_2) = s_{\mathrm{f}}(t_2)$ according to (2.4) and the result that p_{f} is non-decreasing in $[0, t_{\mathrm{f}}^{\mathrm{stop}}]$ above. We omit the proof for the case that t_1 and t_2 straddle on adjacent sets as it is obvious that s_{f} is non-decreasing in this case. With this monotonicity, it is easy to see that $s_{\mathrm{f}}(t) \leq s_f(t_{\mathrm{f}}^{\mathrm{stop}})$ in case $t \leq t_{\mathrm{f}}^{\mathrm{stop}}$. If this is not the case, then we have $s_{\mathrm{f}}(t) = s_f(t_{\mathrm{f}}^{\mathrm{stop}}) \leq s_f(t_{\mathrm{f}}^{\mathrm{stop}})$ according to (2.4). □

One might be tempted to use the same model of movement for the ego vehicle, but the ego vehicle in reality cannot react instantaneously when the front vehicle performs an emergency brake. There will be a slight time delay until the ego vehicle — be it an

[1] Note that Assumption 2.2 specifies the value of a_{f} to be negative.

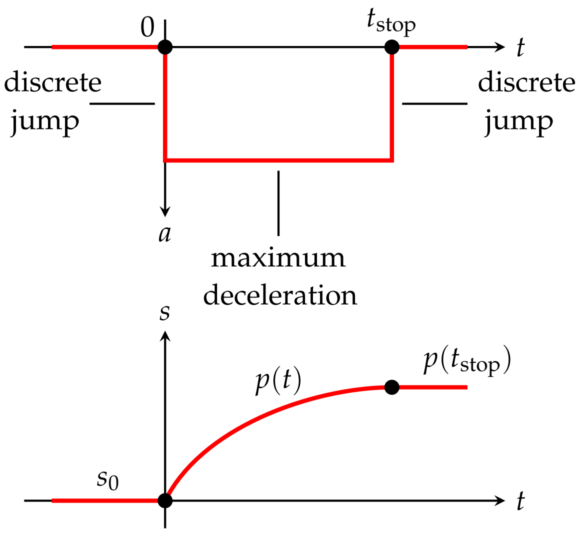


Figure 2.2: A typical plot of deceleration and overall movement of the front vehicle over time.

automatic vehicle or a human driver — perceives the situation and reacts accordingly by performing an emergency brake. To cater for this phenomenon, the overall movement of the ego vehicle is extended from the overall movement of the front vehicle by parameterising (2.4) with a reaction time delay δ as follows:

$$s_{\mathrm{e}}(t) := \begin{cases} s_{\mathrm{e}}^0 & \text{if } t \le 0 \ , \\ s_{\mathrm{e}}^0 + v_{\mathrm{e}} \cdot t & \text{if } 0 \le t \le \delta \ , \\ p_{\mathrm{e}}\,(t - \delta) & \text{if } \delta \le t \le \delta + t_{\mathrm{e}}^{\text{stop}} \ , \\ p_{\mathrm{e}}\,(t_{\mathrm{e}}^{\text{stop}}) & \text{if } \delta + t_{\mathrm{e}}^{\text{stop}} \le t \ , \end{cases} \tag{2.6}$$

After $t_{\mathrm{e}}^{\text{stop}}$, the ego vehicle will be positioned at:

$$s_{\mathrm{e}}^{\text{stop}} := s_{\mathrm{e}}\,(t_{\mathrm{e}}^{\text{stop}}) = s_{\mathrm{e}}^0 + v_{\mathrm{e}} \cdot \delta - \frac{v_{\mathrm{e}}^2}{2 \cdot a_{\mathrm{e}}} \tag{2.7}$$

Similarly to Lemma 2.1, the property of monotonicty and maximum at stopping time also hold for the movement of the ego vehicle.

Lemma 2.2. $t_1 \le t_2 \implies s_e\,(t_1) \le s_e\,(t_2)$ ***and*** $s_e\,(t) \le s_e\,(t_e^{stop})$

The proof for this lemma is similar to that of Lemma 2.1 and hence we omit it here. Note that (2.6) is sensible only for non-negative reaction time. One might argue that this is, of course, a self-evident truth. However, this is part of the modelling of a real world phenomenon which we cannot proved in ISABELLE. Thus, we state it as an assumption as follows:

Assumption 2.4. *Reaction time δ is positive, that is, $0 < \delta$.*

According to these two movement models, a collision will occur if we can find a future time t such that $s_{\mathrm{e}}\, t = s_{\mathrm{o}}\, t$. To generalise this predicate, we define *collision* over a set of real numbers $T :: (\mathbb{R})\mathit{set}$ as follows:

$$\mathit{collision}\,(T) \;:=\; (\exists t \in T.\; s_{\mathrm{e}}(t) = s_{\mathrm{o}}(t)) \;\; . \tag{2.8}$$

Equipped with these definitions and assumptions, the objective of this chapter — finding a sound and complete safe distance expression — can be formalised as finding a term *safe-distance* such that[2]:

$$s_{\mathrm{f}}^{0} - s_{\mathrm{e}}^{0} > \mathit{safe\text{-}distance} \quad \Longrightarrow \quad \neg\mathit{collision}\,[0;\infty) \;\; , \qquad \text{SOUNDNESS}$$

$$\neg\mathit{collision}\,[0;\infty) \quad \Longrightarrow \quad s_{\mathrm{f}}^{0} - s_{\mathrm{e}}^{0} > \mathit{safe\text{-}distance} \;\; . \qquad \text{COMPLETENESS}$$

In principle, we can use quantifier elimination techniques for real arithmetic [35, 36, 37] in order to obtain the safe distance expression such as those implemented in modern computer algebra systems (CASs); there is even a proof-producing procedure implemented in the HOL-Light theorem prover [38]. However, our eight-variable formula appears to be too complex for HOL-Light's quantifier elimination procedure. Therefore, we have to find this expression manually with an interactive theorem prover (Sec. 2.2). This makes the results more robust against changes in the formalisation and more readable compared to those from CASs'.

2.2 Logical analysis of the safe distance problem

A convenient way to find *safe-distance* is by overlaying two plots of vehicle's movement in the same graph (see Fig. 2.3) and deduce graphically, and mathematically, whether there is a collision, a collision freedom, or neither. For example, if we assume that the ego's stopping position lies below the front vehicle's initial position, we can intuitively see that there will be no collision. Likewise, if the ego's stopping position lies above the front vehicle's stopping position, there must be a collision. From these two observations, we can conjecture that certain configurations of stopping positions might cause collision. We therefore need to analyse all possible configurations and, in case of collision freedom, hope that the deduction can fit to the pattern in the soundness condition. That is properly moving terms s_{f}^{0} and s_{e}^{0} to the LHS and remaining terms to the RHS of the inequality; whatever remains in the RHS must be the safe distance.

[2] Note that, for readability purpose, the parentheses surrounding the argument of *collision* are omitted when the argument is an interval.

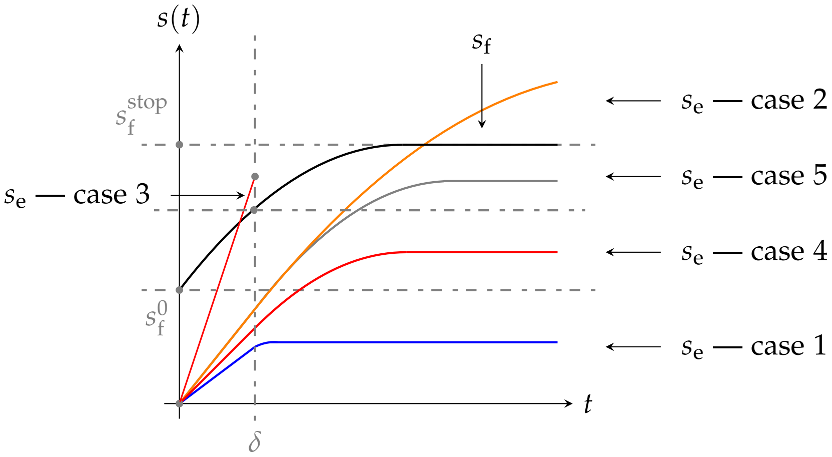


Figure 2.3: Overlay of plots between a fixed movement of the front vehicle (black) and different movements of the ego vehicle (coloured).

We can group all possible configurations of stopping positions into five cases (see Fig. 2.3). The first (second) case is when ego's stopping position is below (above) the initial (stopping) position of the front vehicle. The rest of the cases are when ego's stopping position is in between the initial and the stopping position of the front vehicle. One delineating factor for the remaining three cases is whether at $t = \delta$ the position of the ego vehicle has already been on the above of that of the front vehicle; intuitively this means a collision must have already happened (case three). When this is not the case, we can ignore what has happened in $[0, \delta)$ and focus solely on what could happen in $[\delta, \infty)$. Inspired by case one, case four (five) is characterised by the condition the stopping position of the ego vehicle is lower (higher) than the position of the front vehicle at $t = \delta$.

Theorem 2.1 (All possible cases for safe distance problem.).

$$
\begin{aligned}
& s_e^{stop} < s_f^0 && \text{(CASE 1)}\\
\vee\ & s_f^{stop} \leq s_e^{stop} && \text{(CASE 2)}\\
\vee\ & s_f^0 \leq s_e^{stop} < s_f^{stop} \ \wedge\ s_f(\delta) \leq s_e(\delta) && \text{(CASE 3)}\\
\vee\ & s_f^0 \leq s_e^{stop} < s_f^{stop} \ \wedge\ s_e(\delta) < s_f(\delta) \ \wedge\ s_e^{stop} < s_f(\delta) && \text{(CASE 4)}\\
\vee\ & s_f^0 \leq s_e^{stop} < s_f^{stop} \ \wedge\ s_e(\delta) < s_f(\delta) \ \wedge\ s_f(\delta) \leq s_e^{stop} && \text{(CASE 5)}
\end{aligned}
$$

We can see intuitively that there is no collision in CASE 1. This is formalised by the following theorem.

Theorem 2.2 (CASE 1). $s_e^{stop} < s_f^0 \implies \neg \mathit{collision}\,[0, \infty)$

Proof. This is true because $s_{\mathrm{e}}\,(t) < s_{\mathrm{f}}\,(t)$ holds for every time $t \geq 0$. That is, $s_{\mathrm{e}}\,(t) \leq s_{\mathrm{e}}\,(t_{\mathrm{e}}^{\mathrm{stop}}) = s_{\mathrm{e}}^{\mathrm{stop}} < s_{\mathrm{f}}^{0} = s_{\mathrm{f}}\,(0) \leq s_{\mathrm{f}}\,(t)$ due to transitivity, the premise, monotonicity of s_{f} (s_{e}), and the maximum at the stopping time Lemma 2.1 (2.2). □

The Intermediate Value Theorem (IVT) — readily available in the theory of Analysis in ISABELLE's library — will be helpful for the next case.

Lemma 2.3 (Intermediate Value Theorem). *Suppose that the predicate* $\mathit{isCont}(f, x)$ *evaluates to true for function* f *that is continuous at* x*, then the following property holds.*

$$f(a) \leq y \implies y \leq f(b) \implies a \leq b \implies (\forall x.\, a \leq x \wedge x \leq b \longrightarrow \mathit{isCont}\,(f, x) \implies \exists x.\, a \leq x \wedge x \leq b \wedge f(x) = y$$

Equipped with this lemma, we can prove the following theorem which asserts the occurrence of collision in CASE 2.

Theorem 2.3 (CASE 2). $s_f^{stop} \leq s_e^{stop} \implies \mathit{collision}\,[0, \infty)$.

Proof. We first define a time $b := \max\{t_{\mathrm{e}}^{\mathrm{stop}},\, t_{\mathrm{f}}^{\mathrm{stop}}\}$ in which both vehicles are guaranteed to stop already, and a function $f\, ::\, \mathbb{R} \Rightarrow \mathbb{R}$ where $f := \lambda\tau.\, s_{\mathrm{e}}(\tau) - s_{\mathrm{f}}(\tau)$ which represents the difference between the positions of the ego and the front vehicle. Since we have

1. $f(0) \leq 0$ due to Assumption 2.3;
2. $0 \leq f(b)$ due to premise $s_{\mathrm{f}}^{\mathrm{stop}} \leq s_{\mathrm{e}}^{\mathrm{stop}}$;
3. $0 \leq b$ due to the definition of stopping time in Eq. (2.3), Assumptions 2.1 and 2.2;
4. $\mathit{isCont}(f, x)$ for $x \in [0, b]$ because the difference of two continuous functions is also continuous;

we can deduce $\mathit{collision}\,[0, b]$ with IVT in Lemma 2.3 by substituting a with 0; hence $\mathit{collision}\,[0, \infty)$ since $[0, b] \subseteq [0, \infty)$. □

This theorem confirms our conjecture that, if at some point the position of the ego vehicle is above that of the front vehicle, there must have been a collision previously. In CASE 2, this time is either the stopping time of the ego or the front vehicle — depending on which one stops later. If we look at CASE 3, we can see that at $t = \delta$, this condition also holds and hence we can deduce collision in this case too.

Theorem 2.4 (CASE 3). $s_f^0 \leq s_e^{stop} < s_f^{stop} \implies s_f(\delta) \leq s_e(\delta) \implies \mathit{collision}\,[0, \infty)$

Proof. By defining $b := \delta$ and using the facts that $0 \leq f(b)$, due to the premise $s_\mathrm{f}(\delta) \leq s_\mathrm{e}(\delta)$, and $0 \leq b$ due to Assumption 2.4, we can deduce $\mathit{collision}\,[0, \infty)$ by using the same proof template in Theorem 2.3. □

Figure 2.3 has a horizontal dashed line at $s_\mathrm{f}(\delta)$ to separate CASE 4 and 5. If we focus on the time where $\delta \leq t$, the behaviour of the ego vehicle in CASE 4 becomes similar to that in CASE 1. Indeed, we can deduce collision freedom in this case too; the following lemma will be helpful for proving collision freedom in CASE 4.

Lemma 2.4 (Upward opening quadratic function)**.** *Suppose that we have a quadratic function p :: $\mathbb{R} \Rightarrow \mathbb{R}$ defined as $f = \lambda x.\ a \cdot x^2 + b \cdot x + c$. If we can find x, y, and z such that $f(x) > f(y)$ but $f(y) \leq f(z)$, then a must be positive.*

$$\exists x\, y\, z.\ x < y \wedge y < z\ \wedge\ f(x) > f(y)\ \wedge\ f(y) \leq f(z) \implies a > 0$$

The proof for this lemma is omitted since it can be proved automatically with SOS (sum-of-squares) technique available in ISABELLE. This is true intuitively because, if p opens upward, — $p(x) > p(y)$ but $p(y) \leq p(z)$ for $x < y < z$ — then the first derivative will change from negative to positive; this requires the second derivative — that is $2 \cdot a$ — to be positive.

Theorem 2.5 (CASE 4)**.** $s_f^0 \leq s_e^{stop} < s_f^{stop} \implies s_e(\delta) < s_f(\delta) \implies s_e^{stop} < s_f(\delta) \implies \neg\, \mathit{collision}\,[0, \infty)$

Proof. Firstly, the time set $[0, \infty)$ is divided into three disjoint sets $[0, 0], (0, \delta)$ and $[\delta, \infty)$. From Assumption 2.3, we know that there is no collision at 0, i.e. $\neg \mathit{collision}\,[0, 0]$. By using the same proof template as in Theorem 2.2, we have $\neg\, \mathit{collision}\,[\delta, \infty)$. Hence, we only need to prove $\neg \mathit{collision}\,(0, \delta)$ which we are going to prove by contradiction.

Assume that $\mathit{collision}\,(0, \delta)$ is true. Then there is a t such that $0 < t \wedge t < \delta \wedge s_\mathrm{e}(t) = s_\mathrm{f}(t)$ which, after unfolding the movement according to (2.4) and (2.6), becomes $f(t) = 0$ with $f := \lambda x.\ \frac{1}{2} \cdot a_\mathrm{f} \cdot x^2 + (v_\mathrm{f} - v_\mathrm{e}) \cdot x + (s_\mathrm{f}^0 - s_\mathrm{e}^0)$. Since $f(0) > f(t)$ — by simple substitution and Assumption 2.3 — and $f(t) \leq f(\delta)$ — from premise $s_\mathrm{e}(\delta) < s_\mathrm{f}(\delta)$, deceleration a_f must be positive according to Lemma 2.4, which contradicts with Assumption 2.2. Thus, we have $\neg \mathit{collision}\,(0, \delta)$. □

Up until now, we can deduce either collision or collision freedom. CASE 5 unfortunately is more challenging because the premises are not strong enough to deduce collision or collision freedom. Before we prove the theorem for CASE 5, it will be easier for now to ignore the reaction time and reconsider this later. By ignoring reaction time, the

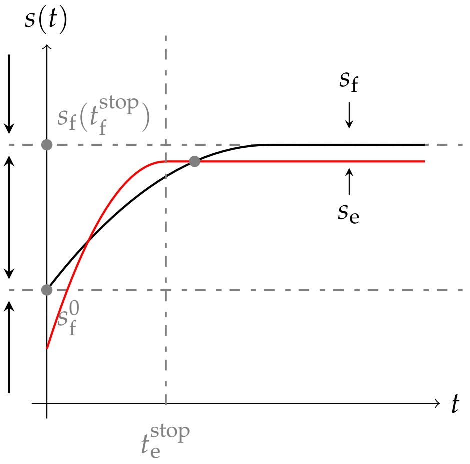


Figure 2.4: Case where the reaction time is ignored and the ego vehicle has stopped before the front vehicle.

movement of the ego vehicle becomes similar to that of the front vehicle (Eq. (2.4)):

$$\hat{s}_{\mathrm{e}}(t) := \begin{cases} s_{\mathrm{e}}^{0} & \text{if } t \le 0 \ , \\ p_{\mathrm{e}}(t) & \text{if } 0 \le t \le t_{\mathrm{e}}^{\mathrm{stop}} \ , \\ p_{\mathrm{e}}(t_{\mathrm{e}}^{\mathrm{stop}}) & \text{if } t_{\mathrm{e}}^{\mathrm{stop}} \le t \ . \end{cases} \tag{2.9}$$

After $t_{\mathrm{e}}^{\mathrm{stop}}$, the ego vehicle will be positioned at

$$\hat{s}_{\mathrm{e}}^{\mathrm{stop}} := \hat{s}_{\mathrm{e}}\left(t_{\mathrm{e}}^{\mathrm{stop}}\right) = s_{\mathrm{e}}^{0} - \frac{v_{\mathrm{e}}^{2}}{2 \cdot a_{\mathrm{e}}} \ ,$$

and the collision predicate is also slightly modified to

$$\textit{collision-no-react}\,(T) := (\exists t \in T.\ \hat{s}_{\mathrm{e}}(t) = s_{\mathrm{o}}(t)) \ .$$

One important property in CASE 5 is that a collision at time t will happen when, and only when, both vehicles have yet to stop, that is, $t \le \min\{t_{\mathrm{e}}^{\mathrm{stop}}, t_{\mathrm{f}}^{\mathrm{stop}}\}$. This is convenient because, in order to find a collision time, the movement of both vehicles s_{f} and $\hat{s}_{\mathrm{e}}$ in *collision-no-react* can be safely replaced by p_{f} and p_{e}, respectively.

Lemma 2.5. *In case $s_f^0 \le \hat{s}_e^{stop} < s_f^{stop}$ is true, a collision will happen before both vehicles stop.*

$$\textit{collision-no-react}\,[0,\infty) \iff \textit{collision-no-react}\,\left(0, \min\{t_e^{stop}, t_f^{stop}\}\right)$$

Proof. The 'if' part of the lemma is true because the predicate *collision-no-react* is 'monotonic' with respect to subset relation.

$$S \subseteq T \implies (\textit{collision-no-react}\ (S) \longrightarrow \textit{collision-no-react}\ (T))$$

As for the 'only if' part of the lemma, we first note that if a collision happens at time t while one of the vehicles has already stopped, then it must be the ego vehicle which has stopped ($t_{\mathrm{e}}^{\mathrm{stop}} < t$). Additionally, we have $s_{\mathrm{f}}(t_{\mathrm{e}}^{\mathrm{stop}}) \leq \hat{s}_{\mathrm{e}}(t_{\mathrm{e}}^{\mathrm{stop}})$ because *1)* $s_{\mathrm{f}}(t_{\mathrm{e}}^{\mathrm{stop}}) \leq s_{\mathrm{f}}(t)$ due to s_{f} being monotonic according to Lemma 2.1; *2)* $s_{\mathrm{f}}(t) = \hat{s}_{\mathrm{e}}(t)$ due to the definition of collision; and *3)* $\hat{s}_{\mathrm{e}}(t) = \hat{s}_{\mathrm{e}}(t_{\mathrm{e}}^{\mathrm{stop}})$ due to the previously deduced fact that the ego vehicle has already stopped (See Fig. 2.4).

By instantiating $f := \lambda x.\ \hat{s}_{\mathrm{e}}(x) - s_{\mathrm{f}}(x)$ with $a := 0, y := 0, b := t_{\mathrm{e}}^{\mathrm{stop}}$ in the Intermediate Value Theorem (Lemma 2.3), we are guaranteed to find a witness x where $f(x) = 0$ and $x \in [0, t_{\mathrm{e}}^{\mathrm{stop}}]$ since *1.)* $f(0) \leq 0$ due to Assumption 2.3; *2.)* $0 \leq f(t_{\mathrm{e}}^{\mathrm{stop}})$ due to the fact $s_{\mathrm{f}}(t_{\mathrm{e}}^{\mathrm{stop}}) \leq \hat{s}_{\mathrm{e}}(t_{\mathrm{e}}^{\mathrm{stop}})$ obtained previously; and *3.)* f is continuous in $[0, t_{\mathrm{e}}^{\mathrm{stop}}]$. Hence, by unfolding the definition of f and the facts that there is no collision at 0 and $t_{\mathrm{e}}^{\mathrm{stop}}$, we can deduce $\mathsf{collision\text{-}no\text{-}react}\,(0, \min\{t_{\mathrm{e}}^{\mathrm{stop}}, t_{\mathrm{f}}^{\mathrm{stop}}\})$. □

The following theorem characterises the condition for ensuring a collision in CASE 5 when the reaction time is ignored.

Theorem 2.6. *When the conditions $s_f^0 \leq s_e^{stop}$ and $s_e^{stop} \leq s_f^{stop}$ are satisfied, the following equality holds.*

$$\begin{aligned} &\mathsf{collision\text{-}no\text{-}react}\,[0, \infty) \quad \Longleftrightarrow \\ &\qquad a_f > a_e \quad \wedge \quad v_f < v_e \quad \wedge \quad s_f^0 - s_e^0 \leq \frac{(v_f - v_e)^2}{2 \cdot (a_f - a_e)} \quad \wedge \quad t_e^{stop} < t_f^{stop} \end{aligned} \tag{2.10}$$

Proof. For the 'if' part, condition on the RHS of the bi-implication ensures the existence of a root to $p_{\mathrm{f}} - p_{\mathrm{e}}$. That is, after unfolding the definition of p_{f} and p_{e} and rearranging the terms, we have quadratic equation $(s_{\mathrm{f}}^0 - s_{\mathrm{e}}^0) + (v_{\mathrm{f}} - v_{\mathrm{e}}) \cdot t + \frac{1}{2} \cdot (a_{\mathrm{f}} - a_{\mathrm{e}}) \cdot t^2 = 0$ whose discriminant $D := (v_{\mathrm{f}} - v_{\mathrm{e}})^2 - 2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}}) \cdot (s_{\mathrm{f}}^0 - s_{\mathrm{e}}^0)$ is guaranteed to be at least zero, $0 \leq D$ (see the first and the third conjunct in RHS). With Lemma 2.5, it is guaranteed that there must be a solution x which is at most $t_{\mathrm{e}}^{\mathrm{stop}}$ and $t_{\mathrm{f}}^{\mathrm{stop}}$. Hence $p_{\mathrm{e}}(x) = p_{\mathrm{f}}(x)$ implies $\hat{s}_{\mathrm{e}}\, x = s_{\mathrm{f}}\, x$ in which case x is the witness to $\mathsf{collision\text{-}no\text{-}react}\,[0, \infty)$.

For the 'only if' part, we know that from $\mathsf{collision\text{-}no\text{-}react}\,[0, \infty)$, we can obtain a root t with $s_{\mathrm{f}}(t) - \hat{s}_{\mathrm{e}}(t) = 0$. Then, Lemma 2.5 allows us to deduce $p_{\mathrm{f}}(t) - p_{\mathrm{e}}(t) = 0$ and Lemma 2.4 $\big($for $p_{\mathrm{o}} - p_{\mathrm{e}}$ at times $0 < t < \min\{t_{\mathrm{e}}^{\mathrm{stop}}, t_{\mathrm{f}}^{\mathrm{stop}}\}\big)$ yields $a_{\mathrm{f}} > a_{\mathrm{e}}$. This gives — together with the fact that the discriminant of $p_{\mathrm{f}} - p_{\mathrm{e}}$ is nonnegative — the remaining conjuncts after some arithmetic manipulations and reasoning. □

If we compare Fig. 2.3 and Fig. 2.4, we can see that the behaviours of both vehicles in CASE 5 after $t \geq \delta$ in the former figure is the same with those in the latter figure — with certain values modified properly. This is the main reason why we analyse CASE 5 without considering the reaction time first; it allows us to separate the analysis into two time sets: $[0, \delta)$ and $[\delta, \infty)$. The following theorem provides the condition for collision (freedom) in CASE 5.

Theorem 2.7 (CASE 5). *By defining*

$$\phi_f := \begin{cases} v_f + a_f \cdot \delta & \text{if } \delta \leq t_f^{stop} \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

and $\tau_f^{stop} := -\frac{\phi_f}{a_f}$, $\sigma_f^0 := s_f(\delta)$, *and* $\sigma_e^0 := s_e^0 + v_e \cdot \delta$, *we have*

$$s_f^0 \leq s_e^{stop} < s_f^{stop} \implies s_e(\delta) < s_f(\delta) \implies s_f(\delta) \leq s_e^{stop} \implies$$
$$\mathit{collision}\,[0, \infty) \iff \left(a_f > a_e \land \phi_f < v_e \land \sigma_f^0 - \sigma_e^0 \leq \frac{(\phi_f - v_e)^2}{2 \cdot (a_f - a_e)} \land t_e^{stop} < \tau_f^{stop} \right) .$$

Proof. For $[0, \delta)$, we can use the same reasoning as performed in case 4 to deduce $\neg\mathit{collision}\,[0, \delta)$. Hence $\mathit{collision}\,[0, \infty)$ if and only if $\mathit{collision}\,[\delta, \infty)$. If we assume that $\delta \leq t_\mathrm{f}^\mathrm{stop}$, then the speed of the front vehicle at $t = \delta$ is $\phi_\mathrm{f} = v_\mathrm{f} + a_\mathrm{f} \cdot \delta$ (obtained from the first derivative of p_f in (2.4)), and the position of the ego and the front vehicle are $\sigma_\mathrm{e}^0 = s_\mathrm{e}^0 + v_\mathrm{e} \cdot \delta$ (obtained from (2.6)) and $\sigma_\mathrm{f}^0 = s_\mathrm{f}(\delta)$ (obtained from (2.4)), respectively. By using Theorem 2.6, we obtain

$$\mathit{collision}\,[\delta, \infty) \iff \left(a_\mathrm{f} > a_\mathrm{e} \land \phi_\mathrm{f} < v_\mathrm{e} \land \sigma_\mathrm{f}^0 - \sigma_\mathrm{e}^0 \leq \frac{(\phi_\mathrm{f} - v_\mathrm{e})^2}{2 \cdot (a_\mathrm{f} - a_\mathrm{e})} \land t_\mathrm{e}^\mathrm{stop} < \tau_\mathrm{f}^\mathrm{stop} \right) ,$$

and hence the conclusion in this theorem. When $t_\mathrm{f}^\mathrm{stop} < \delta$, then the front vehicle has stopped already in $[\delta, \infty)$. By monotonicity in Lemma 2.2, we have for all $t \in [\delta, \infty)$, $s_\mathrm{e}(t) \leq s_\mathrm{e}^\mathrm{stop}$, $s_\mathrm{e}^\mathrm{stop} < s_\mathrm{f}^\mathrm{stop}$ (from the premise) and hence $\neg\mathit{collision}\,[\delta, \infty)$ if $t_\mathrm{f}^\mathrm{stop} < \delta$. Evaluating ϕ_f in Eq. (2.11) results in $\tau_\mathrm{f}^\mathrm{stop} = 0$ and the conjunct $t_\mathrm{e}^\mathrm{stop} < \tau_\mathrm{f}^\mathrm{stop}$ becomes false (because stopping time is positive according to Assumption 2.1 and 2.2). Hence, we have shown the conclusion in this theorem for $t_\mathrm{f}^\mathrm{stop} < \delta$. □

2.3 Designing a sound-and-complete checker

Table 2.1 summarises the logical analyses performed for each case specified in Theorem 2.1. As our first objective is to prove the SOUNDNESS of a checker, we will consider cases where collision freedom can be deduced only; the rest will be useful for proving the COMPLETENESS.

In CASE 1, we can rearrange the deduction by unfolding the definition of $s_\mathrm{e}^\mathrm{stop}$ in (2.7) as follows:

$$s_\mathrm{f}^0 - s_\mathrm{e}^0 \;>\; v_\mathrm{e} \cdot \delta - \frac{v_\mathrm{e}^2}{2 \cdot a_\mathrm{e}} \implies \neg\mathit{collision}\,[0; \infty) \; . \tag{2.12}$$

Table 2.1: Summary of whether collision (freedom) can be deduced in each case.

Case	Deduction	Reference
1	collision freedom	Theorem 2.2
2	collision	Theorem 2.3
3	collision	Theorem 2.4
4	collision freedom	Theorem 2.5
5	conditional	Theorem 2.7

Comparing this with the pattern in (SOUNDNESS), we obtain our first safe distance expression as follows:

$$\mathsf{safe\text{-}distance}_0 \ := \ v_\mathrm{e} \cdot \delta - \frac{v_\mathrm{e}^2}{2 \cdot a_\mathrm{e}} \ .$$

Similarly, we can also unify deduction in CASE 4 with the pattern in (SOUNDNESS). However, instead of one, we obtain three expressions for safe distance — the first is unified with premise $s_\mathrm{e}^\mathrm{stop} < s_\mathrm{f}^\mathrm{stop}$, the second with premise $s_\mathrm{e}(\delta) < s_\mathrm{f}(\delta)$, and the third with premise $s_\mathrm{e}^\mathrm{stop} < s_\mathrm{f}(\delta)$. Theorem 2.5 can be modified as follows (assuming that $\delta \le t_\mathrm{f}^\mathrm{stop}$):

$$s_\mathrm{f}^0 \le s_\mathrm{e}^\mathrm{stop} \implies s_\mathrm{f}^0 - s_\mathrm{e}^0 > \overbrace{v_\mathrm{e} \cdot \delta - \frac{v_\mathrm{e}^2}{2 \cdot a_\mathrm{e}} + \frac{v_\mathrm{f}^2}{2 \cdot a_\mathrm{f}}}^{\mathsf{safe\text{-}distance}_1} \implies s_\mathrm{f}^0 - s_\mathrm{e}^0 > \overbrace{(v_\mathrm{e} - v_\mathrm{f}) \cdot \delta - \frac{1}{2} \cdot a_\mathrm{f} \cdot \delta^2}^{\mathsf{safe\text{-}distance}_2}$$

$$\implies s_\mathrm{f}^0 - s_\mathrm{e}^0 > \underbrace{(v_\mathrm{e} - v_\mathrm{f}) \cdot \delta - \frac{1}{2} \cdot a_\mathrm{f} \cdot \delta^2 - \frac{v_\mathrm{e}^2}{2 \cdot a_\mathrm{e}}}_{\mathsf{safe\text{-}distance}_3} \implies \neg\, \mathsf{collision}\,[0;\infty) \ . \quad (2.13)$$

Although there are three safe distance expressions here, we know that one is bigger than the other two. We only prove $\mathsf{safe\text{-}distance}_3 \ge \mathsf{safe\text{-}distance}_1$ because the other is obvious.

Lemma 2.6. $\delta \le t_f^{stop} \implies \mathsf{safe\text{-}distance}_3 \ge \mathsf{safe\text{-}distance}_1$

Proof. In case $\delta = t_\mathrm{f}^\mathrm{stop}$, we can simply substitute this equality and, after a series of arithmetic manipulation, we obtain $\mathsf{safe\text{-}distance}_3 = \mathsf{safe\text{-}distance}_1$. The following calculation proves a stronger condition when $\delta < t_\mathrm{f}^\mathrm{stop}$ holds.

$$\mathsf{safe\text{-}distance}_3 > \mathsf{safe\text{-}distance}_1$$

$\Longleftarrow$ { unfolding the definition; arithmetic }

$$-v_\mathrm{f} \cdot \delta - \tfrac{1}{2} \cdot a_\mathrm{f} \cdot \delta^2 \ > \ \tfrac{v_\mathrm{f}^2}{2 \cdot a_\mathrm{f}}$$

$\Longleftarrow$ { breaking RHS; unfolding $t_\mathrm{f}^\mathrm{stop}$; arithmetic }

$$
\begin{array}{ll}
& -v_{\mathrm{f}} \cdot \delta - \frac{1}{2} \cdot a_{\mathrm{f}} \cdot \delta^2 \; > \; -v_{\mathrm{f}} \cdot t_{\mathrm{f}}^{\mathrm{stop}} - \frac{1}{2} \cdot a_{\mathrm{f}} \cdot t_{\mathrm{f}}^{\mathrm{stop}} \\
\Longleftarrow & \quad \{ \text{ regrouping } \} \\
& 0 > v_{\mathrm{f}} \cdot (\delta - t_{\mathrm{f}}^{\mathrm{stop}}) + \frac{1}{2} \cdot a_{\mathrm{f}} \cdot (\delta^2 - (t_{\mathrm{f}}^{\mathrm{stop}})^2) \\
\Longleftarrow & \quad \{ \text{ dividing both sides with } \delta - t_{\mathrm{f}}^{\mathrm{stop}}\text{; fact } \delta < t_{\mathrm{f}}^{\mathrm{stop}} \; \} \\
& 0 < v_{\mathrm{f}} + \frac{1}{2} \cdot a_{\mathrm{f}} \cdot (\delta + t_{\mathrm{f}}^{\mathrm{stop}}) \\
\Longleftarrow & \quad \{ \text{ unfolding } t_{\mathrm{f}}^{\mathrm{stop}}\text{; arithmetic } \} \\
& \delta < t_{\mathrm{f}}^{\mathrm{stop}} \\
\Longleftarrow & \quad \{ \text{ premise } \} \\
& \mathit{True}
\end{array}
$$

We have proved, in each case, a stronger condition which can be weakened to the conclusion $\mathit{safe\text{-}distance}_3 \geq \mathit{safe\text{-}distance}_1$. □

By using this lemma and the fact that $\mathit{safe\text{-}distance}_3 \geq \mathit{safe\text{-}distance}_1$, Theorem 2.5 can be simplified further into the following deduction.

$$
s_{\mathrm{f}}^0 \leq s_{\mathrm{e}}^{\mathrm{stop}} \implies \delta \leq t_{\mathrm{f}}^{\mathrm{stop}} \implies s_{\mathrm{f}}^0 - s_{\mathrm{e}}^0 > \mathit{safe\text{-}distance}_3 \implies \neg\, \mathit{collision}\,[0;\infty) \quad (2.14)
$$

Together with the deduction in (2.12), we obtain the following temporary version of our checker.

```
$\mathit{checker}_0$ :=
        let $\mathit{dist} = s_{0,\mathrm{o}} - s_{0,\mathrm{e}}$ in
        if $\underline{\mathit{dist} \; > \; \mathit{safe\text{-}distance}}_0$ orelse $(\delta \leq t_{\mathrm{f}}^{\mathrm{stop}} \wedge \underline{\mathit{dist} \; > \; \mathit{safe\text{-}distance}}_3)$ then $\mathit{True}$
        else $\mathit{undefined}$
```

Lemma 2.7. *$\mathit{checker}_0$ is sound.*

Proof. This checker is sound because, if it returns *True*, then it is either $\mathit{dist} > \mathit{safe\text{-}distance}_0$ or $\mathit{dist} \leq \mathit{safe\text{-}distance}_0$ and $\delta \leq t_{\mathrm{f}}^{\mathrm{stop}} \wedge \mathit{dist} > \mathit{safe\text{-}distance}_3$. For the former case, it is guaranteed to be sound due to (2.12), and the latter due to (2.14) — $\mathit{dist} \leq \mathit{safe\text{-}distance}_0$ is the same with the first premise. □

For CASE 5, we negate both sides of the conclusion in the deduction as follows:

$$
s_{\mathrm{f}}^0 \leq s_{\mathrm{e}}^{\mathrm{stop}} < s_{\mathrm{f}}^{\mathrm{stop}} \implies s_{\mathrm{e}}(\delta) < s_{\mathrm{f}}(\delta) \implies s_{\mathrm{f}}(\delta) \leq s_{\mathrm{e}}^{\mathrm{stop}} \implies
$$
$$
\neg \left(a_{\mathrm{f}} > a_{\mathrm{e}} \wedge \phi_{\mathrm{f}} < v_{\mathrm{e}} \wedge \sigma_{\mathrm{f}}^0 - \sigma_{\mathrm{e}}^0 \leq \frac{(\phi_{\mathrm{f}} - v_{\mathrm{e}})^2}{2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}})} \wedge t_{\mathrm{e}}^{\mathrm{stop}} < \tau_{\mathrm{f}}^{\mathrm{stop}} \right) \iff \neg \mathit{collision}\,[0;\infty) \ ,
$$

and apply De Morgan's rule resulting in

$$s_{\mathrm{f}}^{0} \leq s_{\mathrm{e}}^{\mathrm{stop}} < s_{\mathrm{f}}^{\mathrm{stop}} \implies s_{\mathrm{e}}(\delta) < s_{\mathrm{f}}(\delta) \implies s_{\mathrm{f}}(\delta) \leq s_{\mathrm{e}}^{\mathrm{stop}} \implies \sigma_{\mathrm{f}}^{0} - \sigma_{\mathrm{e}}^{0} > \frac{(\phi_{\mathrm{f}} - v_{\mathrm{e}})^2}{2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}})}$$
$$\implies \left(a_{\mathrm{f}} > a_{\mathrm{e}} \wedge \phi_{\mathrm{f}} < v_{\mathrm{e}} \wedge t_{\mathrm{e}}^{\mathrm{stop}} < \tau_{\mathrm{f}}^{\mathrm{stop}}\right) \implies \neg \mathsf{collision}\,[0;\infty) \ .$$

Unfolding the premise $\sigma_{\mathrm{f}}^{0} - \sigma_{\mathrm{e}}^{0} > \frac{(\phi_{\mathrm{f}} - v_{\mathrm{e}})^2}{2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}})}$ and unifying it with the pattern in SOUNDNESS results in the following safe distance expression (assuming $\delta \leq t_{\mathrm{f}}^{\mathrm{stop}}$):

$$\mathsf{safe\text{-}distance}_4 := \frac{(v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta - v_{\mathrm{e}})^2}{2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}})} + (v_{\mathrm{e}} - v_{\mathrm{f}}) \cdot \delta - \frac{1}{2} \cdot a_{\mathrm{f}} \cdot \delta^2 \ .$$

Note that it is also possible to find unification with the term $s_{\mathrm{e}}^{\mathrm{stop}} < s_{\mathrm{f}}^{\mathrm{stop}}$ and $s_{\mathrm{e}}(\delta) < s_{\mathrm{f}}(\delta)$, but these are $\mathsf{safe\text{-}distance}_1$ and $\mathsf{safe\text{-}distance}_2$, respectively. By assuming $a_{\mathrm{f}} > a_{\mathrm{e}}$, it is very easy to see that $\mathsf{safe\text{-}distance}_4 \geq \mathsf{safe\text{-}distance}_2$ due to the following equality:

$$\mathsf{safe\text{-}distance}_4 = \mathsf{safe\text{-}distance}_2 + \frac{(v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta - v_{\mathrm{e}})^2}{2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}})} \ .$$

Lemma 2.8. $a_f > a_e \implies \mathsf{safe\text{-}distance}_4 \geq \mathsf{safe\text{-}distance}_1$

Proof.

$$\begin{aligned}
& \mathsf{safe\text{-}distance}_1 \leq \mathsf{safe\text{-}distance}_4 \\
\Longleftarrow \quad & \{ \text{ unfolding definitions; multiplying both sides with } 2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}}) \ \} \\
& v_{\mathrm{e}} \cdot \delta \cdot 2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}}) - \tfrac{v_{\mathrm{e}}^2}{a_{\mathrm{e}}} \cdot (a_{\mathrm{f}} - a_{\mathrm{e}}) + \tfrac{v_{\mathrm{f}}^2}{a_{\mathrm{f}}} \cdot (a_{\mathrm{f}} - a_{\mathrm{e}}) \ \leq \\
& \qquad (v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta - v_{\mathrm{e}})^2 + (v_{\mathrm{e}} - v_{\mathrm{f}}) \cdot \delta \cdot 2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}}) - a_{\mathrm{f}} \cdot \delta^2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}}) \\
\Longleftarrow \quad & \{ \text{ arithmetic } \} \\
& \left(-\tfrac{v_{\mathrm{e}}^2 \cdot a_{\mathrm{f}}}{a_{\mathrm{e}}} - \tfrac{v_{\mathrm{f}}^2 \cdot a_{\mathrm{e}}}{a_{\mathrm{f}}}\right) \cdot (a_{\mathrm{e}} \cdot a_{\mathrm{f}}) \ \leq \\
& \qquad (a_{\mathrm{f}} \cdot a_{\mathrm{e}} \cdot \delta^2 - 2 \cdot v_{\mathrm{e}} \cdot v_{\mathrm{f}} - 2 \cdot a_{\mathrm{f}} \cdot \delta \cdot v_{\mathrm{e}} + 2 \cdot a_{\mathrm{e}} \cdot \delta \cdot v_{\mathrm{f}}) \cdot (a_{\mathrm{e}} \cdot a_{\mathrm{f}}) \\
\Longleftarrow \quad & \{ \text{ arithmetic } \} \\
& 0 \leq (-v_{\mathrm{e}} \cdot a_{\mathrm{f}} + v_{\mathrm{f}} \cdot a_{\mathrm{e}} + a_{\mathrm{f}} \cdot a_{\mathrm{e}} \cdot \delta)^2 \\
\Longleftarrow \quad & \{ \text{ property } 0 \leq x^2 \ \} \\
& \mathsf{True}
\end{aligned}$$

□

With these two facts, we can simplify (and weaken) the deduction in CASE 5 into the following deduction:

$$s_{\mathrm{f}}^{0} \leq s_{\mathrm{e}}^{\mathrm{stop}} \implies s_{\mathrm{f}}(\delta) \leq s_{\mathrm{e}}^{\mathrm{stop}} \implies \delta \leq t_{\mathrm{f}}^{\mathrm{stop}} \implies s_{\mathrm{f}}^{0} - s_{\mathrm{e}}^{0} > \mathsf{safe\text{-}distance}_4 \implies$$
$$\left(a_{\mathrm{f}} > a_{\mathrm{e}} \wedge \phi_{\mathrm{f}} < v_{\mathrm{e}} \wedge t_{\mathrm{e}}^{\mathrm{stop}} < \tau_{\mathrm{f}}^{\mathrm{stop}}\right) \implies \neg \mathsf{collision}\,[0;\infty) \ . \quad (2.16)$$

Finally, we can design the whole checker as follows:

```
checker :=
    let $\mathit{dist} = s_{\mathrm{f}}^{0} - s_{\mathrm{e}}^{0}$ in
    if $\underline{\mathit{dist} > \mathit{safe\text{-}distance}_0}$ orelse $(\delta \le \tau_{\mathrm{f}}^{\mathrm{stop}} \wedge \underline{\mathit{dist} > \mathit{safe\text{-}distance}_3})$ then True
    elseif $a_{\mathrm{f}} > a_{\mathrm{e}} \ \wedge\ \phi_{\mathrm{f}} < v_{\mathrm{e}} \ \wedge\ t_{\mathrm{e}}^{\mathrm{stop}} < \tau_{\mathrm{f}}^{\mathrm{stop}}$ then
      $\underline{\mathit{dist} > \mathit{safe\text{-}distance}_4}$  else  $\underline{\mathit{dist} > \mathit{safe\text{-}distance}_1}$
```

