CARL VON OSSIETZKY universität OLDENBURG

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

Differential Dynamic Logics

Automated Theorem Proving for Hybrid Systems

Dissertation zur Erlangung des Grades eines
Doktors der Naturwissenschaften

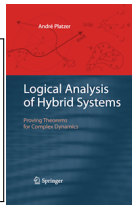
Dipl.-Inform. André Platzer

Gutachter:

Prof. Dr. Ernst-Rüdiger Olderog
Prof. Dr. Tobias Nipkow
Prof. Dr. George J. Pappas

Eingereicht: 23. September 2008
Tag der Disputation: 19. Dezember 2008

Extended version appeared as book:
A. Platzer. Logical Analysis of Hybrid Systems:
Proving Theorems for Complex Dynamics.
Springer, 2010. DOI 10.1007/978-3-642-14509-4



Abstract

Hybrid systems are models for complex physical systems and are defined as dynamical systems with interacting discrete transitions and continuous evolutions along differential equations. With the goal of developing a theoretical and practical foundation for deductive verification of hybrid systems, we introduce *differential dynamic logic* as a new logic with which correctness properties of hybrid systems with parameterized system dynamics can be specified and verified naturally. As a verification technique that is suitable for automation, we introduce a free variable proof calculus with a novel combination of real-valued free variables and Skolemisation for lifting quantifier elimination for real arithmetic to dynamic logic. The calculus is compositional, i.e., it reduces properties of hybrid systems successively to properties of their parts. Our main result proves that this calculus *axiomatises* the transition behaviour of hybrid systems completely relative to differential equations.

Systematically, we develop automated theorem proving techniques for our calculus and present proof procedures to tackle the complexities of integrating decision procedures for real arithmetic. For our logic, we further complement discrete induction with *differential induction* as a new continuous generalization of induction, with which hybrid systems can be verified by exploiting their differential constraints algebraically without having to solve them. Finally, we develop a fixedpoint algorithm for computing the *differential invariants* required for differential induction, and we introduce a *differential saturation procedure* that refines the system dynamics successively with differential invariants until correctness becomes provable. As a systematic combination of logic-based techniques, we obtain a sound verification procedure that is particularly suitable for parametric hybrid systems.

We demonstrate our approach by verifying safety, controllability, liveness, and collision avoidance properties in case studies ranging from train control applications in the *European Train Control System* to air traffic control, where we prove collision avoidance in *aircraft roundabout maneuvers*.

Keywords: dynamic logic, differential equations, logic for hybrid systems, free variable calculus, sequent calculus, axiomatisation, automated theorem proving, real arithmetic, verification of hybrid systems, differential induction, fixedpoint engines, train control, air traffic control

Zusammenfassung

Hybride Systeme sind Modelle für komplexe physikalische Systeme und als dynamische Systeme mit interagierenden diskreten Transitionen und kontinuierlichen Evolutionen längs Differentialgleichungen definiert. Mit dem Ziel, ein theoretisches und praktisches Fundament für die deduktive Verifikation hybrider Systeme zu entwickeln, stellen wir *differentielle dynamische Logik* als eine neue Logik vor, mit der Korrektheitsaussagen von hybriden Systemen mit parametrischer Systemdynamik auf natürliche Art und Weise spezifiziert und verifiziert werden können. Als Verifikationstechnik die sich zur Automatisierung eignet führen wir ferner einen Beweiskalkül ein, der eine neuartige Kombination reell-wertiger freier Variablen mit Skolemisierung besitzt, um Quantorenelimination für reelle Arithmetik auf dynamische Logik zu liften. Der Kalkül arbeitet kompositionell, d.h., er reduziert Eigenschaften von hybriden Systemen sukzessive auf Eigenschaften ihrer Bestandteile. Unser Hauptresultat zeigt, dass dieser Kalkül das Transitionsverhalten hybrider Systeme vollständig relativ zu Differentialgleichungen *axiomatisiert*.

Systematisch entwickeln wir automatische Beweistechniken für den Kalkül und präsentieren Beweisprozeduren, die die Komplexitäten der Integration von Entscheidungsprozeduren reeller Arithmetik bewältigen. Für unsere Logik komplementieren wir weiterhin diskrete Induktion mit *differentieller Induktion* als neuartige kontinuierliche Generalisierung der Induktion, mit der hybride Systeme verifiziert werden können indem deren differentielle Relationen algebraisch ausgenutzt werden ohne sie lösen zu müssen. Schlußendlich entwickeln wir einen Fixpunktalgorithmus um die dafür benötigten *differentiellen Invarianten* zu berechnen und führen eine *differentielle Saturierungsprozedur* ein, welche Systemdynamiken sukzessive mit differentiellen Invarianten solange verfeinert bis Korrektheit beweisbar wird. Als eine systematische Kombination logik-basierter Techniken erhalten wir eine korrekte Verifikationsprozedur, die sich besonders gut für parametrische hybride Systeme eignet.

Wir demonstrieren unseren Ansatz indem wir Sicherheits-, Steuerbarkeits-, Lebendigkeits- und Kollisionsfreiheitseigenschaften in Fallstudien nachweisen, die von Anwendungen in der Zugsteuerung wie dem *European Train Control System* bis hin zur Flugsicherung reichen, wo wir Kollisionsvermeidung von *Kreisverkehrmanövern im Flugverkehr* nachweisen.

Acknowledgements

My sincere thanks go to Prof. Ernst-Rüdiger Olderog for his excellent advise, his support, and for giving me the opportunity to work in one of the most fascinating areas of science within a group of a friendly and productive atmosphere. My advisor, Prof. Ernst-Rüdiger Olderog and the Director of AVACS, Prof. Werner Damm, both deserve my highest gratitude, not only for their continuous support and for their faith, but also for allowing me the freedom to pursue my own research ambitions in the stimulating context of the AVACS project. Ultimately, this made it possible that I developed the theory and practice of differential dynamic logics in all their elegance.

I want to thank the external referees, Prof. Tobias Nipkow from the Technical University of Munich and Prof. George J. Pappas from the University of Pennsylvania. It is an honor for me that they were willing to invest their valuable time and effort into the careful reviewing of my thesis. In fact, I am thankful to all members of my PhD committee, Werner Damm, Ernst-Rüdiger Olderog, George J. Pappas, Tobias Nipkow, and Hardi Hungar for fruitful discussions and for the highest support they offered for my work.

I am especially grateful to Prof. Edmund M. Clarke who invited me to Carnegie Mellon University several times, for his support and interest and for sharing with me parts of his huge knowledge in all areas of formal methods. I further want to acknowledge the help by Prof. Peter H. Schmitt from the University of Karlsruhe (TH), Prof. Bernhard Beckert and Prof. Ulrich Furbach from the University of Koblenz-Landau, Prof. Reiner Hähnle from the Chalmers University of Technology, Gothenburg, Sweden, Prof. Edmund M. Clarke from Carnegie Mellon Univeristy, Pittsburgh, PA, and Prof. Rajeev Goré from the Australian National University, Canberra, at various stages of my career.

For many fruitful discussions I thank my colleagues and friends, Ingo Brückner, Henning Dierks, Johannes Faber, Sibylle Fröschle, Jochen Hoenicke, Stephanie Kemper, Roland Meyer, Michael Möeller, Jan-David Quesel, Tim Strazny and especially my office mate Andreas Schäfer. Ernst-Rüdiger Olderog, Andreas Schäfer, Johannes Faber, Ingo Brückner, Roland Meyer, Henning Dierks, Silke Wagner, Nicole Betz, and Alex Donzé also deserve credit for proofreading some of my papers, many of which have formed the basis for this thesis.

Furthermore, I thank Jan-David Quesel for writing a master's thesis under my

supervision and for much support with the implementation of the verification tool KeYmaera. I am also thankful for indispensable and reliable help from Richard Bubel and Philipp Rümmer with the implementation internals of the KeY basis.

Especially, I thank my parents, Rudolf and Brigitte Platzer, and my sister, Julia, for their continuous support and encouragement, and I thank my wife, Nicole, for her true faith in me.