We prove the soundness and the completeness of the checker (with its supporting lemmas) as follow:

Lemma 2.9. *If $(a_f > a_e \ \wedge\ \phi_f < v_e \ \wedge\ t_e^{stop} < \tau_f^{stop})$ is false, then the following deduction holds.*

$$\delta \le \tau_f^{stop} \implies \phi_f < v_e \implies 0 \le \left(-\frac{v_e^2}{2 \cdot a_e} + \frac{(v_f + a_f \cdot \delta)^2}{2 \cdot a_f} \right)$$

Proof. We prove this lemma by cases: $t_{\mathrm{e}}^{\mathrm{stop}} < \tau_{\mathrm{f}}^{\mathrm{stop}}$ and $t_{\mathrm{e}}^{\mathrm{stop}} \ge \tau_{\mathrm{f}}^{\mathrm{stop}}$. The following calculation is the proof for the first case.

$$0 \le \left(-\tfrac{v_{\mathrm{e}}^2}{2 \cdot a_{\mathrm{e}}} + \tfrac{(v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta)^2}{2 \cdot a_{\mathrm{f}}} \right)$$

$\Longleftarrow$ { arithmetic }

$$(v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta)^2 \cdot \left(-\tfrac{1}{a_{\mathrm{f}}} \right) \ \le\ v_{\mathrm{e}}^2 \cdot \left(-\tfrac{1}{a_{\mathrm{e}}} \right)$$

$\Longleftarrow$ { monotonicity of multiplication; Assumption 2.1 and 2.2 }

$$(v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta)^2 \le v_{\mathrm{e}}^2 \ \wedge\ \left(-\tfrac{1}{a_{\mathrm{f}}} \right) \le \left(-\tfrac{1}{a_{\mathrm{e}}} \right)$$

$\Longleftarrow$ { case $t_{\mathrm{e}}^{\mathrm{stop}} < \tau_{\mathrm{f}}^{\mathrm{stop}}$, premise $\phi_{\mathrm{f}} < v_{\mathrm{e}}$, assumption in this lemma (for the second conjunct) }

$$(v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta)^2 \le v_{\mathrm{e}}^2 \ \wedge\ \mathit{True}$$

$\Longleftarrow$ { square root on both sides of the first conjunct }

$$(v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta) \le v_{\mathrm{e}} \ \wedge\ 0 \le (v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta)$$

$\Longleftarrow$ { premise $\phi_{\mathrm{f}} < v_{\mathrm{e}}$ and unfolding ϕ_{f} }

$$0 \le (v_{\mathrm{f}} + a_{\mathrm{f}} \cdot \delta)$$

$\Longleftarrow$ { dividing both sides with $-a_{\mathrm{f}}$, unfolding $\tau_{\mathrm{f}}^{\mathrm{stop}}$, assumption $0 \le \delta$ and premise $\delta \le \tau_{\mathrm{f}}^{\mathrm{stop}}$ }

True

The following calculation proves the second case.

$$0 \le \left(-\frac{v_\mathrm{e}^2}{2\cdot a_\mathrm{e}} + \frac{(v_\mathrm{f}+a_\mathrm{f}\cdot\delta)^2}{2\cdot a_\mathrm{f}}\right)$$

$$\Longleftarrow \quad \{\ \text{arithmetic}\ \}$$

$$-\frac{v_\mathrm{f}+a_\mathrm{f}\cdot\delta}{a_\mathrm{f}} \times (v_\mathrm{f} + a_\mathrm{f}\cdot\delta) \le -\frac{v_\mathrm{e}}{a_\mathrm{e}} \times v_\mathrm{e}$$

$$\Longleftarrow \quad \{\ \text{monotonicity of multiplication}\ \}$$

$$-\frac{v_\mathrm{f}+a_\mathrm{f}\cdot\delta}{a_\mathrm{f}} \le -\frac{v_\mathrm{e}}{a_\mathrm{e}} \ \wedge\ (v_\mathrm{f} + a_\mathrm{f}\cdot\delta) \le v_\mathrm{e} \ \wedge\ 0 \le v_\mathrm{f} + a_\mathrm{f}\cdot\delta$$

$$\Longleftarrow \quad \{\ \text{premise } \phi_\mathrm{f} < v_\mathrm{e} \text{ and last step in previous calculation}\ \}$$

$$-\frac{v_\mathrm{f}+a_\mathrm{f}\cdot\delta}{a_\mathrm{f}} \le -\frac{v_\mathrm{e}}{a_\mathrm{e}} \ \wedge\ \mathsf{True} \ \wedge\ \mathsf{True}$$

$$\Longleftarrow \quad \{\ \text{folding } t_\mathrm{e}^\mathrm{stop} \text{ and } \tau_\mathrm{f}^\mathrm{stop}\ \}$$

$$\tau_\mathrm{f}^\mathrm{stop} \le t_\mathrm{e}^\mathrm{stop}$$

$$\Longleftarrow \quad \{\ \text{condition in this case}\ \}$$

$$\mathsf{True}$$

□

Lemma 2.10. *If $(a_f > a_e \ \wedge\ \phi_f < v_e \ \wedge\ t_e^{stop} < \tau_f^{stop})$ is false, then the following deduction holds.*

$$\delta \le \tau_f^{stop} \implies s_f^0 - s_e^0 > \mathsf{safe\text{-}distance}_1 \implies \phi_f < v_e \implies s_e(\delta) < s_f(\delta)$$

Proof.

$$\mathsf{True}$$

$$\Longrightarrow \quad \{\ \text{arithmetic}\ \}$$

$$\mathsf{safe\text{-}distance}_1 = \mathsf{safe\text{-}distance}_2 + \left(-\frac{v_\mathrm{e}^2}{2\cdot a_\mathrm{e}} + \frac{(v_\mathrm{f}+a_\mathrm{f}\cdot\delta)^2}{2\cdot a_\mathrm{f}}\right)$$

$$\Longrightarrow \quad \{\ \text{premise } \delta \le \tau_\mathrm{f}^\mathrm{stop}\text{, case } \phi_\mathrm{f} < v_\mathrm{e}\text{, assumption in this lemma, and Lemma 2.9}\ \}$$

$$\mathsf{safe\text{-}distance}_1 \ge \mathsf{safe\text{-}distance}_2$$

$$\Longrightarrow \quad \{\ \text{premise } s_\mathrm{f}^0 - s_\mathrm{e}^0 > \mathsf{safe\text{-}distance}_1\text{; transitivity}\ \}$$

$$s_\mathrm{f}^0 - s_\mathrm{e}^0 > \mathsf{safe\text{-}distance}_2$$

$$\Longrightarrow \quad \{\ \text{folding } s_\mathrm{e} \text{ and } s_\mathrm{f} \text{ with } \delta \le \tau_\mathrm{f}^\mathrm{stop}\ \}$$

$$s_\mathrm{e}(\delta) < s_\mathrm{f}(\delta)$$

□

Lemma 2.11. *If $(a_f > a_e \ \wedge\ \phi_f < v_e \ \wedge\ t_e^{stop} < \tau_f^{stop})$ is false, then the following deduction holds.*

$$\delta \le \tau_f^{stop} \implies s_f^0 - s_e^0 > \mathsf{safe\text{-}distance}_1 \implies v_e \le \phi_f \implies s_e(\delta) < s_f(\delta)$$

Proof.

$$\begin{aligned}
& v_\mathrm{e} \leq \phi_\mathrm{f} \\
\Longrightarrow \quad & \{ \text{ multiplying both sides with } \delta \text{ and } 0 \leq \delta \text{ from Assumption 2.4;} \\
& \quad \text{unfolding } \phi_\mathrm{f} \ \} \\
& v_\mathrm{e} \cdot \delta \leq v_\mathrm{f} \cdot \delta + a_\mathrm{f} \cdot \delta^2 \\
\Longrightarrow \quad & \{ \text{ adding } s_\mathrm{e}^0 \text{ to LHS and } s_\mathrm{f}^0 \text{ on RHS with } s_\mathrm{e}^0 < s_\mathrm{f}^0 \text{ according to} \\
& \quad \text{Assumption 2.3 } \} \\
& s_\mathrm{e}^0 + v_\mathrm{e} \cdot \delta \ \leq \ s_\mathrm{f}^0 + v_\mathrm{f} \cdot \delta + a_\mathrm{f} \cdot \delta^2 \\
\Longrightarrow \quad & \{ \ a_\mathrm{f} \cdot \delta^2 < \tfrac{1}{2} \cdot a_\mathrm{f} \cdot \delta^2 \text{ because } a_\mathrm{f} < 0 \wedge 0 < \delta \text{ according to Assumption 2.2} \\
& \quad \text{and 2.4 } \} \\
& s_\mathrm{e}^0 + v_\mathrm{e} \cdot \delta \ < \ s_\mathrm{f}^0 + v_\mathrm{f} \cdot \delta + \tfrac{1}{2} \cdot a_\mathrm{f} \cdot \delta^2 \\
\Longrightarrow \quad & \{ \text{ folding } s_\mathrm{e} \text{ and } s_\mathrm{f} \text{ with } \delta \leq \tau_\mathrm{f}^\mathrm{stop} \ \} \\
& s_\mathrm{e}(\delta) < s_\mathrm{f}(\delta)
\end{aligned}$$

□

Theorem 2.8. *checker is sound.*

Proof. The *checker* is equal to *True* only if either of the following three cases is satisfied.

1. The condition in the **if** is true.
   The soundness argument for this case is the same with that in Lemma 2.7.

2. The condition in the **elseif** is true and *dist* $>$ *safe-distance*$_4$.
   In this case, the deduction in (2.16) ensures the soundness. Most of the premises can be seen to be true immediately; we only have to show that the first three premises hold. The first and the second premise hold because the first is equal $\neg($*dist* $>$ *safe-distance*$_0)$ and the second is equal to $\neg($*dist* $>$ *safe-distance*$_3)$ — the condition in the **elseif** is evaluated only if the condition in **if** is false. The premise $\delta \leq t_\mathrm{e}^\mathrm{stop}$ must be true because otherwise $\neg(t_\mathrm{e}^\mathrm{stop} < \tau_\mathrm{f}^\mathrm{stop})$.

3. The condition in the **elseif** is false and *dist* $>$ *safe-distance*$_1$.
   The soundness is based on Theorem 2.7. Since the condition in the **elseif** is false, there will be no collision provided that the premises in Theorem 2.7 hold. Premise $s_\mathrm{f}^0 \leq s_\mathrm{e}^\mathrm{stop}$ and $s_\mathrm{f}\,\delta \leq s_\mathrm{e}^\mathrm{stop}$ hold with the same reasoning as in the previous case. Premise $s_\mathrm{e}^\mathrm{stop} \leq s_\mathrm{f}^\mathrm{stop}$ is equal to *dist* $>$ *safe-distance*$_1$ — which is ensured in this case, while $s_\mathrm{e}(\delta) < s_\mathrm{f}(\delta)$ is proved by distinguishing two sub-cases: $\tau_\mathrm{f}^\mathrm{stop} < \delta$ or $\delta \leq \tau_\mathrm{f}^\mathrm{stop}$. For first sub-case, the goal $s_\mathrm{e}(\delta) < s_\mathrm{f}(\delta)$ is equivalent to $s_\mathrm{f}^0 - s_\mathrm{e}^0 > v_\mathrm{e} \cdot \delta + \frac{v_\mathrm{f}^2}{2 \cdot a_\mathrm{f}}$, which is true because

$$s_\mathrm{f}^0 - s_\mathrm{e}^0 \ > \ \textit{safe-distance}_1 \ = \ v_\mathrm{e} \cdot \delta - \frac{v_\mathrm{e}^2}{2 \cdot a_\mathrm{e}} + \frac{v_\mathrm{f}^2}{2 \cdot a_\mathrm{f}} \ \geq \ v_\mathrm{e} \cdot \delta + \frac{v_\mathrm{f}^2}{2 \cdot a_\mathrm{f}} \ .$$

In the second sub-case, the goal $s_{\mathrm{e}}(\delta) < s_{\mathrm{f}}(\delta)$ is also true because of Lemma 2.11 and 2.10. □

Theorem 2.9. *checker is complete.*

Proof. We prove this theorem with proof-by-contraposition technique. That is, we assume that the checker is equal to *False* and deduce collision thereafter. If the condition in the **if** is true, it will always be equal to *True*. Since we assume that the checker returns *False*, this cannot be the case:

$$\mathit{dist} \le \mathit{safe\text{-}distance}_0 \quad \wedge \quad (\delta \le \tau_{\mathrm{f}}^{\mathrm{stop}} \implies \mathit{dist} \le \mathit{safe\text{-}distance}_3) \tag{2.18}$$

The checker is equal to *False* only for the following two cases:

1. The condition in the **elseif** is true and $\mathit{dist} \le \mathit{safe\text{-}distance}_4$
   For the first case, note that $\delta \le \tau_{\mathrm{f}}^{\mathrm{stop}}$; otherwise the conjunct $t_{\mathrm{e}}^{\mathrm{stop}} < \tau_{\mathrm{f}}^{\mathrm{stop}}$ is equal to $t_{\mathrm{e}}^{\mathrm{stop}} < 0$ and this contradicts with Assumptions 2.1 and 2.2. From this, it follows that $\mathit{dist} \le \mathit{safe\text{-}distance}_4$ is equal to the conjunct $\sigma_{\mathrm{f}}^0 - \sigma_{\mathrm{e}}^0 \le \frac{(\phi_{\mathrm{f}} - v_{\mathrm{e}})^2}{2 \cdot (a_{\mathrm{f}} - a_{\mathrm{e}})}$ in Theorem 2.7. With the conjuncts in the **elseif**, we can deduce a collision from Theorem 2.7, provided that we can show the premises hold. The premise $s_{\mathrm{f}}^0 \le s_{\mathrm{e}}^{\mathrm{stop}}$ is equal to $\mathit{dist} \le \mathit{safe\text{-}distance}_0$ which is true according to 2.18. Premises $s_{\mathrm{e}}^{\mathrm{stop}} < s_{\mathrm{f}}^{\mathrm{stop}}$ and $s_{\mathrm{e}}(\delta) < s_{\mathrm{f}}(\delta)$ cannot be deduced in this case. If these are the case, then Theorem 2.7 with second conjunct in (2.18) (for premise $s_{\mathrm{f}}(\delta) \le s_{\mathrm{e}}^{\mathrm{stop}}$) guarantee a collision; otherwise Theorem 2.3 and 2.4 guarantee a collision.

2. The condition is the **elseif** is False and $\mathit{dist} \le \mathit{safe\text{-}distance}_1$; a collision is guaranteed according to Theorem 2.3 in this case. □

2.4 Sound and executable checker using interval arithmetic

The sound-and-complete checker defined previously operates with real numbers datatype $\mathbb{R}$ and by definition they have infinite precision; the results of arithmetic operations such as addition, subtraction, multiplication, and division are exact. This means that the soundness and completeness theorems in Theorem 2.8 and 2.9 have the implicit assumption that every variables such as position, speed, and acceleration can be represented and manipulated with infinite precision. Therefore, simply replacing reals with floating-point numbers $\mathbb{F}$ — the usual way of "representing" reals in computers — in *checker* does not have the formal guarantee of the soundness and completeness conditions anymore.

Isabelle is equipped with approximation package [27] to deal with the implementation (approximation) issue of functions which deal with reals datatype. The idea is to represent each real term with either an interval, $\mathbb{F} \times \mathbb{F}$, or an affine form [39] — written as $\mathbb{F}$ *aform* — and lift each arithmetic operation accordingly and soundly. For example, the addition of two reals $a, b \;::\; \mathbb{R}$ is lifted into the addition of two corresponding intervals $[a_l; a_u], [b_l; b_u] \;::\; \mathbb{F} \times \mathbb{F}$ as follows:

$$[a_l; a_u] + [b_l; b_u] = [a_l + b_l; a_u + b_u] \ .$$

This lifted addition is sound; that is, the addition of $a \in [a_l; a_u]$ and $b \in [b_l; b_u]$ is contained in the RHS of the equality above. Not only arithmetic operations but inequalities (predicates) must also be lifted properly and correctly. For example, the 'less than' predicates $a < b$ is lifted as follows:

$$[a_l; a_u] < [b_l; b_u] \Longleftarrow a_u < b_l \ .$$

Isabelle's approximation function *approx-form* requires two arguments: precision (of type $\mathbb{N}$) and function to be approximated. The first argument — as the name suggests — controls the precision of variable-precision arithmetic operations which will be executed by code (such as SML). The second argument is the concrete syntax of type *form* defined according to the following snippet of formalisation from [27].

```
datatype floatarith = Add floatarith floatarith
                    | Minus floatarith
                    | Mult floatarith floatarith
                    | Inverse floatarith
                    | Var nat
                    | Num float
                      ...

datatype form = Less floatarith floatarith
              | LessEqual floatarith floatarith
                ...
```

The concrete syntax of the function to be approximated can be obtained by using Chaieb's generic reification mechanism [40] according to the following interpretation functions [27].

```
definition interp_fa :: "floatarith => real list => real" where
    "interp_fa (Add a b)        vs   = interp_fa a vs + interp_fa b vs"
    "interp_fa (Minus a b)      vs   = interp_fa a vs - interp_fa b vs"
    "interp_fa (Mult a b)       vs   = interp_fa a vs * interp_fa b vs"
    "interp_fa (Inverse a)      vs   = inverse (interp_fa a vs)"
    "interp_fa (Var n)          vs   = vs ! n"
    "interp_fa (Num f)          vs   = f"
    ...
```

```
definition interp_ie :: "form => real list => bool" where
    "interp_ie (Less a b)       vs   = interp_ie a vs < interp_ie b vs"
    "interp_ie (LessEqual a b)  vs   = interp_ie a vs <= interp_ie b vs"
    ...
```

For example, reifying $s_{\mathrm{f}}^0 - s_{\mathrm{e}}^0 > \; v_{\mathrm{e}} \cdot \delta - \frac{v_{\mathrm{e}}^2}{2 \cdot a_{\mathrm{e}}}$ in Isabelle — with $s_{\mathrm{e}}^0, s_{\mathrm{f}}^0, v_{\mathrm{e}}, \delta$, and a_{e} represented by `se`, `sf`, `ve`, `d`, and `ae` — will give us the following concrete syntax.

```
term "(Less (Add (Mult (Var 4) (Var 5))
        (Minus (Mult (Mult (Var 4) (Var 4))
           (Inverse (Mult (Var 3) (Var 2))))))
      (Add (Var 1) (Minus (Var 0))))
      [se, sf, ae, 2, ve, d]"
```

The approximated version of *checker* can then be defined as

$$\mathit{checker\text{-}approx}\ (\mathit{prec}) \ := \ \mathit{approx\text{-}form}\ (\mathit{prec})\ (\mathit{checker\text{-}concrete})\ ,$$

where *checker-concrete* :: *form* is obtained by reifying *checker* in Sec. 2.3 with the following soundness theorem.

Theorem 2.10. *If $S_e, V_e, A_e, S_f, V_f, A_f, \Delta \ :: \ \mathbb{F} \times \mathbb{F}$ are the intervals (of floating-point numbers) in which actual values of position, speed, maximum deceleration (of both the ego and the front vehicles), and the reaction time are guaranteed to lie, then*

$$\mathit{checker\text{-}approx}\ \mathit{prec}\ S_e\ V_e\ A_e\ S_f\ V_f\ A_f\ \Delta \quad \Longrightarrow \quad \neg \mathit{collision}\ [0; \infty)\ .$$

A direct consequence of using approximation is that, if the checker is equal to *False*, it could either mean that it is really unsafe *collision* $[0; \infty)$ or the precision is not adequate to deduce collision freedom — hence we cannot guarantee the completeness of *checker-approx*. This is generally acceptable from engineering perspective because ensuring collision freedom (soundness) is more important than deducing collision (completeness).

As an example, we use the data from [34] which in turn are obtained from the Next Generation Simulation (NGSIM) project of the U.S. Department of Transportation Federal Highway Administration (FHWA) as follows:

```
definition Se where "Se = float_ivl prec 0"             (* in ft *)
definition Ve where "Ve = float_ivl prec 45.00"         (* in ft / s *)
definition Ae where "Ae = float_ivl prec (-25.72178)"   (* in ft / s / s *)
definition Sf where "Sf = float_ivl prec 66.97"         (* in ft *)
definition Vf where "Vf = float_ivl prec 38.66"         (* in ft / s *)
definition Af where "Af = float_ivl prec (-22.50656)"   (* in ft / s / s *)
definition D  where "D  = float_ivl prec 1"           (* in s *)

theorem "checker_approx prec Se Ve Ae Sf Vf Af D" by eval
```

The function *float-ivl* above is a function to convert a rational number — Isabelle can parse numbers such as `66.97` automatically into `6697 / 1000` — into its corresponding interval of type $\mathbb{F} \times \mathbb{F}$ with respect to a certain precision *prec*. As can be seen, we prove in Isabelle level that the checker is true with the given values, and the proof method `eval` indicates that it is proved via code generation [41].

2.5 Related work

In this section, we compare the formalisation with results from the domain of transportation engineering and formal verification. In general, all related works discussed here except the work by Goodloe et al. [42] are incomplete, and those in the domain of transportation engineering (discussed here) are not formally proved.

Transportation engineering. Doi et al. [43] and Seiler et al. [44] define the safe distance as follow:

$$\textit{safe-distance} = \left(\frac{v_e^2}{2 \cdot a_e} - \frac{v_f^2}{2 \cdot a_f} \right) + v_e \cdot T_1 + (v_f - v_e) \cdot T_2 + d_0 \ , \tag*{Doi et al.}$$

$$\textit{safe-distance} = \left(\frac{v_e^2}{2 \cdot a_e} - \frac{v_f^2}{2 \cdot a_f} \right) + v_e \cdot T + d_0 \ . \tag*{Seiler et al.}$$

Both equations have been modified so that they matches with the notations and assumptions used in this thesis[3]. Symbols T_1, T_2, and T denote system delay, reaction time, and its combination, respectively, while d_0 denotes additional headway distance to make the safe distance definition more conservative. As can be seen from Eq. (2.13), these two are very similar to *safe-distance*$_1$. If we include additional headway distance, *safe-distance*$_1$ is exactly the same with Seiler et al's.

Qu et al. [45] analyse the safe distance problem by applying a technique from molecular dynamics. Unlike the case distinction in our work, they have three cases which depend

[3] Both works assume that the maximum deceleration has positive value while this thesis assumes that it has negative value.

on the relationship between v_e and v_o.

$$\textit{safe-distance} = \begin{cases} v_\mathrm{e} \cdot \delta + \frac{(v_\mathrm{e} - v_\mathrm{f}) \cdot T_2}{2} + - \frac{v_\mathrm{e}^2}{2a_\mathrm{e}} + \frac{v_\mathrm{f}^2}{2a_\mathrm{f}} + d & \text{if } v_\mathrm{e} > v_\mathrm{f} \\ v_\mathrm{e} \cdot \delta + \frac{v_\mathrm{f}^2}{2} \left(\frac{1}{a_\mathrm{f}} - \frac{1}{a_\mathrm{e}} \right) + d & \text{if } v_\mathrm{e} = v_\mathrm{f} \\ \underbrace{v_\mathrm{e} \cdot T - \frac{v_\mathrm{e}}{a_\mathrm{f}} \left(v_\mathrm{f} + \frac{a_\mathrm{f} \cdot T}{2} - v_\mathrm{e} \right)}_{\text{acceleration part}} + & \\ \qquad \left(v_\mathrm{e} \cdot \delta - \frac{v_\mathrm{e}^2 - v_\mathrm{e} \cdot T \cdot a_\mathrm{e}}{2 \cdot a_\mathrm{e}} \right) + \left(\frac{v_\mathrm{f}^2 - v_\mathrm{f} \cdot T \cdot a_\mathrm{f}}{2 \cdot a_\mathrm{f}} \right) + d & \text{if } v_\mathrm{e} < v_\mathrm{f} \end{cases}$$

Their notion of safe distance for case $v_\mathrm{e} > v_\mathrm{f}$ and $v_\mathrm{e} = v_\mathrm{f}$ matches exactly with $\textit{safe-distance}_1$ in Eq. (2.13), if we ignore the headway distance d and build-up time T. However, their notion of safe distance when $v_\mathrm{e} < v_\mathrm{f}$ does not match with any of the definitions of safe distance in this thesis due to different assumptions. They assume that, when $v_\mathrm{e} < v_\mathrm{f}$ holds, the ego vehicle is assumed to accelerate until $v_\mathrm{e} = v_\mathrm{f}$ and, only after that, the ego vehicle performs braking. However, if we do not consider the acceleration part, their definition of safe distance in the third case matches exactly with $\textit{safe-distance}_1$ provided that, again, we ignore the headway distance d and build-up time T.

A more detailed analysis for the safe distance problem is given by Chen et al.[46] They structure their analysis based on three questions: (*1*) whether it is a single lane or multiple-lane scenario; (*2*) whether the other vehicle is stationary, decelerating, or accelerating; and (*3*) whether the speed of the ego vehicle is greater than or less than the other vehicle. One key distinction between this thesis and their work is that this thesis assumes the ego vehicle shall move with constant speed while, in their work, the ego vehicle can accelerate before performing an emergency brake. After taking into account this difference, their safe distance expression is as follows:

$$\textit{safe-distance} = v_\mathrm{e} \cdot \delta - \frac{v_\mathrm{e}^2}{2 \cdot a_\mathrm{e}} + \frac{v_\mathrm{f}^2}{2 \cdot a_\mathrm{f}} \ ,$$

which is exactly the same as $\textit{safe-distance}_1$ in Eq. (2.13).

The related work described up until now always assume that the maximum deceleration for all vehicles is the same — despite they have different names for the ego and the front vehicle. Therefore, none of the works described previously matches $\textit{safe-distance}_4$. Wilson [47] performed case distinction based on the stopping times and graphically identified the region called "envelope of opportunity" for each case. This envelope of opportunity divides the plot between the reaction time and the deceleration of the ego vehicle into safe and unsafe region. The envelope of opportunity for $t_\mathrm{e}^\mathrm{stop} > t_\mathrm{f}^\mathrm{stop}$ and $t_\mathrm{e}^\mathrm{stop} < t_\mathrm{f}^\mathrm{stop}$ match $\textit{safe-distance}_1$ and $\textit{safe-distance}_4$ in Eq. (2.13), respectively.

Formal verification. Loos et al. [48] verify ACC formally in KeYmaera where, in their model of ACC, they axiomatised that a safe distance is formalised as $\textit{safe-distance}_1$

in Eq. (2.13). This safe distance definition is then modified to take into account all possible impacts of control decisions for the future of reaction time, and then setting it as an invariant for the controller. They then use the proof calculus for the quantified differential dynamic logic (Qd$\mathcal{L}$) [49] to prove that the controller maintains this invariant, which in turn implies the axiomatised safe distance in $\textsf{\textit{safe-distance}}_1$ by transitivity. Our work completes theirs by proving that this axiomatised safe distance is indeed safe. However, their controller is safe on the assumption that all vehicles have the same braking performance.

Although Goodloe et al. [42] formally verify programs for aerospace applications, namely airborne conflict detection and resolution (CD&R), their approach is in general very similar to that which is employed in this thesis. The notion of *conflict* is equivalent to the notion of *unsafe distance* in our case; two aircrafts are in conflict if they violate a minimum separation requirement. Their objective is to verify whether a checker correctly determines that two aircraft maintain a minimum separation distance. Similar to this work, they also define an abstract checker, prove its soundness and completeness in PVS theorem prover, derive a concrete checker in C, and prove that the refinement from abstract to concrete checker is correct in Frama-C. This work differs in the step to convert from abstract to concrete checker. Thanks to the code generation facility in Isabelle, the concrete checker can be generated automatically in SML.

3 Predicting the occupancies of other vehicles

Occupancy prediction is one of the core components for planning a safe motion and, hence, it is very crucial to predict the occupancies of other traffic participants correctly. If we suppose that AVs do not utilise the analyses in the previous chapter and that the front vehicle brakes, then they must compute — or predict — the spaces occupied by the front vehicle to plan an emergency manœuvre. This manœuvre is safe only if *1.)* the *actual* occupancies of the plan do not intersect with the *predicted* spaces; and *2.)* the *predicted* spaces enclose the *actual* spaces which will be occupied by the other traffic participants. This chapter focusses on the second point.

This chapter predicts the occupancies of other traffic participants with the technique of so-called *physics-based models* [50]. There are other techniques such as *manœuvre-based* and *interaction-aware* models which are not used here, because we prefer a lower level (more detailed) abstraction. Note that each technique contains the word 'model' which implies that it carries specific hypotheses and assumptions. These assumptions are either not provable in the logical system or taken for granted to simplify the engineering process; this is a standard practice where we use models to approximate natural phenomena or real-world systems. Techniques such as conformance checking [51], which check whether the behaviours in the abstract model could transfer to a real system, could be used to increase the confidence of the formal verification results.

The main contribution of this chapter is to prove formally that the predicted occupancies over-approximate the actual occupancies, assuming that the other traffic participants behave according to the physical model used (Sec. 3.1). We achieve this objective by first finding the analytical functions of the boundaries of the occupancies (Sec. 3.2), and then over-approximate these functions with their respective secant lines (Sec. 3.3). These secant lines are in turned used as the bases for constructing the convex polygon enclosing the actual occupancies over a certain time interval (Sec. 3.4). Similar to the previous chapter, we also consider the numerical correctness in computing the occupancies of other traffic participants (Sec. 3.5).

Occupancy prediction in this chapter is based on the work by Althoff et al. [52] and Althoff and Magdici [53] which culminate in a tool called SPOT [54]. The presented work here can be seen as a (partial) formalisation of these works in ISABELLE. The rectangle enlargement in Sec. 3.5.1 is based on the joint work with Matthias Althoff [55].

3.1 Modelling acceleration-based occupancy

In modelling the occupancy of the other traffic participants, there are two reasonable assumptions to be made [53]:

Assumption 3.1. *Driving backwards is prohibited.*

Assumption 3.2. *Maximum absolute acceleration is limited by a constant a_{max} :: $\mathbb{R}$ where it is assumed to be positive, i.e.,* $0 < a_{max}$.

There are also other assumptions listed in [53] such as the rule about maximum speed, but we stick with these two assumptions; only these two will be relevant for computing the acceleration-based occupancies. As Assumption 3.1 suggests, the work in this chapter applies only to highway scenarios; the equivalent work for urban scenarios is left as future work.

The movement of other traffic participants is modelled by the point-mass model as in the previous chapter, but in two-dimensional Euclidean space:

$$s_x''(t) \;=\; a_x \;, \tag{3.1}$$

$$s_y''(t) \;=\; a_y \;, \tag{3.2}$$

where the acceleration obeys Assumption 3.2 above:

$$\sqrt{a_x^2 + a_y^2} \;\leq\; a_{\max} \;. \tag{3.3}$$

Given that a traffic participant is located at (s_x^0, s_y^0) :: $\mathbb{R}^2$ initially and it has the initial velocity of (v_x^0, v_y^0) :: $\mathbb{R}^2$, then the reachable sets at time t is a circle centred at:

$$(s_x^0 + v_x^0 \cdot t,\ s_y^0 + v_y^0 \cdot t) \;::\; \mathbb{R}^2 \;, \tag{3.4}$$

with a radius of

$$\frac{1}{2} \cdot a_{\max} \cdot t^2 \ . \tag{3.5}$$

We can see this intuitively by first recalling that the reachable sets of a Linear Time-Invariant (LTI) system with input can be obtained by [8, 56]:

1. finding the homogeneous solution of (3.1) and (3.2) — which is (3.4) exactly;

2. finding the set which accounts for the solutions of (3.1) and (3.2) due to the input — which is a circle with the radius of $\frac{1}{2} \cdot a_{\max} \cdot t^2$ due to Assumption 3.2; and finally

3. adding them in the sense of Minkowski sum.

This is valid as long as

$$0 \ \leq \ t \ \leq \ \frac{\|v\|}{a_{\max}} \ , \quad \text{where} \quad \|v\| = \sqrt{(v_x^0)^2 + (v_y^0)^2} \ . \tag{3.6}$$

Otherwise, the centre will turn backwards and hence it violates Assumption 3.1. We shall refer to this time validity often in this chapter, and it is useful to define this predicate for ease of reference. Provided with initial velocity $(v_x^0, v_y^0) \ :: \ \mathbb{R}^2$, the predicate *valid-time* is defined as follows:

$$\mathit{valid\text{-}time}(t) \ \ := \ \ 0 \ \leq \ t \ \wedge \ t \ \leq \ \frac{\|v\|}{a_{\max}} \ . \tag{3.7}$$

For safety verification purposes, we are usually interested with the occupancy over a time interval — not just at a single time instance. Suppose that we have t_{lb} and t_{ub} where $t_{\mathrm{lb}} \leq t_{\mathrm{ub}}$ and both time instances are valid, the occupancies over the interval $[t_{\mathrm{lb}}, t_{\mathrm{ub}}]$ is defined by the following set:

$$\left\{ c \ :: \ \mathit{circle} \ \middle| \ \exists t \in [t_{\mathrm{lb}}, t_{\mathrm{ub}}].\ \mathit{centre}(c) \ = \ (s_x^0 + v_x^0 \cdot t, s_y^0 + v_y^0 \cdot t) \ \wedge \right. \\ \left. \mathit{radius}(c) \ = \frac{1}{2} \cdot a_{\max} \cdot t^2 \right\} \ , \tag{3.8}$$

where *circle* is a concrete data type for circles:

$$\textbf{record} \ \mathit{circle} \ = \ \mathit{centre} \ :: \ \mathbb{R}^2 \ + \ \mathit{radius} \ :: \ \mathbb{R} \ .$$

If we want to check whether the occupancy of the ego vehicle is safe in $[t_{\mathrm{lb}}, t_{\mathrm{ub}}]$, we could not naïvely perform collision freedom tests with all circles in the set above; there are an infinite number of circles even though the interval is of finite duration. To solve this problem, one could find a set which encloses all of these circles, and use this instead for the collision checks with the occupancy of the ego vehicle.

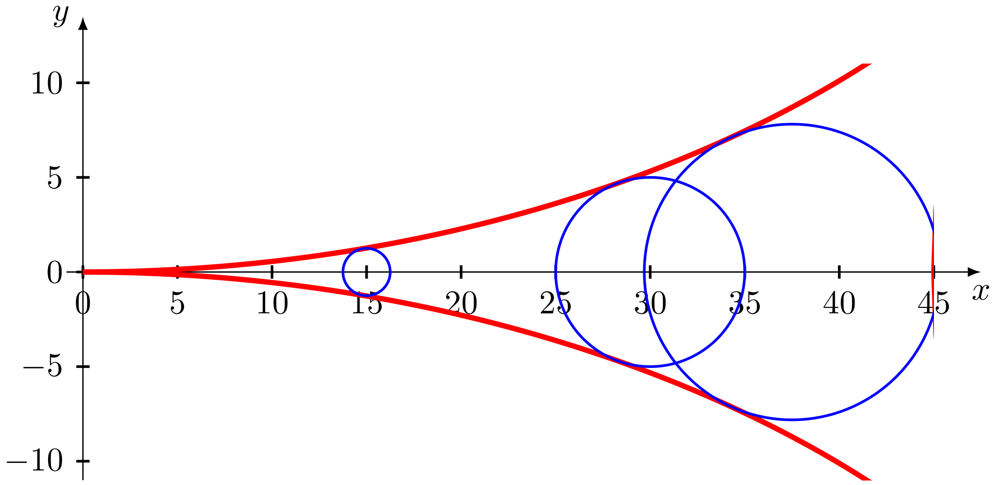


Figure 3.1: Upper and lower boundaries of occupancy circles.

3.2 Constructing boundaries for occupancy circles

Figure 3.1 provides an illustration where we superimpose three occupancy circles — shown in blue colour — at three different time points plotted according to (3.4) and (3.5). From this illustration, we can conjecture that there exists an upper and lower curve which bound the occupancy circles from above and below, respectively. If these upper and lower curves are strictly increasing and decreasing, respectively, we can merely over-approximate the upper and lower boundaries by their secant lines. This desirable property is the primary motivation for deriving these curves formally, which we shall do in this section.

How can we define or obtain such curves? To answer this question, we could start by looking at the last two occupancy circles in Fig. 3.1. Note that these two circles intersect at two points which are below and above the upper and lower curves, respectively. As we move the larger circle towards the smaller circle, we can mentally picture that both intersection points move closer to the points where the smaller circle intersects with the upper and lower curve. To define such curves therefore requires: *1.)* finding a general solution of the intersection of two circles; *2.)* instantiating this solution with circles defined in (3.4) and (3.5); and lastly *3.)* finding the limits as we move the larger circle closer to the smaller circle.

Circle intersection. Suppose that there are two circles described by the following equations:

$$(x - x_1)^2 + (y - y_1)^2 = r_1^2 \ , \tag{3.9}$$

$$(x - x_2)^2 + (y - y_2)^2 = r_2^2 \ , \tag{3.10}$$

where (x_1, y_1) and (x_2, y_2) are their centres and r_1 and r_2 their radii, respectively. We can subtract the second from the first equation, expand the quadratic terms, and rearrange them into the following equation:

$$x \;=\; \underbrace{\frac{y_2 - y_1}{x_1 - x_2}}_{:=\,\alpha_1} \cdot y \;+\; \underbrace{\frac{x_1^2 - x_2^2 \,+\, y_1^2 - y_2^2 \,-\, r_1^2 + r_2^2}{2 \cdot (x_1 - x_2)}}_{:=\,\beta_1} \;. \tag{3.11}$$

By plugging this equation to (3.9) and expanding their quadratic terms, we obtain the following equation:

$$\underbrace{(1 + \alpha_1^2)}_{:=\,\alpha_2} \cdot y^2 \;+\; \underbrace{(2\alpha_1\beta_1 - 2\alpha_1 x_1 - 2y_1)}_{:=\,\beta_2} \cdot y \;+\; \underbrace{(\beta_1 - x_1)^2 + y_1^2 - r_1^2}_{:=\,\gamma_2} \;=\; 0 \;, \tag{3.12}$$

which is a quadratic equation in terms of y. By using the general properties of quadratic equations, we can obtain the following lemma.

Lemma 3.1. *Suppose that two circles centred at (x_1, y_1) and (x_2, y_2) with $x_1 \neq x_2$ and they have the radii of r_1 and r_2, respectively, then they intersect at (x, y) if and only if the following condition is true:*

$$0 \;\leq\; \beta_2^2 - 4 \cdot \alpha_2 \cdot \gamma_2 \quad \wedge \quad y = \frac{-\beta_2 \pm \sqrt{\beta_2^2 - 4 \cdot \alpha_2 \cdot \gamma_2}}{2 \cdot \alpha_2} \quad \wedge \quad x = \alpha_1 \cdot y + \beta_1 \tag{3.13}$$

The first and the second conjuncts are the conditions for the discriminant and the solution of the quadratic equation (3.12), respectively, and the last conjunct is precisely (3.11).

Occupancy intersection. Because of position and orientation invariance of the system described in (3.1) and (3.2), we shall assume first that the other traffic participant is located initially at $(0, 0)$. Additionally, we also assume that the y-component of the velocity is zero, i.e., $v_y^0 = 0$ and $0 < v_x^0$ (see Fig. 3.1 for example). Then, any response of the system due to these assumptions must be shifted by the initial position $(s_x^0, s_y^0) :: \mathbb{R}^2$ and rotated by the initial orientation, i.e., the angle between v_x^0 and v_y^0. Hence, for any two valid time[1] instances t and t' such that $t \leq t'$, we can plug in relevant values in (3.9) and (3.10) with centres and radii defined in (3.4) and (3.5) as follows:

$$x_1 = v_x^0 \cdot t \quad , \quad y_1 = 0 \;, \quad r_1 = \frac{1}{2} \cdot a_{\max} \cdot t^2 \;, \tag{3.14}$$

$$x_2 = v_x^0 \cdot t' \quad , \quad y_2 = 0 \;, \quad r_2 = \frac{1}{2} \cdot a_{\max} \cdot t'^2 \;. \tag{3.15}$$

[1] see (3.7)

These deduced facts due to position and rotation invariance help to simplify the following variables:

$$\begin{aligned}
\alpha_1 &= 0\ , \qquad \alpha_2 = 1\ , \qquad \beta_2 = 0\ , && (3.16)\\
\beta_1 &= \frac{(v_x^0)^2 \cdot (t^2 - t'^2) - \frac{1}{4} \cdot a_{\text{max}}^2 \cdot (t^4 - t'^4)}{2 \cdot v_x^0 \cdot (t - t')}\ , && (3.17)\\
-4 \cdot \gamma_2 &= \underbrace{-\frac{a_{\text{max}}^4 \cdot (t^4 - t'^4)^2}{16 \cdot (v_x^0)^2 \cdot (t - t')^2} + \frac{1}{2} \cdot a_{\text{max}}^2 \cdot (t^4 + t'^4) - (v_x^0)^2 \cdot (t - t')^2}_{:=\ \mathsf{discriminant}}\ . && (3.18)
\end{aligned}$$

Note that β_1, γ_2, and *discriminant* have the type of $\mathbb{R}^2 \Rightarrow \mathbb{R}$ with t and t' as their parameters. We can now obtain a more specific lemma about occupancy circle intersection from Lemma 3.1 as follows:

Lemma 3.2. *Given two time instances t and t' where $t \neq t'$, the occupancies at time t and t' intersect at (x, y) if and only if the following condition is true:*

$$0 \leq \mathsf{discriminant}(t, t') \quad \wedge \quad y = \pm\sqrt{\frac{\mathsf{discriminant}(t, t')}{4}} \quad \wedge \quad x = \beta_1(t, t') \qquad (3.19)$$

Parameterised boundaries. As conjectured previously, the boundaries can be obtained by finding the limits as t' gets closer to t. By looking at the solution in (3.19), finding these limits will result in a parameterised equation of x and y in t. As for β_1, we can find the limit with the help of the following identities:

$$\begin{aligned}
t^2 - t'^2 &= (t - t') \cdot (t + t') && ,\\
t^4 - t'^4 &= (t - t') \cdot (t + t') \cdot (t^2 + t'^2) && .
\end{aligned}$$

Hence, we can define the parametric equations $b_x(t)$ and $b_y(t)$ for the boundaries as follows:

$$\begin{aligned}
b_x(t) &:= \lim_{t' \to t} \quad \beta_1(t, t') &&= \quad v_x^0 \cdot t - \frac{a_{\text{max}}^2 \cdot t^3}{2 \cdot v_x^0} && , \qquad (3.20)\\
b_y(t) &:= \lim_{t' \to t} \pm\sqrt{\frac{\mathsf{discriminant}(t, t')}{4}} &&= \pm\sqrt{-\left(\frac{a_{\text{max}}^2 \cdot t^3}{2 \cdot v_x^0}\right)^2 + \frac{a_{\text{max}}^2 \cdot t^4}{4}} && . \qquad (3.21)
\end{aligned}$$

Figure 3.1 provides an example of plotting these two parametric equations. The positive sign for b_y (denoted by b_y^+) corresponds to the upper boundaries while the negative sign (denoted by b_y^-) to the lower boundaries. Note that in order for b_y to be well-defined, we have to ensure that the expression inside the square root operator in (3.21) is at least zero; we prove this in the following lemma.

Lemma 3.3. *For any* $\mathsf{valid\text{-}time}(t)$*, we have*

$$0 \leq -\left(\frac{a_{max}^2 \cdot t^3}{2 \cdot v_x^0}\right)^2 + \frac{a_{max}^2 \cdot t^4}{4} \quad .$$

Proof. As an initial step, we separate two cases where it is either $t = 0$ or $t \neq 0$. The proof for the former case is trivial, and we are left with the second case which we prove as follows:

$$\begin{array}{ll}
& 0 \ \leq \ -\left(\dfrac{a_{\max}^2 \cdot t^3}{2 \cdot v_x^0}\right)^2 + \dfrac{a_{\max}^2 \cdot t^4}{4} \\
\Longleftarrow & \{ \text{ arithmetic } \} \\
& \dfrac{a_{\max}^4 \cdot t^6}{4 \cdot (v_x^0)^2} \ \leq \ \dfrac{a_{\max}^2 \cdot t^4}{4} \\
\Longleftarrow & \{ \text{ assumption that } a_{\max} \neq 0 \text{ in 3.2 and } t \neq 0 \text{ in this case; dividing both} \\
& \quad \text{sides with the term in RHS } \} \\
& \dfrac{a_{\max}^2 \cdot t^2}{(v_x^0)^2} \ \leq \ 1 \\
\Longleftarrow & \{ \text{ multiplying both sides with } \frac{(v_x^0)^2}{a_{\max}}\text{; taking square root on both sides} \\
& \quad \text{afterwards by noting the facts that } 0 < v_x^0 \text{ and } 0 < a_{\max} \ \} \\
& t \ \leq \ \dfrac{(v_x^0)}{a_{\max}} \\
\Longleftarrow & \{ \text{ definition of } \mathsf{valid\text{-}time}(t) \text{ in (3.7) with } \|v\| = v_x^0 \text{ since } v_y^0 = 0; \\
& \quad \text{assumption } \mathsf{valid\text{-}time}(t) \text{ in this lemma } \} \\
& \mathsf{True}
\end{array}$$

□

Due to the position and rotation invariance property, the direction of travel for a vehicle is solely determined by b_x and, because of Assumption 3.1, we need to ensure that the vehicle would not go backwards; this is formalised in the following lemma.

Lemma 3.4. *For all* $t, t' :: \mathbb{R}$ *where both* t *and* t' *belong to the interval* $[0, t_{max}]$ *with* $t_{max} := \sqrt{\frac{2}{3}} \cdot \frac{v_x^0}{a_{max}}$ *and* $t < t'$*, we have* $b_x(t) < b_x(t')$*.*

Proof. We prove for the case that $t' \in [0, t_{\max})$ initially. The first derivative of b_x is:

$$b_x'(\tau) \ := \ v_x^0 - \frac{3 \cdot a_{\max}^2 \cdot \tau^2}{2 \cdot v_x^0} \quad , \tag{3.22}$$

and it is always positive for $\tau \in [0, t_{\max})$. This is trivially true for the case $\tau = 0$.

Otherwise,

$$
\begin{aligned}
& 0 \; < \; v_x^0 - \frac{3 \cdot a_{\max}^2 \cdot \tau^2}{2 \cdot v_x^0} \\
\Longleftarrow \quad & \{ \text{ moving the minuend to LHS; multiplying both sides with } \tfrac{2 \cdot v_x^0}{3 \cdot a_{\max}} \\
& \quad \text{afterwards } \} \\
& \tau^2 \; < \; \frac{2}{3} \cdot \left(\frac{v_x^0}{a_{\max}} \right)^2 \\
\Longleftarrow \quad & \{ \text{ taking square roots on both sides; unfolding the definition of } t_{\max}; \\
& \quad \text{assumption } 0 < v_x^0 \text{ and } 0 < a_{\max} \text{ due to Assumption 3.1 and 3.2 } \} \\
& 0 \leq \tau \quad \wedge \quad \tau < t_{\max} \\
\Longleftarrow \quad & \{ \text{ assumption } \tau \in [0, t_{\max}) \ \} \\
& \mathit{True}
\end{aligned}
$$

A sufficient condition for showing $b_x(t) < b_x(t')$ is that $0 < b'_x(x)$ for all $x \in [t, t']$. Since both t and t' belongs to the range where b'_x is positive, then $b'_x(x)$ must also be positive. This completes the proof that $b_x(t) < b_x(t')$ for the case $t' \in [0, t_{\max})$. As for the case when $t' = t_{\max}$, note that $b'_x(t_{\max}) = 0$ which shows that $t_{\max}$ is either a local minimum or a local maximum. Because we have shown that b_x is increasing for $[0, t_{\max})$, it must be the case that $b_x(t_{\max})$ is a local maximum and hence $b_x(t) < b_x(t')$. □

To summarise, the boundaries defined in (3.20) and (3.21) exist up to a certain time only. Lemma 3.3 stipulates b_y is defined for those *valid-time*(t), i.e., $t \leq \frac{v_x^0}{a_{\max}}$ in (3.7), to ensure that the discriminant in (3.21) is well-defined. In order to ensure that b_x always increases (does not go backwards), we still have to limit the time to $t \leq t_{\max} = \sqrt{\frac{2}{3}} \cdot \frac{v_x^0}{a_{\max}}$ (see Lemma 3.4).

Proving boundedness. Figure 3.2 provides an arbitrary example of an occupancy circle and the corresponding upper boundary; we shall now prove that occupancy circles are bounded by the boundaries. We first identify the vertical line $b_x(t)$ where it intersects with the upper boundary at $b_y(t)$, and t is the time whose occupancy circle is being considered. To prove the boundedness property, we identify two cases where for all points (x, y) belong to the occupancy circle at time t are either to the left of the intersection line, i.e., $x \leq b_x(t)$ or to the right of the intersection line, i.e., $b_x(t) < x$. We prove the boundedness for the latter case only as the proof for the former can be obtained similarly.

Lemma 3.5. *Suppose that we have $t, \tau \in [0, t_{max}]$ where $t \leq \tau$ and $t \neq 0$. Then, for all points (x, y) in the occupancy circle at time t with $x = b_x(\tau)$, i.e.,*

$$
(b_x(\tau) - v_x^0 \cdot t)^2 + y^2 \;\leq\; \frac{1}{4} \cdot a_{max}^2 \cdot t^4 \;\; , \tag{3.23}
$$

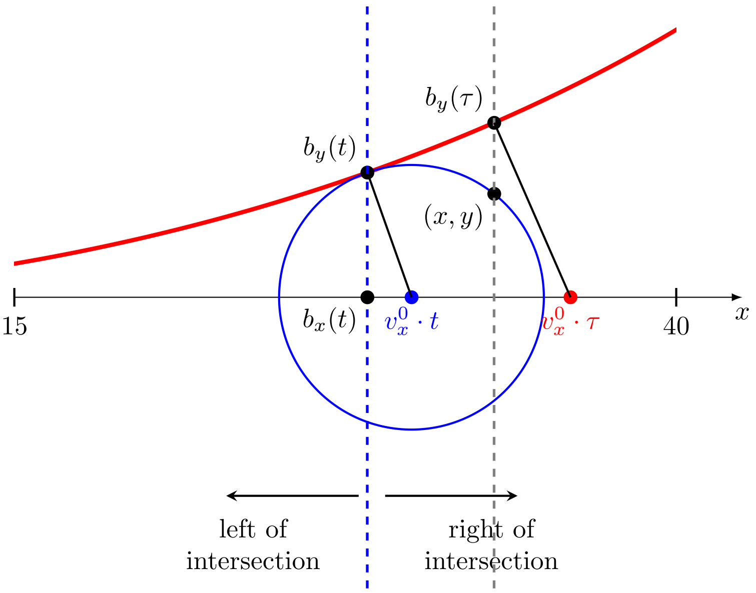


Figure 3.2: Upper and lower boundaries of occupancy circles.

we have

$$y \;\leq\; b_y^+(\tau) \quad \textit{and} \quad y \;\geq\; b_y^-(\tau) \;\; .$$

Proof. First, we identify two cases: either $y < 0$ or $0 \leq y$. Due to the premise $\tau \in [0, t_{\max}]$, it can be easily proved that $0 \leq b_y^+(\tau)$ and we can hence deduce $y \leq b_y^+(t')$ for the former case by using transitivity property. The latter case is proved with the following calculation.

$$
\begin{aligned}
& y \;\leq\; b_y^+(\tau) \\
\Longleftarrow \quad & \{ \text{ unfolding the definition of } b_y^+ \text{ according to (3.21) } \} \\
& y \;\leq\; \sqrt{-\left(\frac{a_{\max}^2 \cdot \tau^3}{2 \cdot v_x^0}\right)^2 + \frac{a_{\max}^2 \cdot \tau^4}{4}} \\
\Longleftarrow \quad & \{ \text{ transitivity with (3.23) and case } 0 \leq y \; \} \\
& \sqrt{-\left(b_x(\tau) - v_x^0 \cdot t\right)^2 + \frac{a_{\max}^2 \cdot t^4}{4}} \;\leq\; \sqrt{-\left(\frac{a_{\max}^2 \cdot \tau^3}{2 \cdot v_x^0}\right)^2 + \frac{a_{\max}^2 \cdot \tau^4}{4}} \\
\Longleftarrow \quad & \{ \text{ unfolding } b_x \text{ according to (3.20); squaring both sides } \} \\
& -\left(v_x^0 \cdot (\tau - t) - \frac{a_{\max}^2 \cdot \tau^3}{2 \cdot v_x^0}\right)^2 + \frac{a_{\max}^2 \cdot t^4}{4} \;\leq\; -\left(\frac{a_{\max}^2 \cdot \tau^3}{2 \cdot v_x^0}\right)^2 + \frac{a_{\max}^2 \cdot \tau^4}{4} \\
\Longleftarrow \quad & \{ \text{ expanding the quadratic term on the LHS; arithmetic } \}
\end{aligned}
$$

$$-\left(v_x^0 \cdot (\tau - t)\right)^2 + a_{\max}^2 \cdot \tau^3 \cdot (\tau - t) \;\leq\; \frac{a_{\max}^2 \cdot (\tau^4 - t^4)}{4}$$

$\Longleftarrow$ { exchange LHS and RHS properly }

$$-\frac{a_{\max}^2 \cdot (\tau^4 - t^4)}{4} \;\leq\; \left(v_x^0 \cdot (\tau - t)\right)^2 - a_{\max}^2 \cdot \tau^3 \cdot (\tau - t)$$

$\Longleftarrow$ { multiply both sides with $\frac{4}{a_{\max}^2}$ }

$$-(\tau^4 - t^4) \;\leq\; \left(\frac{2 \cdot v_x^0}{a_{\max}}\right)^2 \cdot (\tau - t)^2 \;-\; 4 \cdot \tau^3 \cdot (\tau - t)$$

$\Longleftarrow$ { dividing both sides with $\tau - t$; case $\tau = t$ is trivial }

$$-(\tau^2 + t^2) \cdot (\tau + t) \;\leq\; \left(\frac{2 \cdot v_x^0}{a_{\max}}\right)^2 \cdot (\tau - t) \;-\; 4 \cdot \tau^3$$

$\Longleftarrow$ { simplifying LHS and moving $-4 \cdot \tau^3$ to RHS; arithmetic }

$$3 \cdot \tau^3 - \tau^2 \cdot t - \tau \cdot t^2 - t^3 \;\leq\; \left(\frac{2 \cdot v_x^0}{a_{\max}}\right)^2 \cdot (\tau - t)$$

$\Longleftarrow$ { fact LHS equals $(3 \cdot \tau^2 + 2 \cdot \tau \cdot t + t^2) \cdot (\tau - t)$; dividing both sides with $\tau - t$ }

$$3 \cdot \tau^2 + 2 \cdot \tau \cdot t + t^2 \;\leq\; \left(\frac{2 \cdot v_x^0}{a_{\max}}\right)^2$$

$\Longleftarrow$ { premises $\tau \leq t_{\max}$, $t \leq t_{\max}$, and $t_{\max}^2 = \frac{2}{3} \cdot \left(\frac{v_x^0}{a_{\max}}\right)^2$; monotonicity of multiplication }

$$2 \cdot \left(\frac{v_x^0}{a_{\max}}\right)^2 + \frac{4}{3} \cdot \left(\frac{v_x^0}{a_{\max}}\right)^2 + \frac{2}{3} \cdot \left(\frac{v_x^0}{a_{\max}}\right)^2 \;\leq\; 4 \cdot \left(\frac{v_x^0}{a_{\max}}\right)^2$$

$\Longleftarrow$ { arithmetic }

True

The proof for $y \geq b_y^-(\tau)$ can be obtained similarly. □

Note about the formalisation of boundaries in isabelle. We have previously mentioned that the formal construction of the boundaries is obtained by first assuming the position and rotation invariance and then asserting that the actual boundaries are obtained by translating and rotating with respect to the initial conditions; this is for the ease of presentation only. The formalisation in ISABELLE is performed the other way around: the occupancies intersection and the boundaries construction through limits are performed without assuming $s_x^0 = 0$, $s_y^0 = 0$, and $v_y^0 = 0$ which result in a much more complex expression than what are presented in (3.14) – (3.21); then we prove formally in ISABELLE that these boundaries are indeed invariant with respect to translation and rotation.

3.3 Approximating boundaries with secant lines

Performing a collision check between a geometric object and a curve represented by its parametric equations as in (3.20) and (3.21) is difficult. It might be much easier if we can approximate the curve with a line segment, and perform the collision check with this line segment instead. Of course, we need to ensure that this is done over-approximatively: the line segments for the upper boundaries are always above the upper boundaries, and those for the lower boundaries are always below the lower boundaries. This section shows how we can over-approximate these boundaries with their respective secant lines.

Up until now, boundaries for occupancy circles are always presented with their parametric equations. For the reason which will be apparent later in this chapter, it might be helpful to make these parametric equations implicit as follows:

$$\textsf{f-of-x}^{+} \;:=\; b_y^{+} \circ b_x^{-1} \;, \tag{3.24}$$

$$\textsf{f-of-x}^{-} \;:=\; b_y^{-} \circ b_x^{-1} \;. \tag{3.25}$$

Positive superscript signifies that the function $\textsf{f-of-x}^{+}$ (and also b_y^{+}) is the function for the upper boundaries (vice versa for the negative superscript), and notation b_x^{-1} denotes the inverse of function b_x. Function $\textsf{f-of-x}$[2] is well-defined only for the interval of $[b_x(0), b_x(t_{\max})]$; this is because Lemma 3.4 shows that b_x is injective on $[0, t_{\max}]$. Additionally, since $\textsf{f-of-x}$ is composed by b_y and b_x^{-1}, we need to ensure that the range of b_x^{-1} is in the domain of b_y for well-definedness. Since we restrict the domain of $\textsf{f-of-x}$ to $[b_x(0), b_x(t_{\max})]$, we know that the range of b_x^{-1} will be $[0, t_{\max}]$, and all members of this interval are valid according to (3.7).

We can over-approximate the occupancy circles in $[t, t']$ from above (resp. below) by finding the secant lines of $\textsf{f-of-x}^{+}$ (resp. $\textsf{f-of-x}^{-}$) between time points t and t'. Intuitively speaking, secant lines are always above (resp. below) a curve if the curve is opening upward (resp. downward), and to check this, we need to show that the first derivative is non-decreasing (resp. non-increasing). However, we cannot find the derivatives naïvely as we do with other polynomial functions because we have yet to have an explicit form of b_x^{-1}; we only know that such inverse does exist. Finding such inverse is theoretically possible but relatively challenging because it is in cubic form. Instead, we use the following theorem for finding the derivative of the inverse of a function.

Theorem 3.1. *Suppose that $(f^{-1} \circ f)(x) = x$ for all $x \in (a, b)$ and $(f' \circ f^{-1})(x) \neq 0$. Then, for all $x \in (a, b)$ such that f^{-1} is continuous at x, we have*

$$[f^{-1}(x)]' \;=\; \frac{1}{(f' \circ f^{-1})(x)} \;.$$

[2] We drop the superscript when we refer to both $\textsf{f-of-x}^{+}$ and $\textsf{f-of-x}^{-}$ (resp. b_y^{+} and b_y^{-}).

The first derivative of b_x^{-1} is defined and formalised in the following corollary.

Corollary 3.1. *For all $x \in (b_x(0), b_x(t_{max}))$, we have*

$$[b_x^{-1}]'(x) \;=\; \frac{1}{(b_x' \circ b_x^{-1})(x)} \quad . \tag{3.26}$$

Proof. To prove this, we need to show the premises in the theorem above is true. The condition $(b_x^{-1} \circ b_x)(x) = x$ is true because b_x is injective for $t \in [0, t_{\max}]$ while $(b_x' \circ b_x)(x) \neq 0$ is true because we exclude $b_x(t_{\max})$ and $b_x(-t_{\max})$ in the premise of this corollary; times $t_{\max}$ and $-t_{\max}$ are the only time points where $b_x'(\tau) = 0$. The continuity of $b^{-1}(x)$ is guaranteed due to the continuity and the injectivity of b_x on $[0, t_{\max}]$. □

Due to the chain rule for derivative and the expression of *f-of-x* in (3.24) and (3.25), we need to know the first derivative of function b_y if we want to assert whether the first derivative of *f-of-x* is non-decreasing or non-increasing.

Lemma 3.6. *For any valid-time(t) such that $t \neq 0$, the first derivative of b_y is*

$$b_y'(t) \;=\; \frac{t \cdot a_{max} \cdot \left(1 - \frac{3}{2} \cdot \left(\frac{t \cdot a_{max}}{v_x^0}\right)^2\right)}{\sqrt{1 - \left(\frac{t \cdot a_{max}}{v_x^0}\right)^2}} \quad . \tag{3.27}$$

Proof.

$$\begin{aligned}
& b_y'(t) \\
= \quad & \{ \text{ unfolding (3.21) } \} \\
& \left[\sqrt{-\left(\frac{a_{\max}^2 \cdot t^3}{2 \cdot v_x^0}\right)^2 + \frac{a_{\max}^2 \cdot t^4}{4}}\right]' \\
= \quad & \{ \text{ using differentiation rule } \left[\sqrt{f(x)}\right]' \;=\; \tfrac{f'(x)}{2 \cdot \sqrt{f(x)}} \;\; \} \\
& \frac{t^3 \cdot a_{\max}^2 \cdot \left(1 - \frac{6 \cdot t^2 \cdot a_{\max}^2}{(2 \cdot v_x^0)^2}\right)}{2 \cdot b_y(t)} \\
= \quad & \{ \text{ arithmetic manipulation of the denominator } \} \\
& \frac{t^3 \cdot a_{\max}^2 \cdot \left(1 - \frac{6 \cdot t^2 \cdot a_{\max}^2}{(2 \cdot v_x^0)^2}\right)}{t^2 \cdot a_{\max} \cdot \sqrt{1 - \left(\frac{t \cdot a_{\max}}{v_x^0}\right)^2}} \\
= \quad & \{ \text{ simplifying numerator and denominator; premise } t \neq 0 \text{ and} \\
& \quad \text{Assumption 3.2 } \}
\end{aligned}$$

$$\frac{t \cdot a_{\max} \cdot \left(1 - \frac{3}{2} \cdot \left(\frac{t \cdot a_{\max}}{v_x^0}\right)^2\right)}{\sqrt{1 - \left(\frac{t \cdot a_{\max}}{v_x^0}\right)^2}} \qquad \square$$

Note that in the premise of the lemma above, we assume that $t \neq 0$; otherwise $b_y(0) = 0$ and the denominator in the third line of the calculation above will be zero and hence the derivative will be undefined. Equipped with this lemma, we are ready to prove that the first derivative of $\mathit{f\text{-}of\text{-}x}^+$ is non-decreasing.

Lemma 3.7. *For all $x \in (b_x(0), b_x(t_{max}))$, the first derivative of f-of-x is*

$$\mathit{f\text{-}of\text{-}x}'(x) \;=\; \frac{(b'_y \,\circ\, b_x^{-1})(x)}{(b'_x \,\circ\, b_x^{-1})(x)} \;, \tag{3.28}$$

and for all $x_1 \in (b_x(0), b_x(t_{max}))$ and $x_2 \in (b_x(0), b_x(t_{max}))$, condition $x_1 \leq x_2$ implies that

$$[\mathit{f\text{-}of\text{-}x}^+]'(x_1) \;\leq\; [\mathit{f\text{-}of\text{-}x}^+]'(x_2) \;, \tag{3.29}$$

$$[\mathit{f\text{-}of\text{-}x}^-]'(x_1) \;\geq\; [\mathit{f\text{-}of\text{-}x}^-]'(x_2) \;. \tag{3.30}$$

Proof. The derivative in (3.28) is obtained by using the chain rule of differentiation, i.e., $(b_y \circ b_x^{-1})'(x) = (b'_y \circ b_x^{-1})(x) \cdot [b_x^{-1}]'(x)$ and (3.26). We shall prove next the non-decreasing property of $\mathit{f\text{-}of\text{-}x}^+$ only; the non-increasing part can be obtained similarly because $\mathit{f\text{-}of\text{-}x}^-$ is the mirror of $\mathit{f\text{-}of\text{-}x}^+$ with respect to the x-axis. Let us assume that t_1 and t_2 are two time points such that $b_x(t_1) = x_1$ and $b_x(t_2) = x_2$.

$$\mathit{f\text{-}of\text{-}x}^+(x_1) \;\leq\; \mathit{f\text{-}of\text{-}x}^+(x_2)$$

$\Longleftarrow$ { unfolding definition in (3.28); facts $b_x(t_1) = x_1$ and $b_x(t_2) = x_2$ }

$$\frac{[b_y^+]'(t_1)}{b'_x(t_1)} \leq \frac{[b_y^+]'(t_2)}{b'_x(t_2)}$$

$\Longleftarrow$ { unfolding (3.27) and (3.22); arithmetic }

$$\frac{a_{\max}}{v_x^0} \cdot \frac{t_1}{\sqrt{1 - \left(\frac{t_1 \cdot a_{\max}}{v_x^0}\right)^2}} \quad\leq\quad \frac{a_{\max}}{v_x^0} \cdot \frac{t_2}{\sqrt{1 - \left(\frac{t_2 \cdot a_{\max}}{v_x^0}\right)^2}}$$

$\Longleftarrow$ { Assumption 3.2 and $0 < v_x^0$ }

$$t_1 \leq t_2 \quad \wedge \quad \sqrt{1 - \left(\frac{t_2 \cdot a_{\max}}{v_x^0}\right)^2} \;\leq\; \sqrt{1 - \left(\frac{t_1 \cdot a_{\max}}{v_x^0}\right)^2}$$

$\Longleftarrow$ { squaring both sides on the second conjunct }

$$t_1 \leq t_2 \quad \wedge \quad \left(\frac{t_1 \cdot a_{\max}}{v_x^0}\right)^2 \leq \left(\frac{t_2 \cdot a_{\max}}{v_x^0}\right)^2$$

$\Longleftarrow$ { arithmetic; propositional logic property $A \Longrightarrow A \wedge \mathit{True}$ }

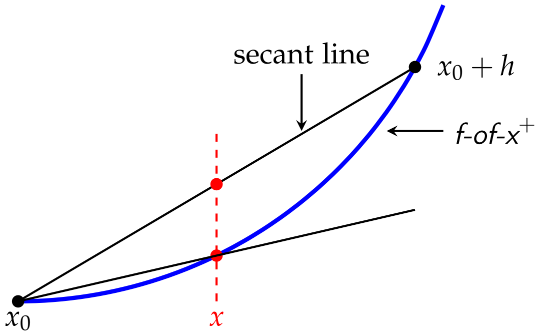


Figure 3.3: Approximating the upper boundary (blue) with a secant line.

$t_1 \leq t_2$

Finally, $t_1 \leq t_2$ must be true or otherwise, we have $t_2 < t_1$ and $x_2 = b_x(t_2) < b_x(t_1) = x_1$ due to the property that b_x is strictly increasing (Lemma 3.4); this contradicts with the premise that $x_1 \leq x_2$. □

After proving that the derivative for *f-of-x*$^+$ is non-decreasing (and *f-of-x*$^-$ is non-increasing), we shall now prove formally that the secant lines over-approximate the boundaries. Given two values of x_0 and $x_0 + h$ with $0 < h$, the slope of the secant line between $(x_0, \textit{f-of-x}(x_0))$ and $(x_0 + h, \textit{f-of-x}(x_0 + h))$ and the corresponding line equation are defined as:

$$\begin{aligned} \textit{secant-slope}(x_0, h) \quad &:= \quad \frac{\textit{f-of-x}\,(x_0 + h) \;-\; \textit{f-of-x}\,(x_0)}{h} \quad , && (3.31) \\ \textit{secant-line}(x) \quad &:= \quad \textit{secant-slope}(x_0, h) \cdot (x - x_0) + \textit{f-of-x}(x_0) \;\; . && (3.32) \end{aligned}$$

Our main objective is to show that for all $x \in [x_0, x_0 + h]$, we have

$$\textit{f-of-x}^+(x) \;\leq\; \textit{secant-line}^+(x) \quad \text{and} \quad \textit{f-of-x}^-(x) \;\geq\; \textit{secant-line}^-(x) \;\; . \qquad (3.33)$$

As Fig. 3.3 shows, these are true if we can show that *secant-slope*$^+$ is non-decreasing and *secant-slope*$^-$ is non-increasing with respect to h; we shall prove these as follows.

Recall first the well-known Mean Value Theorem (MVT) which is readily available in ISABELLE.

Lemma 3.8 (Mean Value Theorem). *For all function* $f \;::\; \mathbb{R} \Rightarrow \mathbb{R}$ *with derivative* $f'(x)$ *for* $a \leq x \leq b$ *and* $a < b$, *we have*

$$\exists z.\; a < z \;\wedge\; z < b \;\wedge\; f(b) - f(a) = (b - a) * f'(z) \;\; .$$

Theorem 3.2. *For all* $x_0 \in (b_x(0), b_x(t_{max}))$, $0 < h_1$, $0 < h_2$ *such that* $h_1 \leq h_2$, $x_0 + h_1 \in (b_x(0), b_x(t_{max}))$ *and* $x_0 + h_2 \in (b_x(0), b_x(t_{max}))$, *we have the following inequalities:*

$$\begin{aligned} \textit{secant-slope}^+(x_0, h_1) \quad &\leq \quad \textit{secant-slope}^+(x_0, h_2) \;\; , \\ \textit{secant-slope}^-(x_0, h_1) \quad &\geq \quad \textit{secant-slope}^-(x_0, h_2) \;\; . \end{aligned}$$

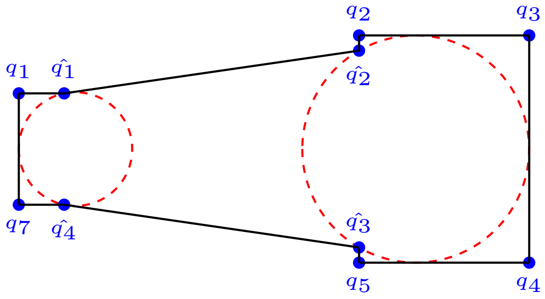


Figure 3.4: First polygon to approximate occupancy circles.

Proof. We shall prove for the non-decreasing part only as the non-increasing part can be obtained similarly. First, we show that the derivative of *secant-slope* is non-negative with respect to the second argument. The derivative of *secant-slope*$^+$ is defined as follows:

$$[\textit{secant-slope}^+]'(x_0, h) \;:=\; \frac{[\textit{f-of-x}^+]'(x_0+h)\cdot h - \textit{f-of-x}^+(x_0+h)}{h^2} + \frac{\textit{f-of-x}^+(x_0)}{h^2} \;.$$

We can see that this definition is indeed the derivative of *secant-slope*$^+$ with respect to h from the quotient rule $(\frac{f}{g})' = \frac{f'\cdot g - f\cdot g'}{g^2}$ and the subtraction rule $(f-g)' = f' - g'$. Next, we proceed with the following calculation.

$$0 \;\le\; \frac{[\textit{f-of-x}^+]'(x_0+h)\cdot h - \textit{f-of-x}^+(x_0+h)}{h^2} + \frac{\textit{f-of-x}^+(x_0)}{h^2} \;.$$

$\Longleftarrow$ { moving the minuend and the right summand to LHS; multiply both sides with h }

$$\frac{\textit{f-of-x}^+(x_0+h) - \textit{f-of-x}^+(x_0)}{h} \;\le\; [\textit{f-of-x}^+]'(x_0+h)$$

$\Longleftarrow$ { instantiating z from Mean Value Theorem (MVT) with $x_0 < z$ and $z < x_0 + h$ }

$$[\textit{f-of-x}^+]'(z) \;\le\; [\textit{f-of-x}^+]'(x_0+h)$$

$\Longleftarrow$ { Lemma 3.7 with the fact $z < x_0 + h$ }

True

Lastly, we use the theorem that non-negative derivative of a function implies that the function is non-decreasing to show that *secant-slope*$^+$ is non-decreasing. □

3.4 Occupancies over a time interval

As mentioned earlier in this chapter, we wish to compute a geometrical object which encloses all occupancy circles of a time interval $[t_{\text{lb}}, t_{\text{ub}}]$ which is easier to represent computationally compared to the definition in (3.8). Figure 3.4 provides the first

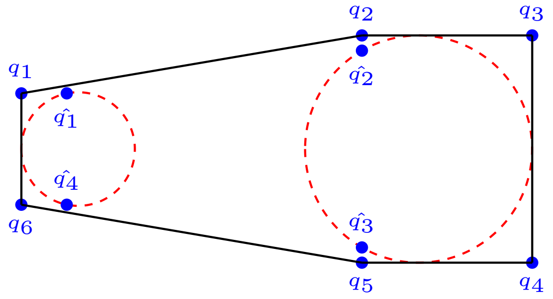


Figure 3.5: Second polygon to approximate occupancy circles.

approximation of such geometrical object. We bound the occupancy circles from left and right with vertical lines positioned at $v_x^0 \cdot t_{\text{lb}} - \frac{1}{2} \cdot a_{\max} \cdot t_{\text{ub}}^2$ and $v_x^0 \cdot t_{\text{ub}} + \frac{1}{2} \cdot a_{\max} \cdot t_{\text{lb}}^2$. Next, we bound the upper and lower sides with a series of straight lines which can be clearly seen to enclose all occupancy circles; line segments $\hat{q_1}\hat{q_2}$ and $\hat{q_4}\hat{q_3}$ are the secant lines described in the previous section.

Definition 3.1. *Given two valid time points t_{lb}, t_{ub} where both time points are at most t_{max} and $t_{lb} \leq t_{ub}$, we can define the following five points.*

$$\begin{aligned}
q_1 &= (v_x^0 \cdot t_{lb} - \frac{1}{2} \cdot a_{max} \cdot t_{lb}^2 \ , \ b_y^+(t_{lb})) \ , \\
\hat{q_1} &= (b_x(t_{lb}) \ , \ b_y^+(t_{lb})) \ , \\
\hat{q_2} &= (b_x(t_{ub}) \ , \ b_y^+(t_{ub})) \ , \\
q_2 &= (b_x(t_{ub}) \ , \ \frac{1}{2} \cdot a_{max} \cdot t_{ub}^2) \ , \\
q_3 &= (v_x^0 \cdot t_{ub} + \frac{1}{2} \cdot a_{max} \cdot t_{ub}^2 \ , \ \frac{1}{2} \cdot a_{max} \cdot t_{ub}^2) \ .
\end{aligned}$$

The next five points $q_4, q_5, \hat{q_3}, \hat{q_4}, q_6$ are defined as the mirror of $q_3, q_2, \hat{q_2}, \hat{q_1}, q_1$, correspondingly, with respect to the x-axis. Then, the first polygon to approximate the occupancy circles between t_{lb} and t_{ub} is defined by the following polygon:

$$\mathsf{poly}_1 := q_1 \circ \hat{q_1} \circ \hat{q_2} \circ q_2 \circ q_3 \circ q_4 \circ q_5 \circ \hat{q_3} \circ \hat{q_4} \circ q_6 \ .$$

Theorem 3.3. *Given two valid time points t_{lb}, t_{ub} where both time points are at most t_{max} and $t_{lb} \leq t_{ub}$, all occupancy circles between t_{lb} and t_{ub} defined in* (3.8) *must be located inside poly_1 defined in Def. 3.1.*

Proof. All points (x, y) in the occupancy circles defined in (3.8) with $b_x(t_{\text{lb}}) \leq x \leq b_x(t_{\text{ub}})$ must be bounded from above and below by b_y^+ and b_y^-, respectively, according to Lemma 3.5. Combined with the facts in (3.33) that b_y^+ and b_y^- are bounded by their respective secant lines, which are line segments $\hat{q_1}\hat{q_2}$ and $\hat{q_4}\hat{q_5}$, we can deduce that (x, y)

is inside poly_1. Next, we are left with two cases: either $v_x^0 \cdot t_{\text{lb}} - \frac{1}{2} \cdot a_{\max} \cdot t_{\text{lb}}^2 \leq x \leq b_x(t_{\text{lb}})$ or $b_x(t_{\text{ub}}) \leq x \leq v_x^0 \cdot t_{\text{ub}} + \frac{1}{2} \cdot a_{\max} \cdot t_{\text{ub}}^2$.

For the former case, all points must be located inside the occupancy circle at time t_{lb}. Otherwise, we can obtain x where $x < v_x^0 \cdot t_{\text{lb}} - \frac{1}{2} \cdot a_{\max} \cdot t_{\text{lb}}^2$ or $b_x(t_{\text{ub}}) < x$ and these two contradict with the condition in the first case. Since in this case b_y^+ is increasing and b_y^- is decreasing, we have $y \leq b_y^+(t_{\text{lb}})$ and $b_y^-(t_{\text{lb}}) \leq y$. We can then deduce that (x, y) is inside poly_1 for this case due to the fact that line segments $q_1\hat{q}_1$ and $q_6\hat{q}_4$ have the values of $b_y^+(t_{\text{lb}})$ and $b_y^-(t_{\text{lb}})$, respectively, in the y-axis.

As for the latter case, (x, y) must be located inside the occupancy circle at t_{ub}. Otherwise, we can find the value x where $x < b_x(t_{\text{ub}})$ or $v_x^0 \cdot t_{\text{ub}} + \frac{1}{2} \cdot a_{\max} \cdot t_{\text{ub}}^2 < x$ which contradict with the condition in the second case. Then, we can deduce that the absolute value of y is at most the radius of the occupancy circle, i.e. $-\frac{1}{2} \cdot a_{\max} \cdot t_{\text{ub}}^2 \leq y \leq \frac{1}{2} \cdot a_{\max} \cdot t_{\text{ub}}^2$. Since line segments q_2q_3 and q_4q_5 have the values of $\frac{1}{2} \cdot a_{\max} \cdot t_{\text{ub}}^2$ and $-\frac{1}{2} \cdot a_{\max} \cdot t_{\text{ub}}^2$, respectively, in the y-axis, it is clear that (x, y) is inside poly_1 by transitivity property. □

The first approximation in Def. 3.1 (Fig. 3.4) is a non-convex polygon, and it is desirable to make this polygon convex; we make this polygon convex as follows (see Fig. 3.5).

Definition 3.2. *Given two valid time points t_{lb}, t_{ub} where both time points are at most t_{max} and $t_{lb} \leq t_{ub}$, the second polygon to approximate the occupancy circles between t_{lb} and t_{ub} is defined by the following polygon:*

$$\mathsf{poly}_2 := q_1 \circ q_2 \circ q_3 \circ q_4 \circ q_5 \circ q_6 \ ,$$

where all of these points are the same with those defined in Def. 3.1.

Theorem 3.4. *Given two valid time points t_{lb}, t_{ub} where both time points are at most t_{max} and $t_{lb} \leq t_{ub}$, all occupancy circles between t_{lb} and t_{ub} defined in* (3.8) *must be located inside poly_2 defined in Def. 3.2.*

Proof. Due to Theorem 3.3, we only need to show that poly_1 is a subset of poly_2 to prove that all points in the occupancy circles define in (3.8) also belong to poly_2. To show this subset relation, note that the difference between poly_1 with poly_2 is that we remove $\hat{q}_1, \hat{q}_2, \hat{q}_3$, and $\hat{q}_4$ from poly_1. Graphically speaking, this implies that we only need to show that *1)* the polyline $q_1 \circ \hat{q}_1 \circ \hat{q}_2 \circ q_2$ is below the line segment q_1q_2; and *2)* the line segment q_6q_5 is below the polyline $q_6 \circ \hat{q}_4 \circ \hat{q}_3 \circ q_5$. The first proof obligation is indeed true because we can see that q_2 is exactly above $\hat{q}_2$ and q_1 is exactly to the left of $\hat{q}_1$; the similar reasoning also applies to showing that the second proof obligation. □

Considering the physical dimension of a vehicle. Up until now, we have been assuming that the other vehicle is a point mass and have yet to consider its physical

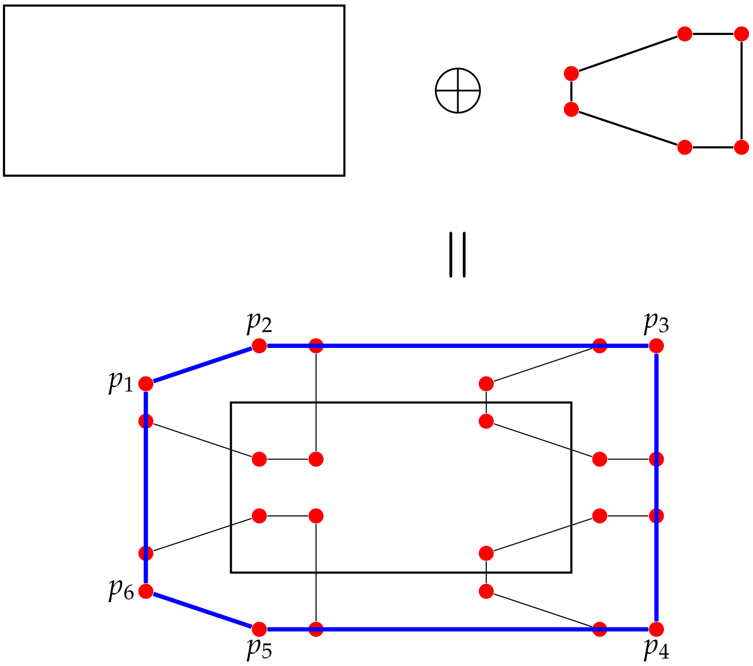


Figure 3.6: Minkowski addition of a vehicle's occupancy polygon with its physical size.

dimension. We can obtain its occupancy over time by performing Minkowski addition between its occupancy as a point mass and its geometrical shape (which can be safely over-approximated with a rectangle). Geometrically speaking, this Minkowski addition can be obtained by shifting the occupancy as a point mass to the four edges of a rectangle and then finding its convex hull (see Fig. 3.6).

Definition 3.3. *Suppose that the physical size of a vehicle is a rectangle with width w and length l such that $w \leq l$ and its occupancy as a point mass for $[t_{lb}, t_{ub}]$ is the polygon $\mathit{poly}_2 = q_1 \circ q_2 \circ q_3 \circ q_4 \circ q_5 \circ q_6$ as defined in Def. 3.2. We could then define:*

$$\begin{aligned} p_1 &= q_1 + (-\frac{1}{2} \cdot l \, , \frac{1}{2} \cdot w) \ , \\ p_2 &= q_2 + (-\frac{1}{2} \cdot l \, , \frac{1}{2} \cdot w) \ , \\ p_3 &= q_3 + \ \ (\frac{1}{2} \cdot l \, , \frac{1}{2} \cdot w) \ , \end{aligned}$$

p_4, p_5, and p_6 is the mirror of p_3, p_2, and p_1, respectively, with respect to the x-axis where each addition is a vector addition in $\mathbb{R}^2$.

3.5 Sound and executable occupancy prediction

Similar to what we have done in Sec. 2.4, this section discusses how we handle the numerical correctness in predicting space occupied by other traffic participants: uncertainties are handled by firstly over-approximating entities such as positions, speeds, and maximum deceleration with intervals, and then lifting all required arithmetic operations — such as addition, multiplication, and division — soundly to their interval arithmetic counterparts. The main difference to that in Sec. 2.4 is that predicting occupancies requires a Minkowski sum with a rectangle (see the previous section) which needs to be enlarged due to uncertainties in its position and orientation.

3.5.1 Enlarging rectangles due to uncertainties

In this section, we shall assume that position x, position y, and orientation θ are bounded by their corresponding interval:

$$x \in [x_{\text{lb}}, x_{\text{ub}}] \ , \quad y \in [y_{\text{lb}}, y_{\text{ub}}] \ , \quad \theta \in [\theta_{\text{lb}}, \theta_{\text{ub}}] \ , \tag{3.34}$$

and the actual width and length of the vehicle are $w :: \mathbb{R}$ and $l :: \mathbb{R}$, respectively. Alternatively, we could represent these uncertainties by their centres and deviations as follows:

$$\begin{aligned}
[x_{\text{lb}}, x_{\text{ub}}] &= [x_c - \delta_x, x_c + \delta_x] \ , \\
[y_{\text{lb}}, y_{\text{ub}}] &= [y_c - \delta_y, y_c + \delta_y] \ , \\
[\theta_{\text{lb}}, \theta_{\text{ub}}] &= [\theta_c - \delta_\theta, \theta_c + \delta_\theta] \ ,
\end{aligned}$$

where

$$x_c = \frac{x_{\text{ub}} + x_{\text{lb}}}{2} \ , \quad \delta_x = x_{\text{ub}} - x_c \ , \tag{3.35}$$

$$y_c = \frac{y_{\text{ub}} + y_{\text{lb}}}{2} \ , \quad \delta_y = y_{\text{ub}} - y_c \ , \tag{3.36}$$

$$\theta_c = \frac{\theta_{\text{ub}} + \theta_{\text{lb}}}{2} \ , \quad \delta_\theta = \theta_{\text{ub}} - \theta_c \ . \tag{3.37}$$

Rectangles are concretely represented by the following datatype:

$$\textbf{record}\ \mathit{rectangle} = \mathit{centre} :: \mathbb{R}^2 + \mathit{width} :: \mathbb{R} + \mathit{length} :: \mathbb{R} + \mathit{ori} :: \mathbb{R} \ ,$$

and we define the function $\mathit{range} :: \mathit{rectangle} \Rightarrow \mathbb{R}^2\ \mathit{set}$ as the function which maps any rectangle to the set of all points in that rectangle. With these definitions, we can formalise the objective of the rectangle enlargement is to enlarge $R_{\text{base}} := (\!|\mathit{centre} = (x_c, y_c), \mathit{width} = w, \mathit{length} = l, \mathit{ori} = \theta_c|\!)$ into $R_{\text{enlarged}} := R_{\text{base}}\ (\!|\mathit{width} := w', \mathit{length} := l'|\!)$ so

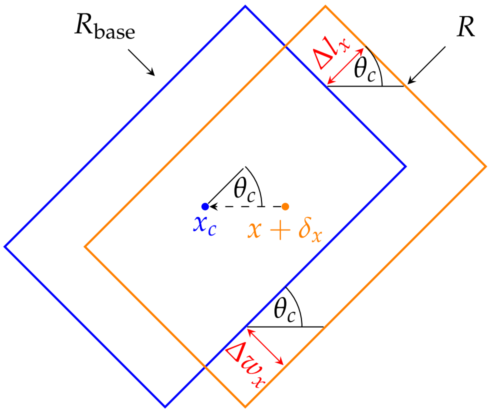


Figure 3.7: Amount of enlargement due to the error in dimension X.

that all rectangles $R := R_{\text{base}}\,(\!|\mathit{centre} := (x,y), \mathit{ori} := \theta|\!)$ where conditions in (3.34) hold are contained by R_{enlarged}, i.e.,

$$\mathit{range}(R) \;\subseteq\; \mathit{range}(R_{\text{enlarged}}) \;\;.$$

In order to find such l' and w', we analyse each uncertainty independently and add them together; this is justified because rotations and translations are linear mappings.

Compensating the position's uncertainty. Figure 3.7 partially illustrates the required enlargement due to the uncertainty in x. As can be seen, we need to increase the length of the rectangle by Δl_x and the width by Δw_x such that rectangle R_{enlarged} could enclose all rectangles R with $x_c \leq x \leq x_c + \delta_x$:

$$\begin{aligned} \Delta l_x &= \delta_x \cdot |\cos(\theta_c)| \;\;, && (3.38) \\ \Delta w_x &= \delta_x \cdot |\sin(\theta_c)| \;\;. && (3.39) \end{aligned}$$

It is not difficult to see that we also need to enlarge the rectangle by the same amount, that is Δl_x and Δw_x, for the case of $x_c - \Delta_x \leq x \leq x_c$.[3] Hence, the total enlargements due to the uncertainty in x are twice of what are defined in (3.38) and (3.39). The amount of enlargements due to the uncertainty in the y-dimension can be obtained similarly too:

$$\begin{aligned} \Delta l_y &= \delta_y \cdot |\sin(\theta_c)| \;\;, && (3.40) \\ \Delta w_y &= \delta_y \cdot |\cos(\theta_c)| \;\;. && (3.41) \end{aligned}$$

[3] Simply switch the interpretation of the two rectangles, i.e., the orange rectangle is R_{base} and the blue rectangle R_x.

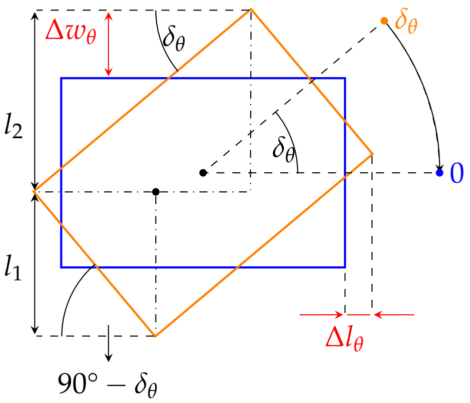


Figure 3.8: Enlargement to compensate error from dimension Θ.