Funding This research has been partly supported by the German Research Council (DFG) under grant SFB/TR 14 AVACS ("Automatic Verification and Analysis of Complex Systems", see `www.avacs.org`) in a Transregional Collaborative Research Center of the Max Planck Institute and the Universities of Oldenburg, Saarbrücken, and Freiburg in Germany, with associated cooperations with the University of Pennsylvania, ETH Zürich, and Academy of Sciences of the Czech Republic. It has further been supported partly by a research fellowship of the German Academic Exchange Service (DAAD), and by a research award of the Floyd und Lili Biava Stiftung.

Part of this work has also been supported by the National Science Foundation under grant nos. CCR-0411152, CNS-0411152, CCF-0429120, CCR-0121547, CCR-0098072, by General Motors and the Carnegie Mellon University-General Motors Collaborative Research Laboratory under grant no. GM9100096UMA, the US Army Research Office (ARO) under grant no. DAAD19-01-1-0485, by the Air Force Research Office (AFRO) under contract no. FA9550-06-1-0312, and the Office of Naval Research (ONR) under grant no. N00014-01-1-0796, and the Naval Research Laboratory (NRL).

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution or government.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

> "Time is defined so that motion looks simple"
> [MTW73, p.23] expressing thoughts of Henri Poincaré

Ensuring correct functioning of complex physical systems is among the most challenging and most important problems in computer science, mathematics, and engineering. In addition to nontrivial underlying physical system dynamics, the behaviour of complex systems is determined increasingly by computerised control and automatic analog or digital decision-making, e.g., in aviation, railway, or automotive applications. At the same time, correct decisions and control of these systems is becoming increasingly important, because more and more safety-critical processes are regulated using automatic or semiautomatic controllers, including the European Train Control System [ERT02], collision avoidance maneuvers in air traffic control [TPS98, LLL00, DMC05, PC07, GMAR07, HKT07], car platooning technology for highways following the California PATH project [HESV91], recent driverless vehicle technology [Bue08], or biomedical applications like automatic glucose regulation for diabetes patients [PDP01]. As a more general phenomenon of complex physical systems that are exemplified in these scenarios, correct system behaviour depends on correct functioning of the *interaction* of control with physical system dynamics and is not just an isolated property of only the control logic or only the physical system dynamics.

To illustrate typical aspects and effects in these application areas, we take a look at two examples in more detail, which will serve as running examples and case studies throughout this thesis. In high-speed trains like ICE (InterCityExpress) or TGV (train à grande vitesse), whose high mass (1000–3000 tons) and high speed (320km/h) causes them to require fairly long braking distances (more than 3.8km), safe driving is impossible just based on sight without automatic technical means that enforce a safe minimum distance between trains. The *European Train Control System* (ETCS), which is currently being developed and installed in Europe [ERT02], regulates and protects train movement according to movement authorities (MA) that are negotiated dynamically in rapid succession by wireless

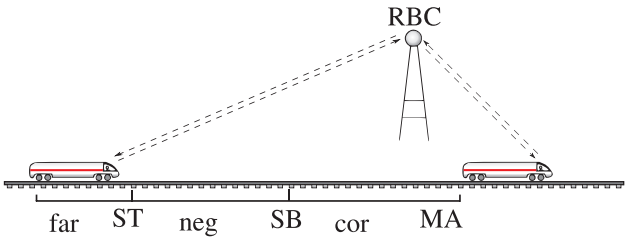
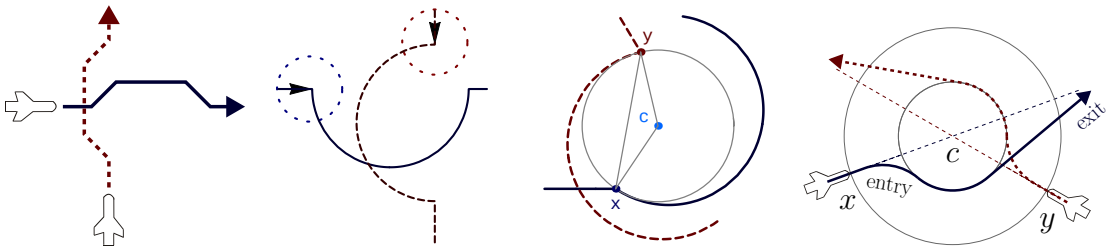


Figure 1.1.: European Train Control System

communication with decentralised radio block controllers (RBC), see Figure 1.1. With the next generation development of ETCS, Level 3, all classical fixed track-side signaling and fixed track segment partitioning with physical separation will become obsolete, thereby advancing to a fully autonomous operation of ETCS in order to achieve its performance goals of maximum speed and density on the track. Yet, a safe operation of ETCS requires that—while achieving these performance goals—the train controllers still always respect their local movement authorities and that the radio block controllers only grant compatible movement authorities to each of the trains. Even in emergency situations, the overall train control system must always ensure that the trains cannot crash into one another. To determine correct functioning of these controllers it has to be shown that the train positions, which evolve dynamically over time, are always safely separated. For this, however, we need to be able to analyse the interaction of the train control logic and the ETCS cooperation protocol with a model of the actual physical train dynamics, because collision freedom is not an isolated property of only the discrete cooperation-layer control protocol, only the local train control decision process, or only the continuous train dynamics, but a joint property of their superposition. In safety-critical complex physical systems like ETCS, full formal verification is indeed quite important and of particular practical relevance for ensuring that they operate safely: Despite careful development and testing, safety violations have recently been reported in ETCS [Gro07] even at its moderate currently deployed level.

In air traffic control, collision avoidance maneuvers [TPS98, LLL00, DMC05, PC07, GMAR07, HKT07] are used to resolve conflicting flight paths that arise during arbitrary free flight of the aircraft, see Figure 1.2. They are last resort means for resolving air traffic conflicts that could lead to collisions and have not been detected by the pilots during free flight or by the flight directors of the Air Route Traffic Control Centres. Consequently, complicated online trajectory prediction, trajectory evaluation, or lengthy maneuver negotiation may no longer be feasible in the short time that remains for resolving the conflict. For instance, in the tragic mid-flight collision in Überlingen [BFU04], only less than one minute of maneuvering time would have been available to prevent the collision after the on-board traffic alert and collision avoidance system TCAS [LLL00] signalled a traffic alert.



1.2a: Linear 1.2b: Circular 1.2c: Tangential 1.2d: Flyable

Figure 1.2.: Collision avoidance maneuvers in air traffic control

Thus, for safe aircraft control we need particularly reliable instant reactions with maneuvers whose correctness has been established previously by a thorough offline analysis. To ensure correct functioning of aircraft collision avoidance maneuvers under all circumstances, the temporal evolution of the aircraft in space must be analysed carefully together with the effects that maneuvering control decisions have on their dynamics, giving again a superposition of physical system dynamics with control.

1.1. Technical Context

1.1.1. Hybrid Systems

As a common mathematical model for complex physical systems, *hybrid systems* [Tav87, ACHH92, NOSY92, ACH+95, Bra95b, Hen96, AHH96, BBM98, LPY99, DN00, PAM+05, DHO06, Lib03] are dynamical systems [Per91, KH96, Sib75] where the system state evolves over time according to interacting laws of discrete and continuous dynamics, with the idea being to capture the superposition of physical system dynamics with control at a natural modelling level. For discrete transitions, the hybrid system changes state instantaneously and possibly discontinuously. During continuous transitions, the system state is a continuous function of continuous time and varies according to a differential equation, which is possibly subject to domain restrictions or algebraic relations resulting from physical circumstances or the interaction of continuous dynamics with discrete control. Continuous dynamics results, e.g., from the continuous movement of a train along the track (train position z evolves with velocity v along the differential equation $z' = v$ where z' is the time-derivative of z) or from the continuous variation of its velocity over time ($v' = a$ with acceleration a). Other behaviour can be modelled more naturally by discrete dynamics, for example, the instantaneous change of control variables like the acceleration (e.g., the changing of a by setting $a := -b$ with braking force $b > 0$) or change of status information in discrete controllers. Both kinds of dynamics in-

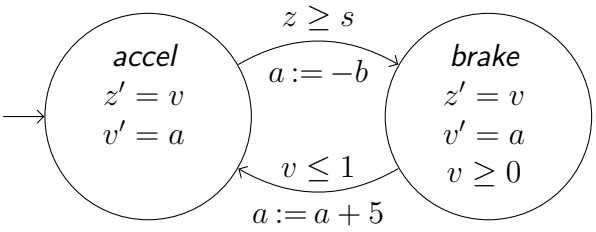


Figure 1.3.: Hybrid automaton for an (overly) simplified train control system

teract, e.g., when measurements of the continuous state affect decisions of discrete controllers (the train switches to braking mode when v is too high). Likewise, they interact when the resulting control choices take effect by changing the control variables of the continuous dynamics (e.g., changing control variable a in $z'' = a$). The superposition of continuous dynamics with analog or discrete control causes complex system behaviour, which can neither be verified by purely continuous reasoning (because of the discontinuities caused by discrete transitions) nor by considering discrete change in isolation (because safety depends on continuous states).