Compensating the orientation's uncertainty. Figure 3.8 illustrates the required amount of enlargements due to uncertainty in the orientation. The width and the length should be enlarged by the following amounts:

$$2 \cdot \Delta l_\theta = l \cdot |\cos(\delta_\theta)| + w \cdot |\sin(\delta_\theta)| - l \ . \tag{3.42}$$

$$2 \cdot \Delta w_\theta = w \cdot |\cos(\delta_\theta)| + l \cdot |\sin(\delta_\theta)| - w \ , \tag{3.43}$$

As a sanity check, we need to ensure that the enlargements are positive; this is indeed the case for $2 \cdot \Delta w_\theta$.

Theorem 3.5. *Given a rectangle with width $0 < w$ and length l where $w < l$, we have for $-\pi < \delta_\theta \leq \pi$,*

$$0 \leq w \cdot |\cos(\delta_\theta)| + l \cdot |\sin(\delta_\theta)| - w \ .$$

Proof. Due to the occurrences of absolute value function in the inequality which we want to prove, we shall prove the theorem with proof-by-cases technique. It is easy to see the inequality holds for the simple cases $\delta_\theta = -\frac{\pi}{2}, \delta_\theta = 0, \delta_\theta = \frac{\pi}{2}$, and $\delta_\theta = \pi$ by substituting standard trigonometry identities. Hence, we are left with four cases: *1)* $-\pi < \delta_\theta < -\frac{\pi}{2}$; *2)* $-\frac{\pi}{2} < \delta_\theta < 0$; *3)* $0 < \delta_\theta < \frac{\pi}{2}$; *4)* $\frac{\pi}{2} < \delta_\theta < \pi$. In each of these cases, we can remove the absolute value function because we can deduce that the functions sin or cos are either positive or negative. We shall, however, prove for the first case only as proofs for the other cases can be obtained similarly.

$$0 \leq w \cdot |\cos(\delta_\theta)| + l \cdot |\sin(\delta_\theta)| - w$$

$$\Longleftarrow \quad \{ \ 0 < \cos(\theta) \text{ and } 0 < \sin(\theta) \text{ due to assumption } 0 < \delta_\theta \leq \tfrac{\pi}{2} \ \}$$

$$0 \leq w \cdot \cos(\delta_\theta) + l \cdot \sin(\delta_\theta) - w$$

$$\Longleftarrow \quad \{ \ \text{arithmetic} \ \}$$

$$w \cdot (1 - \cos(\delta_\theta)) \leq l \cdot \sin(\delta_\theta)$$

$$\Longleftarrow \quad \{ \ \text{divide both sides with } w \cdot \sin(\delta_\theta)\text{; premise } 0 < w \text{ and } w < l \ \}$$

$$\begin{array}{ll}
& \dfrac{1-\cos(\delta_\theta)}{\sin(\delta_\theta)} \leq \dfrac{l}{w} \\
\Longleftarrow & \{ \text{ definition of } \tan(\theta) = \frac{1-\cos(2\cdot\theta)}{\sin(2\cdot\theta)} \ \} \\
& \tan\left(\dfrac{\delta_\theta}{2}\right) \leq \dfrac{l}{w} \\
\Longleftarrow & \{ \text{ transitivity } \} \\
& \tan\left(\dfrac{\delta_\theta}{2}\right) \leq 1 \quad \wedge \quad 1 \leq \dfrac{l}{w} \\
\Longleftarrow & \{ \text{ case } 0 < \delta_\theta \leq \frac{\pi}{2} \text{ and } w < l \ \} \\
& \mathit{True}
\end{array}$$

□

This non-negative property, unfortunately, does not hold for $2 \cdot \Delta l_\theta$; evaluating (3.42) for $\delta_\theta = \frac{\pi}{2}$ results to negative value due to the premise $w < l$. To solve this problem it is necessary to find δ_θ at which the RHS in (3.42) reaches its maximum value, and then we set the enlargement to this maximum value only if δ_θ exceeds this maximum value. To find such maximum, we first find the derivatives of $2 \cdot \Delta l_\theta$ and $2 \cdot \Delta w_\theta$ with respect to δ_θ, assuming that $0 \leq \delta_\theta \leq \frac{\pi}{2}$:

$$\begin{aligned}
[2 \cdot \Delta l_\theta]' &= l \cdot -\sin(\delta_\theta) + w \cdot \cos(\delta_\theta) \ , \\
[2 \cdot \Delta w_\theta]' &= w \cdot -\sin(\delta_\theta) + l \cdot \cos(\delta_\theta) \ .
\end{aligned}$$

These derivatives are zero when $\delta_\theta = \tan^{-1}(\frac{w}{l})$ and $\delta_\theta = \tan^{-1}(\frac{l}{w})$, respectively. Differentiating the two functions above and performing second derivative tests show that $\delta_\theta = \tan^{-1}(\frac{w}{l})$ and $\delta_\theta = \tan^{-1}(\frac{l}{w})$ are local maxima.

Theorem 3.6. *For $0 \leq \delta_\theta \leq \frac{\pi}{2}$, the maximum enlargements due to uncertainty in the orientation are:*

$$(2 \cdot \Delta l_\theta) \leq \sqrt{w^2 + l^2} - l \ , \tag{3.44}$$

$$(2 \cdot \Delta w_\theta) \leq \sqrt{w^2 + l^2} - w \ . \tag{3.45}$$

Proof. We have previously asserted that the local maxima are at $\delta_\theta = \tan^{-1}(\frac{w}{l})$ and $\delta_\theta = \tan^{-1}(\frac{l}{w})$ for $2 \cdot \Delta l_\theta$ and $2 \cdot \Delta w_\theta$, respectively.

$$\begin{array}{ll}
& 2 \cdot \Delta l_\theta \leq l \cdot \cos\left(\tan^{-1}\left(\dfrac{w}{l}\right)\right) + w \cdot \sin\left(\tan^{-1}\left(\dfrac{w}{l}\right)\right) - l \\
\Longrightarrow & \{ \ \sin(\tan^{-1}(\frac{x}{y})) = \frac{x}{\sqrt{x^2+y^2}} \text{ and } \cos(\tan^{-1}(\frac{x}{y})) = \frac{y}{\sqrt{x^2+y^2}} \ \} \\
& 2 \cdot \Delta l_\theta \leq \dfrac{l^2}{\sqrt{w^2+l^2}} + \dfrac{w^2}{\sqrt{w^2+l^2}} - l \\
\Longrightarrow & \{ \text{ arithmetic } \} \\
& 2 \cdot \Delta l_\theta \leq \sqrt{w^2 + l^2} - l
\end{array}$$

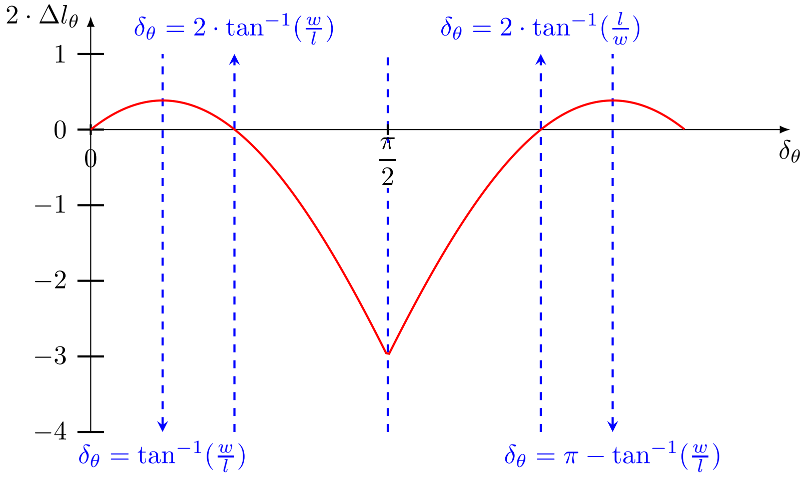


Figure 3.9: Plotting the length enlargement $2 \cdot \Delta l_\theta$ against δ_θ.

This concludes the proof for the first proof obligation; the calculation for $2 \cdot \Delta w_\theta$ can be obtained similarly. □

The discussion about finding the local maxima for $2 \cdot \Delta w_\theta$ and $2 \cdot \Delta l_\theta$ has been limited for the first quadrant only, i.e., $0 \leq \delta_\theta \leq \frac{\pi}{2}$. If we perform the similar analyses for $\frac{\pi}{2} \leq \delta_\theta \leq \pi$, the local maxima would be at $\delta_\theta = \pi - \tan^{-1}(\frac{w}{l})$ and $\Delta_\theta = \pi - \tan^{-1}(\frac{l}{w})$ with the same maximum enlargements as in (3.45) and (3.44). As shown by Fig. 3.9, the length enlargement for $\frac{\pi}{2} \leq \delta_\theta \leq \pi$ is just the mirror of the enlargement in $0 \leq \delta_\theta \leq \frac{\pi}{2}$ with respect to the vertical line $\delta_\theta = \frac{\pi}{2}$. As soon as δ_θ is larger than $\tan^{-1}(\frac{w}{l})$, we set the enlargement to the maximum values as in (3.45) and (3.44); this justifies our previous decision to focus on $0 \leq \delta_\theta \leq \frac{\pi}{2}$ only.

Finally, the total enlargement is the sum of enlargement due to uncertainties in position and orientations:

$$\begin{aligned} \Delta l &:= 2 \cdot \Delta l_x + 2 \cdot \Delta l_y + 2 \cdot \Delta l_\theta \ , \\ \Delta w &:= 2 \cdot \Delta w_x + 2 \cdot \Delta w_y + 2 \cdot \Delta w_\theta \ . \end{aligned}$$

Unfolding the definition of $2 \cdot \Delta l_\theta$ and $2 \cdot \Delta w_\theta$ results in the following expression.

$$
\begin{aligned}
\Delta l \quad = \quad & \textsf{if } \delta_\theta < \tan^{-1}\left(\frac{w}{l}\right) \textsf{ then} \\
& \qquad 2 \cdot \delta_x \cdot |\cos(\theta_c)| \;+\; 2 \cdot \delta_y \cdot |\sin(\theta_c)| \;+\; l \cdot \cos(\delta_\theta) \;+\; w \cdot \sin(\delta_\theta) - l \\
& \textsf{else} \\
& \qquad 2 \cdot \delta_x \cdot |\cos(\theta_c)| \;+\; 2 \cdot \delta_y \cdot |\sin(\theta_c)| \;+\; \sqrt{w^2 + l^2} - l \\
\Delta w \quad = \quad & \textsf{if } \delta_\theta < \tan^{-1}\left(\frac{l}{w}\right) \textsf{ then} \\
& \qquad 2 \cdot \delta_x \cdot |\sin(\theta_c)| \;+\; 2 \cdot \delta_y \cdot |\cos(\theta_c)| \;+\; w \cdot \cos(\delta_\theta) \;+\; l \cdot \sin(\delta_\theta) - w \\
& \textsf{else} \\
& \qquad 2 \cdot \delta_x \cdot |\sin(\theta_c)| \;+\; 2 \cdot \delta_y \cdot |\cos(\theta_c)| \;+\; \sqrt{w^2 + l^2} - w
\end{aligned}
$$

These two functions will be reified (refined) into those involving floating-point numbers (see Sec. 2.4) during the code generation process. During the evaluation of the guard $\delta_\theta < \tan^{-1}(\_)$,[4], both δ_θ and $\tan^{-1}(\_)$ will be replaced with $(d_{\text{lb}}, d_{\text{ub}}) :: \mathbb{F} \times \mathbb{F}$ and $(t_{\text{lb}}, t_{\text{ub}}) :: \mathbb{F} \times \mathbb{F}$, respectively, and Δl and Δw evaluate to the **then**-part only if $d_{\text{ub}} < t_{\text{lb}}$. This guarantees that even in the case that δ_θ has its maximum deviation, i.e., $\delta_\theta = d_{\text{ub}}$, it is still lower than the lower bound of the interval approximation of $\tan^{-1}(\_)$. In case the condition $d_{\text{ub}} < t_{\text{lb}}$ does not hold, it could be the case that the guard cannot be deduced — because the uncertainties are too large and overlap — or δ_θ is definitely larger than $\tan^{-1}(\_)$; in either case, we safely over-approximate the enlargements by their maximum values.

3.5.2 Interval arithmetic-based occupancy prediction

In summary, there are four steps to accomplish in order to compute the predicted occupancy of the other traffic participants: *1)* over-approximate the actual dimension of vehicles by intervals; *2)* enlarge the actual dimensions to account for uncertainties in position and orientation; *3)* compute the six vertices of the polygon representing the predicted occupancy; and *4)* rotate and translate the polygon by the orientation and centre of the rectangle, respectively. This section provides a running example and discusses how to ensure the numerical correctness in each step.

Over-approximating a vehicle's dimension. We assume that the traffic participant we want to predict is represented by a rectangle — to represent its physical shape — with the length of $L :: \mathbb{F} \times \mathbb{F}$ and width of $W :: \mathbb{F} \times \mathbb{F}$.

[4] The underscore sign "_" signifies that it can be replaced with either $\frac{w}{l}$ or $\frac{l}{w}$.

```
(* length and width cannot be represented with finite precision *)
definition "ex_length = 3 / 2 * pi"
definition "ex_width  = 1 / 2 * pi"

(* precision *)
definition "p = 20"

(* approximating the lower and upper bounds   *)
definition L :: "float * float" where "L = the (ex_length_approx p)"
definition W :: "float * float" where "W = the (ex_width_approx p)"

(* theorem showing approximations are correct *)
theorem "fst W <= ex_width"  and  "ex_width  <= snd W" and
        "fst L <= ex_length" and  "ex_length <= snd L"
  unfolding ex_width_def' w_def' ex_length_def' l_def'
  by (simp_all)(approximation 20)+
```

We also assume that variables X, Y, Θ :: $\mathbb{F} \times \mathbb{F}$ are the intervals bounding the exact centre (x, y) :: $\mathbb{R} \times \mathbb{R}$ and orientation θ :: $\mathbb{R}$.

```
                            (* [4.875; 5.125] *)
definition "X = (Float 39       (- 3), Float 41       (- 3))"
                            (* [0.875; 1.125] *)
definition "Y = (Float 7        (- 3), Float 9        (- 3))"
                            (* pi/4           *)
definition "T = (Float 1647098 (- 21), Float 1647101 (- 21))"
```

From these variables (intervals), we can then obtain the centre (x_c, y_c) and orientation θ_c and their corresponding deviations δ_x, δ_y, and δ_θ via (3.35), (3.36), and (3.37) as follows:

```
theorem
              (*  5 *)                    (* 0.125 *)
"xc = Float       5     0"   and "dx = Float 1  (-3)"    and
              (*  1 *)                    (* 0.125 *)
"yc = Float       1     0"   and "dy = Float 1  (-3)"    and
"Tc = Float 3294199 (-22)"           and "dt = Float 3 (-22)"
    unfolding ex_simps by auto
```

Rectangle enlargements. We reify the rectangle enlargement functions Δl and Δw (formalised as total_dw_abstract) in Sec. 3.5.1 into the corresponding functions involving intervals of floating-point numbers with the technique explained in Sec. 2.4. If we name the approximated version of Δw as total_dw, then we have the following correctness condition[5]:

```
theorem complete_checker_correctness':
(* assuming dx, l, dt, w, theta, dy are bounded by their intervals *)
assumes "dxl    <= dx"         and      "dx    <= dxu"
assumes "ll     <= l"          and      "l     <= lu"
assumes "dtl    <= dt"         and      "dt    <= dtu"
assumes "wl     <= w"          and      "w     <= wu"
assumes "thetal <= theta"      and      "theta <= thetau"
assumes "dyl    <= dy"         and      "dy    <= dyu"
(* upper and lower bound of the total enlargement (approximated ver.) *)
assumes "Some (lo, up) =
  total_dw p (dxl, dxu) (dyl, dyu) (dtl, dtu) (wl, wu) (ll, lu)
                                                   (thetal, thetau)"
assumes "0 < w" and "0 < l" and "0 <= dt"
shows "total_dw_abstract dx dy dt w l theta <= up"
```

With this executable rectangle enlargement functions, we can obtain the over-approximated width $w' :: \mathbb{F}$ and length $l' :: \mathbb{F}$ which accounts for uncertainties in position and orientation (and possibly the case that the actual width and length cannot be represented with finite precision) as follows:

```
definition
"l' = snd L + (snd o the) (total_dl p (dx, dx) (dy, dy) (dt, dt) W L T)"
definition
"w' = snd W + (snd o the) (total_dw p (dx, dx) (dy, dy) (dt, dt) W L T)"
```

The numerical results for the enlarged width and length are defined in the following theorem.

```
theorem
  "l' = Float 21248259 (- 22)" (* approximately 5.066 *) and
  "w' = Float  8071455 (- 22)" (* approximately 1.924 *)
  by eval+
```

Computing the six vertices of *poly*$_2$. The vertices of *poly*$_2$ can be computed by finding the approximated version of each point defined in Def. 3.3 first. Reifying the function for computing p_1 into `point1_size` (as explained in Sec. 2.4) ensures the following correctness condition[6]:

[5] We only show the correctness condition for `total_dw`; the similar condition can also be obtained for Δl.
[6] Similar theorems for p_2, p_3, p_4, p_5, and p_6 also exist.

```
  theorem correctness_point1_approx:
    assumes "tmin <= t" and "t <= max"
    assumes "point1_approx p (tmin, tmax) = (Some (l1,u1), Some (l2,u2))"
    shows "(l1, l2) <= point1_size t t'" and "point1_size t t' <= (u1, u2)"
```

In this example, we assume that the maximum acceleration and the initial speed are defined as follows:

```
    abbreviation a_max_ex :: "real" where "a_max_ex = 10" (* m / s^2 *)
    abbreviation vx_ex    :: "real" where "vx_ex    = 30" (* m / s   *)
```

Admittedly, it is rarely the case that the maximum acceleration and the initial speed can be conveniently represented by integers. In case we cannot represent these values exactly, we can find their lower and upper bounds and use these intervals instead (as what we did for W and L in the beginning of this section). By using these values, the vertices of the polygon representing the predicted occupancy between $t = 0.125\,\mathrm{s}$ and $t' = 0.25\,\mathrm{s}$ are defined as follows:

```
theorem
(* x =  1.138  *)(* y =  1.040  *)
"point1_approx p (t, t)   =
                   ((Float    597103  (- 19), Float    597104  (- 19)),
                    (Float   1090851  (- 20), Float   1090852  (- 20)))"
(* x =  4.941  *)(* y =  1.274  *)
"point2_approx p (t', t') =
                    ((Float   1295244  (- 18), Float   1295246  (- 18)),
                     (Float   1336611  (- 20), Float   1336612  (- 20)))"
(* x =  10.345 *)(* y =   1.274 *)
"point3_approx p (t', t') =
                    ((Float   1356004  (- 17), Float   1356005  (- 17)),
                     (Float   1336611  (- 20), Float   1336612  (- 20)))"
(* x =  10.345 *)(* y =  -1.274 *)
"point4_approx p (t', t') =
                    ((Float   1356004  (- 17), Float    1356005 (- 17)),
                     (Float (-1336612) (- 20), Float (-1336611) (- 20)))"
(* x =   4.941 *)(* y =  -1.274 *)
"point5_approx p (t', t') =
                    ((Float   1295244  (- 18), Float   1295246  (- 18)),
                     (Float (-1336612) (- 20), Float (-1336611) (- 20)))"
(* x =   1.138 *)(* y =  -1.040 *)
"point6_approx p (t, t)   =
                    ((Float    597103  (- 19), Float    597104  (- 19)),
                     (Float (-1090852) (- 20), Float (-1090851) (- 20)))"
```

```
  by eval+
```

Numerical results for p_4, p_5, and p_6 can be obtained by mirroring p_3, p_2, and p_1, correspondingly, with respect to the y-axis (same x values with y values inverted).

Rotating and translating. The main challenge for rotating and translating rectangles is that each vertex in the polygon is not known exactly but bounded by a box (see the numerical results above). To rotate each of these boxes over-approximatively, we rotate all four vertices in each box which will produce another four boxes, and then bound all of these boxes by finding the smallest and largest values in each dimension; this is called Axis Aligned Bounding Box (AABB) in the literature.

```
    theorem
    (*   x = 0.069;   y = 1.541   *)
    "rotate_box p1_zono = ((Float 1169159 (- 24), Float 1169529 (- 24)),
                           Float 1615769 (- 20), Float 1615791 (- 20))"
    (*   x = 2.592;   y = 4.395   *)
    "rotate_box p2_zono = ((Float 1359173 (- 19), Float 1359205 (- 19)),
                           Float 1152147 (- 18), Float 1152167 (- 18))"
    (*   x = 6.413;   y = 8.216   *)
    "rotate_box p3_zono = ((Float 1681384 (- 18), Float 1681412 (- 18)),
                           Float 1076971 (- 17), Float 1076989 (- 17))"
    (*   x = 8.216;   y = 6.413   *)
    "rotate_box p4_zono = ((Float 1076972 (- 17), Float 1076988 (- 17)),
                           Float 1681379 (- 18), Float 1681415 (- 18))"
    (*   x = 4.395;   y = 2.592   *)
    "rotate_box p5_zono = ((Float 1152150 (- 18), Float 1152166 (- 18)),
                           Float 1359169 (- 19), Float 1359207 (- 19))"
    (*   x = 1.541;   y = 0.069   *)
    "rotate_box p6_zono = ((Float 1615769 (- 20), Float 1615791 (- 20)),
                           Float 1169134 (- 24), Float 1169506 (- 24))"
    by eval+
```

As a sanity check, note that p_3 is the mirror of p_4 with respect to y-axis. Since the rectangle in this example has the orientation of 45°, then the mirror relation with respect to y-axis changes into that with respect to the line $y = x$ after the rotation. This means the values of x and y for p_3 and p_4 should be swapped and, as the numerical results show, it is indeed the case. The same sanity check applies to the other pairs of vertices too, i.e., (p_2, p_5) and (p_1, p_6).

The translation can be achieved by adding x_c to both the lower and upper bounds of x-values and y_c to both upper and lower bounds of y-values. The numerical results after translations are shown as follows:

```
theorem
  (*  x = 5.069;    y = 2.541  *)
  "p1_translated = ((Float 85055239 (- 24), Float 85055609 (- 24)),
                    Float  2664345 (- 20), Float  2664367 (- 20))"
  (*  x = 7.592;    y = 5.395  *)
  "p2_translated = ((Float  3980613 (- 19), Float  3980645 (- 19)),
                    Float  1414291 (- 18), Float  1414311 (- 18))"
  (*  x = 11.413;   y = 9.216  *)
  "p3_translated = ((Float  2992104 (- 18), Float  2992132 (- 18)),
                    Float  1208043 (- 17), Float  1208061 (- 17))"
  (*  x = 13.216;   y = 7.413  *)
  "p4_translated = ((Float  1732332 (- 17), Float  1732348 (- 17)),
                    Float  1943523 (- 18), Float  1943559 (- 18))"
  (*  x = 9.395;    y = 3.592  *)
  "p5_translated = ((Float  2462870 (- 18), Float  2462886 (- 18)),
                    Float  1883457 (- 19), Float  1883495 (- 19))"
  (*  x = 6.541;    y = 1.069  *)
  "p6_translated = ((Float  6858649 (- 20), Float  6858671 (- 20)),
                    Float 17946350 (- 24), Float 17946722 (- 24))"
  by eval+
```

3.6 Related work

To the best of my knowledge, this chapter is the first work on the formalisation of occupancy prediction in a theorem prover. This chapter only formalises the acceleration-based occupancy [52] and neglects other abstractions such as lane-following occupancy [53] and safe distance occupancy [54]. However, this does not mean that the occupancy prediction formalised in this chapter is not sound. The occupancy prediction formalised here over-approximates the actual reachable set of a vehicle, and further abstractions mentioned above are meant to improve the accuracy of the occupancy prediction so that the resulting set is as close as possible to the actual reachable sets. Compared to the works mentioned above, this chapter provides a formal proof in Isabelle/HOL of how the computed set (polygon) over-approximates the actual occupancy soundly (Theorem 3.4).

This chapter also presents the first formalisation — to the best of my knowledge — of rectangle enlargements due to the uncertainties in vehicle's orientation and position; the formalisation is heavily based on the works by Althoff et al. [57] and Rizaldi et al. [55]. The key difference is that the length enlargement in Sec. 3.5.1 is valid for $0 \leq \Delta_\theta \leq \pi$

while those in the previous work are only valid for $0 \leq \Delta_\theta \leq \tan^{-1}(\frac{w}{l})$. To see this, we first note that the length enlargements in [57] and [55] due to uncertain orientation are the absolute value of the enlargement in (3.42), i.e., $\Delta l_\theta^{\text{prev}} := |\Delta l_\theta|$. If we define Δ'_θ as the value at which $2 \cdot \Delta l_\theta$ is exactly the negative of the maximum length enlargement, i.e. $2 \cdot \Delta l_\theta = -\left(\sqrt{w^2 + l^2} - l\right)$, then $2 \cdot \Delta l_\theta^{\text{prev}}$ is less than the maximum length enlargement for $\tan^{-1}(\frac{w}{l}) < \Delta_\theta < \Delta'_\theta$ (see Fig. 3.9). This is unsound because it does not cover the case of local maximum, i.e. $\theta = \tan^{-1}(\frac{w}{l})$, which requires the maximum length enlargement.

4 Monitoring overtaking traffic rules with LTL

Formal monitoring of autonomous vehicles is the first formal analysis which we shall explore in this thesis — the other being formal verification. Why would one be interested in monitoring? Monitoring is a compromise between formal verification and testing: it gives a more formal guarantee than testing but requires less effort than formal verification. Additionally, monitoring conforms with the traditional development process in which *development* and *post-hoc verification* are performed independently. In black-box testing, for example, verification engineers deliberately ignore the system's inner workings to avoid inadvertently introduced biases in development. In monitoring, similarly, they only need the traces of the system for verification without understanding how the system works in detail. Thus, practitioners prefer monitoring to formal verification.

Consider again the following traffic rules about overtaking taken from the German traffic code StVO §5(4):

> When changing the lane to the left lane during overtaking, no following road users shall be endangered. During overtaking, a sufficient side clearance must be provided to other road users, especially pedestrians and cyclist. The driver who overtakes has to change from the fast lane to the right lane as soon as possible. The road user being overtaken shall not be obstructed.

If we wish to monitor these traffic rules, it is necessary to define all necessary concepts concretely. From a motion planner's perspective — as it only knows about the position, speed, and orientation — the most sensible way to interpret the concepts of *endangered* or *obstructed* is in terms of safe distance (see Ch. 2): a vehicle is endangered or obstructed if the ego vehicle leaves no sufficient safe distance to that vehicle. The concept of *overtaking*, on the other hand, is relatively more challenging to concretise. This chapter concretises the concept of overtaking in terms of occupancy (see Ch. 3) and the geometrical shape of the road in which the vehicles operate.

Traces provided for monitoring cyber-physical systems are usually time-sampled due to the continuous dynamics of the system. Monitoring these traces raises the question of *robustness*: how likely does the monitoring procedure change the satisfaction result of a property if we know the behaviour in between these time points? Maler and Ničković [58] provided an example where shifting the sampling points changes the satisfaction of an STL (Signal Temporal Logic) property — the property is false before shifting but true afterwards. This chapter provides a novel way to address this issue by using reachability analysis to enclose the behaviours in between these time points with sets.

The main contribution of this chapter is to define a framework for monitoring traffic rules. This objective is first achieved by codifying the overtaking traffic rules in linear temporal logic (Sec. 4.1). To concretise the concept of overtaking, we need to define first the physical environment in which an autonomous vehicle operates (Sec. 4.2) and the function to determine the lane an autonomous vehicle currently occupies (Sec. 4.3). Only then can we valuate the truth value of the atomic propositions (Sec. 4.4) elicited in the codification step. Section 4.5 discusses the numerical soundness of this monitoring framework.

This chapter is based on the joint work with Jonas Keinholz, Monika Huber, Jochen Feldle, Fabian Immler, Matthias Althoff, and Tobias Nipkow [59]. The related work in this chapter is partly based on the collaboration work with Matthias Althoff [60].

4.1 Codifying overtaking traffic rules in LTL

Prior to concretising the concept of overtaking, it is helpful to start with the codification — the process of identifying relevant atomic propositions and then use them to represent the natural language requirements in a logical language. In codification, we assume that we know how to valuate each atomic proposition. Taking this valuation for granted helps us to understand the overall picture of how the monitoring process will look like. As for the logical language, we choose Linear Temporal Logic (LTL) here because it makes the formalisation more concise than first-order logic.

Figure 4.1 illustrates the typical process of overtaking, annotated with four important time points to help us defining overtaking. We define time t_1 as the first time where the occupancy of the ego vehicle — the tip of the rectangle — touches the lane divider. Time t_2 meanwhile is the first time since t_1 where the ego vehicle is completely located in the next lane — all points in the rectangle are located inside the next lane. Time t_3 and t_4

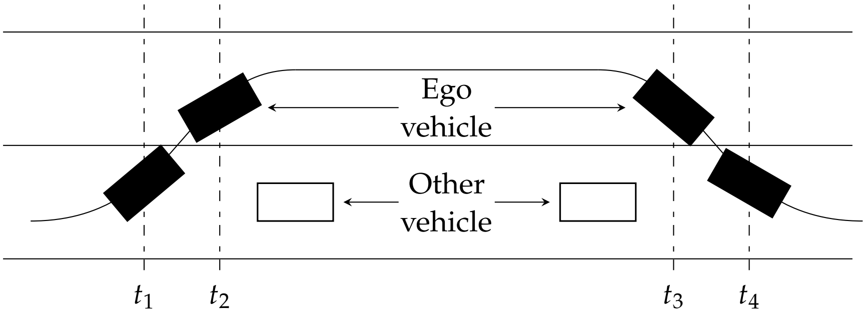


Figure 4.1: Illustration of overtaking. The curve represents the overtaking trajectory. We show the positions of the ego vehicle (filled rectangle) at four different time points t_1, t_2, t_3, and t_4. The positions of the other vehicle (empty rectangle) are shown only for t_1 and t_4.

Table 4.1: Atomic propositions and their intended interpretations

atomic proposition	intended interpretation
`overtaking`	performing an overtaking manoeuvre, i.e., $[t_1, t_4)$
`begin-overtaking`	overtaking and starting to move to the next lane, i.e., $[t_1, t_2)$
`merging`	starting to merge to the original lane, i.e., t_3
`finish-overtaking`	overtaking and returning to the original lane, i.e., $[t_3, t_4)$
`sd-rear`	maintaining a safe distance to the rear vehicle on all lanes
`safe-to-return`	leave large enough distance for merging to the original lane

are defined similarly to t_1 and t_2 except that the left lane and right lane are inverted in their definitions. That is, t_3 is the first time since t_2 where the occupancy of the ego vehicle touches the lane divider. Time t_4 meanwhile is the time when the ego vehicle is completely located inside the original lane.

The first sentence in StVO §5(4) can be interpreted as ensuring a safe distance for the following vehicle during $[t_1, t_2)$. As for the second sentence, the ego vehicle needs to maintain some side clearance to the vehicle being overtaken when their position in x-axis overlaps; it does not need to maintain such side clearance when the ego vehicle is still behind the front vehicle. Legal experts conclude that this side clearance is 1 m. The third sentence meanwhile enforces the ego vehicle to merge (time t_3) as soon as possible. The word "as soon as possible here" is interpreted as the time when ego vehicle has provided enough safe distance to the vehicle being overtaken. The safe distance not only needs to be maintained at t_3, but also during the time interval $[t_3, t_4)$.

To support this interpretation of StVO §5(4), we define six atomic propositions with their intended interpretations in Tab. 4.1. The traffic rules are codified as follows:

1. *W*hen changing the lane to the left lane during overtaking, no following road user shall be endangered.

$$\Phi_1 \ := \ G\,(\texttt{begin-overtaking} \ \longrightarrow \ \texttt{sd-rear})$$

2. *D*uring overtaking, the driver has to change from the fast lane to the right lane as soon as possible.

$$\Phi_2 \ := \ G\,(\texttt{merging} \ \longleftrightarrow \ \texttt{safe-to-return})$$

3. *T*he road user being overtaken shall not be obstructed.

$$\Phi_3 \ := \ G\,(\texttt{finish-overtaking} \ \longrightarrow \ \texttt{sd-rear})$$

With the traffic rules codified, we now need to define interpretation functions for each atomic proposition. That is, we need a way to translate a *run* (the evolution of continuous values such as position, orientation, speed, and acceleration) into a *trace* (the word over the set of all atomic propositions). The interpretation functions for atomic propositions such as `sd-rear` and `safe-to-return` can be easily obtained by using the safe distance checker in Ch. 2. The more challenging part is to define the interpretation functions for atomic propositions related to the overtaking qualifiers, such as `begin-overtaking` and `merging`, because they are defined with respect to the road model.

Just as the safe distance checker is paramount for atomic propositions `sd-rear` and `safe-to-return`, so is a *lane detection* function for atomic propositions `merging` and `begin-overtaking`. Suppose that we label the right lane and left lane (the direction of the travel is to the right) in Fig. 4.1 with 0 and 1, respectively. Additionally, we define that the lane detection primitive returns both 0 and 1 if the ego vehicle touches the lane divider. Time t_1 is thus the time where the lane detection returns both 0 and 1. Also, the atomic proposition `begin-overtaking` should be true at this time until the lane detection returns 1, which signifies that the vehicle is completely located at the left lane (time t_2). Time points t_3 and t_4 can be defined similarly from the lane detection function, and consequently, atomic propositions `merging` and `finish-overtaking` can be valuated appropriately.

4.2 Modelling lanelets and lanes for lane detection

To detect the lanes a vehicle currently occupies, one needs a computational (formal) model of the road in which it operates. This section formalises two models of lanes: a

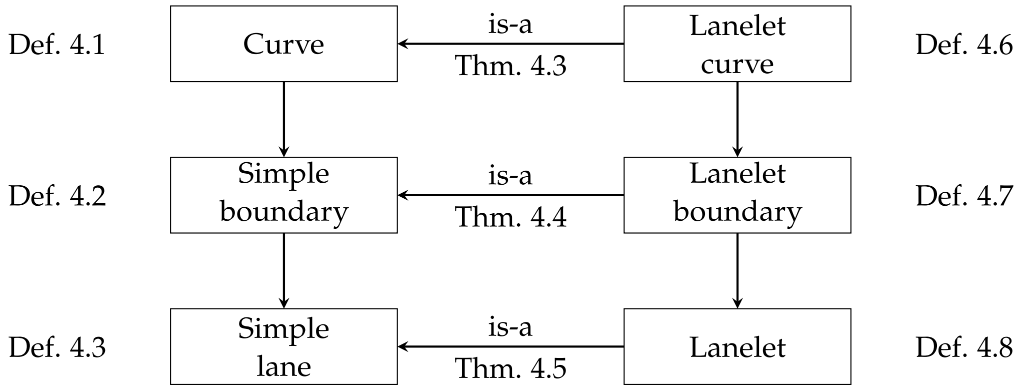


Figure 4.2: Locales relationship between several definitions related to lanes and lanelets.

generalised lane and *lanelets* [61] (see Fig. 4.2); the former is closer to a mathematical definition (more abstract) while the latter to the implementation level (more concrete). We formalise on two different abstraction levels because it is easier to prove theorems for abstract models while it is more convenient to generate code for concrete models. If we show that the concrete model refines the abstract model, then any theorem in the abstract model is also a theorem in the concrete model. In most cases, proving refinement is easier than proving a theorem directly in concrete models.

4.2.1 Generalised lanes

A *simple lane* is uniquely characterised by its left and right boundaries, and we define a *boundary* as a *curve* which has no common point (not self-intersecting).

Definition 4.1 (Curves). *A function* $\mathsf{curve\text{-}eq} :: \mathbb{R} \Rightarrow \mathbb{R}^2$ *is a curve if it is continuous on a convex and compact set* $\mathsf{domain} :: (\mathbb{R})\mathsf{set}$*.*

Parametric function *curve-eq* defines a curve on two-dimensional Euclidean space only. This restriction is because we are focussing on traffic road scenarios in which we do not gain much by generalising it to n-dimensional Euclidean space. The conditions of being compact and convex ensure that the *domain* is an interval.

Definition 4.2 (Simple boundaries). *A curve is a simple boundary if both of its parametric function* curve-eq *and its projection to the x-axis, i.e.* $\mathsf{curve\text{-}eq\text{-}x} := \mathsf{fst} \circ \mathsf{curve\text{-}eq}$*, are injective on the set* domain*.*

Forcing *curve-eq* to be injective ensures that the corresponding curve is not self-intersecting while demanding *curve-eq-x* to be injective guarantees that *curve-eq-x* has an inverse; we need this to define *f-of-x* in Theorem 4.1. Similar to the function *curve-eq-x*, the function $\mathsf{curve\text{-}eq\text{-}y}$ is defined as the projection of $\mathsf{curve\text{-}eq}$ to the y-axis, i.e., $\mathsf{curve\text{-}eq\text{-}y} := \mathsf{snd} \circ \mathsf{curve\text{-}eq}$.

Theorem 4.1. *By defining* setX *and* setY *as the image of* $\mathsf{curve\text{-}eq\text{-}x}$ *and* $\mathsf{curve\text{-}eq\text{-}y}$ *on the set* domain*, respectively, we have*

$$\mathsf{f\text{-}of\text{-}x} \in \mathsf{setX} \to \mathsf{setY}$$

where $\mathsf{f\text{-}of\text{-}x} := \mathsf{curve\text{-}eq\text{-}y} \circ \mathsf{curve\text{-}eq\text{-}x}^{-1}$.

Note that the arrow '$\rightarrow$' in $\mathsf{setX} \to \mathsf{setY}$ denotes the set of all functions with domain setX and codomain setY.

Definition 4.3 (Simple lanes)**.** *A simple lane is characterised by two continuous functions*

$$\mathsf{f\text{-}of\text{-}x}_l \in (\mathsf{setX}_l \to \mathsf{setY}_l) \quad \textit{and} \quad \mathsf{f\text{-}of\text{-}x}_r \in (\mathsf{setX}_r \to \mathsf{setY}_r)$$

representing its left and right boundaries which satisfy the following two conditions:

1. $\mathsf{setX}_l \cap \mathsf{setX}_r \neq \emptyset$ *, and*
2. $\forall x \in \mathsf{setX}_l \cap \mathsf{setX}_r .\ \ \mathsf{f\text{-}of\text{-}x}_l(x) \neq \mathsf{f\text{-}of\text{-}x}_r(x)$

These conditions imply that there is a set along the x-axis where both boundaries overlap, and in this set, both boundaries do not intersect.

Definition 4.4. *For any* $x \in \mathsf{setX}_l \cap \mathsf{setX}_r$*, function* $\mathsf{between\text{-}setY}$ *returns the set of real numbers between the left and right boundaries (an interval):*

$$\mathsf{between\text{-}setY}(x) := \left[\min(\mathsf{f\text{-}of\text{-}x}_l(x), \mathsf{f\text{-}of\text{-}x}_r(x));\ \max(\mathsf{f\text{-}of\text{-}x}_l(x), \mathsf{f\text{-}of\text{-}x}_r(x)\right] \ .$$

Definition 4.5 (Drivable area)**.** *The* $\mathsf{drivable\text{-}area}$ *of a simple lane is the set of points between left and right boundaries:*

$$\mathsf{drivable\text{-}area} := \left\{(x,y) \mid x\,y :: \mathbb{R}.\ \ x \in \mathsf{setX}_l \cap \mathsf{setX}_r \ \wedge \ y \in \mathsf{between\text{-}setY}(x)\right\} \ .$$

One important theorem we can prove from these definitions is that the drivable area is path connected (any two points in the area can always be connected with a path whose image does not leave that area).

Theorem 4.2. *The* $\mathsf{drivable\text{-}area}$ *is path connected.*

Proof. The trick to prove this theorem is to define a mid curve which is always halfway between the left and right boundaries. By doing so, we can always ensure that the points in this mid curve are always located in the drivable area; this is because real numbers

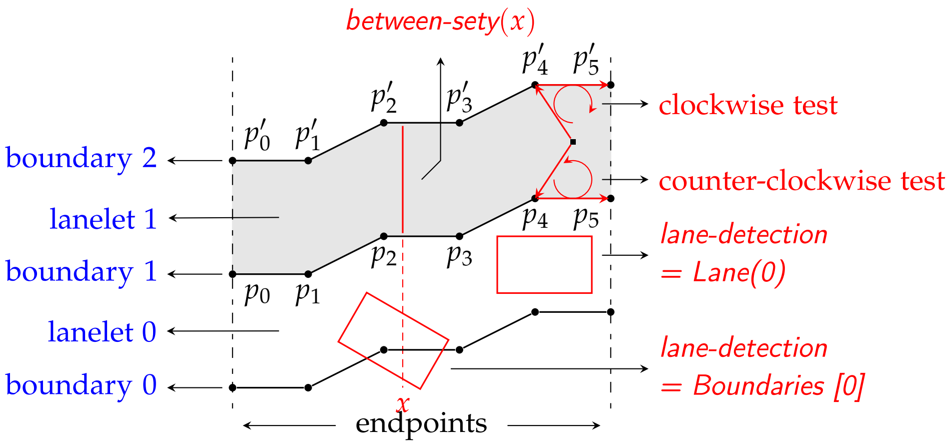


Figure 4.3: An example of two lanelets with the direction to the *right*. The upper and lower polygonal chains for lanelet 1 is *points-le* $= [(p'_0, p'_1), (p'_1, p'_2), \ldots, (p'_4, p'_5)]$ and *points-ri* $= [(p_0, p_1), (p_1, p_2), \ldots, (p_4, p_5)]$, respectively. One restriction used in this formalisation is that the endpoints have the same value in x-dimension, i.e., $\mathit{fst}(p_0) = \mathit{fst}(p'_0)$ and $\mathit{fst}(p_5) = \mathit{fst}(p'_5)$. The grey area is the drivable area for lanelet 1. Both the rightmost lanelet and the rightmost boundary are identified with 0, and they increase as we move to the leftmost lanelet and boundary.

are dense. Then, for any two points (x_1, y_1) and (x_2, y_2) in the drivable area, we first connect a vertical line from (x_1, y_1) to the mid curve, then follow through this mid curve until we can connect a vertical line to (x_2, y_2). Hence, we can always find a path between any two points in the drivable area whose image is a subset of the drivable area. □

4.2.2 Lanelets

A lanelet [61] consists of two monotone lanelet curves (left and right). We define lanelet curves and their monotonicity as follows.

Definition 4.6 (Lanelet curves). *A lanelet curve is a nonempty polygonal chain; an xs :: $(\mathbb{R}^2 \times \mathbb{R}^2)$ list is a polygonal chain if*

$$\forall i.\ i + 1 < |xs| \longrightarrow \mathit{snd}\,(xs\,!\,i) = \mathit{fst}\,(xs\,!\,(i+1))\ .$$

A line segment x :: $\mathbb{R}^2 \times \mathbb{R}^2$ in a polygonal chain xs can be easily transformed into a line path of type $\mathbb{R} \Rightarrow \mathbb{R}^2$ with the domain of $[0;1]$ as follows:

$$\mathit{linepath}(a, b) \ := \ (\lambda x.\ (1 - x) \cdot a + x \cdot b)\ .$$

Two line paths can be joined into a single path with the domain [0; 1] as follows:

$$\mathit{joinpaths}(f_1, f_2) \ := \ \left(\lambda x.\ \textbf{if } x \leq \frac{1}{2} \textbf{ then } f_1(2 \cdot x) \textbf{ else } f_2(2 \cdot x - 1)\right) \ .$$

We define *lanelet-curve-eq* as a series of *joinpaths* of line segment in a lanelet curve as follows:

$$\mathit{lanelet\text{-}curve\text{-}eq}\ \ [x] \quad = \quad \mathit{linepath}(x) \tag{4.1}$$

$$\mathit{lanelet\text{-}curve\text{-}eq}\ \ (x\#xs) \quad = \quad \mathit{joinpaths}(\mathit{linepath}(x), \mathit{lanelet\text{-}curve\text{-}eq}(xs)) \tag{4.2}$$

Theorem 4.3. *A lanelet curve is a curve.*

Proof. Associated with a lanelet curve is the function *lanelet-curve-eq* with the domain of [0, 1] defined in (4.1) and (4.2). This theorem is true due to the facts that [0, 1] is both compact and convex, a line path is a continuous function on [0, 1], and joining a series of line paths preserves continuity on [0, 1]. □

Note that in order for a curve to be a simple boundary in Def. 4.2, we need to ensure that the curve equation is injective on its domain. This property can be ensured if the polygonal chain is monotone.