Among several other models for hybrid systems [BBM98], the model of *hybrid automata* [Hen96, ACH+95] is the most widely used notation. They specify discrete and continuous dynamics in a graph, see Figure 1.3 for a (much too) simple train control example. Each node corresponds to a continuous dynamical system and is decorated by its differential equation and an invariant region specifying the maximum domain of evolution. In the node *brake* of Figure 1.3, the differential equations $z' = v, v' = a$ only apply within the invariant region $v \geq 0$ (the train does not move backwards when braking). Edges specify the discrete switching behaviour between the respective modes of continuous evolution. They can be decorated with conditions (*guards*) that need to hold and with discrete state transformations (*jumps*) that take instantaneous effect when the system follows the edge. For example, the automaton in Figure 1.3 can take an edge to leave node *accel* when train position z passed point s, which sets the acceleration to braking by $a := -b$, and enter node *brake*.

1.1.2. Model Checking

As a standard verification technique, model checking [EC82, CGP99] has been used successfully for verifying temporal logic properties [Pnu77, EC82, EH86, ACD90] of finite-state abstractions of automata-based transition structures by exhaustive state space exploration [ACH+95, HNSY92, Hen96, MPM05]. The continuous state spaces of hybrid automata, however, do not admit equivalent finite-state abstractions [Hen96]. Because of this, model checkers for hybrid automata use various approximations [HNSY92, ACH+95, Hen96, CK03, Frä99, AW01, CFH+03, ADG03,

Tiw03, MPM05] and are still more successful in falsification than in verification. Furthermore, for hybrid systems with symbolic parameters in the dynamics, correctness crucially depends on the free parameters (e.g., b and s in Figure 1.3). It is, however, quite difficult to determine corresponding symbolic parameter constraints from concrete values of a counterexample trace produced by a model checker, especially if they rely on nonstructural state splitting [CK03, CFH$^+$03, ADG03, Fre05]. Finally, in hybrid systems with nontrivial interaction of discrete and continuous dynamics, parameters also have a nontrivial impact on the system behaviour, leading to nonlinear parameter constraints and nonlinearities in the discrete and continuous dynamics. For instance, the nonlinear constraint $s \geq \frac{v^2}{2b}$ will turn out to be important for Figure 1.3. Thus, standard model checking approaches [Hen96, ACH$^+$95, CK03, Fre05] cannot be used, as they require at most linear discrete dynamics.

1.1.3. Deductive Verification

Deductive approaches [BHS07, BP06, HLS$^+$96, HKT00, Har79, ZRH92, Dav97, DN00] have been used for verifying systems by proofs instead of by state space exploration and, thus, do not require finite-state abstractions. Davoren and Nerode [DN00] further argue that deductive methods support formulas with free parameters. First-order logic, for instance, has widely proven its power and flexibility in handling symbolic parameters as free or quantified logical variables. However, first-order logic has no built-in means for referring to state transitions, which are crucial for verifying dynamical systems where states change over time.

In *temporal logics* [Pnu77, EC82, EH86, ACD90, Sti92], state transitions can be referred to using modal operators. In deductive approaches, temporal logics have been used to prove validity of formulas in calculi [DN00, ZRH92]. Valid formulas of temporal logic, however, only express generic facts that are true for all systems, regardless of their actual behaviour. Hence, the behaviour of a specific hybrid system would need to be characterised declaratively with temporal formulas to obtain meaningful results. Then, however, equivalence of declarative temporal representations and actual system operations needs to be proven separately using other techniques. Furthermore, even for finite-state systems, direct temporal characterisations can be computationally infeasible, e.g., direct temporal characterisations transform the linear-time CTL model checking problem into an EXPTIME-complete satisfiability problem [Eme90].

Dynamic logic (DL) [Pra76, Har79, HKT00] is a successful approach for verifying infinite-state discrete systems deductively [BHS07, BP06, HLS$^+$96, HKT00, Har79]. Like model checking, DL does not need declarative characterisations of system behaviour but can analyse the transition behaviour of actual operational system models directly. Yet, operational models are fully *internalised* within DL-formulas, and DL is closed under logical operators. Within a single specification and verification language, it combines operational system models with means to talk about the

states that are reachable by system transitions. DL provides parameterised modal operators $[\alpha]$ and $\langle\alpha\rangle$ that refer to the states reachable by system α and can be placed in front of any formula. The formula $[\alpha]\phi$ expresses that all states reachable by system α satisfy formula ϕ. Likewise, $\langle\alpha\rangle\phi$ expresses that there is at least one state reachable by α for which ϕ holds. These modalities can be used to express necessary or possible properties of the transition behaviour of α in a natural way. They can be nested or combined propositionally. In first-order dynamic logic with quantifiers, $\exists p\,[\alpha]\langle\beta\rangle\phi$ says that there is a choice of parameter p such that for all possible behaviour of system α there is a reaction of system β that ensures ϕ. Likewise, $\exists p\,([\alpha]\phi \wedge [\beta]\psi)$ says that there is a choice of parameter p that makes both $[\alpha]\phi$ and $[\beta]\psi$ true, simultaneously.

On the basis of first-order logic over the reals, which we use to describe safe regions of hybrid systems and to quantify over parameter choices, we introduce a first-order dynamic logic over the reals with modalities that directly quantify over the possible transition behaviour of hybrid systems. Since hybrid systems are subject to both continuous evolution and discrete state change, we generalise dynamic logic so that operational models α of hybrid systems can be used in modal formulas like $[\alpha]\phi$.

1.1.4. Compositional Verification

As a verification technology for our logic, we devise a compositional proof calculus for verifying properties of a hybrid system by proving properties of its parts. The calculus decomposes $[\alpha]\phi$ symbolically into an equivalent formula,, for instance, $[\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2$ about subsystems α_i of α and subproperties ϕ_i of ϕ. With this, $[\alpha]\phi$ can simply be verified by proving the $[\alpha_i]\phi_i$ separately and combining the results conjunctively. In particular, synthesised parameter constraints carry over from the latter to the former just by conjunction.

Unfortunately, hybrid automata are not suitably compositional for this purpose. Their graph structures cannot be decomposed into subgraphs α_i such that the formula $[\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2$ is equivalent to $[\alpha]\phi$, because of the dangling edges between the subgraphs α_i. For instance, the automaton in Figure 1.3 cannot simply be verified by proving $[\mathit{accel}]\phi \wedge [\mathit{brake}]\phi$, because the effects of edges between the nodes need to be taken into account.

Consequently, we do not impose an automaton structure on the system. Instead, we introduce *hybrid programs* as a textual program notation for hybrid systems by extending conventional discrete program notations in classical DL. They allow for flexible programmatic combinations of elementary discrete or continuous transitions by structured control programs with a perfectly compositional semantics: The semantics of a compound hybrid program is a simple function of the semantics of its parts and does not further depend on automata graph structures. The resulting first-order dynamic logic for hybrid programs is called *differential dynamic*

logic ($\mathsf{d}\mathcal{L}$) and constitutes a natural specification and verification logic for hybrid systems. With the goal of developing a solid theoretical, practical, and applicable foundation for deductive verification of hybrid systems by automated theorem proving, the focus of this thesis is a thorough analysis of the logic $\mathsf{d}\mathcal{L}$ and its calculus.