Definition 4.7 (Lanelet boundaries)**.** *A lanelet boundary is a monotone polygonal chain w.r.t x-axis, i.e., a polygonal chain whose x-element always increases:*

$$\forall i < |xs|. \quad (\mathit{fst} \circ \mathit{fst})\,(xs\,!\,i) \ < \ (\mathit{fst} \circ \mathit{snd})\,(xs\,!\,i) \ .$$

The property of being monotone for a polygonal chain ensures that for each x, we have a unique y such that (x, y) is in the polygonal chain. Therefore, given a polygonal chain *points*, we can always create a function *f-of-x* from the set of all real numbers in x-dimension to the set of real numbers in y-dimension.

Theorem 4.4. *A lanelet boundary is a simple boundary (cf. Def. 4.2).*

Proof. Associated with a lanelet curve is the function *lanelet-curve-eq* with the domain of [0, 1] defined in (4.1) and (4.2). To show that this function is a simple boundary we have to show that the x-element of the parametric function is injective. This fact is a direct consequence of the condition for a polygonal chain being monotone in Def. 4.7. □

Definition 4.8 (Lanelets)**.** *A lanelet is characterised by two lanelet boundaries points-le and points-ri which do not intersect and have the same endpoints in x-dimension. That is,*

1. *For any two chains, $c_{left} \in \mathsf{set}(\mathsf{points\text{-}le})$ and $c_{right} \in \mathsf{set}(\mathsf{points\text{-}ri})$, and any two points p_{left} and p_{right} such that*

   $$\exists t_1, t_2 \in [0,1]. \quad \mathsf{linepath}(c_{left}, t_1) = p_{left} \ \wedge \ \mathsf{linepath}(c_{right}, t_2) = p_{right} \quad ,$$

   we have $p_{left} \neq p_{right}$.
2. $\mathsf{fst}\ (\mathsf{points\text{-}le.lanelet\text{-}curve\text{-}eq}\ 0) \ = \ \mathsf{fst}\ (\mathsf{points\text{-}ri.lanelet\text{-}curve\text{-}eq}\ 0)$
3. $\mathsf{fst}\ (\mathsf{points\text{-}le.lanelet\text{-}curve\text{-}eq}\ 1) \ = \ \mathsf{fst}\ (\mathsf{points\text{-}ri.lanelet\text{-}curve\text{-}eq}\ 1)$

Similar to what is defined in [61], there is no requirement of the relative placement between the two polygonal chains; *points-le* could be positioned above *points-ri* (from a bird's-eye view) or vice-versa. If it is the former then the lanelet has the direction to the right, and the left if it is the latter. Two polygonal chains points_1 and points_2 are called non-intersecting if there does not exist any two intersecting chains $c_1 \in \mathsf{set}(\mathsf{points}_1)$, $c_2 \in \mathsf{set}(\mathsf{points}_2)$.

Theorem 4.5. *A lanelet is a simple lane (cf. Def. 4.3).*

Proof. Suppose that the left and right lanelet boundaries are *points-le* and *points-ri*. Because both polygonal chains are nonempty (due to the definition of lanelet boundaries), the corresponding *setX* for both polygonal chains are nonempty too. In fact, both are the same set because we assume that they are monotone and have the same endpoints in x-dimension (cf. Def. 4.8). Hence $\mathsf{setX}_{\mathsf{points\text{-}le}} \cap \mathsf{setX}_{\mathsf{points\text{-}ri}} \neq \emptyset$. Because these two polygonal chains are non-intersecting too, we have the second condition in Def. 4.3. □

Note that, with this definition, we could not model a lane which has a 90° turn. This condition is because our definition of a monotone polygonal chain is fixed w.r.t. x-dimension. Lanelets in [61] do not have this restriction, but we can circumvent this problem by using a more general definition of monotone polygonal chains w.r.t to line l and split a polygonal chain into a minimal number of monotone polygonal chains [62]; each with its own coordinate system.

Theorem 4.6 (Drivable area). *By using the functional representation of the left and right boundaries $\mathsf{f\text{-}of\text{-}x}_l$ and $\mathsf{f\text{-}of\text{-}x}_r$ (Theorem 4.1), and defining*

$$\begin{aligned} \mathsf{first\text{-}point} &:= (\mathsf{fst} \circ \mathsf{hd})\ \mathsf{points\text{-}le} \quad , \\ \mathsf{last\text{-}point} &:= (\mathsf{snd} \circ \mathsf{last})\ \mathsf{points\text{-}le} \ , \end{aligned}$$

the following equalities hold.

$$\begin{aligned} \mathsf{setX} &= \big\{x \mid x. \quad \mathsf{fst}\,(\mathsf{first\text{-}point}) \leq x \leq \mathsf{fst}\,(\mathsf{last\text{-}point})\big\} \quad , \\ \mathsf{between\text{-}setY}(x) &= \big[\min(\mathsf{f\text{-}of\text{-}x}_l(x), \mathsf{f\text{-}of\text{-}x}_r(x)), \max(\mathsf{f\text{-}of\text{-}x}_l(x), \mathsf{f\text{-}of\text{-}x}_r(x))\big] \ , \\ \mathsf{drivable\text{-}area} &= \big\{(x,y) \ \big|\ x\,y.\ x \in \mathsf{setX} \wedge y \in \mathsf{between\text{-}setY}(x)\big\} \quad . \end{aligned}$$

Proof. This theorem is true due to Theorem 4.5 and the definition of *between-setY* and drivable area for a simple lane in Def. 4.4 and 4.5, respectively. □

By using the path connectedness in Theorem 4.2, we also have the property that the drivable area of a lanelet is also path connected.

Definition 4.9 (Lane). *A lane bs is a list of at least two lanelet boundaries where any two adjacent lanelet boundaries form a lanelet:*

$$\forall i.\ i+1 < |bs| \longrightarrow \textit{is-lanelet}\ (bs\,!\,i)\ (bs\,!\,(i+1))\ ,$$

and all lanelets have the same direction:

$$\begin{aligned} & \forall i.\ i+1 < |bs| \longrightarrow \textit{direction-left}\ \ (bs\,!\,i)\ (bs\,!\,(i+1)) \\ \vee\ \ & \forall i.\ i+1 < |bs| \longrightarrow \textit{direction-right}\ (bs\,!\,i)\ (bs\,!\,(i+1))\ . \end{aligned}$$

4.3 Designing and verifying a lane detection function

Given a rectangle representing the occupancy of a vehicle and a lane defined in Def. 4.9, the purpose of this section is to define a verified function to determine which lanelet the vehicle currently occupies. There are three possibilities of the result of this lane detection:

$$\textbf{datatype}\ \textit{detection-opt}\ =\ \textit{Outside}\ |\ \textit{Lanelet}\ (n\ ::\ \mathbb{N})\ |\ \textit{Boundaries}\ (ns\ ::\ \mathbb{N}\ \textit{list})$$

If the lane detection is equal to *Lanelet*(n), it has the interpretation that the vehicle is completely located inside a lanelet bounded by lanelet boundaries n and $n+1$. If it is equal to *Boundaries* ns, it means that the vehicle intersects with all lanelet boundaries identified in ns. If the physical size of the vehicle is relatively small, it usually intersects only with one lanelet boundary. However, if the size is relatively big (e.g. lorries), then the vehicle could intersect with at least two lanelet boundaries when it moves to the next lanelet.

Since we can view both a rectangle and a lanelet boundary as a list of chains (segments), we can check whether a rectangle intersects with a lanelet boundary by performing segment intersection tests. Of course, we only need to test relevant segments in the lanelet boundary; those segments whose x-values are located to the left or the right of the rectangle can be omitted. If we want to ensure further that the rectangle is located inside *Lanelet*(n), we also need to check whether all four vertices of the rectangle are located inside lane n. If none of these tests is true, then the vehicle must be located outside of the lane. To summarise, the lane detection function needs *segment intersection* test and *point-in-lanelet* test for lane detection.

4.3.1 Segments intersection test

A segment is concretely represented by a pair of points $(p,q) :: \mathbb{R}^2 \times \mathbb{R}^2$ and the set of all points in the segment is formalised as follows:

$$\textit{closed-segment}\,(p,q) \;=\; \left\{(1-u)\cdot p + u\cdot q \;\middle|\, u :: \mathbb{R}.\;\; 0 \le u \le 1\right\}\;. \tag{4.3}$$

Given a segment $((x_1,y_1),(x_2,y_2))$, we can obtain three coefficients $a, b,$ and c as follows:

$$a := y_2 - y_1; \quad b := x_1 - x_2; \quad c := x_1\cdot y_2 - x_2\cdot y_1\;. \tag{4.4}$$

The set of all points in a *line* formed by a segment (p,q) is formalised as follows:

$$\textit{points-in-lines}(p,q) \;=\; \{(x,y).\;\; a\cdot x + b\cdot y = c\} \tag{4.5}$$

where coefficients a, b, c are defined according to (4.4); we denote the equation $a\cdot x + b\cdot y = c$ as the characteristic equation for the line formed by the segment (p,q). The following theorem formalised the relationship between a segment and its corresponding line.

Theorem 4.7. $\textit{closed-segment}\,(p_1,p_2) \;\subseteq\; \textit{points-in-line}\,(p_1,p_2)$

Proof. Suppose that $p \in \textit{closed-segment}\,(p_1,p_2)$. Then, we can obtain u where $0 \le u \le 1$ and

$$\begin{aligned} \textit{fst}(p) &= (1-u)\cdot \textit{fst}(p_1) \;+ u\cdot \textit{fst}(p_2) & (4.6)\\ \textit{snd}(p) &= (1-u)\cdot \textit{snd}(p_1) + u\cdot \textit{snd}(p_2) & (4.7)\end{aligned}$$

according to the definition of segments in (4.3). Multiplying the first equation with $\textit{snd}(p_2) - \textit{snd}(p_1)$, the second with $\textit{fst}(p_1) - \textit{fst}(p_2)$ and adding them together result in the following equality:

$$(\textit{snd}(p_2) - \textit{snd}(p_1))\cdot \textit{fst}(p) + (\textit{fst}(p_1) - \textit{fst}(p_2))\cdot \textit{snd}(p) \;=\; \textit{fst}(p_1)\cdot \textit{snd}(p_2) - \textit{fst}(p_2)\cdot \textit{snd}(p_1)\;\;,$$

which implies that $p \in \textit{points-in-line}(p_1,p_2)$ according to the definition in (4.5). □

To check whether two segments intersect, we could first view segments as lines and try to find their intersection as follows: represent the two lines with their respective characteristic equations, $a_1\cdot x + b_1\cdot y = c_1$ and $a_2\cdot x + b_2\cdot y = c_2$, and combine these two equations in the matrix form:

$$\underbrace{\begin{bmatrix} a_1 & b_1\\ a_2 & b_2\end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x\\ y\end{bmatrix}}_{\mathbf{p}} = \underbrace{\begin{bmatrix} c_1\\ c_2\end{bmatrix}}_{\mathbf{b}}\;. \tag{4.8}$$

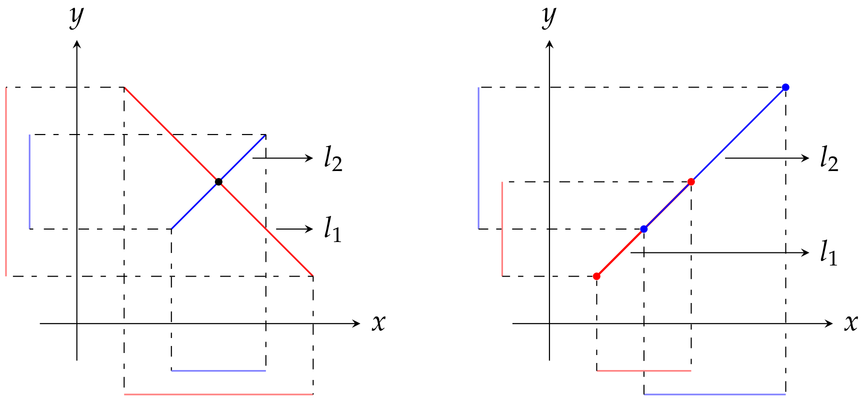


Figure 4.4: Two out of three possible segment intersection cases. The left figure denotes the case where both segments are not aligned and not parallel and the right parallel and aligned.

If the inverse A^{-1} does exist then the intersection point is uniquely defined as $\mathbf{p} = A^{-1}\mathbf{b}$.

The fundamental theorem in linear algebra about a system of equations states that there are three possible cases regarding the number of solutions: *1)* there is no solution, *2)* there is a unique solution, or *3)* there are infinitely many solutions. The first two cases happen when both lines are not aligned. More specifically, it has no solution when both lines are parallel, and a unique solution when both lines are not parallel (left of Fig. 4.4); the third case happens when both lines are aligned (right of Fig. 4.4). However, since we are dealing with segments instead of lines, we need to check the intersection — if it does exist — is indeed located on both segments.

In order to identify these three cases, we define the following functions:

$$\textsf{parallel-and-not-aligned}\,(l_1, l_2) \;:=\; a_1 \cdot b_2 - a_2 \cdot b_1 = 0 \;\wedge\; b_1 \cdot c_2 - b_2 \cdot c_1 \neq 0 \;, \tag{4.9}$$

$$\textsf{not-aligned}(l_1, l_2) \;:=\; a_1 \cdot b_2 - a_2 \cdot b_1 \neq 0 \;\vee\; b_1 \cdot c_2 - b_2 \cdot c_1 \neq 0 \;, \tag{4.10}$$

where a, b, and c are obtained according to (4.4).

Theorem 4.8 (No solution)**.** *If the function* $\textsf{parallel-and-not-aligned}(l_1, l_2)$ *is true, then*

$$\neg\exists p.\quad p \in \textsf{closed-segment}(l_1) \;\wedge\; p \in \textsf{closed-segment}(l_2) \;. \tag{4.11}$$

Proof. We first show that the system of equations in (4.8) has no solution:

$$\neg\exists p :: \mathbb{R} \times \mathbb{R}.\quad p \in \textsf{points-in-line}(l_1) \;\wedge\; p \in \textsf{points-in-line}(l_2) \;. \tag{4.12}$$

Suppose that we can find such p. Then, according to Def. 4.5, we have

$$a_1 \cdot \textsf{fst}(p) + b_1 \cdot \textsf{snd}(p) = c_1 \;\wedge\; a_2 \cdot \textsf{fst}(p) + b_2 \cdot \textsf{snd}(p) = c_2 \;.$$

Multiplying the first equality with b_2 and the second equality with b_1 and then subtracting the latter from the former results in the following equation:

$$(a_1 \cdot \mathit{fst}(p) + b_1 \cdot \mathit{snd}(p)) \cdot b_2 - (a_2 \cdot \mathit{fst}(p) + b_2 \cdot \mathit{snd}(p)) \cdot b_1 = c_1 \cdot b_2 - c_2 \cdot b_1 \ .$$

After several arithmetic manipulations, we can deduce the following equality:

$$(a_1 \cdot b_2 - a_2 \cdot b_1) \cdot \mathit{fst}(p) = c_1 \cdot b_2 - c_2 \cdot b_1 \ .$$

However, the premise in this theorem and Def. 4.9 imply $a_2 \cdot b_1 - a_1 \cdot b_2 = 0$ and $b_1 \cdot c_2 - b_2 \cdot c_1 \neq 0$ which contradicts with the equality above; hence we have proved (4.12). The condition in (4.11) is then a direct consequence of (4.12) and Theorem 4.7. □

Theorem 4.9 (Unique solution)**.** *If the function $\mathit{parallel\text{-}and\text{-}not\text{-}aligned}(l_1, l_2)$ is false but the function $\mathit{not\text{-}aligned}(l_1, l_2)$ is true, then the point $p := \mathit{find\text{-}intersection}(l_1, l_2)$ belongs to both lines l_1 and l_2:*

$$p \in \mathit{points\text{-}in\text{-}line}(l_1) \ \wedge \ p \in \mathit{points\text{-}in\text{-}line}(l_2) \ . \tag{4.13}$$

Proof. By unfolding the definitions in (4.9) and (4.10) and using the resolution proof rule, we can deduce that the determinant in (4.8) is non-zero. This implies that point p is well-defined and it is equal to $A^{-1}\mathbf{b}$. Basic results in linear algebra show that this point satisfies equations $a_1 \cdot x + b_1 \cdot y = c_1$ and $a_2 \cdot x + b_2 \cdot y = c_2$ which in turns implies (4.13) according to (4.5). □

Theorem 4.10 (Infinitely many solutions)**.** *If the function $\mathit{not\text{-}aligned}(l_1, l_2)$ is false, then*

$$\exists p :: \mathbb{R} \times \mathbb{R}. \quad p \in \mathit{points\text{-}in\text{-}line}(l_1) \ \wedge \ p \in \mathit{points\text{-}in\text{-}line}(l_2) \ . \tag{4.14}$$

Proof. From the premise that $\neg \mathit{not\text{-}aligned}(l_1, l_2)$, we know that $a_1 \cdot b_2 - a_2 \cdot b_1 = 0$ and $b_1 \cdot c_2 - b_2 \cdot c_1 = 0$. These imply that $a_1 = \frac{b_1}{b_2} \cdot a_2$, $b_1 = \frac{b_1}{b_2} \cdot b_2$, and $c_1 = \frac{b_1}{b_2} \cdot c_2$, i.e., the corresponding lines of segment l_1 and l_2 are the same; hence the condition in (4.14). □

By using these two functions and these three theorems, we can structure the temporary function $\mathit{segment\text{-}intersect}\,(l_1, l_2)$ as follows:

```
$\mathit{segment\text{-}intersect}(l_1, l_2)$ :=
    let $p = \mathit{find\text{-}intersection}(l_1, l_2)$ in
    if $\mathit{parallel\text{-}and\text{-}not\text{-}aligned}(l_1, l_2)$ then False
    else if $\mathit{not\text{-}aligned}(l_1, l_2)$ then undefined else undefined
```

The reason why the second **then** and **else** parts of the function are still undefined is because we only know that the corresponding *lines* intersect but not the *segments* (cf. Theorem 4.9 and 4.10). If the two segments are not parallel and not aligned (left of

Fig. 4.4), we need to check whether the intersection point p is $\textsf{in-x-interval}(l_1, \textsf{fst}(p)) \wedge \textsf{in-y-interval}(l_1, \textsf{snd}(p))$ and $\textsf{in-x-interval}(l_2, \textsf{fst}(p)) \wedge \textsf{in-y-interval}(l_2, \textsf{snd}(p))$. These two predicates have the following properties.

$$\textsf{in-x-interval}(l :: \mathbb{R}^2 \times \mathbb{R}^2, x) \iff x \in \textsf{closed-segment}\left((\textsf{fst} \circ \textsf{fst})\,(l), (\textsf{fst} \circ \textsf{snd})\,(l)\right), \tag{4.15}$$

$$\textsf{in-y-interval}(l :: \mathbb{R}^2 \times \mathbb{R}^2, y) \iff y \in \textsf{closed-segment}\left((\textsf{snd} \circ \textsf{fst})\,(l), (\textsf{snd} \circ \textsf{snd})\,(l)\right). \tag{4.16}$$

Theorem 4.11. *If the function* $\textsf{parallel-and-not-aligned}(l_1, l_2)$ *is false but* $\textsf{not-aligned}(l_1, l_2)$ *is true, then*

$$\textsf{in-x-interval}(l_1, \textsf{fst}\ p) \implies p \in \textsf{closed-segments}(l_1) ,$$

where $p := \textsf{find-intersection}(l_1, l_2)$ *and assuming* $(\textsf{fst} \circ \textsf{fst})(l_1) \neq (\textsf{fst} \circ \textsf{snd})(l_1)$.

Proof. For ease of reference, we label the x and y elements of l_1 and p as follows:

$$x_1 = (\textsf{fst} \circ \textsf{fst})(l_1), \quad x_2 = (\textsf{fst} \circ \textsf{snd})(l_1), \quad y_1 = (\textsf{snd} \circ \textsf{fst})(l_1), \quad y_2 = (\textsf{snd} \circ \textsf{snd})(l_1), \tag{4.17}$$

$$x = \textsf{fst}(p), \quad y = \textsf{snd}(p) . \tag{4.18}$$

From Theorem 4.9 we know that p is located on the line of segment l_1 and so are both endpoints $\textsf{fst}(l_1)$ and $\textsf{snd}(l_1)$. With (4.5), we can deduce the following equalities:

$$a \cdot x_2 + b \cdot y_2 = a \cdot x_1 + b \cdot y_1 \quad \wedge \quad a \cdot x + b \cdot y = a \cdot x_1 + b \cdot y_1 ,$$

where a, b, c are the coefficients obtained from (4.4) for l_1, and hence

$$y_2 - y_1 = (x_1 - x_2) \cdot \frac{a}{b} \quad \wedge \quad y - y_1 = (x_1 - x) \cdot \frac{a}{b} , \tag{4.19}$$

after some arithmetic manipulations. Note that b cannot be zero due to the premises that the endpoints are not the same.

$$\begin{aligned}
& \textsf{in-x-interval}(l_1, x) \\
\implies & \quad \{ \text{ (4.15) } \} \\
& x \in \textsf{closed-segment}\,(x_1, x_2) \\
\implies & \quad \{ \text{ (4.3) } \} \\
& \exists t \geq 0. \quad t \leq 1 \ \wedge \ x = (1 - t) \cdot x_1 + t \cdot x_2 \\
\implies & \quad \{ \text{ arithmetic } \} \\
& \exists t \geq 0. \quad t \leq 1 \ \wedge \ t \cdot (x_2 - x_1) = x - x_1 \\
\implies & \quad \{ \text{ multiply both sides with } \tfrac{a}{b} \ \}
\end{aligned}$$

$$
\begin{array}{ll}
& \exists t \geq 0. \quad t \leq 1 \ \wedge \ t \cdot (x_2 - x_1) \cdot \dfrac{a}{b} = (x - x_1) \cdot \dfrac{a}{b} \\
\Longrightarrow & \quad \{ \text{ using equalities in (4.19) } \} \\
& \exists t \geq 0. \quad t \leq 1 \ \wedge \ t \cdot (y_2 - y_1) = y - y_1 \\
\Longrightarrow & \quad \{ \text{ arithmetic } \} \\
& \exists t \geq 0. \quad t \leq 1 \ \wedge \ y = (1 - t) \cdot y_1 + t \cdot y_2 \\
\Longrightarrow & \quad \{ \text{ (4.3) } \} \\
& y \in \textit{closed-segment}\,(y_1, y_2) \\
\Longrightarrow & \quad \{ \text{ with fact } x \in \textit{closed-segment}\,(x_1, x_2) \text{ in step 2 of this calculation and fact} \\
& \quad\quad p = \textit{find-intersection}(l_1, l_2) \ \} \\
& p \in \textit{closed-segment}\,(l_1)
\end{array}
$$

□

Theorem 4.12. *If the function parallel-and-not-aligned(l_1, l_2) is false but not-aligned(l_1, l_2) is true and $(\textit{fst} \circ \textit{fst})\, l_1 = (\textit{fst} \circ \textit{snd})\, l_1$ (segment l_1 stands vertical), then*

$$\textit{in-x-interval}(l_1, \textit{fst}(p)) \ \wedge \ \textit{in-y-interval}(l_1, \textit{snd}(p)) \quad \Longrightarrow \quad p \in \textit{closed-segment}(l_1) \ ,$$

where $p := \textit{find-intersection}(l_1, l_2)$.

Proof. For ease of reference, we reuse the definitions in (4.17) and (4.18). Due to the premises $(\textit{fst} \circ \textit{fst})(l_1) = (\textit{fst} \circ \textit{snd})(l_1)$ and $\textit{in-x-interval}(l_1, x)$, we can deduce $x = x_1$ and $x = x_2$.

$$
\begin{array}{ll}
& \textit{in-y-interval}(l_1, y) \\
\Longrightarrow & \quad \{ \text{ (4.16) } \} \\
& y \in \textit{closed-segment}\,(y_1, y_2) \\
\Longrightarrow & \quad \{ \text{ (4.3) } \} \\
& \exists t \geq 0. \quad t \leq 1 \ \wedge \ y = (1 - t) \cdot y_1 + t \cdot y_2 \\
\Longrightarrow & \quad \{ \text{ arithmetic } \} \\
& \exists t \geq 0. \quad t \leq 1 \ \wedge \ y = y_1 + t \cdot (y_2 - y_1) \\
\Longrightarrow & \quad \{ \text{ facts } x = x_2 \text{ and } x = x_1 \ \} \\
& \exists t \geq 0. \quad t \leq 1 \ \wedge \ y = y_1 + t \cdot (y_2 - y_1) \ \wedge \ x = x_1 + t \cdot (x_2 - x_1) \\
\Longrightarrow & \quad \{ \text{ (4.3) } \} \\
& p \in \textit{closed-segments}(l_1)
\end{array}
$$

□

These two theorems show that testing both $\textit{in-x-interval}(l_1, \textit{fst}(p))$ and $\textit{in-y-interval}(l_1, \textit{snd}(p))$ are required to ensure $p \in \textit{closed-segment}(l_1)$. Similarly, we can adapt Theorem 4.11 and Theorem 4.12 by replacing l_1 with l_2 to guarantee $p \in \textit{closed-segment}(l_2)$. With these four theorems, we can then update *segment-intersect* as follows:

```
$\mathit{segment\text{-}intersect}(l_1, l_2)$ :=
    let $p = \mathit{find\text{-}intersection}(l_1, l_2);\quad p_x = \mathit{fst}(p);\quad p_y = \mathit{snd}(p)$ in
    if $\mathit{parallel\text{-}and\text{-}not\text{-}aligned}(l_1, l_2)$ then $\mathit{False}$
    else if $\mathit{not\text{-}aligned}(l_1, l_2)$ then
      $\mathit{in\text{-}x\text{-}interval}(l_1, p_x) \wedge \mathit{in\text{-}x\text{-}interval}(l_2, p_x) \wedge$
      $\mathit{in\text{-}y\text{-}interval}(l_1, p_y) \wedge \mathit{in\text{-}y\text{-}interval}(l_2, p_y)$
    else $\mathit{undefined}$
```

If two segments are aligned (right figure of Fig. 4.4), we only need to check whether their x-interval and y-interval overlaps — $\mathit{overlaps\text{-}x}(l_1, l_2)$ and $\mathit{overlaps\text{-}y}(l_1, l_2)$. Given two intervals $[l_1, u_1]$ and $[l_2, u_2]$ where $l_1 \leq u_1$ and $l_2 \leq u_2$, we can check whether these two overlap by testing $l_2 \leq u_1 \wedge l_1 \leq u_2$. Functions $\mathit{overlaps\text{-}x}(l_1, l_2)$ and $\mathit{overlaps\text{-}y}(l_1, l_2)$ have the following correctness conditions:

$$\begin{aligned} \mathit{overlaps\text{-}x}(l_1, l_2) \iff \exists p.\ & p \in \mathit{closed\text{-}segment}\,((\mathit{fst} \circ \mathit{fst})(l_1), (\mathit{fst} \circ \mathit{snd})(l_1)) \quad \wedge \\ & p \in \mathit{closed\text{-}segment}\,((\mathit{fst} \circ \mathit{fst})(l_2), (\mathit{fst} \circ \mathit{snd})(l_2)) \qquad (4.20) \\ \mathit{overlaps\text{-}y}(l_1, l_2) \iff \exists p.\ & p \in \mathit{closed\text{-}segment}\,((\mathit{snd} \circ \mathit{fst})(l_1), (\mathit{snd} \circ \mathit{snd})(l_1)) \quad \wedge \\ & p \in \mathit{closed\text{-}segment}\,((\mathit{snd} \circ \mathit{fst})(l_2), (\mathit{snd} \circ \mathit{snd})(l_2)) \qquad (4.21) \end{aligned}$$

Theorem 4.13. *If both functions $\mathit{parallel\text{-}and\text{-}not\text{-}aligned}(l_1, l_2)$ and $\mathit{not\text{-}aligned}(l_1, l_2)$ are false, then $\mathit{overlaps\text{-}x}(l_1, l_2)$ and $\mathit{overlaps\text{-}y}(l_1, l_2)$ imply*

$$\exists p.\quad p \in \mathit{closed\text{-}segment}\,(l_1) \ \wedge\ p \in \mathit{closed\text{-}segment}\,(l_2)\ ,$$

assuming $(\mathit{fst} \circ \mathit{fst})(l_1) \leq (\mathit{fst} \circ \mathit{fst})(l_2)$, $\mathit{fst}(l_1) \neq \mathit{snd}(l_1)$, and $\mathit{fst}(l_2) \neq \mathit{snd}(l_2)$.

Proof. Suppose that a_1, b_1, c_1 and a_2, b_2, c_2 are the coefficients (obtained from (4.4)) of the corresponding line equations for segments l_1 and l_2, respectively. Two cases are considered here: $b_1 \cdot b_2 \neq 0$ or $b_1 \cdot b_2 = 0$. In the former case, we can deduce $b_1 \neq 0$ and $b_2 \neq 0$ and we define a constant $k := \frac{a_1}{a_2}$ when $a_2 \neq 0$ or else $k := \frac{b_1}{b_2}$ when $a_2 = 0$; it cannot be the case that both $a_2 = 0$ and $b_2 = 0$ as this would mean $\mathit{fst}\ l_2 = \mathit{snd}\ l_2$. Choosing $p := \mathit{fst}(l_2)$ ensures that $p \in \mathit{closed\text{-}segment}\,(l_2)$ and $a_2 \cdot \mathit{fst}(p) + b_2 \cdot \mathit{snd}(p) = c_2$. Multiplying both sides of this equality with k results in $a_1 \cdot \mathit{fst}(p) + b_1 \cdot \mathit{snd}(p) = c_1$ because $\mathit{not\text{-}aligned}(l_1, l_2)$ is false (see the proof of Theorem 4.10). This equality and the assumption $\mathit{overlaps\text{-}x}(l_1, l_2)$ (and the correctness condition in (4.20)) entail $p \in \mathit{closed\text{-}segment}\,(l_1)$.

In the latter case, both b_1 and b_2 must be equal to zero. This is because $\mathit{not\text{-}aligned}\,(l_1, l_2)$ is false implies $a_1 \cdot b_2 = a_2 \cdot b_1$ and hence $b_1 = 0 \longrightarrow b_2 = 0$ and $b_2 = 0 \longrightarrow b_1 = 0$ (a_1 and b_1 cannot be zero at the same time and so do a_2 and b_2). This means both l_1 and

l_2 stand vertical, i.e., $(\mathit{fst} \circ \mathit{fst})\ l_1 = (\mathit{fst} \circ \mathit{snd})\ l_1 = (\mathit{fst} \circ \mathit{fst})\ l_2 = (\mathit{fst} \circ \mathit{snd})\ l_2$. Premise $\mathit{overlaps\text{-}y}(l_1, l_2)$ ensures that we can obtain y which are located on both segments (cf. the correctness condition in (4.21)). Then, choosing $p := ((\mathit{fst} \circ \mathit{fst})\, l_1, y)$ ensures that $p \in \mathit{closed\text{-}segment}\,(l_1)$ and $p \in \mathit{closed\text{-}segment}\,(l_2)$. □

Theorem similar to Theorem 4.13 with the premise $(\mathit{fst} \circ \mathit{fst})(l_2) \leq (\mathit{fst} \circ \mathit{fst})(l_1)$ instead of $(\mathit{fst} \circ \mathit{fst})(l_1) \leq (\mathit{fst} \circ \mathit{fst})(l_2)$ also exists; choose $p := \mathit{fst}(l_1)$ instead of $p := \mathit{fst}(l_2)$ for the witness of the first case. The complete form of the segment intersection function can now be defined as follows:

```
$\mathit{segment\text{-}intersect}(l_1, l_2)\ :=$
    let $p = \mathit{find\text{-}intersection}(l_1, l_2);\quad p_x = \mathit{fst}\ p;\quad p_y = \mathit{snd}\ p$   in
    if $\mathit{parallel\text{-}and\text{-}not\text{-}aligned}(l_1, l_2)$ then $\mathit{False}$
    else if $\mathit{not\text{-}aligned}(l_1, l_2)$ then
      $\mathit{in\text{-}x\text{-}interval}(l_1, p_x)\ \wedge\ \mathit{in\text{-}x\text{-}interval}(l_2, p_x) \wedge$
      $\mathit{in\text{-}y\text{-}interval}(l_1, p_y)\ \wedge\ \mathit{in\text{-}y\text{-}interval}(l_2, p_y)$
    else
      $\mathit{overlaps\text{-}x}(l_1, l_2)\ \wedge\ \mathit{overlaps\text{-}y}(l_1, l_2)$
```

Theorem 4.14. *For any two segments $l_1, l_2 :: (\mathbb{R}^2 \times \mathbb{R}^2)$ where $\mathit{fst}(l_1) \neq \mathit{snd}(l_1)$ and $\mathit{fst}(l_2) \neq \mathit{snd}(l_2)$, we have*

$$\mathit{segment\text{-}intersect}(l_1, l_2) \iff \exists\, p.\ p \in \mathit{closed\text{-}segment}(l_1)\ \wedge\ p \in \mathit{closed\text{-}segment}(l_2)\ .$$

Proof. We shall prove the soundness ($\Longrightarrow$) part only since we cannot guarantee completeness ($\Longleftarrow$) any longer when we reify this function to that with intervals of floating-point numbers. The soundness of this function is a direct consequence of Theorem 4.11, 4.12, and 4.13. □

4.3.2 Point-in-lanelet test

Consider a lanelet which has the direction to the right and parametrised by *points-le* and *points-ri* as its left and its right boundary, respectively, as shown in Fig. 4.3. Firstly, we need a function to filter irrelevant segments in *points-le* and *points-ri*; this is achieved with the function *find-segment* with the following correctness condition:

$$\begin{aligned} \textsf{find-segment}(\textsf{points} :: (\mathbb{R}^2 \times \mathbb{R}^2)\ \textsf{list}, x :: \mathbb{R}) = \textsf{Some}\ c \implies& \\ c \in \textsf{set}(\textsf{points}) \ \wedge\ \textsf{in-x-interval}(c, x) & \quad (4.22) \\ \textsf{find-segment}(\textsf{points} :: (\mathbb{R}^2 \times \mathbb{R}^2)\ \textsf{list}, x :: \mathbb{R}) = \textsf{None} \implies& \\ \neg \exists c \in \textsf{set}(\textsf{points}).\ \textsf{in-x-interval}(c, x) & \\ & \quad (4.23) \end{aligned}$$

Theorem 4.15. *For any lanelet boundary* $\textsf{points} :: (\mathbb{R}^2 \times \mathbb{R}^2)\ \textsf{list}$*, we have*

$$\exists c.\ \textsf{find-segment}(\textsf{points}, x) = \textsf{Some}\ c \iff \textsf{fst}\,(\textsf{first-point}) \ \leq\ x\ \leq\ \textsf{fst}\,(\textsf{last-point})\ ,$$

where $\textsf{first-point} := (\textsf{fst} \circ \textsf{hd})(\textsf{points})$ *and* $\textsf{last-point} := (\textsf{snd} \circ \textsf{last})(\textsf{points})$*.*

Proof. We shall only prove the '$\Longrightarrow$' part as the '$\Longleftarrow$' part is trivial due to (4.23). From the correctness condition in (4.22), we know that the list $\textsf{points}$ is not empty and hence $\textsf{hd}(\textsf{points})$ and $\textsf{last}(\textsf{points})$ are well-defined. Additionally, we can deduce also that c is one of the chains in $\textsf{points}$ where $(\textsf{fst} \circ \textsf{fst})(c) \ \leq\ x\ \leq\ (\textsf{fst} \circ \textsf{snd})(c)$; this inequality is due to the property of lanelet boundaries being monotone (see Def. 4.7) and (4.15). Monotonicity in lanelet boundaries also ensures that $\textsf{fst}(\textsf{first-point}) \ \leq\ (\textsf{fst} \circ \textsf{fst})(c)$ and $(\textsf{fst} \circ \textsf{snd})(c) \ \leq\ \textsf{fst}(\textsf{last-point})$ which in turn ensures that $\textsf{fst}(\textsf{first-point}) \ \leq\ x\ \leq\ \textsf{fst}(\textsf{last-point})$ by transitivity. □

Secondly, after we know that the x-value of the point is in between *first-point* and *last-point*, we need to ascertain whether the y-value is inside the interval $\textsf{between-setY}(x)$ (see Fig. 4.3). This is achieved by performing a clockwise test to the segment above and a counter-clockwise test to the segment below. Clockwise test (*cw*) for a triple (p_1, p_2, p_3) checks whether the sequence of points in the triple has a clockwise orientation; the counter-clockwise (*counter-cw*) test does the opposite (see Fig. 4.3). Both functions are defined as follows [63]:

$$\textsf{counter-cw}(p_1, p_2, p_3) \ :=\ |p_1 p_2 p_3| > 0\ , \quad (4.24)$$

$$\textsf{cw}(p_1, p_2, p_3) \ :=\ |p_1 p_2 p_3| < 0\ , \quad (4.25)$$

$$|p_1 p_2 p_3| \ :=\ \begin{vmatrix} p_1^x & p_1^y & 1 \\ p_2^x & p_2^y & 1 \\ p_3^x & p_3^y & 1 \end{vmatrix} \ =\ \begin{array}{l} p_1^x \cdot p_2^y + p_1^y \cdot p_3^x + p_2^x \cdot p_3^y - \\ (p_3^x \cdot p_2^y + p_3^y \cdot p_1^x + p_2^x \cdot p_1^y) \end{array} \quad (4.26)$$

Theorem 4.16. *Provided that* $\textsf{fst}(e_1) < \textsf{fst}(e_2)$ *and the corresponding line equation for the segment* (e_1, e_2) *is* $f(x) := m \cdot (x - \textsf{fst}(e_1)) + \textsf{snd}\ e_1$ *with* $m := \frac{\textsf{snd}\ e_2 - \textsf{snd}\ e_1}{\textsf{fst}(e_2) - \textsf{fst}(e_1)}$*, we have*

$$\textsf{counter-cw}(p, e_1, e_2) \iff f(\textsf{fst}(p)) < \textsf{snd}\ p\ , \quad (4.27)$$

$$\textsf{cw}(p, e_1, e_2) \iff \textsf{snd}\ p \ < f(\textsf{fst}(p))\ . \quad (4.28)$$

Proof.

$$\begin{array}{ll}
 & \mathit{counter\text{-}cw}(p, e_1, e_2) \\
\iff & \quad \{ \text{ Definition of } \mathit{counter\text{-}cw} \ \} \\
 & |pe_1e_2| > 0 \\
\iff & \quad \{ \text{ property } |abc| = |\mathbf{0}(b-a)(c-a)| \ \} \\
 & |\mathbf{0}(e_1 - p)(e_2 - p)| > 0 \\
\iff & \quad \{ \text{ Definition in (4.26); arithmetic } \} \\
 & \mathit{fst}(e_2 - p) \cdot \mathit{snd}(e_1 - p) + \mathit{fst}(p - e_1) \cdot \mathit{snd}\,(e_2 - p) < 0 \\
\iff & \quad \{ \text{ arithmetic } \} \\
 & \mathit{snd}\,(e_2 - e_1) \cdot \mathit{fst}(p - e_1) + \mathit{snd}\,(e_1 - p) \cdot \mathit{fst}(e_2 - e_1) < 0 \\
\iff & \quad \{ \text{ divide both sides with } \mathit{fst}(e_2 - e_1)\text{; premise } \mathit{fst}(e_1) < \mathit{fst}(e_2) \ \} \\
 & \dfrac{\mathit{snd}\,(e_2 - e_1)}{\mathit{fst}(e_2 - e_1)} \cdot \mathit{fst}(p - e_1) + \mathit{snd}\,e_1 < \mathit{snd}\ p \\
\iff & \quad \{ \text{ Definition of the line equation } f \ \} \\
 & f(\mathit{fst}(p)) \ < \ \mathit{snd}\ p
\end{array}$$

The proof for (4.28) can be obtained similarly. □

The point-in-lanelet test for a right direction lanelet is formalised by the following function.

```
$point\text{-}in\text{-}lanelet(p)$ :=  let $c_1 = find\text{-}segment(points\text{-}le, p)$;
                            $c_2 = find\text{-}segment(points\text{-}ri, p)$
                        in  if $c_1 = None \lor c_2 = None$ then $False$
                            else $cw(p, (fst \circ the)\ c_1, (snd \circ the)\ c_1) \land$
                                 $counter\text{-}cw(p, (fst \circ the)\ c_2, (snd \circ the)\ c_2)$
```

Theorem 4.17. *For a right-direction lanelet defined in Def. 4.8 with* points-le *and* points-ri *as its left and its right boundary, respectively, we have*

$$\mathit{point\text{-}in\text{-}lanelet}(p) \iff p \in \mathit{drivable\text{-}area}\ .$$

Proof. When the function $\mathit{point\text{-}in\text{-}lanelet}(p)$ is true, Theorem 4.15 ensures that $\mathit{fst}(p) \in \mathit{setX}$. Because we assume that the lanelet has a direction to the right, we have

$$\mathit{between\text{-}setY}(x) \ = \ \left[\mathit{f\text{-}of\text{-}x_r}(x),\ \mathit{f\text{-}of\text{-}x_l}(x)\right]\ .$$

Correctness condition in (4.22) also ensures that $\mathit{in\text{-}x\text{-}interval}(c_1, x)$ and $\mathit{in\text{-}x\text{-}interval}(c_2, x)$ which in turn allows us to replace $\mathit{f\text{-}of\text{-}x_l}(x)$ and $\mathit{f\text{-}of\text{-}x_r}(x)$ with the line equations corresponding to chain c_1 and c_2, respectively. Theorem 4.16 then ensures that $\mathit{snd}\ p \in$

$\mathit{between\text{-}setY}(\mathit{fst}(p))$. Together with the fact $\mathit{fst}(p) \in \mathit{setX}$ obtained previously, we can deduce $p \in \mathit{drivable\text{-}area}$ by unfolding the definition of drivable area in Theorem 4.6.

From the definition of *point-in-lanelet*, if this function is false then it could be the case that either both c_1 and c_2 are *None* or the conjunction in the **else** part is false. If it is the former, Theorem 4.15 ensures that $\mathit{fst}(p) \notin \mathit{setX}$ and, if it is the latter Theorem 4.16 ensures that $\mathit{snd}\ p \notin \mathit{between\text{-}setY}(\mathit{fst}(p))$. In either case, we can deduce that $p \notin \mathit{drivable\text{-}area}$ by using Theorem 4.6. □

4.3.3 Lane detection function

Detecting whether a rectangle intersects with a lanelet boundaries can be checked with the following function.

```
$\mathit{rectangle\text{-}intersect}\,(r, bs\ ::\ (\mathbb{R}^2 \times \mathbb{R}^2)\ \mathit{list}) \quad :=$
        let $ls = \mathit{get\text{-}segments}\ (r);\quad is = \mathit{map}\ (\mathit{segment\text{-}intersect\text{-}boundary}\ bs)\ ls$
        in $((is\ !\ 0)\ \vee\ (is\ !\ 1)\ \vee\ (is\ !\ 2)\ \vee\ (is\ !\ 3))$
$\mathit{segment\text{-}intersect\text{-}boundary}\ (cs\ ::\ \big(\mathbb{R}^2 \times \mathbb{R}^2\big)\ \mathit{list}))\ (s\ ::\ \mathbb{R}^2 \times \mathbb{R}^2) \quad :=$
        if $\mathit{is\text{-}empty}\ cs$ then $\mathit{False}$ else if $\mathit{segment\text{-}intersect}\ (\mathit{hd}\ cs)\ s$ then $\mathit{True}$
        else $\mathit{segment\text{-}intersect\text{-}boundary}\ (\mathit{tl}\ cs)\ s$
```

The function *get-segments*, as its name implies, is the function which returns the list of four edges of a rectangle. Similarly, we also have the function *get-vertices* which returns the list of four extreme points of a rectangle.