1.1.5. Lifting Quantifier Elimination

When proving $\mathsf{d}\mathcal{L}$ formulas, interacting hybrid dynamics causes interactions of arithmetic quantifiers and dynamic modalities, which both affect the values of symbols. For continuous evolutions, we have to prove formulas like $\forall t\,[\alpha]x{\geq}0$ expressing that, for all durations t of some evolution in α, $x \geq 0$ holds after all executions of system α. Standard first-order quantifier rules [HS94, Fit96, FM99] are incomplete for handling these situations, because they are based on instantiation or unification, which is already insufficient for proving the tautology $\forall z\,(z^2 \geq 0)$. Unfortunately, decision procedures for real arithmetic like real quantifier elimination [Tar51, CH91] cannot handle $\forall t$ either, because of the modality $[\alpha]$. The actual algebraic constraints on t still depend on how the system variables evolve along the dynamics of α. This effect inherently results from the interacting dynamics of hybrid systems, where the duration t of a continuous evolution determines the resulting state and, hence, affects all subsequent discrete or continuous evolutions in α. Thus, the effect of α first needs to be analysed with respect to the arithmetical constraints it imposes on t for $x \geq 0$ to hold, before the quantifier $\forall t$ can be handled.

In this thesis, we present a calculus that is suitable for automation and combines deductive and arithmetical quantifier reasoning within a single proof. It introduces real-valued free variables and Skolem terms to postpone quantifier elimination and continue reasoning beyond the occurrence of a real quantifier in front of a modality. Later, however, our calculus reintroduces a corresponding quantifier into the proof when its algebraic constraints have been discovered completely. For $\forall t\,[\alpha]x \geq 0$, our calculus will, for instance, continue with the unquantified kernel $[\alpha]x \geq 0$ after replacing t by a Skolem term $s(x)$. Once all arithmetical constraints on $s(x)$ are known, a quantifier for $s(x)$ is reintroduced and handled by real quantifier elimination [Tar51, CH91]. In a similar manner, our calculus combines quantifier elimination with deduction for handling existential real quantifiers using real-valued free variables.

We introduce a calculus that makes this intuition formally precise. Crucially, we exploit the relationship of Skolem terms and free variables in order to keep track of the lost quantifier nesting to prohibit unsound rearrangements of quantifiers when they are reintroduced. The corresponding calculus rules are perfectly natural and comply with the prerequisites of quantifier elimination over the reals. Further, the $\mathsf{d}\mathcal{L}$ semantics and calculus are fully compositional so that properties of a hybrid program can be proven by reduction to properties of its parts following a structural symbolic decomposition within the $\mathsf{d}\mathcal{L}$ calculus.

1.1.6. Differential Induction and Differential Strengthening

In Chapter 3, we extend our logic $\mathsf{d}\mathcal{L}$ to the differential-algebraic dynamic logic DAL, which is the logic of general hybrid change. DAL provides differential-algebraic programs (DA-programs) as general models for hybrid systems by allowing for propositional operators and quantifiers in discrete and continuous transitions. The standard approach to dealing with continuous dynamics for hybrid systems is to use symbolic or numerical solutions of their respective differential equations. Unfortunately, the range of systems that is amenable to these techniques is fairly limited, because even solutions of simple linear differential equations quickly fall into undecidable classes of arithmetic. For instance, the solutions of $s' = c, c' = -s$ are trigonometric functions like sin and cos. As a means for verifying hybrid systems with challenging continuous dynamics without having to solve differential equations, we further complement discrete induction with a new form of differential induction and add differential strengthening for refining the system dynamics with auxiliary invariants.

1.2. Related Work

Model Checking of Hybrid Automata Model checking approaches work by state space exploration and—due to their undecidable reachability problem—require [Hen96] various abstractions or approximations [HNSY92, ACH+95, Hen96, Frä99, AW01, CFH+03, Tiw03, MPM05] for hybrid automata, including numerical approximations [CK03, ADG03].

Beyond standard approaches [ACH+95, Hen96, Fre05] for linear systems with constant dynamics, Lafferriere et al. [LPY99, LPS00, LPY01] presented a decision procedure for o-minimal hybrid automata and classes of linear dynamics with a homogeneous eigenstructure. They analyse the discrete and continuous dynamics independently, which requires completely decoupled dynamics with forgetful jumps, i.e., where the outcome of a jump is completely independent of the continuous state.

Chutinan and Krogh [CK03] presented polyhedral approximations of hybrid automata with polyhedral discrete dynamics, invariants, and initial state sets. Fränzle [Frä99] showed that reachability is decidable for specific classes of robust polynomial hybrid automata, where the safe and unsafe states are sufficiently separate and the safe region is bounded. Asarin et al. [ADG03] used piecewise linear numerical approximations in an approximate reachability algorithm for continuous systems with known Lipschitz bounds. Mysore et al. [MPM05] showed decidability of bounded-time and bounded switching reachability prefixes of semi-algebraic hybrid automata.

Model checking tools like HyTech [HH94], PHAVer [Fre05], CheckMate [CK03], or other approaches [Hen96, Frä99, PAM+05] cannot handle our applications with non-

linear switching, nonlinear discrete and continuous dynamics, and high-dimensional state spaces.

Because hybrid systems do not admit equivalent finite-state abstractions [Hen96] and due to general limits of numerical approximation [PC07], model checkers are still more successful in falsification than in verification. To obtain a sound verification approach and for improved handling of free parameters [DN00], we follow a symbolic logic-based approach and support $\mathsf{d}\mathcal{L}$ as a significantly more expressive specification language. Finally, we introduce hybrid programs as a more uniform model for hybrid systems that is amenable to compositional symbolic verification.

Logics for Real-time Systems Logics for real-time systems [HNSY92, SRH02] are not expressive enough to capture the dynamics of hybrid systems, particularly their differential equations, which are the main focus of this thesis. For instance, Schobbens et al. [SRH02] give complete axiomatisations of two decidable dense time propositional linear temporal logics. Unfortunately, in these propositional logics one cannot express that relevant separation properties like $(x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2$ hold always during the flight of aircraft guided by specific flight controllers.

Logics for Hybrid Systems Zhou et al. [ZRH92] extended duration calculus with mathematical expressions in derivatives of state variables. They use a multitude of calculus rules and a non-constructive oracle that requires external mathematical reasoning about the notions of derivatives and continuity.

Davoren and Nerode [Dav97, DN00] presented a semantics of modal μ-calculus in hybrid systems and examine topological aspects. They provided Hilbert-style calculi to prove formulas that are valid for all hybrid systems simultaneously. With this, however, only limited information can be obtained about a particular system: In propositional modal logics, system behaviour needs to be axiomatised declaratively in terms of abstract actions a, b, c of unknown effect.

The strength of our logic primarily is that it is an expressive *first-order* dynamic logic: It handles actual operational models of hybrid systems like $a := a + 5; z'' = a$ instead of abstract propositional actions of unknown effect. The advantage of our calculus in comparison to others [ZRH92, Dav97, DN00] is that it provides a constructive modular combination of arithmetic reasoning with reasoning about hybrid transitions and works by structural decomposition. With this, our calculus can be used easily for verifying actual operational hybrid system models, which is of considerable practical interest [Hen96, BBM98, CK03, CFH$^+$03, MPM05, DHO06, PC07, DMO$^+$07]. It supports free parameters and first-order definable flows, which are well-suited for verifying the coordination of train dynamics. First-order approximations of more general flows can be used according to [AW01, PC07, Per91].

Specification Languages for Hybrid Systems Inspired by He [Jif94], Zhou et al. [CJR95] presented a hybrid variant of CSP as a language for describing hybrid systems. They gave a semantics in extended duration calculus [ZRH92] but no verification technique.

Rönkkö et al. [RRS03] extended guarded command programs with differential relations and gave a weakest-precondition semantics in higher-order logic with built-in derivatives. Without providing a means for verification of this higher-order logic, this approach is still limited to providing a notational variant of classical mathematics.