Theorem 4.18. *The predicate* $\mathit{rectangle\text{-}intersect}\,(r, \mathit{boundary})$ *is true if and only if*

$$\exists\, c\ p.\quad c \in \mathit{set}(\mathit{boundary})\ \wedge\ p \in \mathit{closed\text{-}segment}(c)\ \wedge\ \mathit{in\text{-}perimeter}(p, r)\ \ ,$$

where in-perimeter *is defined as follows:*

$$\mathit{in\text{-}perimeter}(p, r)\ :=\ \exists l.\quad l \in \mathit{set}(()\mathit{get\text{-}segments}(r))\ \wedge\ p \in \mathit{closed\text{-}segment}\,(l)\ \ .$$

The proof for this theorem can be obtained by using the standard technique of induction over the boundary and the correctness of *segment-intersect* in Theorem 4.14. Detecting whether a rectangle is inside a lanelet meanwhile is formalised by the following function:

$$\begin{aligned}
\textit{rectangle-inside}\,(r) \;&:=\; \textit{vertices-inside}\,(r) \;\wedge\; \textit{segments-no-touch}\,(r)\;\;, \\
\textit{vertices-inside}\,(r) \;&:=\; \textbf{let}\; vs = \textit{get-vertices}\,(r);\quad is = \textit{map point-in-lanelet}\; vs \\
&\;\; \textbf{in}\; (is\;!\;0) \;\wedge\; (is\;!\;1) \;\wedge\; (is\;!\;2) \;\wedge\; (is\;!\;3)\;\;, \\
\textit{segments-no-touch}\,(r) \;&:=\; \textbf{let}\; ls = \textit{get-segments}\,(r);\quad is = \textit{map intersect-boundaries}\; ls \\
&\;\; \textbf{in}\; \neg\,((is\;!\;0) \;\vee\; (is\;!\;1) \;\vee\; (is\;!\;2) \;\vee\; (is\;!\;3))\;\;, \\
\textit{intersect-boundaries} \;&:=\; \lambda x.\;\; \textit{segment-intersect-boundary}\,(\textit{points-ri})\;(x) \\
&\qquad\qquad\qquad \vee\; \textit{segment-intersect-boundary}\,(\textit{points-le})\;(x)\;\;.
\end{aligned}$$

Lemma 4.1. *If there are two points $p_1, p_2 \in$ drivable-area of a lanelet, then*

$$\textit{in-x-interval}((p_1, p_2), x) \quad\Longrightarrow\quad x \;\in\; \textit{setX}\;\;.$$

Proof. Without the loss of generality, we can assume further that $\textit{fst}(p_1) < x < \textit{fst}(p_2)$ from the premise $\textit{in-x-interval}((p_1, p_2), x)$ and (4.15). The proofs for cases where $x = \textit{fst}(p_1)$ or $x = \textit{fst}(p_2)$ are trivial, and the proof for the case where $\textit{fst}(p_2) \leq \textit{fst}(p_1)$ can be obtained similarly. Since we know that $p_1, p_2 \in \textit{drivable-area}$, Theorem 4.6 guarantees that $\textit{fst}(\textit{first-point}) \leq \textit{fst}(p_1)$ and $\textit{fst}(p_2) \leq \textit{fst}(\textit{last-point})$. With the assumption in the beginning of this proof, we can deduce that $x \in \textit{setX}$ by using the transitivity property. □

Lemma 4.2. *If there are two points $p_1, p_2 \in$ drivable-area of a lanelet and the segment (p_1, p_2) does not intersect with either the left or right boundary, i.e., $\neg\textit{intersect-boundaries}\;(p_1, p_2)$, then*

$$p \in \textit{closed-segment}\,(p_1, p_2) \quad\Longrightarrow\quad \textit{snd}\;p \in \textit{between-setY}(\textit{fst}(p))\;\;.$$

Proof. Without the loss of generality, we shall prove this lemma for lanelets with the direction to the right; the proof for the lanelets with the direction to the left can be obtained similarly. Assume that the consequence of the deduction above is false. Then, it is either the case that p is located above the left boundary, $\textit{snd}\;p \geq \textit{f-of-x}_l\;(\textit{fst}(p))$ or below the right boundary, $\textit{snd}\;p \leq \textit{f-of-x}_r\;(\textit{fst}(p))$. In either case, we can use the Intermediate Value Theorem (IVT) in Lem. 2.3 to deduce that there is a point $p \in \textit{closed-segment}\,(p_1, p)$ that intersects the respective boundary; this contradicts with the assumption that there is no intersection with both boundaries. Note that the continuities of both $\textit{f-of-x}_l$ and $\textit{f-of-x}_r$ are guaranteed since both are curves (via Theorem 4.3 and Def. 4.1). □

Theorem 4.19. *If rectangle-inside(r) is true in a lanelet and seg $:: \mathbb{R}^2 \times \mathbb{R}^2$ is one of the four edges of rectangle r, then*

$$p \in \textit{closed-segment}(\textit{seg}) \quad\Longrightarrow\quad p \in \textit{drivable-area}\;\;.$$

Proof. The premise $\mathsf{rectangle\text{-}inside}(r)$ implies the segment seg does not intersect with either the left or the right boundary and both predicates $\mathsf{point\text{-}in\text{-}lanelet}\,(\mathsf{fst}(\mathsf{seg}))$ and $\mathsf{point\text{-}in\text{-}lanelet}\,(\mathsf{snd\ seg})$ are true. With Theorem 4.17, we can deduce further that both $\mathsf{fst}(\mathsf{seg})$ and $\mathsf{snd\ seg}$ are in the drivable area and, together with premise $p \in \mathsf{closed\text{-}segment}\,(\mathsf{seg})$, Lemma 4.1 and 4.2, we can deduce that $p \in \mathsf{drivable\text{-}area}$ by unfolding the definition of drivable area in Theorem 4.6. □

Theorem 4.20. *If $\mathsf{rectangle\text{-}inside}(r)$ is true in a lanelet, then the interior of rectangle r is a subset of the drivable area.*

Proof. Assume that p is located at the interior of rectangle r. This condition implies that there are two different edges of rectangle r — seg_1 and seg_2 — where $\mathsf{in\text{-}x\text{-}interval}(\mathsf{seg}_1, \mathsf{fst}(p))$, $\mathsf{in\text{-}x\text{-}interval}(\mathsf{seg}_2, \mathsf{fst}(p))$, and seg_1 is located above of seg_2. Since the premise $\mathsf{rectangle\text{-}inside}(r)$ guarantees that both endpoints of seg_1 are inside the drivable area, we can deduce further that $\mathsf{fst}(p) \in \mathsf{setX}$ from Lem. 4.1. A vertical line $x = \mathsf{fst}(p)$ will intersect two segments seg_1 and seg_2 at p_1 and p_2, respectively, and $\mathsf{snd}\ p \in \mathsf{closed\text{-}segment}\,(\mathsf{snd}\ p_2, \mathsf{snd}\ p_1)$. Theorem 4.19 stipulates that both $p_1 \in \mathsf{drivable\text{-}area}$ and $p_2 \in \mathsf{drivable\text{-}area}$ and, together with the facts that $\mathsf{fst}(p_1) = \mathsf{fst}(p) = \mathsf{fst}(p_2)$ and seg_1 is above seg_2, we can deduce that $\mathsf{snd}\ p \in \mathsf{between\text{-}setY}\,(\mathsf{fst}(p))$ and hence $p \in \mathsf{drivable\text{-}area}$ according to the definition of drivable area in Theorem 4.6. □

Detecting whether a rectangle is located outside of a lanelet can be formalised as follows:

$$\begin{aligned}
\mathsf{rectangle\text{-}outside}\,(r) &:= \mathsf{vertices\text{-}outside}\,(r) \ \wedge\ \mathsf{segments\text{-}no\text{-}touch}\,(r)\ \ , \\
\mathsf{vertices\text{-}outside}\,(r) &:= \textbf{let}\ vs = \mathsf{get\text{-}vertices}\,(r);\quad is = \mathsf{map\ point\text{-}in\text{-}lanelet}\ vs \\
&\ \textbf{in}\ \neg\,((is\ !\ 0)\ \vee\ (is\ !\ 1)\ \vee\ (is\ !\ 2)\ \vee\ (is\ !\ 3))\ \ .
\end{aligned}$$

Theorem 4.21. *If $\mathsf{rectangle\text{-}outside}(r)$ is true in a lanelet, then $p \notin \mathsf{drivable\text{-}area}$ for all points p located at either at the perimeter or in the interior of rectangle r.*

We omit the proof for this theorem because the proof sketch is very much similar to the proofs of Theorem 4.19 and 4.20. Equipped with these primitives, and their associated lemmas and theorems, function $\mathsf{lane\text{-}detection}\ ::\ \mathsf{rectangle}\ \Rightarrow$

$(\mathbb{R}^2 \times \mathbb{R}^2)$ *list list* $\Rightarrow$ *detection-opt* can be defined as follows:

```
lane-detection (rect, bss) :=
    if rectangle-outside (hd bss) (last bss) rect then Outside
    else if check-inside bss rect $0$ = Some ($n$) then Lanelet ($n$)
    else Boundaries (check-boundaries bss rect $0$)
check-inside (r # l # bss) rect $n$ :=
    if rectangle-inside r l rect then Some ($n$)
    else if $\neg$ (is-empty bss) then check-inside (l # bss) rect $(n+1)$ else None
check-boundaries (bs # bss) rect $n$ :=
    if rectangle-intersect (rect, bs) then $n$ # check-boundaries bss rect $(n+1)$
    else check-boundaries bss rect $(n+1)$
```

The lane detection function starts by checking whether the rectangle is located outside the whole lane. We utilise the previously defined *rectangle-outside* function for checking whether a rectangle is located outside of a specific lanelet. Due to the way a lane is defined (Def. 4.9), we check whether a rectangle is outside by using the rightmost and leftmost lane of a lane (*hd bss* and *last bss* in the definition above) as *points-ri* and *points-le*. Theoretically speaking, the result of a lane detection must be either *Lanelet* (n), *Boundaries* (ns), or *Outside*, where *ns* is not an empty list. However, due to limited numerical precision, it might be the case that none of these cases is returned by this function; this happens when the argument for the constructor *Boundaries* is an empty list.

4.4 Interpretation with lane detection and safe distance checker

Provided with a trace (a sequence of values such as position, orientation, and speed) and a lane (a list of lanelet boundaries) how can we determine time points t_1, t_2, t_3 and t_4 with *lane-detection*? Remember that we need these time points to define atomic propositions such as *begin-overtaking* and *merging*. Since the function *lane-detection* requires a list of rectangles as its second argument (see the last part of Sec. 4.3.3), we must first convert the trace into a list of rectangles[1]. Taking the values of position and orientation from the trace, we translate a rectangle with fixed length and width by this position and then rotate the resulting rectangle with the orientation.

Provided with a list of rectangles *rects* and *lane*, we can approximate t_1, t_2, t_3, and t_4 by using linear search. Time t_1 is the first time a vehicle touches the lane boundary

[1] Note that the lane detection check is only done for discrete time

which can be found by the following function (the base case where the list is empty is straightforward):

Definition 4.10.

```
start-inc-lane (rect # rects) $l$ idx  :=
    case lane-detection (rect, lane) of
        Boundaries $[l + 1]$ $\Rightarrow$ Some(idx, rects)
    |   Lanelet $l$ $\Rightarrow$ start-inc-lane rects $l$ $(idx + 1)$
    |   _ $\Rightarrow$ None
```

Theorem 4.22. *If* $\mathit{start\text{-}inc\text{-}lane}(\mathit{rects}, \mathit{ori\text{-}lane}, \mathit{start\text{-}idx}) = \mathit{Some}(\mathit{idx}_1, r)$ *then the smallest* n *to satisfy the following conditions is* idx_1.

$$\begin{aligned}
& \mathit{start\text{-}idx} \leq n \\
\wedge \quad & \mathit{lane\text{-}detection}(\mathit{rects} \; ! \; (n - \mathit{start\text{-}idx})) = \mathit{Boundaries} \; [\mathit{ori\text{-}lane} + 1] \\
\wedge \quad & \forall m. \quad 0 \leq (m - \mathit{start\text{-}idx}) \; < \; (n - \mathit{start\text{-}idx}) \; \longrightarrow \\
& \qquad\qquad \mathit{lane\text{-}detection} \, (\mathit{rects} \; ! \; (m - \mathit{start\text{-}idx})) = \mathit{Lane}(\mathit{ori\text{-}lane})
\end{aligned}$$

Proof. We first show that idx_1 satisfies the condition above. The definition of *start-inc-lane* clearly shows that $\mathit{start\text{-}idx} \leq \mathit{idx}_1$ as successive evaluations of this function either increases the time parameter (the third argument) or leave it unchanged. Lane detection at $\mathit{rects}\,!\,(\mathit{idx}_1 - \mathit{start\text{-}idx})$ is $\mathit{Boundaries}\,[\mathit{ori\text{-}lane} + 1]$ because the guard condition for $\mathit{Some}(\mathit{idx}_1, \mathit{rects})$ in the case distinction is $\mathit{Boundary}\;[l + 1]$ and, according to the premise, $l = \mathit{ori\text{-}lane}$; the third condition is true by using induction over *rects*.

Next, we show that for all n which satisfies the condition above, we can deduce $\mathit{idx}_1 \leq n$. Suppose that n satisfies those three conditions and $n < \mathit{idx}_1$ and we subtract both sides with *start-idx* into $n - \mathit{start\text{-}idx} < \mathit{idx}_1 - \mathit{start\text{-}idx}$ ($\mathit{start\text{-}idx} \leq n$ according to the first condition). Then, by using the third condition (the universal quantification), we can deduce that the lane detection at $n - \mathit{start\text{-}idx}$ is $\mathit{Lane}(\mathit{ori\text{-}lane})$ which contradicts with the second condition in the assumption. Hence, it must be the case that $\mathit{idx}_1 \leq n$. With the first sub-proof, we can deduce that idx_1 is the smallest index to satisfy the three conditions in the theorem. □

Apart from returning the index pointing to t_1, the function *start-inc-lane* also returns the rest of the list where it has yet to process; we feed this list to the next function to find t_2.

Definition 4.11.

```
finish-inc-lane (rect # rects) $b$ $t$    :=
    case lane-detection (rect, lane) of
        Boundaries [$b$]  ⇒  finish-inc-lane rects $b$ $(t+1)$
    |   Lanelet $b$  ⇒  Some($t$, rects)
    |   _  ⇒  None
```

Theorem 4.23. *If finish-inc-lane(rects, bound-id, start-idx) = Some(idx_2, r) then the smallest n to satisfy the following conditions is idx_2.*

$$\begin{aligned} & \textit{start-idx} \le n \\ \wedge\ & \textit{lane-detection}(\textit{rects} \,!\, (n - \textit{start-idx})) = \textit{Lane}\ (\textit{bound-id}) \\ \wedge\ & \forall m.\ \ 0 \le (m - \textit{start-idx}) < (n - \textit{start-idx}) \longrightarrow \\ & \qquad \textit{lane-detection}\,(\textit{rects} \,!\, (m - \textit{start-idx})) = \textit{Boundaries}\ [\textit{bound-id}] \end{aligned}$$

Since times t_3 and t_4 are similar to t_1 and t_2, respectively, we can define functions *start-dec-lane* and *finish-dec-lane* to detect t_3 and t_4, correspondingly, as follows:

Definition 4.12.

```
start-dec-lane (rect # rects) $l$ $t$ :=
    case lane-detection (rect, lane) of
        Boundaries [$l$]  ⇒  Some($t$, rects)
    |   Lanelet $l$  ⇒  start-dec-lane rects $l$ $(t+1)$
    |   _  ⇒  None
```

Theorem 4.24. *If start-dec-lane(rects, next-lane, start-idx) = Some(idx_3, r) then the smallest n to satisfy the following conditions is idx_3.*

$$\begin{aligned} & \textit{start-idx} \le n \\ \wedge\ & \textit{lane-detection}(\textit{rects} \,!\, (n - \textit{start-idx})) = \textit{Boundaries}\ [\textit{next-lane}] \\ \wedge\ & \forall m.\ \ 0 \le (m - \textit{start-idx}) < (n - \textit{start-idx}) \longrightarrow \\ & \qquad \textit{lane-detection}\,(\textit{rects} \,!\, (m - \textit{start-idx})) = \textit{Lane}(\textit{next-lane}) \end{aligned}$$

Definition 4.13.

```
finish-dec-lane (rect # rects) $b$ $t$    :=
    case lane-detection (rect, lane) of
        Boundaries [$b$]  ⇒  finish-dec-lane rects $b$ $(t+1)$
    |   Lanelet $(b-1)$  ⇒  Some($t$, rects)
    |   _  ⇒  None
```

Theorem 4.25. *If $\textit{finish-dec-lane}(\textit{rects}, \textit{bound-id}, \textit{start-idx}) = \textit{Some}(\textit{idx}_4, r)$ and $0 < \textit{bound-id}$, then the smallest n to satisfy the following conditions is $\textit{idx}_4$.*

$$\begin{aligned} & \textit{start-idx} \le n \\ \wedge\quad & \textit{lane-detection}(\textit{rects} \,!\, (n - \textit{start-idx})) = \textit{Lane}\ (\textit{bound-id} - 1) \\ \wedge\quad & \forall m.\ \ 0 \le (m - \textit{start-idx}) \ <\ (n - \textit{start-idx}) \ \longrightarrow \\ & \qquad\qquad \textit{lane-detection}\,(\textit{rects} \,!\, (m - \textit{start-idx})) = \textit{Boundaries}\,[\textit{bound-id}] \end{aligned}$$

The difference between *start-inc-lane* and *start-dec-lane* is that the relationship between the origin and the destination lane are inverted. In case the constructor *Boundary* is matched during a pattern matching, the former stops searching when the argument to this constructor is $l + 1$ while the latter l; similar argument also applies when we compare *finish-inc-lane* and *finish-dec-lane*. Assuming that the initial lanelet is n, we can combine these four functions to detect t_1, t_2, t_3, and t_4 as follows:

Definition 4.14.

$$\boxed{\begin{array}{llllllr} \textit{overtaking}\ \textit{rects}_0 \ := & \textbf{\textit{do}}\,\{ & (\textit{idx}_1, \textit{rects}_1) \ \leftarrow & \textit{start-inc-lane} & (\textit{rects}_0, & n, & 0); \\ & & (\textit{idx}_2, \textit{rects}_2) \ \leftarrow & \textit{finish-inc-lane} & (\textit{rects}_1, & n+1, & \textit{idx}_1 + 1); \\ & & (\textit{idx}_3, \textit{rects}_3) \ \leftarrow & \textit{start-dec-lane} & (\textit{rects}_2, & n+1, & \textit{idx}_2 + 1); \\ & & (\textit{idx}_4, \textit{rects}_4) \ \leftarrow & \textit{finish-dec-lane} & (\textit{rects}_3, & n, & \textit{idx}_3 + 1); \\ & & \multicolumn{4}{l}{\textit{Some}\ (\textit{idx}_1, \textit{idx}_2, \textit{idx}_3, \textit{idx}_4, \textit{rects}_4)} & \} \end{array}}$$

So what are t_1, t_2, t_3, and t_4 exactly? If we have $\textit{Some}(\textit{idx}_1, \textit{idx}_2, \textit{idx}_3, \textit{idx}_4, \textit{rects}_4)$ as the result of *overtaking rects*, then we can deduce the following condition with Thm. 4.22:

$$\begin{aligned} 0 \le \textit{idx}_1 \ \wedge\ (\forall m \ge 0.\ \ m < \textit{idx}_1 \ \longrightarrow\ & \textit{lane-detection}\,(\textit{rects} \,!\, m) = \textit{Lane}(n)) \\ \wedge\quad & \textit{lane-detection}(\textit{rects} \,!\, \textit{idx}_1) = \textit{Boundaries}\ [n+1]\ . \end{aligned}$$

Mentally replacing $\textit{idx}_1$ with t_1 — even though these two identifiers are of different types — provides a rough approximation of translating the definition of t_1 (the time at which the ego vehicle touches the lane boundary for the first time) in a logical formula. Similarly, Thm. 4.23 allows us to deduce the following condition:

$$\begin{aligned} \textit{idx}_1 < \textit{idx}_2 \wedge (\forall m \ge \textit{idx}_1.\ \ m < \textit{idx}_2 \ \longrightarrow\ & \textit{lane-detection}\,(\textit{rects} \,!\, m) = \textit{Boundaries}\ [n+1]) \\ \wedge\quad & \textit{lane-detection}(\textit{rects} \,!\, \textit{idx}_2) = \textit{Lane}(n+1)\ , \end{aligned}$$

which is the formal translation of time t_2 (the first time since t_1 the ego vehicle is completely located at the next lane). As for t_3, we can use Thm. 4.24 to translate its informal definition (the earliest time since t_2 to touch the lanelet boundary) as follows:

$$\begin{aligned} \textit{idx}_2 < \textit{idx}_3 \ \wedge\ (\forall m \ge \textit{idx}_2.\ \ m < \textit{idx}_3 \ \longrightarrow\ & \textit{lane-detection}\,(\textit{rects} \,!\, m) = \textit{Lane}(n+1)) \\ \wedge\quad & \textit{lane-detection}(\textit{rects} \,!\, \textit{idx}_3) = \textit{Boundaries}\ [n+1]\ . \end{aligned}$$

Finally, Thm. 4.25 allows us to translate time t_4 (the earliest time since t_3 to be completely located inside the original lane) as follows:

$$\begin{aligned} \mathit{idx}_3 \leq \mathit{idx}_4 \wedge (\forall m \geq \mathit{idx}_3.\quad m < \mathit{idx}_4 \longrightarrow \textsf{lane-detection}\,(\mathit{rects}\ !\ m) = \textsf{Boundaries}\,[n+1]) \\ \wedge \quad \textsf{lane-detection}(\mathit{rects}\ !\ \mathit{idx}_4) = \textsf{Lane}(n)\ . \end{aligned}$$

Labelling. After we have obtained these time indices, it is easy to construct a run which is suitable for LTL monitoring. For example, the run for `overtaking` is

$$\textsf{replicate}(\mathit{idx}_1, \emptyset)\quad @\quad \textsf{replicate}(\mathit{idx}_4 - \mathit{idx}_1, \{\texttt{overtaking}\})\quad @\quad \textsf{replicate}(|\mathit{rects}| - \mathit{idx}_4, \emptyset)\ ,$$

and the run for `begin-overtaking` is

$$\textsf{replicate}(\mathit{idx}_1, \emptyset)\quad @\quad \textsf{replicate}(\mathit{idx}_2 - \mathit{idx}_1, \{\texttt{begin-overtaking}\})\quad @\quad \textsf{replicate}(|\mathit{rects}| - \mathit{idx}_2, \emptyset)\ .$$

It is not difficult to translate these constructions to those corresponding to atomic propositions `finish-overtaking` and `merging`. Finally, to obtain a complete run with respect to these four atomic propositions, we could use a pointwise union between all of these four runs.

In addition to these four atomic propositions, we have also identified atomic propositions `sd-rear` and `safe-to-return` in our codification (cf. Sec. 4.1). For `sd-rear`, we need a function to identify traffic participants that are behind the ego vehicle and occupy relevant lanes and boundaries, i.e., vehicles which might be endangered if it performs an overtaking manoeuvre. The former can be done by comparing the location of other traffic participants with respect to the location of the ego vehicle, i.e., checking whether $(\textsf{fst} \circ \textsf{centre})(\mathit{rect}_{tp})) < (\textsf{fst} \circ \textsf{centre})(\mathit{rect}_{ego})$ (see Sec. 3.5.1 for the concrete data type of rectangle). The latter can be achieved by using the following function:

$$\begin{array}{l}
\textbf{fun}\ \textsf{is-relevant}\ ::\ \textsf{detection-opt} \Rightarrow \textsf{detection-opt} \Rightarrow \textsf{bool}\ \textbf{where} \\
\quad \textsf{is-relevant Outside \_} \longleftrightarrow \textsf{False} \\
|\quad \textsf{is-relevant \_ Outside} \longleftrightarrow \textsf{False} \\
|\quad \textsf{is-relevant}\ (\textsf{Lanelet}\ x)\ (\textsf{Lanelet}\ y) \longleftrightarrow (x = y) \\
|\quad \textsf{is-relevant}\ (\textsf{Lanelet}\ x)\ (\textsf{Boundaries}\ ys) \longleftrightarrow (x \in \textsf{set}((\textsf{relevant-lanelets}\ ys))) \\
|\quad \textsf{is-relevant}\ (\textsf{Boundaries}\ xs)\ (\textsf{Lanelet}\ y) \longleftrightarrow (y \in \textsf{set}((\textsf{relevant-lanelets}\ xs))) \\
|\quad \textsf{is-relevant}\ (\textsf{Boundaries}\ xs)\ (\textsf{Boundaries}\ ys) \longleftrightarrow \\
\qquad\qquad\qquad \textsf{set}((\textsf{relevant-lanelets}\ xs)) \cap \textsf{set}((\textsf{relevant-lanelets}\ ys)) \neq \emptyset
\end{array}$$

The first and the second argument to the function *is-relevant* is the location of the ego and the other vehicle, respectively. This function makes use of function $\textsf{relevant-lanelets}(xs)$ which finds lanelets whose boundaries are listed in the list of boundary identifiers xs.

- When either of the vehicles are located outside of the lane, then the other vehicle is not relevant for checking `sd-rear` any more.
- When the ego vehicle is located at *Lanelet*(x), then other vehicles which are located at the same lanelet or touches one of the boundaries of *Lanelet*(x) are relevant.
- When the ego and the other vehicle are occupying *Boundaries*(xs) and *Boundaries*(ys), respectively, then the other vehicle is relevant for `sd-rear` if and only if their sets of affected lanelets overlap.

After relevant traffic participants are identified with these two functions, the atomic proposition `sd-rear` is asserted at the current time only if the ego vehicle maintains safe distances to all of these traffic participants. For atomic proposition `safe-to-return`, we need to keep track of the other vehicle being overtaken. This vehicle can be easily identified as the closest vehicle in front of the ego vehicle occupying the original lane at time t_1. The atomic proposition `safe-to-return` is asserted when the ego vehicle has left a safe distance for the vehicle being overtaken.

The safe distance checker in Ch. 2 is parameterised by the position, speed, and maximum deceleration of the ego vehicle and the front vehicle. For asserting atomic proposition `safe-to-return`, parameters for the ego vehicle in the safe distance checker are actually replaced by the position, speed, and maximum deceleration of the vehicle being overtaken and, vice versa, parameters for the front vehicle by those values of the ego vehicle. Therefore, Assumption 2.3 in Sec. 2.1 are not satisfied before time t_1 and we circumvent this situation by returning *False* to the situations where Assumption 2.3 are not met; this fits naturally with our interpretation of `safe-to-return`.

4.5 Sound and executable LTL monitoring of traffic rules

In this section, we shall provide a numerical example of how to use the monitoring framework explained in previous sections. The purpose of this example is to show that what we have formalised is not a mere intellectual exercise, but one can use it to evaluate real scenarios. Similar to what we have done in previous chapters, the primary challenge in this section is to handle real numbers soundly. However, unlike what we have previously done, this chapter uses a slightly different approach: instead of representing each variable with a pair of floating-point numbers enclosing the real value, we assume that each real variable is a floating-point number.

Figure 4.5 illustrates the scenario used in this section. The solid curve in the figure is the trajectory of the ego vehicle we want to monitor (obtained from an off-the-shelf motion planner), and all vehicles drive in the right direction. Ego vehicle (solid rectangle) is positioned at $(0, 0)$ and the first vehicle (double rectangle) at $(-25, 4.5)$ initially. Both

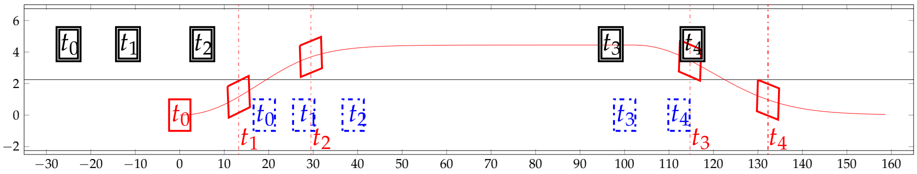


Figure 4.5: An example of concrete overtaking scenario.

vehicles have a constant speed of $16.7\,\mathrm{m\,s^{-1}}$. The second vehicle (dashed rectangle) is located at $(19, 0)$ with an initial speed of $11.1\,\mathrm{m\,s^{-1}}$. All vehicles have the length l of $5\,\mathrm{m}$, width w of $2\,\mathrm{m}$, maximum deceleration of $-8\,\mathrm{m\,s^{-2}}$, and reaction time of $1\,\mathrm{s}$. For each vehicle we show the position at time $t_1 = 0.8\,\mathrm{s}$, $t_2 = 1.8\,\mathrm{s}$, $t_3 = 7.3\,\mathrm{s}$, and $t_4 = 8.4\,\mathrm{s}$, which are the time points detected by the lane detection function.

Obtaining over-approximative rectangles. The previous section assumes that the trace is an ideal trace. In practice, the trace contains only data at certain sampled time points only. This situation is problematic because we do not know how the vehicle behaves in between these time points. For example, suppose that the lane detection return $\mathit{Lane}(0)$ at both sampled times t and t' and it records nothing in between these time points. If a definition requires a vehicle to stay in lane 0 during $[t, t']$, its validity could be jeopardised because it is possible that the vehicle touches $\mathit{Boundaries}[0]$ in between these time points. Of course, the smaller the sampling interval is, the less likely this will happen. However, only reducing the likelihood of these unsound behaviours is still not acceptable in any formal analysis.

Chapter 3 safely approximates the behaviours in between these sampled time points. By feeding the function with two time points t and t' alongside with other pertinent data such as position, speed, and maximum acceleration, it computes the polygon which is formally guaranteed to contain all possible behaviours of the vehicle in between time t and t'. Some people would argue that the dynamics used to guarantee this claim is too simplistic for such a claim. However, note that variables used in this computation of occupancy are essentially intervals which account for uncertainties; simple dynamics models which account for uncertainties could be as valid as high fidelity models [64]. The polygon computed by the occupancy prediction can be over-approximated further by a rectangle to fit the first argument of *lane-detection*.

Another threat to the validity of this numerical example is the choice of using single floating-point numbers to represent the values of variables of type $\mathbb{R}$. There are four measured variables of type $\mathbb{R}$ in this numerical example: position, orientation, speed, and maximum deceleration. For the first two variables, we mitigate this threat by enlarging the corresponding rectangle soundly as what we have done in Sec. 3.5.1. Hence, the vehicle's dimension of $5\,\mathrm{m} \times 2\,\mathrm{m}$ is actually the enlarged version of the actual vehicle's dimension due to uncertain position and orientation. As for the variables of speed and maximum deceleration, we argue that it is not that difficult to extend to that of the interval version since the checker in Ch. 2 can handle intervals already.

Handling rotations soundly. Note that the trace consists of the positions and orientations of each traffic participant over time. The lane detection, however, operates on rectangles instead of positions and orientations. To bridge the gap between these representations, we first create a template rectangle with the length and width specified above, position at $(0, 0)$ (its centre), and orientation of zero. Therefore, given a position and an orientation, we translate this template rectangle to this position and rotate it by this orientation. Translations can be done exactly for floating-point numbers without any rounding operations, but rotations are more involved due to the occurrences of transcendental function sine and cosine in the rotation matrix. As Ch. 3.5.2 shows, we can over-approximate these transcendental functions, but we shall have four boxes instead of four vertices in the rotated version due to the uncertainties.

The root cause of this rotation problem is that rectangles are not closed under the approximated rotation function. In order to rectify this problem, we resolve to Affine Arithmetic which has been formalised in ISABELLE by Immler [28]. Immler's formalisation ensures that an affine form is closed under the transcendental functions sine and cosine. Since we can view the template rectangle in its affine form, i.e. its centre at $(0, 0)$ and its two generators $g_1 = (\frac{l}{2}, 0)$ and $g_2 = (0, \frac{w}{2})$, we can utilise this formalisation to provide us with an over-approximated version of the rotated rectangle. However, there is no guarantee that the affine form obtained from this rotation is of order two: the number of vertices and edges could be more than those of rectangles. This situation poses no difficulties for lane detection because the underlying principles for checking either a rectangle or a convex object is inside a lanelet is the same: *1)* check whether all of their vertices are inside the lanelet, and *2)* ensure none of their edges intersects with any segment of the lanelet boundaries.

Up to now, we can always over-approximate the values of variables of type $\mathbb{R}$ with floating-point numbers so that the function is executable and its soundness is still guaranteed by sacrificing the completeness property. This is not the case for detecting overtaking: when we want to determine t_1, which is the first time a vehicle touches the lane divider at the beginning of overtaking phase, over-approximating rectangles causes the actual time a vehicle touches the lane divider to be slightly later than t_1. This is acceptable in practice because touching the lane divider entails an *obligation* to maintain a safe distance to the closest vehicle behind in the fast lane. Hence, if we detect t_1 earlier than it supposed to be (as in our case here), one obtains a larger distance at the actual time, which leads to a safer situation. Had we under-approximated the rectangle, t_1 would be detected later, and the distance would be shorter at the actual time, which leads to a more dangerous situation.

Executable semantics of LTL. We have previously codified the traffic rules in LTL without discussing the semantics of LTL. Standard LTL [65] is interpreted over infinite length word, and the monitoring process usually requires the construction of monitoring automata [66]. However, because the traces we wish to monitor usually have a finite duration, we use the finite-length semantics of LTL [67] as the basis for monitoring:

Definition 4.15 (Semantics of LTL over finite-duration traces)**.** *Given a list of a set of atomic proposition ξ, an LTL formula ϕ from the grammar*

$$\phi \;::=\; \mathsf{True} \mid q \mid \neg\phi \mid \phi \wedge \phi \mid X\phi \mid \phi\, U\, \phi \enspace ,$$

where q is an atomic proposition, a function drop(i,xs) *which drops i elements from the beginning of the list xs, we define $\xi \models \phi$ as follows:*

$$\begin{aligned}
\xi &\models \mathsf{True} && \iff && \mathit{true} \enspace , \\
\xi &\models q && \iff && q \in (\xi \,!\, 0) \enspace , \\
\xi &\models \neg\phi && \iff && \mathit{not}\ \xi \models \phi \enspace , \\
\xi &\models \phi \wedge \psi && \iff && \xi \models \phi\ \mathit{and}\ \xi \models \psi \enspace , \\
\xi &\models X\phi && \iff && \mathit{if}\ \mathsf{drop}(1,\xi)\ \mathit{is\ defined\ then}\ \mathsf{drop}(1,\xi) \models \phi \enspace , \\
\xi &\models \phi\, U\, \psi && \iff && (\exists i < |\xi|.\ \mathsf{drop}(i,\xi) \models \psi \ \wedge\ (\forall j < i.\ \mathsf{drop}(j,\xi) \models \phi)) \enspace .
\end{aligned}$$

We can also define other operators $\mathsf{False}, \vee, \longrightarrow, \longleftrightarrow, F, G$ *as follows:*

$$\mathsf{False} \;:=\; \neg\,\mathsf{True}, \quad \phi \vee \psi \;:=\; \neg(\neg\phi \wedge \neg\psi), \quad \phi \longrightarrow \psi \;:=\; \neg\phi \vee \psi,$$
$$\phi \longleftrightarrow \psi \;:=\; \phi \longrightarrow \psi \wedge \psi \longrightarrow \phi, \quad F\phi \;:=\; \mathsf{False}\, U\, \phi, \quad \mathit{and} \quad G\phi \;:=\; \neg F(\neg\phi) \enspace .$$

Since the semantics above is defined recursively and traces ξ are represented by lists, ISABELLE can generate codes easily (to Standard ML) for this definition, and we use them to monitor our numerical example. Thus, we only have to trust ISABELLE's code generator for the correctness of the code. The result of the simulation shows that all overtaking traffic rules except the merging rule (Φ_2) are satisfied. Particularly, rule Φ_2 is not satisfied due to the 'if' ($\longleftarrow$) fragment. If we weaken rule Φ_2 into $\Phi_2' := G(\texttt{merging} \longrightarrow \texttt{safe-to-return})$, the trace will satisfy Φ_2'.

4.6 Related Work

The monitoring part of our work belongs to the research area called *runtime verification*; Küster [68] provides a complete overview of this research area. Specifically, our work can be categorised as *runtime monitoring*. Our work does not construct a monitor automaton as in most monitoring techniques [69, 70, 66] but simply executes the semantics of LTL over finite-length traces; our approach belongs to the category of rewriting-based technique [67]. Of course, it is possible to use the monitor automata approach mentioned above for monitoring traffic rules formalised in LTL. This thesis however opts for the rewriting-based technique because we want to fully utilise the code generation technology in ISABELLE. Rewriting-basd technique is sufficient because we wish to verify traces produced by autonomous vehicles' planners whose duration are usually not very long. The other intended application of our work is to perform automated offline checking of a recorded trace for compliance with traffic rules.

In terms of the logic used for specifying properties, there is signal temporal logic (STL) [71] which is expressive enough to specify real-time properties. This is particularly useful for relaxing the requirement to satisfy a rule within a certain duration of time such as in the 'if' part of Φ_2. Another expressive logic for runtime monitoring is metric first-order temporal logic (MFOTL) [72] which is capable of handling relations that change over time such as safe distance. However, we stick to logic without first-order fragment because we do not need to reason about first-order structure for formalising the presented subset of traffic rules.

We could also argue that this work belongs to the category of computational law. Buchanan and Headrick [73] could be considered as the first to propose a serious effort in formalising law. Historically speaking, this is the period when expert systems were popular in the domain of Artificial Intelligence (cf. the history of AI in [74]). Therefore, it is not surprising that they suggested to formalise law with the technique used in expert systems, that is, with *rules*. Additionally, there are also other works which use other computational structures such as algorithm, flow chart, and decision nets to formalise law; interested readers should consult the work of Sergot [75] for further details.

Two major milestones for formalising law are the works of Sergot et al. [76] and Bench-Capon et al. [77] in which they formalised the British Nationality Act and the Supplementary Benefit Act, respectively, with Horn fragments of First-Order Logic (Horn clauses) in PROLOG [78]. Compared to our approach, Horn clauses have limited expressiveness for defining a primitive concept [79]. For example, using Horn clauses to formalise the legal sentence "Driving should be on the *rightmost lane* if possible, except for *overtaking*" with Horn clauses, forces us to assume that the primitive concepts *rightmost lane* and *overtaking* can always be determined. Unfortunately, for monitoring purpose, we have to base our formalisation with primitive concepts such as positions, orientations, speeds, and accelerations; not only the high-level concepts such as 'overtaking' and 'rightmost lane' directly.

Apart from propositional and first-order logic, there is also deontic logic [80] for formalising law which can express the notions of *permission* and *obligation* explicitly. These two notions can be easily recognised in legal documents by identifying the keywords 'may' and 'must', respectively. In relation to the formalisation of traffic rules, this logic was extended by Royakkers [81], and he showed how that the extended logic can address the issue of conflicting speed limits in Dutch Traffic Regulation 1990. Compared to our approach, deontic logic is more suitable to prove the consistency of a legal text than to monitor the compliance of an autonomous vehicle. This observation is in-line with the study from Jones and Sergot [82] in which they discuss when to use and not to use deontic logic for formalising law.

5 Formal verification of motion planners based on manœuvre automata

Autonomous vehicles' planning and control are hard. Not only are they required to consider complex vehicle dynamics, but they must also deal with possibly unknown and dynamically changing environments. To tackle these complexities, most symbolic motion planners abstract continuous systems by discrete representations in either an *environment-driven* [83, 84] or a *controller-driven* manner [85, 86]. The former partitions the environment into cells, such as triangles or squares, while the latter partitions the controller into several primitives, such as `turn-left` or `turn-right`. Which discretisation is preferred for autonomous vehicles?

Environment-driven discretisation is preferred when *1)* we have static, *a priori* known, and geometrically complex environments; or *2)* we have to handle expressive specifications, such as those expressed in Linear Temporal Logic (LTL). However, environment-driven discretisation usually works only for systems with simple dynamics [87]. On the contrary, controller-driven discretisation is preferred when we have dynamic, possibly unknown, and geometrically simple environments. Controllers designed with this discretisation can handle complex dynamics and navigate the environment by chaining a series of well-tested motion primitives [86]. However, specification languages for this discretisation, such as in [88], are very close to the implementation level; often, we want to specify *what* to achieve rather than *how* to achieve it.

Most vehicle models for autonomous vehicles are complex, making controller-driven discretisation a natural choice. In this chapter, we shall use manoeuvre automata-based motion planners [89] (Sec. 5.2), where each motion primitive is encoded as a state in our manoeuvre automaton. However, autonomous vehicles operate in dynamic and possibly unknown environments, where they could benefit from specification languages such as LTL (Sec. 5.3) which is usually associated with environment-driven discretisation. This chapter aims to combine the advantages of both discretisation strategies by interpreting LTL over manoeuvre automata. To the best of our knowledge, this is the first work to tackle this challenge.

This chapter is based on joint work with Fabian Immler, Bastian Schürmann, and Matthias Althoff [90].

5.1 Affine forms and zonotopes for representing sets

This chapter deals with the formal verification of manœuvre-automata-based motion planners for autonomous vehicles using *reachability analysis*. Given a particular system represented by its Ordinary Differential Equations (ODE), one can use reachability analysis to compute the *sets* of states reachable from a particular set of initial states. Since we shall frequently refer to the concrete data structure used to represent these sets, i.e., *affine forms* and their geometrical shape *zonotopes*, we introduce them briefly.

An affine form A is defined by a sequence $(A_i)_{i \in \mathbb{N}}$ with only finitely many nonzero elements. We write A_i to refer to the i-th element of the affine form A; A_0 is its centre and the remaining A_i are its generators. An affine form is interpreted for a valuation $\varepsilon :: \mathbb{N} \to [-1, 1]$ as:

$$[\![A]\!]_\varepsilon := A_0 + \sum_i \varepsilon_i \cdot A_i \ .$$

One calls the terms ε_i noise symbols which are usually unknown and they represent independent components of uncertainties. Noise symbols could be shared between affine forms and they are treated symbolically: the sum of two affine forms is given by the pointwise sum of their generators, and multiplication with a constant factor is also done componentwise:

$$[\![(A+B)]\!]_\varepsilon \ := \ (A_0 + B_0) + \sum_i \varepsilon_i \cdot (A_i + B_i) \ , \tag{5.1}$$

$$[\![(k \cdot A)]\!]_\varepsilon \ := \ k \cdot A_0 + \sum_i \varepsilon_i \cdot (k \cdot A_i) \ . \tag{5.2}$$

To illustrate this symbolic treatment, compare the result of subtracting variable A by itself when it has intervals and affine forms as its concrete data structure. Subtracting $A = [A_l, A_u]$ by itself, according to the subtraction rule in interval arithmetic, results in $[A_l - A_u, A_u - A_l]$ which has the measure[1] twice as large as its original measure. This result does not conform to our intuition that subtracting a variable by itself should

[1] The measure of an interval is obtained by subtracting the lower bound from the upper bound.

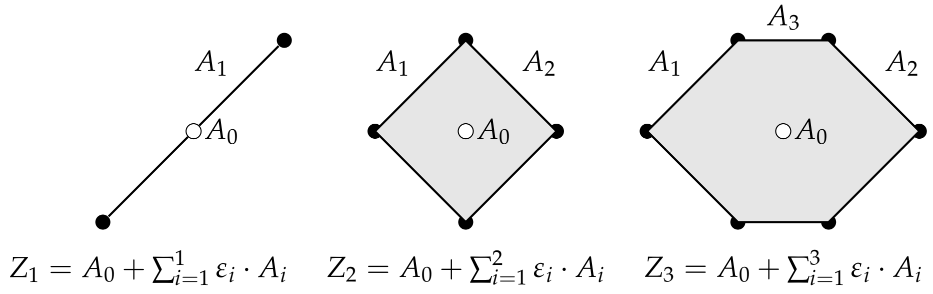

A_1
A_0
A_1
A_2
A_0
A_3
A_1
A_2
A_0
$Z_1 = A_0 + \sum_{i=1}^{1} \varepsilon_i \cdot A_i$
$Z_2 = A_0 + \sum_{i=1}^{2} \varepsilon_i \cdot A_i$
$Z_3 = A_0 + \sum_{i=1}^{3} \varepsilon_i \cdot A_i$

Figure 5.1: Three zonotopes with $A_0 = (0,0), A_1 = (1,1), A_2 = (1,-1)$, and $A_3 = (1,0)$. Black circles represent the extreme points of each zonotope.

be zero. Subtracting $[\![A]\!]_\varepsilon = A_0 + \sum_i \varepsilon_i \cdot A_i$ by itself meanwhile equals to zero for any valuation ε due to identities $A - B = A + (-1) \cdot B$, (5.1), and (5.2). Intuitively, this is the result of tracking the dependencies between variables through noise symbols ε_i and hence subtracting a variable by itself is zero.

For $A_0, A_i :: \mathbb{R}^n$, the affine form A has the geometric shape of a zonotope $\mathcal{Z} :: (\mathbb{R}^n)\mathit{set}$ — a special class of polytopes. By defining a function *range* as the set of all possible valuations of an affine form, the relationship between an affine form A and a zonotope $\mathcal{Z}$ is formalised as $\mathit{range}(A) = \mathcal{Z}$. Figure 5.1 provides a graphical illustration of the set of all points belonging to a zonotope with one, two, and three generators. For affine form with one generator, the range is the set of all points in the line connecting the two solid black circles. For affine forms with two and three generators, the range is the set of all points on the edges and all interior points (grey coloured area including the centre A_0).

Zonotopes are convex sets. To prove this, according to the definition of convex sets, we have to show that any element in between $p_1, p_2 \in \mathcal{Z}$ must also be an element of the zonotope $\mathcal{Z}$. The set of all elements in between p_1 and p_2 is exactly $\mathit{closed\text{-}segment}\,(p_1, p_2)$ defined in (4.3). Any $p \in \mathit{closed\text{-}segment}\,(p_1, p_2)$ has a parameter t where $0 \leq t \leq 1$ and $p = t \cdot p_1 + (1-t) \cdot p_2$. Since p_1 and p_2 are located inside the zonotope $\mathcal{Z}$, there must be ε and ε' such that $p_1 = A_0 + \sum_i \varepsilon_i \cdot A_i$ and $p_2 = A_0 + \sum_i \varepsilon'_i \cdot A_i$. Substituting these equations to the definition of p, we have $p = A_0 + \sum_i (t \cdot \varepsilon_i + (1-t) \cdot \varepsilon'_i) \cdot A_i$ by using identities (5.1) and (5.2). Since the term $t \cdot \varepsilon_i + (1-t) \cdot \varepsilon'_i$ must also be in the range of $[-1, 1]$, we can deduce that p is also inside $\mathcal{Z}$.

Zonotopes have extreme points which are shown as solid black circles in Fig 5.1. A point p is an extreme point of zonotope $\mathcal{Z}$ if there are no two points $p_1, p_2 \in \mathcal{Z}$ such that $p \in \mathit{closed\text{-}segment}\,(p_1, p_2)$. Immler [63] provides a more detailed discussion about the algorithm to obtain all extreme points of a zonotope. Zonotopes can be defined alternatively as the *convex hull* — the set of all *convex combinations* — of all of its extreme points. That is, suppose that $p_1, p_2, \ldots p_n$ are all the extreme points of zonotope $\mathcal{Z}$, then for every point $p \in \mathcal{Z}$ there exist noise symbols ε_i where $p = \sum_i \varepsilon_i \cdot p_i$, $0 \leq \varepsilon_i \leq 1$, and $\sum_i \varepsilon_i = 1$. The notion of extreme points is central to the construction of manoeuvre automata which we shall discuss in Sec. 5.2.

An important operation for sets considered in this chapter is the *Minkowski sum* which is defined as $\mathsf{msum}(A, A') = \{a + a' \mid a \in A \ \wedge a' \in A'\ \}$. When both A and A' are zonotopes, the conditions $a \in A$ and $a' \in A'$ are true if and only if there exist two sequences of noise symbols ε and ε' such that $a = A_0 + \sum_i \varepsilon_i \cdot A_i$ and $a' = A'_0 + \sum_i \varepsilon'_i \cdot A'_i$ and the expression $a + a'$ becomes $A_0 + A'_0 + \sum_i \varepsilon_i \cdot A_i + \sum_i \varepsilon'_i \cdot A'_i$. The last two summations can be grouped together by concatenating the two sequences A_i and A'_i into $\hat{A}_i$ such that the nonzero elements of A_i and A'_i does not share the same noise symbols in $\hat{A}$. Concretely speaking, if we represent an affine form (a zonotope) by a pair of its centre c and a list of its generators gs, then the *Minkowski sum* of two affine forms $A = (\mathsf{c}, \mathsf{gs})$ and $A' = (\mathsf{c}', \mathsf{gs}')$ is defined as:

$$\begin{aligned} \mathsf{msum}(A, A') &= (\mathsf{c} + \mathsf{c}',\ \mathsf{msum\text{-}gens}(A, A'))\ , \\ \mathsf{msum\text{-}gens}(A, A') &= \mathsf{gs}\ @\ \mathsf{gs}'\ , \end{aligned}$$

where function @ denotes list concatenation. Figure 5.1 provides graphical illustrations of the Minkowski sum: $Z_2 = \mathsf{msum}(Z_1, Z'_2)$, $Z_3 = \mathsf{msum}(Z_2, Z'_3)$ where $Z'_2 = \mathbf{0} + A_2$ and $Z'_3 = \mathbf{0} + A_3$.

5.2 Constructing Manoeuvre Automata

A manoeuvre automaton (MA) [86] is an automaton whose states represent manoeuvres (motion primitives) which an autonomous system could execute. For helicopters, these could be standard manoeuvres such as `hover` and `land`, or more aggressive movements such as `hammerhead` and `loop`. For autonomous vehicles, these could be basic manoeuvres such as `turn-left` and `turn-right`, or more ambitious manoeuvres such as `hard-left` and `hard-right`. A transition between two states in an MA indicates that the system which the MA represents can execute those two manoeuvres successively. Formally, they are defined as follows:

Definition 5.1. *A manoeuvre automaton is a tuple* $MA = (M, \mathsf{jump}, \mathsf{ode})$ *where*

- *M is a predefined type for manoeuvre labels;*
- $\mathsf{jump} :: ((M \times M))\mathsf{set}$ *is the transition relation between manoeuvre labels; and*
- $\mathsf{ode}(m) :: \mathbb{R} \times \mathbb{R}^n \Rightarrow \mathbb{R}^n$ *is the corresponding ordinary differential equation (ODE) for manoeuvre m.*

If we assume that the $\mathit{ode}(m)$ has the general form of

$$\dot{x} = f(x, u_m) \ , \tag{5.3}$$

then the $\mathit{ode}(m)$ represents a fixed system model f — such as a point-mass or as a kinematic single-track model for autonomous vehicles [91] — with a fixed input trajectory u_m for manoeuvre m. For an initial state x_{init} and a final state x_{final}, a controller must choose a trajectory $u_m \in \mathcal{U}_m$ which steers x_{init} to x_{final}. Depending on whether f represents a closed-loop or open-loop, u_m is either a reference or a control output.

For safety verification purposes, it is paramount to formally compute the *reachable set* of a manoeuvre m — denoted by $\mathit{reach}(m)$. This set represents the set of all states x which could be reached by the system f in (5.3) from an initial set denoted by $\mathit{init}(m)$ with any trajectory u_m from $\mathcal{U}_m$. A manoeuvre m is considered to be safe with respect to a given unsafe set $\mathcal{D}$ if and only if $\mathit{reach}(m)$ does not intersect with the unsafe set: $\mathit{reach}(m) \cap \mathcal{D} = \emptyset$ (see Fig. 5.2). This unsafe set could either be composed of obstacles that a vehicle has to avoid or a range of speeds at which a vehicle must not drive. Due to its important role in safety verification, $\mathit{reach}(m)$ is computed with the aid of a formally verified implementation of reachable sets computation in the theorem prover ISABELLE [14].

Algorithm 1: Computing reachable sets of a manoeuvre in MA

```
$(x_c, u_c) \leftarrow \mathit{centre\text{-}optimal\text{-}control}(\mathit{centre}(\mathit{init}(m)), x_{\text{final}}, \mathit{steps})$
$S_{\text{reach},1} \leftarrow \mathit{init}(m)$
for $k = 1, \ldots, \mathit{steps}$ do
    $P_k \leftarrow \mathit{approx\text{-}with\text{-}parallelotope}(S_{\text{reach}})$
    $Z_k \leftarrow \mathit{corner\text{-}control\text{-}linear\text{-}approx}(P_k, x_c, u_c)$
    $S_{\text{reach},k+1} \leftarrow \mathit{compute\text{-}reachable}(P_k, Z_k)$
end
$\mathit{reach}(m) \leftarrow \bigcup_{k=1}^{\mathit{steps}} S_{\text{reach},k}$
```

In what way do reachable sets $\mathit{reach}(m)$ play an important role for safety verification of a path from a manoeuvre automaton? A safe path from an MA is defined as a series of manoeuvres $m = m_0, m_1, \ldots, m_n$ which: *a)* respects the transition relation jump in Def. 5.1, i.e., $(m_i, m_{i+1}) \in \mathit{jump}$ for $0 \leq i < n$; *b)* ensures that the reachable set of each manoeuvre does not intersect with an unsafe set; and *c)* for every chain (m_i, m_{i+1}) in the series, the final set of m_i — denoted by $\mathit{final}(m_i)$ — must be contained by the initial set of m_{i+1}, i.e., $\mathit{final}(m_i) \subseteq \mathit{init}(m_{i+1})$ (see [89]). The first two requirements are obvious but the last one might not be: Fig. 5.2 illustrates these requirements of ensuring the safety of a path from an MA. If the last requirement does not hold, there might be a trajectory enclosed initially by the first manoeuvre $x_m \in \mathit{reach}(m_0)$, ending outside of the initial set $\mathit{init}(m_1)$, and visits the unsafe set eventually (we make no guarantees about trajectories which start from the outside of the specified initial sets).

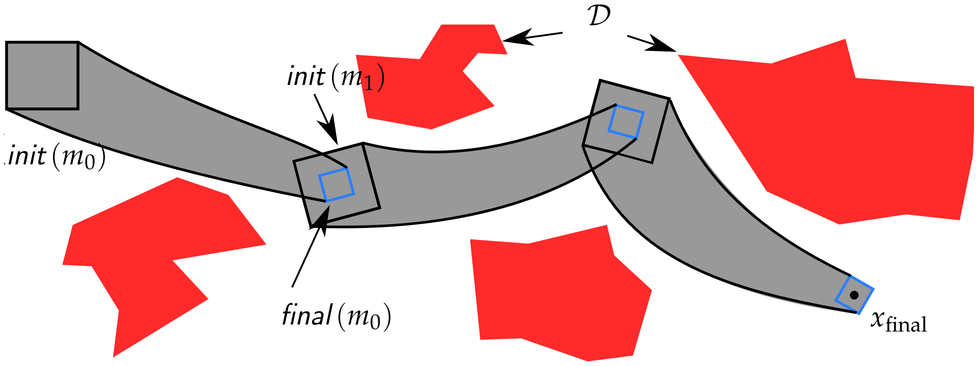


Figure 5.2: Ensuring the safety of a path from an MA.

Intertwining controller design and reachable sets computation. It might seem that designing a controller and computing its reachable sets for a manoeuvre automaton are two separate processes. Schürmann et al. [89] provide an approach to combine these two processes whose algorithm is shown in Alg. 1, illustrated in Fig. 5.3, and briefly explained as follows.

1. *Computing the centre reference trajectory (line 1).*
   From the initial set $init\,(m)$, we find an optimal reference trajectory x_{centre} which starts from the centre of $init\,(m)$ to a final state x_{final}. The qualifier 'optimal' here means that the reference trajectory will be as close as possible to the designated final set x_{final}; other optimality measures include control effort. This reference trajectory serves as a subgoal for the optimisation solver at each time step. Rather than finding an optimal solution over the whole duration to reach x_{final}, we solve an optimisation problem to reach $x_{\text{centre}}(k)$ at each time step k. We could use solvers such as the ACADO toolkit [92] or MATLAB's function `fmincon` for this purpose. Note that the optimisation also has parameter *steps* as an argument; this denotes how many times reachability analysis must be performed for each manoeuvre.

2. *Approximating reachable sets with parallelotope (line 4).*
   The set representation used to represent reachable sets in this work is the *affine form* whose set of valuations forms a *zonotope* (cf. Sec. 5.1). However, the zonotope representing the current reachable set is over-approximated by a parallelotope before it proceeds to the next iteration. This is because there exists a closed-form expression to represent each element in the parallelotope as convex combinations of its extreme points [93]. This fact is very convenient for the construction process because the optimal controller for each point (state) in the parallelotope is also a convex combination of the extreme points' optimal controller (see the next step). Another reason for choosing parallelotopes is that the number of extreme points for parallelotopes is at most 2^n (n is as in Def. 5.1) while that of zonotopes could be more than 2^n; this fact reduces the complexity of constructing the optimal

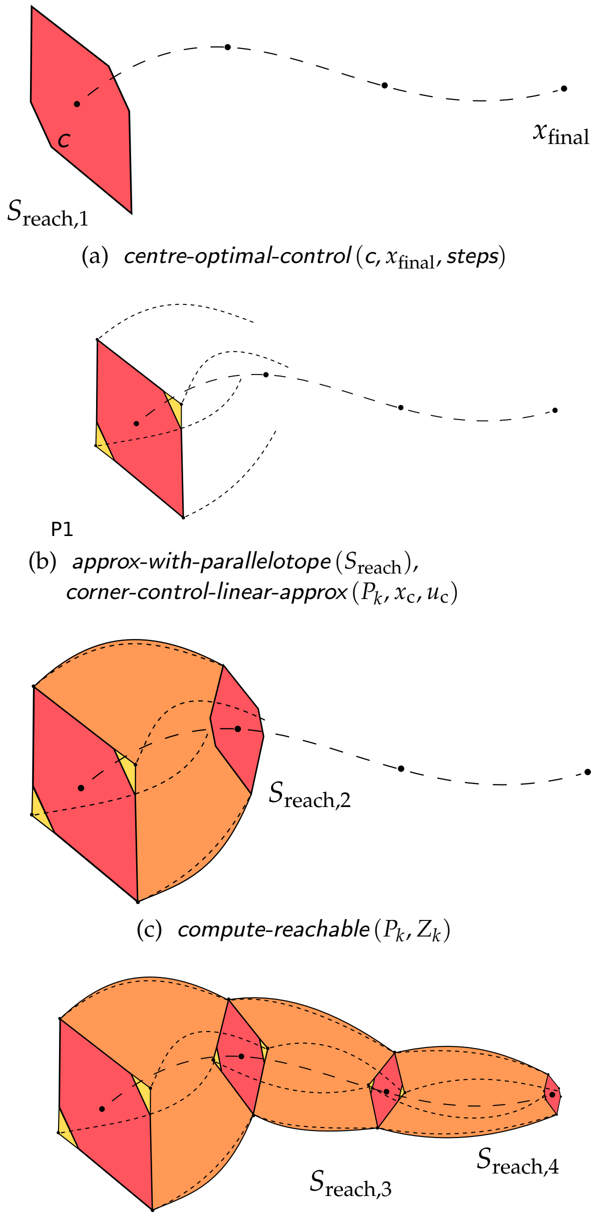
controller.

3. *Finding optimal controllers for extreme points and computing reachable sets (line 5 and 6).*
   For each extreme point identified in the previous step, we find an optimal controller which steers the extreme point as close as possible to the centre trajectory. Suppose that *1)* for each state $x \in S_{\mathrm{reach},i}$ we have coefficients $\gamma_1, \gamma_2, \dots \gamma_{2^n}$ (cf. [93]) and extreme points $p_1, p_2, \dots p_{2^n}$ such that $x = \sum_i \gamma_i \cdot p_i$, $0 \leq \gamma_i \leq 1$, and $\sum_i \gamma_i = 1$; and *2)* function u_i for $0 < i \leq 2^n$ is the optimal controller for extreme point p_i, then the optimal controller for x is $u = \sum_i \gamma_i \cdot u_i$. This is approximated further by a zonotope Z_k (in the input space) — that is u_i is a constant value instead of a function — to reduce the non-linearity and computation errors in the reachability analysis performed next. By setting this optimal controller as u in (5.3), we can then obtain an explicit expression of the $\mathsf{ode}(m)$ and use the formalised reachability analysis in [14] to obtain the reachable set.

Formal construction of manoeuvre automata. How do we formally construct a manoeuvre automaton for formal verification purposes? Given the algorithm for constructing manoeuvre automata in Alg. 1, one might be tempted to formalise each step in the ISABELLE theorem prover. This is a tall order since the algorithm in Alg. 1 frequently uses an optimisation solver which does not exist yet in the ISABELLE theorem prover — formalising an optimisation solver in a theorem prover is worth another PhD project as shown in [94]. However, since the focus of this thesis is on the *safety verification* aspect, it does not hurt if we obtain a non-optimal controller, while the reachable set is certified. Hence, the qualifier 'formal' here refers only to the reachability analysis part of the construction.

With this design decision, we are left with the challenge of interfacing the reachability analysis in ISABELLE [14] with the rest of the algorithm in MATLAB [89]. Figure 5.4 illustrates how we interface ISABELLE and MATLAB by using the C programming language as a *lingua franca*. Functions programmed in MATLAB are callable from C by using the MATLAB API. ISABELLE, on the other hand, can call functions in Standard ML (SML) directly, but not those in C. Fortunately, there is a Foreign Function Interface (FFI) between SML and C which enables us to call functions in C and, hence, MATLAB indirectly. Therefore, we need to provide the corresponding wrapper for each MATLAB function required by ISABELLE at the SML and C level.

From isabelle's to IEEE-754's floating-point representation. In ISABELLE, a floating point is represented by two arbitrary-precision integers: mantissa (or significand) man and exponent exp; together they represent the real number $\mathsf{man} \cdot 2^{\mathsf{exp}}$. However, since we need to interface with MATLAB, we need to convert ISABELLE's representation of



(a) *centre-optimal-control* $(c, x_{\text{final}}, \textit{steps})$

(b) *approx-with-parallelotope* (S_{reach}),
corner-control-linear-approx $(P_k, x_{\text{c}}, u_{\text{c}})$

(c) *compute-reachable* (P_k, Z_k)

(d) Repeat until $k = \textit{steps}$

Figure 5.3: Graphical illustration of computing reachable sets for MA

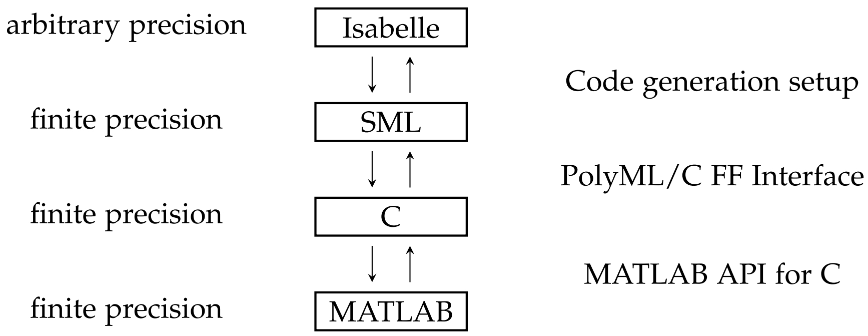


Figure 5.4: Block diagram for interfacing ISABELLE and MATLAB.

floating-points to that of IEEE-754[2] — the representation used by MATLAB. Unlike in ISABELLE, a floating-point number in IEEE-754 has fixed numbers of bits for its mantissa and exponent. In case of double-precision floating-point numbers, the length of the mantissa and exponent is 52 and 11 bits, respectively, and the remaining bit is reserved for the sign bit.

To understand better the conversion we need to do, it is necessary to discuss the three different modes IEEE-754 has: *1.) denormalised* mode to represent numbers which are very close to zero; *2.) special* mode to represent numbers such as infinity and `NaN`; and *3.) normalised* mode to represent the numbers in between. Each mode has different semantics for computing the corresponding real numbers and, therefore, we cannot simply just "map" the mantissa and exponent in ISABELLE's format to those of IEEE-754; we have to identify firstly in which mode an ISABELLE's floating point number resides. The set of floating-point numbers in denormalised, normalised, and special modes have linear orders. Therefore, we can identify the mode by checking whether it is larger or smaller than the smallest and the largest normalised floating-point numbers. If we assume that the exponent field takes up k bits and the mantissa has the precision of prec[3], then the smallest and the largest normalised values are

$$\begin{aligned} \mathit{smallest\text{-}normalised} &:= 2^{\mathit{prec}-1} \times 2^{\mathit{exp\text{-}min}}, & (5.4)\\ \mathit{largest\text{-}normalised} &:= (2^{\mathit{prec}} - 1) \times 2^{\mathit{exp\text{-}max}}, & (5.5)\end{aligned}$$

where $\mathit{exp\text{-}min} := 1 - \mathit{bias} - (\mathit{prec} - 1)$ and $\mathit{exp\text{-}max} := \mathit{bias} - (\mathit{prec} - 1)$ and $\mathit{bias} = 2^{k-1} - 1$. Note that the $\mathit{exp\text{-}min}$ and $\mathit{exp\text{-}max}$ are defined differently from the literature. For example, in [95] $\mathit{exp\text{-}min}$ and $\mathit{exp\text{-}max}$ are defined as $1 - \mathit{bias}$ and bias, respectively, but here we have subtracted $\mathit{prec} - 1$ from these values. This is because the mantissa m in [95] is implicitly multiplied by $\frac{1}{2^{\mathit{prec}-1}}$ so that the mantissa m actually represents the

[2] Bryant and O'Hallaron [95] provide a comprehensive introduction to IEEE-754 representation.

[3] Note the difference between precision and the number of bits required to represent the mantissa. IEEE-754 has the leading-one assumption which means that although it uses 52 bits for the mantissa, it has the precision of 53 bits.

value of $\frac{m}{2^{prec-1}}$. We make this multiplication explicit by carrying over the value of $\frac{1}{2^{prec-1}}$ to the 2^{1-bias} and 2^{bias}, and hence the subexpression $prec - 1$ in *exp-min* and *exp-max*.

In order to test the order relation with respect to these boundary values, we use the following two functions:

$$\begin{aligned} \textit{in-denormalised}\,(m,e) \;&:=\; m = 0 \;\vee\; (e - (\textit{prec} - \textit{bitlen}\,(m)) < \textit{exp-min}) \;\;, \\ \textit{in-normalised}\,(m,e) \;&:=\; m = 0 \;\vee\; (e - (\textit{prec} - \textit{bitlen}\,(m)) \leq \textit{exp-max}) \;\;, \end{aligned}$$

where *bitlen* (m) is a function to find the number of bits required to represent the number m. We prove in ISABELLE that these two functions detect the range of denormalised and normalised mode correctly.

Theorem 5.1. *For any mantissa m and exponent e, we have*

$$\begin{aligned} \textit{in-denormalised}\,(m,e) \;&\Longrightarrow\; |m| \cdot 2^e < \textit{smallest-normalised}, \textit{ and} \\ \textit{in-normalised}\,(m,e) \;&\Longrightarrow\; |m| \cdot 2^e \leq \textit{largest-normalised} \;\;. \end{aligned}$$

Proof. Case $m = 0$ is trivial; we only prove the case $m \neq 0$.

$$\begin{aligned} & |m| \cdot 2^e \;<\; \textit{smallest-normalised} \\ \Longleftarrow \quad & \{ \text{ unfolding the definition of } \textit{smallest-normalised} \text{ according to (5.4) } \} \\ & |m| \cdot 2^e \;<\; 2^{prec-1} \times 2^{exp\text{-}min} \\ \Longleftarrow \quad & \{ \text{ arithmetic } \} \\ & |m| \cdot 2^{e-exp\text{-}min} \times 2^{exp\text{-}min} \;<\; 2^{prec-1} \times 2^{exp\text{-}min} \\ \Longleftarrow \quad & \{ \text{ dividing both sides with } 2^{exp\text{-}min} \} \\ & |m| \cdot 2^{e-exp\text{-}min} \;<\; 2^{prec-1} \\ \Longleftarrow \quad & \{ \text{ fact } |m| \;<\; 2^{bitlen\,(m)} \} \\ & 2^{bitlen\,(m)} \cdot 2^{e-exp\text{-}min} \;<\; 2^{prec-1} \\ \Longleftarrow \quad & \{ \text{ taking log on both sides } \} \\ & \textit{bitlen}\,(m) + e - \textit{exp-min} \;<\; \textit{prec} - 1 \\ \Longleftarrow \quad & \{ \text{ definition of } \textit{in-denormalised}\,(m,e)\text{; arithmetic } \} \\ & \textit{in-denormalised}\,(m,e) \end{aligned}$$

The proof for *in-normalised* $(m,e) \implies |m| \cdot 2^e \leq$ *largest-normalised* is similar. □

After we know in which mode the ISABELLE's floating-point number resides, we need to adjust the mantissa and exponent correctly according to the specification in IEEE-754. For those in the denormalised mode, the exponent *exp* has to be set to *exp-min*, whereas for those in the normalised mode, the mantissa *man* has to be set into *man'* so that

$2^{prec-1} \leq |man'|$. The former is obtained with the following function:

$$\begin{aligned} \mathit{denormalise}\,(m,e) \;:=\; & \textbf{let } s = \mathit{exp\text{-}min} - e \textbf{ in} \\ & \textbf{if } s \leq 0 \textbf{ then } (\mathit{bitshift\text{-}left}\,(m,-s), e+s) \\ & \textbf{else if } 0 \leq m \textbf{ then } (\mathit{bitshift\text{-}right}\,(m,s), e+s) \\ & \textbf{else}(-\mathit{bitshift\text{-}right}\,(-m,s), e+s)) \;\;, \end{aligned}$$

where $\mathit{bitshift\text{-}left}\,(x,b)$ and $\mathit{bitshift\text{-}right}\,(x,b)$ are the operations for performing b-bit shift to the left and right, respectively. The correctness theorem for this function is as follows:

Theorem 5.2. *For any two mantissas m, m' and two exponents e, e', the denormalise function has the following invariant property:*

$$\mathit{denormalise}\,(m,e) = (m',e') \implies e' = \mathit{exp\text{-}min} \,\wedge\, m \cdot 2^e = m' \cdot 2^{e'} \;\;.$$

Proof. We only prove for the normative case where $0 \leq s$ and $0 \leq m$ as the proofs for other cases can be obtained similarly.

$$\begin{aligned} & m' \cdot 2^{e'} \\ \iff & \quad \{ \text{ case } 0 \leq s \text{ and } 0 \leq m \text{ in } \mathit{denormalise}\,(m,e) \; \} \\ & \mathit{bitshift\text{-}right}\,(m,s) \cdot 2^{e+s} \\ \iff & \quad \{ \text{ definition of } \mathit{bitshift\text{-}right}\,(m,s) \; \} \\ & \left\lfloor \frac{m}{2^s} \right\rfloor \cdot 2^{e+s} \\ \iff & \quad \{ \text{ property of floor function; } 2^{e+s} \text{ is multiple of } 2^s \; \} \\ & \left\lfloor \frac{m}{2^s} \cdot 2^{e+s} \right\rfloor \\ \iff & \quad \{ \text{ arithmetic } \} \\ & m \cdot 2^e \end{aligned}$$

The condition $e' = \mathit{exp\text{-}min}$ is obvious from the definition $s := \mathit{exp\text{-}min} - e$ and equality $e' = e + s$. □

For the normalised mode, we use the following function:

$$\begin{aligned} \mathit{normalise}\,(m,e) \;:=\; & \textbf{if } \mathit{bitlen}\,(|m|) < \mathit{prec} \textbf{ then} \\ & (m \cdot 2^{\mathit{prec}-\mathit{bitlen}\,(|m|)}, e - (\mathit{prec} - \mathit{bitlen}\,(|m|))) \textbf{ else } (m,e) \end{aligned}$$

with the following correctness theorem:

Theorem 5.3. *For any two mantissas m, m' and two exponents e, e', the normalise function has the following invariant property:*

$$\mathit{normalise}\,(m,e) = (m',e') \implies 2^{prec-1} \leq |m'| \,\wedge\, m \cdot 2^e = m' \cdot 2^{e'} \;\;.$$

Proof. For the case where $\mathsf{bitlen}\,(|m|) < \mathsf{prec}$, we have $m' = m \cdot 2^{\mathsf{prec}-\mathsf{bitlen}\,(|m|)}$. By using the identity $\mathsf{bitlen}\,(b \cdot 2^c) = \mathsf{bitlen}\,(b) + c$, we have $\mathsf{bitlen}\,(|m'|) = \mathsf{bitlen}\,(|m|) + \mathsf{prec} - \mathsf{bitlen}\,(|m|) = \mathsf{prec}$. Since we now know that $|m|$ has *prec*-bit precision, we can deduce that $2^{\mathsf{prec}-1} \leq |m'|$. This a basic inequality involving $\mathsf{bitlen}\,(\_)$ operation; we also use this inequality to prove $2^{\mathsf{prec}-1} \leq |m|$ when $\mathsf{prec} \leq \mathsf{bitlen}\,(m)$. To prove the second conjunct, we simply replace the definition of m' and e' such that $m' \cdot 2^{e'} = m \cdot 2^{\mathsf{prec}-\mathsf{bitlen}\,(|m|)} \cdot 2^{e-(\mathsf{prec}-\mathsf{bitlen}\,(|m|))} = m \cdot 2^e$ when $\mathsf{bitlen}\,(|m|) < \mathsf{prec}$; case $\mathsf{prec} \leq \mathsf{bitlen}\,(m)$ is trivial. □

If we know that (m, e) is in the normalised range, then we can guarantee that the new mantissa m' after being normalised, i.e. $\mathsf{normalise}\,(m, e) = (m', e')$, will have exactly *prec* bits.

Theorem 5.4.

$$\mathsf{in\text{-}normalised}\,(m, e) \implies \mathsf{normalise}\,(m, e) = (m', e') \implies \mathsf{bitlen}\,(|m'|) = \mathsf{prec}\ .$$

Proof. Following the proof of Thm. 5.3, we know that $\mathsf{bitlen}\,(|m'|) = \mathsf{prec}$ when $\mathsf{bitlen}\,(m) < \mathsf{prec}$. When $\mathsf{prec} \leq \mathsf{bitlen}\,(|m|)$, we have $m' = m$ and, from Thm. 5.1 and assumption *in-normalised* (m,e), the mantissa m is at most $2^{\mathsf{prec}} - 1$. Hence $\mathsf{bitlen}\,(|m'|) = \mathsf{bitlen}\,(|m|) \leq \mathsf{bitlen}\,(2^{\mathsf{prec}} - 1) = \mathsf{prec}$. For the case $\mathsf{prec} \leq \mathsf{bitlen}\,(|m|)$, we have $\mathsf{bitlen}\,(|m'|) = \mathsf{prec}$. □

After we have obtained the proper mantissa *man'* and exponent *exp'* from the either denormalise or normalise function, we are left with only a few things to do. Since we have the leading one assumption for the normalised mode, an amount of $2^{\mathsf{prec}-1}$ has to be subtracted from *man'*. Then, we have to add either $\mathsf{bias} - 1$ (denormalised mode) or *bias* (normalised mode) to the exponent *exp'* because IEEE-754 use a biased representation for negative numbers. Note that since ISABELLE's floating-point implementation can have arbitrary precision, we have to ensure that the floating-point numbers used in ISABELLE's theories are guaranteed to have 53-bit precision before being passed down to SML, C, and MATLAB. Lastly, it is worth to mention that the verified function works one way only — from ISABELLE's to IEEE-754's representation. We implement the translation for the other direction in MATLAB, but not verify it formally. However, this translation is very straightforward and is very easy to inspect.

5.3 Motion Planning with Manoeuvre Automata

Motion planning with MA falls into the category of temporal-logic motion planning techniques. Fainekos et al. [96], for example, show how to use temporal logic for motion planning by first discretising the environment and labelling them with meaningful atomic propositions. From these atomic propositions, we can formalise the plan such as sequencing (visit π_1 and then π_2), reach-avoid (eventually reach π_1 and simultaneously avoid π_2), and coverage (reach π_1, π_2, and π_3 in any order). An LTL formula in this setting is always interpreted over a continuous trajectory $\xi :: \mathbb{R} \Rightarrow \mathbb{R}^2$ where an atomic proposition π is true if and only if ξ is located initially inside the area labelled by π, i.e., $[\![\pi]\!]$. If we can utilise a model checker — with this physical interpretation — to *falsify* the *negated* plan, we can then obtain a discrete plan to *satisfy* the *original* plan. Lastly, we are left with the task of finding a continuous controller for the discrete plan.

The previously explained approach works if we consider a single trajectory only and fails miserably when we consider a set of trajectories. Consider the formalisation of reach-avoid plans in standard LTL which Fainekos et al. [96] formalised as $\neg \textsf{obstacle}\ U\ \textsf{goal}$. Supposed that we naïvely lift the denotation for atomic propositions used by Fainekos et al. [96] into $\sigma = \mathcal{A}_0, \mathcal{A}_1, \ldots \mathcal{A}_n \models \pi \iff \mathcal{A}_0 \subseteq [\![\pi]\!]$, where $\mathcal{A}_i$ is the sequence of reachable sets of manoeuvre m (see [97]). This denotation implies that avoiding obstacles which is syntactically formalised as $\sigma \models \neg \textsf{obstacle}$ is true if and only if $\mathcal{A}_0 \not\subseteq [\![\textsf{obstacle}]\!]$. However, $\mathcal{A}_0 \not\subseteq [\![\textsf{obstacle}]\!]$ does not mean that $\mathcal{A}_0$ does not intersect with $[\![\textsf{obstacle}]\!]$ at all, which is formalised more aptly with $\mathcal{A}_0 \cap [\![\textsf{obstacle}]\!] = \emptyset$. Hence, naïvely lifting the existing semantics into that for reachable sets to determine $\sigma \models \neg \textsf{obstacle}\, U\, \textsf{goal}$ means that there could be a trajectory which visits the obstacle before reaching the goal, i.e., this decision procedure is unsound; the following section explains how to solve this problem.

5.3.1 Interpreting LTL over manoeuvre automata

Definition 5.2 (Linear Temporal Logic for MA). *If AP is the type for all atomic propositions, then we can create a new compound data type*

$$\textbf{datatype}\ atom \ = \ AP^{+} \mid AP^{-} \ ,$$

where we label an atomic proposition with either a positive or negative sign. The syntax of LTL for manoeuvre automata is defined by the following grammar:

$$\phi \ ::= \ \textsf{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid X\,\phi \mid \phi_1\, U\, \phi_2 \ , \tag{5.6}$$

where π :: atom. Constant false*, logical operators disjunction and implication, and temporal operators F and G are defined as usual [96].*

Atomic propositions in path planning with LTL are used to represent objects of interest. For example, atomic propositions in this chapter could be defined as follows:

$$\textbf{datatype}\ AP \ = \textsf{left-boundary} \mid \textsf{right-boundary} \mid \textsf{obstacle} \mid \textsf{goal} \ .$$

These atomic propositions have geometrical interpretations, but atomic propositions in this chapter generally could also be interpreted non-geometrically, such as *speed-range*, which has the interpretation of the sets of allowable speeds, or *safe-distance* which can be interpreted with the safe distance function formally verified in Ch. 2, or any atomic propositions related to overtaking explained in Ch. 4.

Definition 5.3 (Semantics of LTL for MA over finite-length traces)**.** *Suppose that the state space for the model* $\mathsf{ode}(m)$ *in Def. 5.1 is of type* $\mathbb{R}^n$*, and there is an interpretation function* $[\![\_]\!]$ :: $AP \Rightarrow (\mathbb{R}^n)\mathsf{set}$*. Additionally, for a finite sequence of sets* $\sigma = \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$*, we denote the j-th suffix of* σ *by* $\sigma[j..] := \mathcal{A}_j, \ldots, \mathcal{A}_k$ *for* $0 \leq j \leq k$*. We can define a semantics of LTL for MA over a finite sequence of sets* $\sigma = \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k$*, where* $\mathcal{A}_i$ :: $(\mathbb{R}^n)\mathsf{set}$ *for* $0 \leq i \leq k$*, as follows:*

$$\begin{array}{lcl}
\sigma \models \mathit{true} & & \\
\sigma \models \pi^+ & \Longleftrightarrow & \mathcal{A}_0 \subseteq [\![\pi]\!] \\
\sigma \models \pi^- & \Longleftrightarrow & \mathcal{A}_0 \cap [\![\pi]\!] = \emptyset \\
\sigma \models \neg\phi & \Longleftrightarrow & \mathit{not}\ \sigma \models \phi \\
\sigma \models \phi_1 \wedge \phi_2 & \Longleftrightarrow & \sigma \models \phi_1\ \mathit{and}\ \sigma \models \phi_2 \\
\sigma \models X\,\phi & \Longleftrightarrow & \mathit{if}\ \sigma[1..] \mathit{is\ defined\ then}\ \sigma[1..] \models \phi \\
\sigma \models \phi_1\, U\, \phi_2 & \Longleftrightarrow & \exists j.\ \sigma[j..] \models \phi_2\ \wedge\ \forall i.0 \leq i < j \longrightarrow \sigma[i..] \models \phi_1\ .
\end{array}$$

The syntax and semantics in Defs. 5.2 and 5.3 provide a solution to the problem explained earlier. Each atomic proposition is now labelled with either a positive or negative sign, and the root cause of the unsafety in the previous argument is because $\mathcal{A}_0 \not\subseteq [\![\mathit{obstacle}]\!]$ does not imply $\mathcal{A}_0 \cap [\![\mathit{obstacle}]\!] = \emptyset$; the semantics in 5.3 enforces all negatively labelled atomic propositions have the denotation that all trajectories in $\mathcal{A}_0$ cannot be located at $[\![\pi]\!]$, i.e., $\mathcal{A}_0 \cap [\![\pi]\!] = \emptyset$. Positively labelled atomic propositions, meanwhile, have the obvious denotation that all trajectories in the initial set $\mathcal{A}_0$ must also be located inside $[\![\pi]\!]$, i.e., $\mathcal{A}_0 \subseteq [\![\pi]\!]$. Note that it is possible to have a case where $\mathcal{A}_0 \not\subseteq [\![\pi]\!]$ and $\mathcal{A}_0 \cap [\![\pi]\!] \neq \emptyset$ — we could deduce neither $\sigma \models \pi^+$ nor $\sigma \models \pi^-$.

Checking zonotope inclusion and intersection freedom. The semantics in Def. 5.3 does not stipulate any concrete type of sets. Since the reachable sets computation explained in the previous section use affine forms whose sets of valuations are zonotopes as their concrete data type, we shall also use them to check $\sigma \models \pi^+$ (using an inclusion check) and $\sigma \models \pi^-$ (checking for intersection freedom) in $\mathbb{R}^2$ since higher dimensions are not required in this work. We define the function *zono-contain2D*(Z, Z') to check whether the zonotope *range*(Z) is a subset of zonotope *range*(Z'). This is performed[4] by first enumerating all extreme points of zonotope *range*(Z) and then checking whether each of these extreme points belongs to the zonotope *range*(Z').

[4] For high-dimensional zonotopes, please consult the technique described in CORA [98].

Enumerating extreme points is achieved indirectly via the function *segments-of-zonotopes* which outputs a polygonal chain (cf. Def. 4.6) whose first elements are sorted in a counter-clockwise manner; this function is formalised by Immler [63] which we directly use here. This function is implemented such that the result is always a polygon:

$$\textit{segments-of-zonotope}(A) = \textit{cs} \implies \textit{cs} \neq \textit{[]} \longrightarrow \textit{fst}(\textit{hd}(\textit{cs})) = \textit{snd}(\textit{last}(\textit{cs})) \ .$$

As such, it is impossible for the function to return a list of a single element only due to the definition of polygonal chains in Def. 4.6 which forces both endpoints of the chain to be different. Then, the function for enumerating all extreme points can be defined as follows:

$$\textit{extreme-pts}\,(A) := \textit{map fst}\ (\textit{segments-of-zonotope}\,(A)) \ .$$

Checking whether a point belongs to a zonotope is achieved via the following function:

$$\begin{aligned} p \in_{\textit{zono}} A := \ & \textbf{case}\ \textit{segments-of-zonotope}(A)\ \textbf{of} \\ & \texttt{[]} \quad \Rightarrow p = A_0 \\ & \mid [x, y] \Rightarrow p \in \textit{points-in-line}(x) \ \wedge\ \textit{in-x-interval}(x, \textit{fst}(p)) \\ & \mid \textit{cs} \quad \Rightarrow \forall c \in \textit{set}(\textit{cs}) \longrightarrow \textit{counter-cw}(p, \textit{fst}(c), \textit{snd}(c)) \ . \end{aligned}$$

Theorem 5.5. $p \in_{\textit{zono}} A \implies p \in \textit{range}(A)$.