Uses of Deduction for Hybrid Systems Manna et al. [MS98, KMP00] and Ábrahám et al. [ÁMSH01] used theorem provers for checking invariants of hybrid automata in STeP [MS98] or PVS [ÁMSH01], respectively. Their working principle is, however, quite different from ours. Given a hybrid automaton and given a global system invariant, they compile, in a single step, a verification condition expressing that the invariant is preserved under all transitions of the hybrid automaton. Hence, hybrid aspects and transition structure vanish completely before the proof starts. All that remains is a flat quantified mathematical formula. Which hybrid systems can be verified with this approach in practice strongly depends on the general mathematical proving capabilities of STeP and PVS, which typically require user interaction.

In contrast, we follow a fully symbolic approach using a genuine specification and verification logic for hybrid systems. Our dynamic logic works deductively by symbolic decomposition and preserves the transition structure during the proof, which simplifies traceability of results considerably. Further, the structure in this symbolic decomposition can be exploited for deriving invariants or parametric constraints. Consequently, in $\mathsf{d}\mathcal{L}$, invariants do not necessarily need to be given beforehand. Moreover, in practice, guiding quantifier elimination procedures along natural splitting possibilities of the structural decomposition performed by the $\mathsf{d}\mathcal{L}$ calculus turns out to be important for successful automatic proof strategies (Chapter 5).

1.3. Contributions

Our main conceptual contribution is a series of differential dynamic logics for hybrid systems ($\mathsf{d}\mathcal{L}$, DAL, and dTL), which capture the logical quintessence of the dynamics of hybrid systems succinctly. Our logics provide a uniform semantics and concise language for specifying and verifying correctness properties of general hybrid systems with sophisticated dynamics. Our main practical contribution is a concise *free variable calculus* that axiomatises the transition behaviour of hybrid systems relative to differential equation solving. With our generalisation of free variable calculi to dynamic logic over the reals, the calculus is suitable for

automated theorem proving and for verifying hybrid superpositions of interacting discrete and continuous dynamics compositionally.

Our main theoretical contribution is that we prove our calculi to be complete relative to the handling of differential equations. To the best of our knowledge, this is the first relative completeness proof for a logic of hybrid systems, and even the first formal notion of hybrid completeness. Our results fully align hybrid and continuous reasoning proof-theoretically and show that hybrid systems with interacting repetitive discrete and continuous evolutions can be verified whenever differential equations can.

We further contribute a verification calculus that includes uniform proof rules for differential induction along differential equations or more general differential constraints, using a combination of differential invariants, differential variants, and differential strengthening for verifying hybrid systems without having to solve their differential constraints. Based on these calculi, we develop a fixedpoint verification algorithm that computes the required invariants and differential invariants for a formula and refines the underlying system dynamics as needed during the proof.

As applied contributions, we demonstrate the capabilities of our logics, calculi, and algorithms by verifying collision avoidance in realistic train control applications and challenging air traffic control maneuvers. Overall, our logic-based verification approach for hybrid systems can successfully verify realistic applications that were out of scope for other approaches, both for theoretical reasons and for scalability reasons.

1.4. Structure of this Thesis

This thesis consists of three parts that basically correspond to the theory, practice, and applications, respectively, of differential dynamic logics for hybrid systems. You are now reading the introduction.

Logics and Calculi In Part I, which is the core part of this thesis, we introduce novel logics and proof calculi that form the new conceptual, formal, and technical basis for logic-based verification of hybrid systems. In Chapter 2, we introduce the differential dynamic logic $\mathsf{d}\mathcal{L}$ as a variant of dynamic logic that is suitable for specifying and verifying properties of hybrid systems. It generalises dynamic logic to dynamic logic over the reals in the presence of continuous state transitions along differential equations. As a verification technique, we present a new compositional sequent calculus for $\mathsf{d}\mathcal{L}$ that is suitable for automation and integrates handling of real quantifiers by generalising Skolemisation and free variables to the reals. In Chapter 2, we also show completeness relative to differential equations as the most fundamental theoretical result in this thesis.

In Chapter 3, we introduce the differential-algebraic logic DAL that extends the class of hybrid system models by allowing more general differential-algebraic equations and quantified nondeterminism. Further, we present a uniform theory of differential induction, differential invariants, differential variants, and differential strengthening as central symbolic verification techniques for handling challenging continuous dynamics in hybrid systems without having to solve their differential equations.

In Chapter 4, we address the handling of temporal properties and introduce the differential temporal dynamic logic dTL along with a calculus that reduces temporal properties to $\mathsf{d}\mathcal{L}$ properties. The extensions of $\mathsf{d}\mathcal{L}$ that we present in Chapter 3 and Chapter 4 are complementary and compatible. Their direct modular combination immediately defines the differential-algebraic temporal dynamic logic DATL.

Automated Theorem Proving In Part II, we focus on the practical aspects of implementing the verification calculi from Part I. The calculi in Part I have already been designed for the practical needs of automated theorem proving, most notably so the free variable and Skolemisation techniques from Chapter 2 and the compositional calculi from Part I. Immediate implementations of the calculi from Part I in automated theorem provers can directly prove examples of medium complexity. Yet, more complex case studies still require additional algorithmic techniques for achieving very high degree automation and good scalability properties. In Chapter 5, we refine the calculi from Part I to tableau procedures and present proof strategies that navigate among their nondeterminisms to overcome the complexity issues of integrating real quantifier elimination as a decision procedure for real arithmetic.

In Chapter 6, which results from joint work with Edmund M. Clarke [PC08a], we refine the differential induction techniques from Chapter 3 to a fully automatic verification algorithm for computing the required discrete and differential invariants of a hybrid system locally in a logic-based fixedpoint loop.

Applications In Part III, we shift our attention to application scenarios for our logic-based verification approach for hybrid systems. Extending smaller hybrid systems which have served as running examples throughout this thesis, we show full case studies of the European Train Control System in Chapter 7, which is based on joint work with Jan-David Quesel [PQ08b], and for aircraft collision avoidance maneuvers in Chapter 8, which is based on joint work with Edmund M. Clarke [PC07, PC08a, PC08b].

Finally, Chapter 9 concludes this thesis with a discussion of the results and perspectives for future research.

Appendix In Appendix A, we briefly characterise the verification tool KeYmaera that implements the logics and automated theorem proving techniques presented in

this thesis and has been implemented in joint work with Jan-David Quesel [PQ08a]. We also survey various techniques that can be used to handle real arithmetic, including a description of techniques for using Gröbner basis computations to handle nonlinear real arithmetic.

In Appendix B, we formally analyse the close relationship of hybrid automata and hybrid programs by embedding hybrid automata into hybrid programs. For reference, Appendix C summarises some classical results about differential equations that we need in this thesis. In the interest of a comprehensive presentation, theorems that we reference from the book by Walter [Wal98] in this thesis can be found in Appendix C.

Sources This thesis is based on several sources. Chapter 2 is based on an article in the Journal of Automated Reasoning [Pla08b] and also covers most of the material from other sources of previous work at TABLEAUX [Pla07b, Pla07c] and HSCC [Pla07d]. Chapter 3 is an extended version of an article in the Journal of Logic and Computation [Pla08a], where we now add a relative completeness argument and prove that DAL is a conservative extension of the sublogic $\mathsf{d}\mathcal{L}$. We further combine the solution based techniques from Chapter 2 with differential induction based techniques from Chapter 3 by introducing the new extension of differential monotonicity relaxations. Chapter 4 is a substantially extended version of a previous paper at LFCS [Pla07e, Pla07f], where we now add a complete and more elegant calculus and provide a modular relative completeness proof.

In Chapter 5, we extend a previous paper at VERIFY [Pla07a] with more detail on iterative background closure strategies, including experimental evaluation, and complement this proof technique with a new iterative inflation strategy. Chapter 6 is based on joint work with Edmund M. Clarke at CAV [PC08a, PC08b].

Chapter 7 is a substantially revised and improved version of joint work with Jan-David Quesel at HSCC [PQ08b]. Chapter 8 is a significantly improved and detailed case study developed on the basis of joint work with Edmund M. Clarke at HSCC [PC07] and CAV [PC08b, PC08a].

Finally, Appendix A uses excerpts from joint work with Jan-David Quesel at IJCAR [PQ08a], adding thorough descriptions of computational backends and overall discussion of the KeYmaera verification tool. Appendix B is a substantially detailed version of a proof sketch in previous work [Pla07c]. Appendix C summarises classical results from the theory of differential equations [Wal98].

Further Material For extensions of the logic $\mathsf{d}\mathcal{L}$ with nominals we refer to our work at HyLo [Pla07g], where we introduce state-based reasoning as a paradigm for delaying expansion of transitions using nominals as symbolic state labels. For numerical approximation-refinement model checking techniques for hybrid systems, general (un)decidability results for numerical hybrid systems image computation,

and probabilistic verification techniques for hybrid systems, we refer to joint work with Edmund M. Clarke at HSCC [PC07].

For an overall embedding of this work in the context of the AVACS project, with a special emphasis on the overall verification flow within AVACS, we refer to joint work with Werner Damm, Alfred Mikschl, Jens Oehlerking, Ernst-Rüdiger Olderog, Jun Pang, Marc Segelken, and Boris Wirtz [DMO+07].

Due to their different focus, some other material is also not covered in this thesis, including joint work with Bernhard Beckert [BP06] on dynamic logic for object-oriented programming languages and with Stephanie Kemper [KP07] on SAT-based abstraction-refinement model checking for real-time systems using Craig interpolants [Cra57].

Part I.

Logics and Proof Calculi for Hybrid Systems

Chapter 2.

Differential Dynamic Logic d$\mathcal{L}$

Contents

Synopsis

Hybrid systems are models for complex physical systems and are defined as dynamical systems with interacting discrete transitions and continuous evolutions along differential equations. With the goal of developing a theoretical and practical foundation for deductive verification of hybrid systems, we introduce a dynamic logic for hybrid programs, which is a program notation for hybrid systems. As a verification technique that is suitable for automation, we introduce a free variable proof calculus with a novel combination of real-valued free variables and Skolemisation for lifting quantifier elimination for real arithmetic to dynamic logic. The calculus is compositional, i.e., it reduces properties of hybrid programs to properties of their parts. Our main result proves that this calculus axiomatises the transition behaviour of hybrid systems completely relative to differential equations. In a case study with cooperating traffic agents of the European Train Control System, we further show that our calculus is well-suited for verifying realistic hybrid systems with parametric system dynamics.

2.1. Introduction

In this chapter, we introduce the differential dynamic logic dℒ, its syntax, semantics, and proof calculus. It forms the core of this thesis and is the basis for the extensions and applications in subsequent chapters of this thesis. As the most fundamental theoretical result in this work, we prove that our calculus is a complete axiomatisation of hybrid systems reachability relative to differential equations.

Contributions

Our main conceptual contribution is the differential dynamic logic dℒ for hybrid programs, which captures the logical quintessence of the dynamics of hybrid systems succinctly. Our main practical contribution is a concise free variable calculus for dℒ that axiomatises the transition behaviour of hybrid systems relative to differential equation solving. It is suitable for automated theorem proving and for verifying hybrid interacting discrete and continuous dynamics compositionally. Our main theoretical contribution is that we prove the dℒ calculus to be complete relative

to the handling of differential equations. To the best of our knowledge, this is the first relative completeness proof for a logic of hybrid systems, and even the first formal notion of hybrid completeness. Our results fully align hybrid and continuous reasoning proof-theoretically and show that hybrid systems with interacting repetitive discrete and continuous evolutions can be verified whenever differential equations can. As an applied contribution, we further demonstrate that our logic and calculus can be used successfully for verifying collision avoidance in realistic train control applications.

Structure of this Chapter

After introducing syntax and semantics of the differential dynamic logic $\mathsf{d}\mathcal{L}$ in Section 2.2 and Section 2.3, we introduce a free variable sequent calculus for $\mathsf{d}\mathcal{L}$ in Section 2.5 and prove soundness and relative completeness in Section 2.6 and Section 2.7, respectively. We present relatively semidecidable fragments of $\mathsf{d}\mathcal{L}$ in Section 2.8 In Section 2.9, we use our calculus to prove an inductive safety property of the train control system that we present in Section 2.4. We draw conclusions and discuss future work in Section 2.10.

2.2. Syntax of Differential Dynamic Logic

In this section, we introduce the differential dynamic logic $\mathsf{d}\mathcal{L}$ in which operational models of hybrid systems are internalised as first-class citizens, so that correctness statements about the transition behaviour of hybrid systems can be expressed as formulas. As a basis, $\mathsf{d}\mathcal{L}$ includes (nonlinear) real arithmetic for describing concepts like safe regions of the state space. Further, $\mathsf{d}\mathcal{L}$ supports real-valued quantifiers for quantifying over the possible values of system parameters or durations of continuous evolutions. For talking about the transition behaviour of hybrid systems, $\mathsf{d}\mathcal{L}$ provides modal operators like $[\alpha]$ or $\langle\alpha\rangle$ that refer to the states reachable by following the transitions of hybrid system α.

The logic $\mathsf{d}\mathcal{L}$ is a first-order dynamic logic over the reals for hybrid programs, which is a compositional program notation for hybrid systems. Hybrid programs provide:

Discrete jump sets Discrete transitions are represented as instantaneous assignments of values to state variables, which are, essentially, difference equations. They can express resets $a := -b$ or adjustments of control variables like $a := a + 5$, as occurring in the discrete transformations attached to edges in hybrid automata, see Figure 1.3. Likewise, implicit discrete state changes like the changing of evolution modes from one node of an automaton to the other can be expressed uniformly

as, e.g., $q := brake$, where variable q remembers the current node. To handle simultaneous changes of multiple variables, discrete jumps can be combined to sets of jumps with simultaneous effect following corresponding techniques in the discrete case [BP06]. For instance, the discrete jump set $a := a + 5, A := 2a^2$ expresses that a is increased by 5 and, simultaneously, variable A is set to $2a^2$, which is evaluated *before* a receives its new value.

Differential equation systems Continuous variation in system dynamics is represented using differential equation systems as evolution constraints. For example the differential equation $z'' = -b$ describes deceleration and $z' = v, v' = -b \,\&\, v \geq 0$ expresses that the evolution only applies as long as the speed is $v \geq 0$, which represents mode *brake* of Figure 1.3. This is an evolution along the differential equation system $z' = v, v' = -b$ that is restricted to remain within the region $v \geq 0$, i.e., to stop braking before $v < 0$. Such an evolution can stop at any time within $v \geq 0$, it could even continue with transient grazing along the border $v = 0$, but it is never allowed to enter $v < 0$.

Control structure Discrete and continuous transitions—represented as difference or differential equations, respectively—can be combined to form a hybrid program with interacting hybrid dynamics using regular expression operators $(\cup, ^*, ;)$ of regular programs [HKT00] as control structure. For example, $q := accel \cup z'' = -b$ describes a train controller that can either choose to switch to acceleration mode or brake by the differential equation $z'' = -b$, by a nondeterministic choice $(\cup)$. In conjunction with other regular combinations, control constraints can be expressed using tests like $?z \geq s$ as guards for the system state.

2.1 Example (Embedding hybrid automata). With these operations, hybrid systems can be represented naturally as hybrid programs. For example, Figure 2.1 depicts a hybrid program rendition of the hybrid automaton in Figure 1.3. We represent each discrete and continuous transition of the automaton as a sequence of statements with a nondeterministic choice between these transitions. Line 4 represents a continuous transition. It tests if the current node q is *brake*, and then follows a differential equation system restricted to the invariant region $v \geq 0$. Line 3 characterises a discrete transition of the automaton. It tests the guard $z \geq s$ when in node *accel*, resets $a := -b$, and then switches q to node *brake*. By the semantics of hybrid automata [ACH+95, Hen96], an automaton in node *accel* is only allowed to make a transition to node *brake* if the invariant of *brake* is true when entering the node, which is expressed by the additional test $?v \geq 0$. In order to obtain a fully compositional model, hybrid programs make these implicit side-conditions explicit. Finally, the *-operator at the end of Figure 2.1 expresses that the transitions of a hybrid automaton can repeat indefinitely.