Proof. In case that the returned list of segments is empty, we know that the zonotope does not have any generator and hence the zonotope consists of one element only, i.e., $\textit{range}(A) = \{A_0\}$. Hence $p \in_{\textit{zono}} A$ if and only if $p = A_0$ which in turn true if and only if $p \in \textit{range}(A)$. If it returns two chains $[x, y]$, then chain x and y are actually the same except that their endpoints are reversed because $[x, y]$ is a polygon. Hence, the zonotope $\textit{range}(A)$ is a closed segment of either x or y, i.e., $\textit{range}(A) = \textit{closed-segment}(x)$. In this case, $p \in_{\textit{zono}} A$ is equal to $p \in \textit{points-in-line}(x)$ and $\textit{in-x-interval}(x, \textit{fst}(p))$. By unfolding the definitions in (4.5) and (4.15), we can deduce that $p \in \textit{closed-segment}(x)$ and subsequently $p \in \textit{range}(A)$. In case there are at least three chains, we check whether the point p has a counter-clockwise relation with each chain. The fact that this equals $p \in \textit{range}(A)$ is the correctness property of *segments-of-zonotope* proved in [28]. □

Checking whether zonotope $\textit{range}(Z)$ is a subset of zonotope $\textit{range}(Z')$ is formalised by the following function:

$$\begin{aligned} \textit{zono-contain2D}\,(Z, Z') := \ & \textbf{case}\ \textit{extreme-pts}\,(Z)\ \textbf{of} \\ & \texttt{[]} \Rightarrow Z_0 \in_{\textit{zono}} Z' \\ & \mid \textit{ps} \Rightarrow \forall p.\ p \in \textit{set}(\textit{ps}) \longrightarrow p \in_{\textit{zono}} Z' \ . \end{aligned}$$

Theorem 5.6. *For any two zonotopes Z, Z' of type $\mathbb{R}^2$, we have*

$$\textit{zono-contain2D}\,(Z, Z') \implies \textit{range}\,(Z) \subseteq \textit{range}\,(Z') \ .$$

Proof. In case that the list of the extreme points of Z is an empty list, $\textsf{zono-contain2D}\,(Z, Z')$ is equal to $Z_0 \in_{zono} Z'$ and Thm. 5.5 allows us to deduce $Z_0 \in \textsf{range}(Z')$. With the equality $\textsf{range}(Z) = \{Z_0\}$ in this case, we can deduce that $\textsf{range}(Z) \subseteq \textsf{range}(Z')$. In case that the list of the extreme points of Z is not an empty list, we can obtain m extreme points $[p_0, p_1, \ldots, p_{m-1}]$ such that $\forall i.\ 0 \le i < m \longrightarrow p_i \in_{zono} Z'$ from the definition of $\textsf{zono-contain2D}\,(Z, Z')$. From this fact, we can deduce further that $\forall i.\ 0 \le i < m \longrightarrow p_i \in \textsf{range}(Z')$ by using Thm. 5.5. With the definition of zonotope, there exists a noise symbol ε_i such that $p_i = Z'_0 + \sum_j \varepsilon_{i,j} \cdot Z'_j$ where $\varepsilon_{i,j} \in [-1, 1]$ for every $0 \le i < m$. Then,

$$
\begin{array}{ll}
 & p \in \textsf{range}(Z) \\
\Longrightarrow & \{\text{ definition of membership via extreme points }\} \\
 & \exists c.\quad p = \sum_i c_i \cdot p_i \ \wedge\ \sum_i c_i = 1 \\
\Longrightarrow & \{\text{ unfolding } p_i \ \} \\
 & \exists c.\quad p = \sum_i c_i \cdot \left(Z'_0 + \sum_j \varepsilon_{i,j} \cdot Z'_j\right) \ \wedge\ \sum_i c_i = 1 \\
\Longrightarrow & \{\text{ arithmetic involving summation }\} \\
 & \exists c.\quad p = \sum_i c_i \cdot Z'_0 \ +\ \sum_i \sum_j c_i \cdot \varepsilon_{i,j} \cdot Z'_j \ \wedge\ \sum_i c_i = 1 \\
\Longrightarrow & \{\text{ using the right conjunct and switching the summation }\} \\
 & \exists c.\quad p = Z'_0 \ +\ \sum_j \left(\sum_i c_i \cdot \varepsilon_{i,j}\right) \cdot Z'_j \ \wedge\ \sum_i c_i = 1\ .
\end{array}
$$

From the right conjunct $\sum_i c_i = 1$, we also have $\sum_i c_i \cdot \varepsilon_{i,j} \le \sum_i c_i \cdot |\varepsilon_{i,j}| \le \sum_i c_i \cdot 1 = 1$ and $\sum_i c_i \cdot \varepsilon_{i,j} \ge \sum_i c_i \cdot -|\varepsilon_{i,j}| \ge \sum_i c_i \cdot -1 = -1$. With these bounds and the calculation above, we can deduce that $p \in \textsf{range}(Z')$ and hence $\textsf{range}(Z) \subseteq \textsf{range}(Z')$. □

Checking whether zonotope $\textsf{range}\,(Z)$ does *not* intersect with $\textsf{range}\,(Z')$ is performed via the following function:

$$\textsf{collision-freedom2D}\,(Z, Z') \ :=\ \neg\left(Z'_0 - Z_0 \in_{zono} \left(0, \textsf{msum-gens}\,(Z, Z')\right)\right)\ .$$

Theorem 5.7. *For any two zonotopes Z, Z' of type $\mathbb{R}^2$, we have*

$$\textsf{collision-freedom2D}\,(Z, Z') \ \Longrightarrow\ \textsf{range}\,(Z) \cap \textsf{range}\,(Z') = \varnothing\ .$$

Proof. From Thm. 5.5, we can deduce that $Z'_0 - Z_0 \notin \textsf{range}\,((0, \textsf{msum-gens}\,(Z, Z')))$. Hence,

$$
\begin{array}{ll}
 & \neg\exists \varepsilon\, \varepsilon'.\quad Z'_0 - Z_0 = \sum_i \varepsilon_i \cdot Z_i + \sum_i \varepsilon'_i \cdot Z'_i \\
\Longleftrightarrow & \{\text{ renaming } \varepsilon' \text{ into } \hat{\varepsilon} \text{ such that } \hat{\varepsilon}_i = -\varepsilon'_i \ \} \\
 & \neg\exists \varepsilon\, \hat{\varepsilon}.\quad Z'_0 - Z_0 = \sum_i \varepsilon_i \cdot Z_i - \sum_i \hat{\varepsilon}_i \cdot Z'_i \\
\Longleftrightarrow & \{\text{ arithmetic }\}
\end{array}
$$

$$
\begin{aligned}
& \neg\exists \varepsilon\, \hat{\varepsilon}.\quad Z'_0 + \textstyle\sum_i \hat{\varepsilon}_i \cdot Z'_i = Z_0 + \sum_i \varepsilon_i \cdot Z_i \\
\Longleftrightarrow\quad & \{\ \text{definition of zonotope}\ \} \\
& \neg\exists p.\quad p \in \mathsf{range}(Z)\ \wedge\ p \in \mathsf{range}(Z') \\
\Longleftrightarrow\quad & \{\ \text{set theory}\ \} \\
& \mathsf{range}(Z) \cap \mathsf{range}(Z') = \emptyset\ .
\end{aligned}
$$

□

5.3.2 Satisfiability checking of LTL over manoeuvre automata

Given a formally verified manoeuvre automaton and a formally defined LTL interpreted over a sequence of reachable sets, how can we use these to plan a motion for autonomous vehicles? Planning a motion, as in any typical temporal logic motion planning, starts with encoding the plan we wish to achieve — such as sequencing, reach-avoid, or coverage — in LTL (Def. 5.2), and proceeds with searching a valid path in the manoeuvre automaton according to the semantics defined in Def. 5.3. The problem of finding a path in MA which satisfies a plan formalised in LTL can be stated formally as satisfiability checking:

Definition 5.4 (Satisfiability checking). *An LTL formula ϕ is satisfiable with respect to a manoeuvre automaton $MA = (M, \mathsf{jump}, \mathsf{ode})$ if there is a path $\tau = m_0, m_1, \ldots m_{n-1}$ such that $m_i :: M$ for all $0 \leq i < n$ and*

$$\mathsf{reach}(m_0), \mathsf{reach}(m_1), \ldots, \mathsf{reach}(m_{n-1}) \models \phi\ .$$

Satisfiability checking is a search problem and since *1.)* time efficiency is paramount, and *2.)* a path satisfying a plan usually has a finite duration, we use a depth-limited search strategy for satisfiability checking. Since each manoeuvre lasts for 1 s and a sensible duration for a plan is supposed to be less than 10 s, the maximum depth is set to be 10. Note that the search strategy can be improved further by using an informed search strategy. However, since our main focus is on correctness, we choose a simpler, yet sufficient depth-limited search strategy for satisfiability checking.

As an example, we construct an intentionally simple, formally verified manoeuvre automaton with three motion primitives which last for 1 s each:

$$\textbf{datatype}\ M\ =\ \mathsf{go\text{-}straight} \mid \mathsf{turn\text{-}left} \mid \mathsf{turn\text{-}right}\ , \tag{5.7}$$

where any two manoeuvres can be composed, i.e., $\mathsf{jump} := \mathit{UNIV} :: M \times M$. Note that the duration for each motion primitive does not need to be the same; some primitives could last for, e.g., 0.1 s and others could last for 5 s. We use the following kinematic model of autonomous vehicles:

$$\dot{v} = a; \quad \dot{\Psi} = b; \quad \dot{x} = v \cdot \cos(\Psi); \quad \dot{y} = v \cdot \sin(\Psi)\ . \tag{5.8}$$

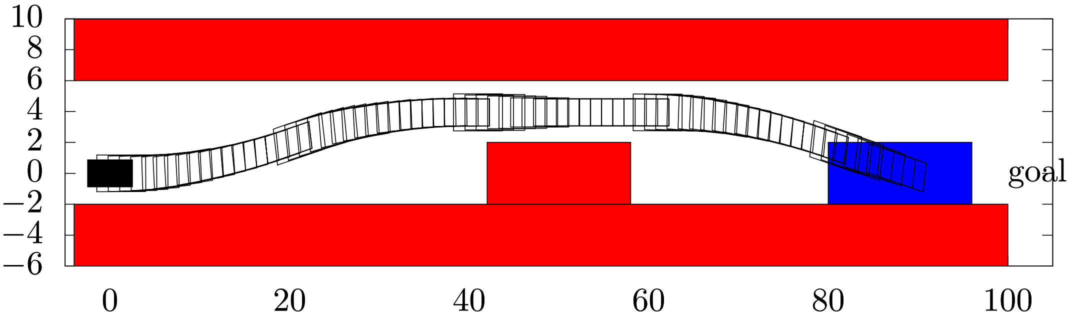


Figure 5.5: Example of a reach-avoid scenario. The vehicle is represented as the solid black rectangle. Red-coloured rectangles are the objects the vehicle has to avoid. The blue-coloured rectangle is the area which the vehicle has to reach eventually.

State variables v and Ψ are speed and orientation, respectively, while x and y are the positions in Cartesian coordinates. Inputs to the system are a and b, which denote acceleration and normalised steering angle, respectively. The initial set $\mathit{init}(m)$ is set to be the same for all manoeuvres:

$$[19.8; 20.2]\,\mathrm{m\,s^{-1}} \times [-0.02; 0.02]\,\mathrm{rad} \times [-0.2; 0.2]\,\mathrm{m} \times [-0.2; 0.2]\,\mathrm{m}\ .$$

Meanwhile, the final states are (20, 0, 20, 0) for *go-straight*, (20.2, 0.2, 19.87, 0.2) for *turn-left*, and (20.2, −0.2, 19.87, −0.2) for *turn-right*. We use the controller design in [89] to obtain a set of trajectories for each manoeuvre and use the verified implementation in [14] to compute the reachable sets for each time.

According to the third requirement to ensure the safety of a path from an MA in Sect. 5.2, we must ensure the enclosure property $\mathit{final}(m_1) \subseteq \mathit{init}(m_2)$ holds for any two manoeuvres $(m_1, m_2) \in \mathit{jump}$. However, the concrete numbers above show that $\mathit{final}(\textit{go-straight}) \not\subseteq \mathit{init}(\textit{go-straight})$; this does not mean that we cannot compose two *go-straight* primitives. This is because the system modelled in (5.8) is invariant with respect to position and orientation. Hence, the second *go-straight* manoeuvre can be shifted so that the centre position of the first final set is the same as the centre position of the second initial set. Additionally, each reachable set in the second *go-straight* manoeuvre is also rotated by the centre orientation of the first final set. Hence, in this specific example, we can think of each primitive identified in (5.7) as a kind of template which can be translated and rotated accordingly.

Example scenario. We consider the reach-avoid scenario for autonomous vehicles (Fig. 5.5) for motion planning which is identical to the numerical example used in Sec. 4.5. Unlike the requirement to maintain a safe distance in Sec. 4.5, here we consider a static obstacle which has to be avoided. The road in this example is divided into two four-meter-wide lanes and bounded by left and right boundaries; both boundaries are specified as 104 m × 4 m-rectangles. The obstacle is a 16 m × 4 m-rectangle located at $(50, 0)$ and the vehicle has the length and width of 5 m and 1.75 m, respectively. It is located initially at $(0, 0)$ and must reach the goal represented by a 16 m × 4 m rectangle located at $(80, 0)$.

The reach-avoid plan is formalised with the following LTL formula:

$$\phi \ := \ (\textit{left-boundary}^{-} \ \wedge \ \textit{right-boundary}^{-} \ \wedge \ \textit{obstacle}^{-}) \ \mathcal{U} \ \textit{goal}^{+} \ \ ,$$

and after performing satisfiability checking, the search returned the following plan as shown in Fig. 5.5:

$$\tau \ := \ \textit{turn-left} \ , \ \textit{turn-right} \ , \ \textit{go-straight} \ , \ \textit{turn-right} \ , \ \textit{go-straight} \ \ .$$

Regarding the search strategy for satisfiability checking, there are two properties we proved: termination and soundness. The former is proved with the aid of the function package in ISABELLE [99] by specifying a measure function which decreases after each recursive call. Meanwhile, the latter is ensured due to the following two facts: *1)* we use the formalised LTL monitoring function from our previous work [59] to check whether current nodes satisfy the LTL formula, and *2)* we interpret each atomic proposition over-approximatively either due to inherent uncertainty or numerical round-offs.

Two remarks are worth mentioning here. Firstly, note that the main scientific dimension considered in this work is the correctness of a motion planner achieved with the aid of a theorem prover. Hence, we prioritise correctness over other dimensions such as coverage, efficiency, and scalability. The example provided in this section should be perceived as an evidence that the formalisation in Isabelle is implementable (code generation); this section by no means is an evaluation of the coverage of our framework which we plan to do in future with other scenarios in [91]. Secondly, readers might question the fidelity of the model in (5.8). However, Schürmann et al. [64] have provided a framework such that a relatively simple model like ours with added uncertainties to incorporate the behaviour a higher fidelty model or a real vehicle could adequately ensure the safety of a plan in a real vehicle.

5.4 Related work

Fainekos et al. [84] and Plaku et al. [100] use satisfiability checking (or falsification) of temporal logic for finding a path which satisfies a plan formalised in (a fragment of) LTL. Fainekos et al. [84] expand and contract objects which must be avoided and reached, respectively, in order to have a robust interpretation of LTL. Plaku et al. [100] ignore the issue of numerical soundness when checking whether a path satisfies an LTL formula. Our approach, meanwhile, uses sets (zonotopes) as the main data structure which means we can handle robustness and numerical soundness simultaneously.

Interpreting LTL formulae over a set of trajectories has also been studied by Roehm et al. [97]. The difference between our semantics is in the way we treat the negation operator. In their work, the negation operator is allowed for formulae without any temporal operators only. Our approach, however, does not have this restriction — hence ours is more expressive — but it comes with an additional requirement of labelling each atomic proposition with a positive or negative sign.

Mitsch et al. [101] use the theorem prover KeYmaera X [13] to prove safety properties of autonomous vehicles. The main difference to our work is the approach to formal reasoning. Theirs is *proof-theoretic*: *a)* they specify the physical model of autonomous vehicles with hybrid programs and the property with differential dynamic logic [12]; then *b)* they use the proof system's inference rules to deduce that the hybrid program indeed satisfies the specified property. As pointed out by Anand and Knepper [102], KeYmaera X does not consider the possibility of round-off errors in floating-point numbers. This issue has been addressed by Bohrer et al. [103] where they introduce a framework called VeriPhy.

Our approach is *model-theoretic*: *1)* we model autonomous vehicles with manoeuvre automata in which each state (manoeuvre) is assigned with reachable sets of the physical behaviour; *2)* we specify the property in a modified LTL which takes the reachable sets into account; and *3)* we enumerate all possible paths in the manoeuvre automaton and find a path which satisfies the property according to the predefined semantics of the modified LTL. The role of the Isabelle theorem prover in our work is to prove that each step is implemented correctly. Compared to VeriPhy, we use affine arithmetic and VeriPhy uses interval arithmetic — a special case of affine arithmetic. However, our approach needs to trust the code generation setup provided by Isabelle, whereas VeriPhy uses a sound compilation technique to generate code in CAKEML [104].

Anand and Knepper [102] use the COQ theorem prover to implement a framework to specify the physical model and controller of robots for the Robot Operating System (ROS). Compared to our formalisation, theirs is closer to the implementation level; ours assumes that the optimal controller can be implemented correctly in the hardware. However, their implementation assumes that the high-level plan is given, whereas we derive a high-level plan and a low-level controller. Both works guarantee numerical soundness, but with a different technique; theirs uses constructive reals, whereas we use floating-point numbers.

Belta et al. [87] have outlined that the challenge for symbolic motion planning and control is to tie the *top-down approaches*, which use temporal logic on rather abstract models, and *bottom-up approaches*, whose aim is to construct manoeuvre automata effectively for formal analysis. We solve this challenge by adapting the syntax and semantics of LTL for manoeuvre automata. The main finding for this work is that *reachability analysis* is the key ingredient to solve this problem. It allows us to compute the reachable sets of each motion primitive and subsequently to define the satisfaction relation of motion primitives with formulae in LTL. We also address the challenge of formal verification of cyber-physical systems, where numerical soundness is largely ignored. By using a generic theorem prover such as ISABELLE, we can guarantee both mathematical correctness and numerical soundness.

6 Conclusion

We revisit again the main message of this thesis: modern proof assistants, such as ISABELLE *1.)* could be used to clarify and specify system-level requirements for autonomous vehicles faithfully and systematically; *2.)* could combine different automated reasoning techniques for hybrid systems such as satisfiability checking and reachability analysis without sacrificing the soundness of each technique; and *3.)* could be used to perform formal analyses such as runtime monitoring and formal verification without compromising numerical correctness.

Chapter 2 shows how we can use proof assistants such as ISABELLE to clarify vague requirements such as maintaining a safe distance. We take the descriptive definition of safe distance, model it mathematically, devise a safe distance expression, and prove mathematically that it is sound. This notion of safe distance apparently is a versatile one too: we use it to concretise the notions such as *endangered* and *obstructed* for formalising overtaking traffic rules in Ch. 4. This versatility is due to the abstraction level considered in this thesis: in the level where we record only variables such as position, speed, and orientation, interpreting safety-related notions with safe distance is arguably the most sensible way.

Chapter 4 demonstrates the faithful codification of overtaking traffic rules in linear temporal logic (LTL) where concepts such as *overtaking* and *safe distance* are modelled as atomic propositions. This idea of representing vague concepts such as *overtaking* as atomic propositions is an important one: doing so allows us to figure out first the structure of the sentences in terms of conjunctions, disjunction, implication, universal or existential quantification over time, without being stifled with the exact definitions of those concepts. This initial codification (analysis) proves useful to make the monitoring more tractable as evidenced by replacing a bi-implication with an implication in the requirement to return to the original lane as soon as possible. Only after we are satisfied with the codification in this level should we concern with the definition of each atomic proposition.

There are many attempts to formalise system-level requirements for autonomous vehicles in LTL, but the approach in this thesis does not clutter the LTL specifications with function and predicate symbols from real analysis. For example, Wongpiromsarn et al. [105] formalise the requirement of safe distance as[1] $\mathsf{G}(\mathit{dist}(x, \mathrm{Obs}) \geq X_{\mathrm{obs}} \wedge \mathit{dist}(x, \mathit{Loc}(\mathrm{Veh})) \geq X_{\mathrm{obs}})$. Although it is understandable what this codification means, the standard syntax of LTL does not recognise function symbols, such as *dist* and *Loc*, and predicate symbols, such as $\geq$; that would be the realm of first-order LTL. This thesis meanwhile formalises the safe distance rule simply with $\mathsf{G}(\texttt{sd-front})$ in which case `sd-front` is an atomic proposition with the interpretation of the safe distance function between the ego vehicle and the front vehicle expressed in Ch. 2; such separation of concerns between codification and concretisation is helpful for formalising traffic rules systematically.

The monitoring in Ch. 4 is particularly challenging to ensure its numerical correctness because traces provided by motion planners will be time-sampled, and we do not know the behaviours in between these time points. This is where the prediction framework in Ch. 3 plays a pivotal role in this thesis: we use the prediction to over-approximate the behaviours in between these time points soundly. Admittedly, the prediction framework in Ch. 3 applies to spatial behaviours only — positions and orientations — but we can use reachability analysis to over-approximate the speed and acceleration in between these time points. The interpretation functions for atomic propositions also present the risk of numerical errors too. We address this risk by correctly transforming each interpretation function (of each atomic proposition) to handle intervals rather than single values (using interval analysis).

Formally verified manoeuvre automata in Ch. 5 could bring the allusion that *all* paths in those automata will always be correct. This is not the case in this thesis, and it could only be true if we want the manoeuvre automaton to satisfy a fixed property only — just as a specific finite automaton is designed to recognise a specific pattern in compilers. A manoeuvre automaton is a template only: its motion primitives represent predefined actions an autonomous vehicle could take, and its transitions represent two primitives that can be executed successively. The motion planner is actually a *search* function in the space generated by unfolding the manoeuvre automaton, and a formally verified motion planner based on manoeuvre automata means a *search* function proved to be sound: it always returns a sequence of motion primitives that satisfies a given property,

[1] *1.)* $\mathit{dist}(x :: \mathbb{R}^2, Y :: (\mathbb{R}^2)\mathit{set}) := \inf_{y \in Y} \|x - y\|$ is the function returning the closest distance between a point and a set of points; *2.)* $x :: \mathbb{R}^2$ is the current centre position of the ego vehicle; *3.)* $\mathrm{Obs} :: (\mathbb{R}^2)\mathit{set}$ is the set of all points belong to any obstacle; *4.)* $X_{\mathrm{obs}} :: \mathbb{R}$ is a parameterised safe distance, e.g., 1 m; *5.)* $\mathit{Loc} :: \mathit{Veh} \Rightarrow \mathbb{R}^2$ is a function which returns the (centre) position of the other vehicle; and *6.)* Veh is universally quantified implicitly.

provided that the search function can find such sequence.

Future work

Here are some potential directions for future work:

- **Formalising traffic rules relevant to traffic codes of other countries.**
  Even though the number of formalised traffic rules in this thesis is limited, it has already opened a new research direction in autonomous vehicles research. The next research objective to tackle is to continue formalising traffic codes specific to countries such as US, UK, Australia, and Singapore. From an academic perspective, this serves the purpose of evaluating the scalability of the previous concretisations. From a practical standpoint, this helps to clarify those traffic codes and provide feedbacks for accommodating autonomous vehicles operating on those countries legally legally.

- **Formal monitoring machine learning-based motion planners.**
  Machine learning can solve many problems efficiently, but it still cannot explain why it does what it does. This is problematic for finding a plan which must obey traffic rules because it cannot provide the reasoning behind the chosen plan. If it is involved in an accident, we cannot justify in front of the court that the reasoning behind the plan is embedded in the coefficients of the neural networks. To solve this, we can ensure that plans from machine learning-based motion planners will be executed only if they are certified by the formal monitoring framework presented in this thesis. In doing so, we can obtain motion planners that are both efficient and explainable.

- **Formal verification of motion planner's code.**
  Current state-of-the-art of formal verification for autonomous vehicles is done at modelling level only. If we want to have a higher degree of dependability, we still need to formally verify the codes used for controlling autonomous vehicles. We can generate codes automatically from the ISABELLE theorem prover, but the translation is not formally verified. There is an ongoing effort for certifying this translation, but it is for a variant of ML programming language called CAKEML. Hence, an interesting research direction is to see how to program robots with CAKEML code generated from ISABELLE. Another interesting direction is to integrate the code with a formally verified operating system such as SEL4 — providing both safety and security.

Bibliography

[1] R. Alur, T. A. Henzinger, and P.-H. Ho. “Automatic symbolic verification of embedded systems”. In: *IEEE Transactions on Software Engineering* 22.3 (Mar. 1996), pp. 181–201. ISSN: 0098-5589. DOI: `10.1109/32.489079`.

[2] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. “HyTech: A model checker for hybrid systems”. In: *Computer Aided Verification*. Ed. by O. Grumberg. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 460–463. ISBN: 978-3-540-69195-2.

[3] A. Chutinan and B. H. Krogh. “Verification of Polyhedral-Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations”. In: *Hybrid Systems: Computation and Control*. Ed. by F. W. Vaandrager and J. H. van Schuppen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 76–90. ISBN: 978-3-540-48983-2.

[4] E. Asarin, O. Bournez, T. Dang, and O. Maler. “Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems”. In: *Hybrid Systems: Computation and Control*. Ed. by N. Lynch and B. H. Krogh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 20–31. ISBN: 978-3-540-46430-3.

[5] A. B. Kurzhanski and P. Varaiya. “Ellipsoidal Techniques for Reachability Analysis”. In: *Hybrid Systems: Computation and Control*. Ed. by N. Lynch and B. H. Krogh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 202–214. ISBN: 978-3-540-46430-3.

[6] I. Mitchell and C. J. Tomlin. “Level Set Methods for Computation in Hybrid Systems”. In: *Hybrid Systems: Computation and Control*. Ed. by N. Lynch and B. H. Krogh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 310–323. ISBN: 978-3-540-46430-3.

[7] A. Girard. “Reachability of Uncertain Linear Systems Using Zonotopes”. In: *Hybrid Systems: Computation and Control*. Ed. by M. Morari and L. Thiele. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 291–305. ISBN: 978-3-540-31954-2.

[8] A. Girard, C. Le Guernic, and O. Maler. "Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs". In: *Hybrid Systems: Computation and Control*. Ed. by J. P. Hespanha and A. Tiwari. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 257–271.

[9] C. Le Guernic and A. Girard. "Reachability Analysis of Hybrid Systems Using Support Functions". In: *Computer Aided Verification*. Ed. by A. Bouajjani and O. Maler. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 540–554. ISBN: 978-3-642-02658-4.

[10] M. Althoff. "Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars". Dissertation. München: Technische Universität München, 2010.

[11] X. Chen, E. Ábrahám, and S. Sankaranarayanan. "Flow*: An Analyzer for Non-linear Hybrid Systems". In: *Computer Aided Verification*. Ed. by N. Sharygina and H. Veith. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 258–263. ISBN: 978-3-642-39799-8.

[12] A. Platzer and J.-D. Quesel. "KeYmaera: A Hybrid Theorem Prover for Hybrid Systems (System Description)". In: *Automated Reasoning*. Ed. by A. Armando, P. Baumgartner, and G. Dowek. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 171–178. ISBN: 978-3-540-71070-7.

[13] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer. "KeYmaera X: An Axiomatic Tactical Theorem Prover for Hybrid Systems". In: *Automated Deduction*. Ed. by A. P. Felty and A. Middeldorp. Cham: Springer International Publishing, 2015, pp. 527–538. ISBN: 978-3-319-21401-6.

[14] F. Immler. "Verified Reachability Analysis of Continuous Systems". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by C. Baier and C. Tinelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 37–51. ISBN: 978-3-662-46681-0.

[15] S. Kong, S. Gao, W. Chen, and E. Clarke. "dReach: δ-Reachability Analysis for Hybrid Systems". In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by C. Baier and C. Tinelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 200–205. ISBN: 978-3-662-46681-0.

[16] United Nations Economic Commission for Europe. *Vienna Convention on Road Traffic*. Nov. 1968.

[17] S. M. Loos, A. Platzer, and L. Nistor. "Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified". In: *Formal Methods*. Ed. by M. Butler and W. Schulte. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 42–56. ISBN: 978-3-642-21437-0.

[18] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Vol. 2283. LNCS. Springer, 2002.

[19] K. G. Larsen, M. Mikučionis, and J. H. Taankvist. "Safe and Optimal Adaptive Cruise Control". In: *Correct System Design*. Ed. by R. Meyer, A. Platzer, and H. Wehrheim. Cham: Springer International Publishing, 2015, pp. 260–277. ISBN: 978-3-319-23506-6. DOI: 10.1007/978-3-319-23506-6_17. URL: https://doi.org/10.1007/978-3-319-23506-6_17.

[20] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi. "UPPAAL — a tool suite for automatic verification of real-time systems". In: *Hybrid Systems III*. Ed. by R. Alur, T. A. Henzinger, and E. D. Sontag. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 232–243. ISBN: 978-3-540-68334-6.

[21] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang. "Perception, Planning, Control, and Coordination for Autonomous Vehicles". In: *Machines* 5.1 (2017). ISSN: 2075-1702. DOI: 10.3390/machines5010006. URL: http://www.mdpi.com/2075-1702/5/1/6.

[22] E. Plaku and S. Karaman. "Motion planning with temporal-logic specifications: Progress and challenges". In: *AI Communications* 29 (Aug. 2015), pp. 151–162. DOI: 10.3233/AIC-150682.

[23] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli. "A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles". In: *CoRR* abs/1604.07446 (2016). arXiv: 1604.07446. URL: http://arxiv.org/abs/1604.07446.

[24] *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia*. Tech. rep. GAO/IMTEC-92-26. United States General Accounting Office, 1992.

[25] J.-L. Lions, L. Lübeck, J.-L. Fauquembergue, G. Kahn, W. Kubbat, S. Levedag, L. Mazzini, D. Merle, and C. O'Hallaron. *ARIANE 5: Flight 501 Failure, report by the inquiry board*. Tech. rep. 33-1996. European Space Agency (ESA) and Centre national d'études spatiales (CNES), 1996.

[26] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. "Basic concepts and taxonomy of dependable and secure computing". In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (Jan. 2004), pp. 11–33. ISSN: 1545-5971. DOI: 10.1109/TDSC.2004.2.

[27] J. Hölzl. "Proving Inequalities over Reals with Computation in Isabelle/HOL". In: *Programming Languages for Mechanized Mathematics Systems*. Ed. by G. D. Reis and L. Théry. Munich, Aug. 2009, pp. 38–45.

[28] F. Immler. "Affine Arithmetic". In: *Archive of Formal Proofs* (Feb. 2014). http://isa-afp.org/entries/Affine_Arithmetic.html, Formal proof development. ISSN: 2150-914x.

[29] L. Lamport and L. C. Paulson. "Should Your Specification Language Be Typed". In: *ACM Trans. Program. Lang. Syst.* 21.3 (May 1999), pp. 502–526. ISSN: 0164-0925. DOI: 10.1145/319301.319317. URL: http://doi.acm.org/10.1145/319301.319317.

[30] E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics*. Berlin, Heidelberg: Springer-Verlag, 1990. ISBN: 0-387-96957-8.

[31] A. Kaldewaij. *Programming: The Derivation of Algorithms*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1990. ISBN: 0-13-204108-1.

[32] D. Gries and F. B. Schneider. *A Logical Approach to Discrete Math*. Berlin, Heidelberg: Springer-Verlag, 1993. ISBN: 0-387-94115-0.

[33] R. C. Backhouse. *Program Construction: Calculating Implementations from Specifications*. New York, NY, USA: John Wiley & Sons, Inc., 2003. ISBN: 0470848820.

[34] A. Rizaldi, F. Immler, and M. Althoff. "A Formally Verified Checker of the Safe Distance Traffic Rules for Autonomous Vehicles". In: *NASA Formal Methods*. Ed. by S. Rayadurgam and O. Tkachuk. Cham: Springer International Publishing, 2016, pp. 175–190.

[35] J. Ferrante and C. Rackoff. "A Decision Procedure for the First Order Theory of Real Addition with Order". In: *SIAM J. Comput.* 4.1 (1975), pp. 69–76. DOI: 10.1137/0204006. URL: https://doi.org/10.1137/0204006.

[36] R. Loos and V. Weispfenning. "Applying Linear Quantifier Elimination". In: *The Computer Journal* 36.5 (1993), pp. 450–462. DOI: 10.1093/comjnl/36.5.450. eprint: /oup/backfile/content_public/journal/comjnl/36/5/10.1093/comjnl/36.5.450/2/360450.pdf. URL: +%20http://dx.doi.org/10.1093/comjnl/36.5.450.

[37] T. Nipkow. "Linear Quantifier Elimination". In: *Journal of Automated Reasoning* 45.2 (Oct. 2010), pp. 189–212. ISSN: 1573-0670. DOI: 10.1007/s10817-010-9183-0. URL: https://doi.org/10.1007/s10817-010-9183-0.

[38] S. McLaughlin and J. Harrison. "A Proof-Producing Decision Procedure for Real Arithmetic". In: *Automated Deduction – CADE-20*. Ed. by R. Nieuwenhuis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 295–314. ISBN: 978-3-540-31864-4.

[39] F. Immler. "Affine Arithmetic". In: *Archive of Formal Proofs* (Feb. 2014). http://isa-afp.org/entries/Affine_Arithmetic.html, Formal proof development. ISSN: 2150-914x.

[40] A. Chaieb. "Automated methods for formal proofs in simple arithmetics and algebra". Dissertation. München: Technische Universität München, 2008.

[41] F. Haftmann and T. Nipkow. "Code Generation via Higher-Order Rewrite Systems". In: *Functional and Logic Programming*. Ed. by M. Blume, N. Kobayashi, and G. Vidal. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 103–117. ISBN: 978-3-642-12251-4.

[42] A. E. Goodloe, C. Muñoz, F. Kirchner, and L. Correnson. "Verification of Numerical Programs: From Real Numbers to Floating Point Numbers". In: *NASA Formal Methods*. Ed. by G. Brat, N. Rungta, and A. Venet. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 441–446. ISBN: 978-3-642-38088-4.

[43] A. Doi, T. Butsuen, T. Niibe, T. Takagi, Y. Yamamoto, and H. Seni. "Development of a rear-end collision avoidance system with automatic brake control". In: *JSAE Review* 15.4 (1994), pp. 335–340. ISSN: 0389-4304. URL: `http://www.sciencedirect.com/science/article/pii/038943049490216X`.

[44] P. Seiler, B. Song, and K. Hedrick. "Development of a Collision Avoidance System". In: *SAE Conference*. 1998, Paper No. 980853. DOI: `10.4271/980853`. URL: `./1998/SeilerEtAl_98SAE_CollisionAvoidanceSystem.pdf`.

[45] D. Qu, X. Chen, W. Yang, and X. Bian. "Modeling of Car-Following Required Safe Distance Based on Molecular Dynamics". In: *Mathematical Problems in Engineering* 2014.Article ID 604023 (2014). DOI: `10.1155/2014/604023`.

[46] C. Chen, L. Liu, X. Du, Q. Pei, and X. Zhao. "Improving Driving Safety Based on Safe Distance Design in Vehicular Sensor Networks". In: *International Journal of Distributed Sensor Networks* 2012.Article ID 469067 (2012). DOI: `10.1155/2012/469067`.

[47] B. H. Wilson. "How soon to brake and how hard to brake: Unified analysis of the envelope of opportunity for rear-end collision warnings". In: *Enhanced Safety of Vehicles*. Vol. 47. 2001.

[48] S. M. Loos, A. Platzer, and L. Nistor. "Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified". In: *FM 2011: Formal Methods*. Ed. by M. Butler and W. Schulte. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 42–56. ISBN: 978-3-642-21437-0.

[49] A. Platzer. "A Complete Axiomatization of Quantified Differential Dynamic Logic for Distributed Hybrid Systems". In: *Logical Methods in Computer Science* 8.4 (2012), pp. 1–44. DOI: `10.2168/LMCS-8(4:17)2012`.

[50] S. Lefèvre, D. Vasquez, and C. Laugier. "A survey on motion prediction and risk assessment for intelligent vehicles". In: *ROBOMECH Journal* 1.1 (July 2014), p. 1. ISSN: 2197-4225. DOI: `10.1186/s40648-014-0001-z`. URL: `https://doi.org/10.1186/s40648-014-0001-z`.

[51] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. "Reachset Conformance Testing of Hybrid Automata". In: *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. HSCC '16. Vienna, Austria: ACM, 2016, pp. 277–286. ISBN: 978-1-4503-3955-1. DOI: `10.1145/2883817.2883828`. URL: `http://doi.acm.org/10.1145/2883817.2883828`.

[52] M. Althoff, D. Heß, and F. Gambert. "Road occupancy prediction of traffic participants". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. Oct. 2013, pp. 99–105. DOI: `10.1109/ITSC.2013.6728217`.

[53] M. Althoff and S. Magdici. "Set-Based Prediction of Traffic Participants on Arbitrary Road Networks". In: *IEEE Transactions on Intelligent Vehicles* 1.2 (June 2016), pp. 187–202.

[54] M. Koschi and M. Althoff. "SPOT: A Tool for Set-Based Prediction of Traffic Participants". In: *Proc. of the IEEE Intelligent Vehicles Symposium*. 2017, pp. 1679–1686.

[55] A. Rizaldi, S. Söntges, and M. Althoff. "On time-memory trade-off for collision detection". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. June 2015, pp. 1173–1180.

[56] M. Koschi, C. Pek, M. Beikirch, and M. Althoff. "Set-Based Prediction of Pedestrians in Urban Environments Considering Formalized Traffic Rules". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2018, pp. 2704–2711. doi: `10.1109/ITSC.2018.8569434`.

[57] M. Althoff and J. M. M. Dolan. "Online Verification of Automated Road Vehicles Using Reachability Analysis". In: *IEEE Transactions on Robotics* 30.4 (2014), pp. 903–918.

[58] O. Maler and D. Ničković. "Monitoring properties of analog and mixed-signal circuits". In: *International Journal on Software Tools for Technology Transfer* 15.3 (June 2013), pp. 247–268. issn: 1433-2787. doi: `10.1007/s10009-012-0247-9`. url: `https://doi.org/10.1007/s10009-012-0247-9`.

[59] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow. "Formalising and Monitoring Traffic Rules for Autonomous Vehicles in Isabelle/HOL". In: *Integrated Formal Methods*. Ed. by N. Polikarpova and S. Schneider. Cham: Springer International Publishing, 2017, pp. 50–66.

[60] A. Rizaldi and M. Althoff. "Formalising Traffic Rules for Accountability of Autonomous Vehicles". In: *Proc. of the IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 1658–1665.

[61] P. Bender, J. Ziegler, and C. Stiller. "Lanelets: Efficient map representation for autonomous driving". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. June 2014, pp. 420–425. doi: `10.1109/IVS.2014.6856487`.

[62] S. C. Park and H. Shin. "Polygonal chain intersection". In: *Computers & Graphics* 26.2 (2002), pp. 341–350. issn: 0097-8493. doi: `http://doi.org/10.1016/S0097-8493(02)00060-2`.

[63] F. Immler. "A Verified Algorithm for Geometric Zonotope/Hyperplane Intersection". In: *Proc. of International Conference on Certified Programs and Proofs*. 2015, pp. 129–136.

[64] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff. "Ensuring Drivability of Planned Motions Using Formal Methods". In: *Proc. of the Intelligent Transportation Systems Conference*. 2017, pp. 1661–1668.

[65] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008. ISBN: 9780262026499.

[66] A. Bauer, M. Leucker, and C. Schallhart. "Runtime Verification for LTL and TLTL". In: *ACM Trans. Softw. Eng. Methodol.* 20.4 (2011), 14:1–14:64. DOI: `10.1145/2000799.2000800`. URL: `https://doi.org/10.1145/2000799.2000800`.

[67] G. Roşu and K. Havelund. "Rewriting-Based Techniques for Runtime Verification". In: *Autom. Softw. Eng.* 12.2 (2005), pp. 151–197. DOI: `10.1007/s10515-005-6205-y`. URL: `https://doi.org/10.1007/s10515-005-6205-y`.

[68] J.-C. Küster. "Runtime Verification on Data-Carrying Traces". PhD thesis. The Australian National University, Oct. 2016.

[69] O. Kupferman and M. Y. Vardi. "Model Checking of Safety Properties". In: *Formal Methods in System Design* 19.3 (Nov. 2001), pp. 291–314. ISSN: 1572-8102. DOI: `10.1023/A:1011254632723`. URL: `https://doi.org/10.1023/A:1011254632723`.

[70] M. d'Amorim and G. Roşu. "Efficient Monitoring of ω-Languages". In: *Computer Aided Verification*. Ed. by K. Etessami and S. K. Rajamani. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 364–378. ISBN: 978-3-540-31686-2.

[71] O. Maler and D. Nickovic. "Monitoring Temporal Properties of Continuous Signals". In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings*. 2004, pp. 152–166. DOI: `10.1007/978-3-540-30206-3_12`.

[72] D. A. Basin, F. Klaedtke, S. Müller, and E. Zalinescu. "Monitoring Metric First-Order Temporal Properties". In: *J. ACM* 62.2 (2015), 15:1–15:45. DOI: `10.1145/2699444`.

[73] B. G. Buchanan and T. Headrick. "Some speculation about artificial intelligence and legal reasoning. Stanford Law Review (1970): 40-62". In: *Stanford Law Review* 23.1 (1970), pp. 40–62.

[74] S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Pearson Education, 2010. ISBN: 978-0-13-207148-2.

[75] M. Sergot. "The Representation of Law in Computer Programs". In: *Knowledge-Based Systems and Legal Applications*. Ed. by T. Bench-Capon. APIC. Elsevier Science, 1991. ISBN: 9781483295343.

[76] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. "The British Nationality Act as a Logic Program". In: *Commun. ACM* 29.5 (1986), pp. 370–386.

[77] T. J. M. Bench-Capon, G. O. Robinson, T. W. Routen, and M. J. Sergot. "Logic programming for large scale applications in law: A formalisation of Supplementary Benefit legislation". In: *Proceedings of the 1st International Conference on Artificial Intelligence and Law*. ICAIL '87. Boston, Massachusetts, USA: ACM, 1987, pp. 190–198. ISBN: 0-89791-230-6. DOI: 10.1145/41735.41757.

[78] A. Colmerauer and P. Roussel. "The birth of Prolog". In: *The Second ACM SIGPLAN Conference on History of Programming Languages*. HOPL-II. Cambridge, Massachusetts, USA: ACM, 1993, pp. 37–52. ISBN: 0-89791-570-4. DOI: 10.1145/154766.155362.

[79] M. J. Sergot. "Machine Intelligence 11". In: ed. by J. E. Hayes, D. Michie, and J. Richards. New York, NY, USA: Oxford University Press, Inc., 1988. Chap. Representing Legislation As Logic Programs, pp. 209–260. ISBN: 0-19-853718-2.

[80] L. T. McCarty. "Permissions and obligations". In: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI'83. Karlsruhe, West Germany: Morgan Kaufmann Publishers Inc., 1983, pp. 287–294.

[81] L. M. Royakkers. *Extending Deontic Logic for the Formalisation of Legal Rules*. Law and Philosophy Library. Springer Netherlands, 1998. ISBN: 9780792349822.

[82] A. J. Jones and M. Sergot. "Deontic logic in the representation of law: Towards a methodology". English. In: *Artificial Intelligence and Law* 1.1 (1992), pp. 45–64. ISSN: 0924-8463. DOI: 10.1007/BF00118478.

[83] C. Belta, V. Isler, and G. J. Pappas. "Discrete abstractions for robot motion planning and control in polygonal environments". In: *IEEE Transactions on Robotics* 21.5 (2005), pp. 864–874. ISSN: 1552-3098. DOI: 10.1109/TRO.2005.851359.

[84] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. "Temporal logic motion planning for dynamic robots". In: *Automatica* 45.2 (2009), pp. 343–352. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2008.08.008.

[85] V. Gavrilets, B. Mettler, and E. Feron. "Human-inspired control logic for automated maneuvering of miniature helicopter". In: *Journal of Guidance Control and Dynamics* 27.5 (2004), pp. 752–759.

[86] E. Frazzoli, M. A. Dahleh, and E. Feron. "Maneuver-based motion planning for nonlinear systems with symmetries". In: *IEEE Transactions on Robotics* 21.6 (2005), pp. 1077–1091. ISSN: 1552-3098. DOI: 10.1109/TRO.2005.852260.

[87] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. "Symbolic planning and control of robot motion [Grand Challenges of Robotics]". In: *IEEE Robotics Automation Magazine* 14.1 (2007), pp. 61–70. ISSN: 1070-9932. DOI: 10.1109/MRA.2007.339624.

[88] M. B. Egerstedt and R. W. Brockett. "Feedback can reduce the specification complexity of motor programs". In: *IEEE Transactions on Automatic Control* 48.2 (2003), pp. 213–223. ISSN: 0018-9286. DOI: 10.1109/TAC.2002.808466.

[89] B. Schürmann and M. Althoff. "Convex Interpolation Control with Formal Guarantees for Disturbed and Constrained Nonlinear Systems". In: *Proc. Hybrid Systems: Computation and Control*. 2017, pp. 121–130.

[90] A. Rizaldi, F. Immler, B. Schürmann, and M. Althoff. "A Formally Verified Motion Planner for Autonomous Vehicles". In: *Automated Technology for Verification and Analysis*. Ed. by S. K. Lahiri and C. Wang. Cham: Springer International Publishing, 2018, pp. 75–90.

[91] M. Althoff, M. Koschi, and S. Manzinger. "CommonRoad: Composable Benchmarks for Motion Planning on Roads". In: *Proc. of the IEEE Intelligent Vehicles Symposium*. 2017, pp. 719–726.

[92] B. Houska, H. J. Ferreau, and M. Diehl. "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization". In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312.

[93] B. Schürmann, A. El-Guindy, and M. Althoff. "Closed-Form Expressions of Convex Combinations". In: *Proc. of the American Control Conference*. 2016, pp. 2795–2801.

[94] V. Magron, X. Allamigeon, S. Gaubert, and B. Werner. "Formal proofs for Nonlinear Optimization". In: *Journal of Formalized Reasoning* 8.1 (2015), pp. 1–24.

[95] R. E. Bryant and D. R. O'Hallaron. *Computer Systems: A Programmer's Perspective*. 2nd. USA: Addison-Wesley Publishing Company, 2010. ISBN: 9780136108047.

[96] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. "Temporal Logic Motion Planning for Mobile Robots". In: *Proc. of the IEEE International Conference on Robotics and Automation*. 2005, pp. 2020–2025. DOI: 10.1109/ROBOT.2005.1570410.

[97] H. Roehm, J. Oehlerking, T. Heinz, and M. Althoff. "STL Model Checking of Continuous and Hybrid Systems". In: *Proc. of 14th International Symposium on Automated Technology for Verification and Analysis*. 2016, pp. 412–427.

[98] M. Althoff. "An Introduction to CORA 2015". In: *Applied Verification for Continuous and Hybrid Systems*. 2015, pp. 120–151.

[99] A. Krauss. "Automating recursive definitions and termination proofs in higher-order logic". PhD thesis. Technical University Munich, 2009.

[100] E. Plaku, L. E. Kavraki, and M. Y. Vardi. "Falsification of LTL safety properties in hybrid systems". In: *International Journal on Software Tools for Technology Transfer* 15.4 (2013), pp. 305–320.

[101] S. Mitsch, K. Ghorbal, D. Vogelbacher, and A. Platzer. "Formal verification of obstacle avoidance and navigation of ground robots". In: *The International Journal of Robotics Research* 36.12 (2017), pp. 1312–1340. DOI: 10.1177/0278364917733549.

[102] A. Anand and R. A. Knepper. "ROSCoq: Robots Powered by Constructive Reals". In: *Proc. of the 6th International Conference on Interactive Theorem Proving*. 2015, pp. 34–50.

[103] B. Bohrer, Y. K. Tan, S. Mitsch, M. Myreen, and A. Platzer. "VeriPhy: Verified Controller Executables from Verified Cyber-Physical System Models". In: *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2018.

[104] R. Kumar, M. O. Myreen, M. Norrish, and S. Owens. "CakeML: A Verified Implementation of ML". In: *Proc. of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 2014, pp. 179–191.

[105] T. Wongpiromsarn, S. Karaman, and E. Frazzoli. "Synthesis of provably correct controllers for autonomous vehicles in urban environments". In: *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2011, pp. 1168–1173. DOI: `10.1109/ITSC.2011.6083056`.