$$
\begin{array}{l}
q := \mathit{accel}; \qquad /^{*}\ \mathit{initial\ mode\ is\ node\ accel}\ ^{*}/ \\
\Big(\quad (?q = \mathit{accel};\ \ z' = v, v' = a) \\
\cup\ (?q = \mathit{accel} \wedge z \geq s;\ \ a := -b;\ \ q := \mathit{brake};\ \ ?v \geq 0) \\
\cup\ (?q = \mathit{brake};\ \ z' = v, v' = a\,\&\,v \geq 0) \\
\cup\ (?q = \mathit{brake} \wedge v \leq 1;\ \ a := a + 5;\ \ q := \mathit{accel})\Big)^{*}
\end{array}
$$

Figure 2.1.: Hybrid program rendition of hybrid automaton for (overly) simplified train control

2.2.1. Terms

The formulas of $\mathsf{d}\mathcal{L}$ are built over a set V of real-valued logical variables and a (finite) signature Σ of real-valued function and predicate *symbols*, with the usual function and predicate symbols for real arithmetic, such as $0, 1, +, -, \cdot, /, =, \leq, <, \geq, >$. State variables of hybrid systems, like, e.g., positions and velocities, are represented as real-valued constant symbols of Σ, i.e., function symbols of arity 0. Unlike fixed symbols like 1, state variables are *flexible*, i.e., their interpretation can change from state to state during the execution of a hybrid program. Flexibility of symbols will be used to represent the progression of system values along states over time during a hybrid evolution. Symbols like 1, instead, are *rigid*, i.e., they have the same value at all states.

There is no need to distinguish between discrete and continuous variables in $\mathsf{d}\mathcal{L}$. The distinction between logical variables in V, which can be quantified, and state variables in Σ, which can change their value by discrete jumps and differential equations in modalities, is not strictly required. For instance, quantification of state variables is definable using auxiliary logical variables. The distinction makes the semantics less subtle, though. Our calculus assumes that V contains sufficiently many variables and Σ contains additional Skolem function symbols, which are reserved for use by the calculus.

The set $\mathrm{Trm}(\Sigma, V)$ of *terms* is defined as in classical first-order logic yielding polynomial (or rational) expressions over V and over additional Skolem terms $s(t_1, \ldots, t_n)$ with terms t_i. Our calculus only uses Skolem terms $s(X_1, \ldots, X_n)$ with logical variables $X_i \in V$.

2.1 Definition (Terms). $\mathrm{Trm}(\Sigma, V)$ is the set of all *terms*, which is the smallest set such that:

- If $x \in V$, then $x \in \mathrm{Trm}(\Sigma, V)$.
- If $f \in \Sigma$ is a function symbol of arity $n \geq 0$ and, for $1 \leq i \leq n$, $\theta_i \in \mathrm{Trm}(\Sigma, V)$, then $f(\theta_1, \ldots, \theta_n) \in \mathrm{Trm}(\Sigma, V)$. The case $n = 0$ is permitted.

The set of formulas of *first-order logic* is defined as usual, giving first-order real arithmetic [Tar51] augmented with Skolem terms. We will show the precise relation-

ship to standard first-order real arithmetic without Skolem terms in Lemma 2.13 of Section 2.5.2.

2.2 Definition (First-order formulas). The set $\mathrm{Fml}_{\mathrm{FOL}}(\Sigma, V)$ of *formulas of first-order logic* is the smallest set with:

- If $p \in \Sigma$ is a predicate symbol of arity $n \geq 0$ and $\theta_i \in \mathrm{Trm}(\Sigma, V)$ for $1 \leq i \leq n$, then $p(\theta_1, \ldots, \theta_n) \in \mathrm{Fml}_{\mathrm{FOL}}(\Sigma, V)$.
- If $\phi, \psi \in \mathrm{Fml}_{\mathrm{FOL}}(\Sigma, V)$, then $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi) \in \mathrm{Fml}_{\mathrm{FOL}}(\Sigma, V)$.
- If $\phi \in \mathrm{Fml}_{\mathrm{FOL}}(\Sigma, V)$ and $x \in V$, then $\forall x\, \phi, \exists x\, \phi \in \mathrm{Fml}(\Sigma, V)$.

2.2.2. Hybrid Programs

As uniform compositional models for hybrid systems, discrete and continuous transitions can be combined by structured control programs.

2.3 Definition (Hybrid programs). The set $\mathrm{HP}(\Sigma, V)$ of *hybrid programs*, with typical elements α, β, is defined inductively as the smallest set such that

1. If $x_i \in \Sigma$ is a state variable and $\theta_i \in \mathrm{Trm}(\Sigma, V)$ for $1 \leq i \leq n$, then the *discrete jump set* $(x_1 := \theta_1, \ldots, x_n := \theta_n) \in \mathrm{HP}(\Sigma, V)$ is a hybrid program.
2. If $x_i \in \Sigma$ is a state variable and $\theta_i \in \mathrm{Trm}(\Sigma, V)$ for $1 \leq i \leq n$, then, $x_i' = \theta_i$ is a *differential equation* in which x_i' represents the time-derivative of variable x_i. If χ is a first-order formula, then $(x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi) \in \mathrm{HP}(\Sigma, V)$.
3. If χ is a first-order formula, then $(?\chi) \in \mathrm{HP}(\Sigma, V)$.
4. If $\alpha, \beta \in \mathrm{HP}(\Sigma, V)$, then $(\alpha \cup \beta) \in \mathrm{HP}(\Sigma, V)$.
5. If $\alpha, \beta \in \mathrm{HP}(\Sigma, V)$, then $(\alpha; \beta) \in \mathrm{HP}(\Sigma, V)$.
6. If $\alpha \in \mathrm{HP}(\Sigma, V)$, then $(\alpha^*) \in \mathrm{HP}(\Sigma, V)$.

The effect of jump set $x_1 := \theta_1, \ldots, x_n := \theta_n$ is to simultaneously change the interpretations of the x_i to the respective θ_i by performing a discrete jump in the state space. In particular, the θ_i are evaluated before changing the value of any x_j. The effect of $x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi$ is an ongoing continuous evolution respecting the differential equation system $x_1' = \theta_1, \ldots, x_n' = \theta_n$ while remaining within the region χ. The evolution is allowed to stop at any point in χ. It is, however, required to stop before it leaves χ. For unconstrained evolutions, we write $x' = \theta$ in place of $x' = \theta \,\&\, \mathit{true}$. For structural reasons, we expect both difference equations (discrete jump sets) and differential equations to be given in explicit form, i.e., with the affected variable on the left. The dℒ semantics allows

arbitrary differential equations. To retain feasible arithmetic, some of our calculus rules assume that, like in [ACH+95, Frä99, MPM05, Hen96], the differential equations have first-order definable flows or approximations. We assume that standard techniques are used to determine corresponding solutions or approximations, e.g., [AW01, LPY99, PC07, Per91, Wal98].

The test action $?\chi$ is used to define conditions. Its semantics is that of a no-op if χ is true in the current state; otherwise, like *abort*, it allows no transitions. Note that, according to Definition 2.3, we only allow first-order formulas as tests. Instead, we could allow *rich tests*, i.e., arbitrary $\mathsf{d}\mathcal{L}$ formulas χ with nested modalities as tests $?\chi$ inside hybrid programs (and even in invariant regions χ of differential equations). The calculus and our meta-results directly carry over to rich test $\mathsf{d}\mathcal{L}$. To simplify the presentation, however, we refrain from allowing arbitrary $\mathsf{d}\mathcal{L}$ formulas as tests, because that requires simultaneous inductive handling of hybrid programs and $\mathsf{d}\mathcal{L}$ formulas in syntax, semantics, and completeness proofs, because $\mathsf{d}\mathcal{L}$ formulas would then be allowed to occur in hybrid programs and vice versa.

The non-deterministic choice $\alpha \cup \beta$, sequential composition $\alpha;\beta$, and non-deterministic repetition α^* of programs are as usual but generalised to a semantics in hybrid systems. Choices $\alpha \cup \beta$ are used to express behavioural alternatives between the transitions of α and β. The sequential composition $\alpha;\beta$ says that the hybrid program β starts executing after α has finished (β never starts if α does not terminate). Observe that, like repetitions, continuous evolutions within α can take longer or shorter, which already causes uncountable nondeterminism. This nondeterminism is inherent in hybrid systems and as such reflected in hybrid programs. Repetition α^* is used to express that the hybrid process α repeats any number of times, including zero times. The control flow operations of choice, sequential composition, and repetition can be combined with $?\chi$ to form all other control structures [HKT00]. For instance, $(?\chi;\alpha)^*;?\neg\chi$ corresponds to a while loop that repeats α while χ holds and only stops when χ ceases to hold.

Hybrid programs are designed as a minimal extension of conventional discrete programs. They characterise hybrid systems succinctly by adding continuous evolution along differential equations as the only additional primitive operation to a regular basis of conventional discrete programs. To yield hybrid systems, their operations are interpreted over the domain of real numbers. This gives rise to an elegant syntactic hierarchy of discrete, continuous, and hybrid systems. Hybrid automata [Hen96] can be represented as hybrid programs using a straightforward generalisation of standard program encodings of automata, see Appendix B for formal details. The fragment of hybrid programs without differential equations corresponds to conventional discrete programs generalised over the reals or to discrete-time dynamical systems [Bra95b]. The fragment without discrete jumps corresponds to switched continuous systems [Bra95b, BBM98], whereas the fragment of differential equations gives purely continuous dynamical systems [Sib75]. Only the composition of mixed discrete jumps and continuous evolutions gives rise

to truly hybrid behaviour.

2.2.3. Formulas of Differential Dynamic Logic

The formulas of $\mathsf{d}\mathcal{L}$ are defined as in first-order dynamic logic [HKT00]. That is, they are built using propositional connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and quantifiers $\forall, \exists$ (first-order part). In addition, if ϕ is a $\mathsf{d}\mathcal{L}$ formula and α a hybrid program, then $[\alpha]\phi, \langle\alpha\rangle\phi$ are formulas (dynamic part).

2.4 Definition ($\mathsf{d}\mathcal{L}$ formulas). The set $\mathrm{Fml}(\Sigma, V)$ of *formulas* of $\mathsf{d}\mathcal{L}$, with typical elements ϕ, ψ, is the smallest set such that

1. If p is a predicate symbol of arity $n \geq 0$ and $\theta_i \in \mathrm{Trm}(\Sigma, V)$ for $1 \leq i \leq n$, then $p(\theta_1, \ldots, \theta_n) \in \mathrm{Fml}(\Sigma, V)$.
2. If $\phi, \psi \in \mathrm{Fml}(\Sigma, V)$, then $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi) \in \mathrm{Fml}(\Sigma, V)$.
3. If $\phi \in \mathrm{Fml}(\Sigma, V)$ and $x \in V$, then $\forall x\, \phi, \exists x\, \phi \in \mathrm{Fml}(\Sigma, V)$.
4. If $\phi \in \mathrm{Fml}(\Sigma, V)$ and $\alpha \in \mathrm{HP}(\Sigma, V)$, then $[\alpha]\phi, \langle\alpha\rangle\phi \in \mathrm{Fml}(\Sigma, V)$.

We consider $\phi \leftrightarrow \psi$ as an abbreviation for $(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$ to simplify the calculus. When *train* denotes the hybrid program in Figure 2.1, the following $\mathsf{d}\mathcal{L}$ formula states that the train is able to leave region $z < m$ when it starts in the same region:

$$z < m \rightarrow \langle \mathit{train} \rangle z \geq m \ .$$

Note that, according to Definition 2.4, hybrid programs are not additional external objects but fully internalised [Bla00] as first-class citizens within the logic $\mathsf{d}\mathcal{L}$ itself and the logic is closed. That is, modalities can be combined propositionally, by quantifiers, or nested. For instance, $[\alpha]\langle\beta\rangle x \leq c$ says that, whatever α is doing, β can react in some way to reach a controlled state where x is less than some critical value c. Dually, $\langle\beta\rangle[\alpha]x \leq c$ expresses that β can stabilise $x \leq c$, i.e., behave in such a way that $x \leq c$ remains true no matter how α reacts. Accordingly, $\exists p\, [\alpha]x \leq c$ says that there is a choice of parameter p such that α remains in $x \leq c$.

During our analysis, we assume differential equations and discrete transitions to be well-defined. In particular, we assume that all divisions p/q are guarded by conditions that ensure $q \neq 0$ as, otherwise, the system behaviour is not well-defined due to an undefined value at a singularity. It is simple but tedious to augment the semantics and the calculus with corresponding side conditions to show that this is respected. For instance, we assume that $x := p/q$ is guarded by $?q \neq 0$ and that continuous evolutions are restricted such that the differential equations are well-defined as, e.g., $x' = p/q \,\&\, q \neq 0$. Also see [BP06] for techniques how such exceptional behaviour can be handled by program transformation while avoiding partial valuations in the semantics. In logical formulas, partiality can be avoided by writing, e.g., $p = c \cdot q \wedge q \neq 0$ rather than $p/q = c$.

2.3. Semantics of Differential Dynamic Logic

We define the semantics of $\mathsf{d}\mathcal{L}$ as a Kripke semantics with worlds representing the possible system states and with reachability along the hybrid transitions of the system as accessibility relation. The interpretations of $\mathsf{d}\mathcal{L}$ consist of states (worlds) that are essentially first-order structures over the reals. In particular, real values are assigned to state variables, possibly different values in each state. A potential behaviour of a hybrid system corresponds to a succession of states that contain the observable values of system variables during its hybrid evolution.

2.3.1. Valuation of Terms

An *interpretation* I assigns functions and relations over the reals to the respective (rigid) symbols in Σ. The function and predicate symbols of real arithmetic are interpreted as usual by I. A *state* is a map $\nu : \Sigma_{fl} \to \mathbb{R}$; the set of all states is denoted by $\mathrm{Sta}(\Sigma)$. Here, Σ_{fl} denotes the set of (flexible) state variables in Σ (they have arity 0). Finally, an *assignment for logical variables* is a map $\eta : V \to \mathbb{R}$. It contains the values for logical variables, which are not subject to change by modalities but only by quantification. Observe that flexible symbols (which represent state variables), are allowed to assume different interpretations in different states. Logical variable symbols, however, are rigid in the sense that their value is determined by η alone and does not depend on the state.

The *valuation* $val_{I,\eta}(\nu, \cdot)$ of terms is defined as usual [FM99, HKT00] with a distinction of rigid and flexible functions [BP06].

2.5 Definition (Valuation of terms). The *valuation of terms* with respect to interpretation I, assignment η, and state ν is defined by

1. $val_{I,\eta}(\nu, x) = \eta(x)$ if $x \in V$ is a logical variable.
2. $val_{I,\eta}(\nu, a) = \nu(a)$ if $a \in \Sigma$ is a state variable (flexible function symbol of arity 0).
3. $val_{I,\eta}(\nu, f(\theta_1, \ldots, \theta_n)) = I(f)\big(val_{I,\eta}(\nu, \theta_1), \ldots, val_{I,\eta}(\nu, \theta_n)\big)$ when $f \in \Sigma$ is a rigid function symbol of arity $n \geq 0$.

2.3.2. Valuation of Formulas

The *valuation* $val_{I,\eta}(\nu, \cdot)$ of formulas is defined as usual for first-order modal logic [FM99, HKT00] with a distinction of rigid and flexible functions [BP06]. Modalities parameterised by a hybrid program α follow the accessibility relation spanned by the respective hybrid state transition relation $\rho_{I,\eta}(\alpha)$, which is simultaneously inductively defined in Definition 2.7.

We will use $\nu[x \mapsto d]$ to denote the *modification* of a state ν that agrees with ν except for the interpretation of the symbol $x \in \Sigma_{fl}$, which is changed to $d \in \mathbb{R}$. Similarly, $\eta[x \mapsto d]$ agrees with the assignment η except on $x \in V$, which is assigned $d \in \mathbb{R}$.

2.6 Definition (Valuation of dℒ formulas). The *valuation, $val_{I,\eta}(\nu, \cdot)$, of formulas* with respect to interpretation I, assignment η, and state ν is defined as

1. $val_{I,\eta}(\nu, p(\theta_1, \ldots, \theta_n)) = I(p)\big(val_{I,\eta}(\nu, \theta_1), \ldots, val_{I,\eta}(\nu, \theta_n)\big)$
2. $val_{I,\eta}(\nu, \phi \wedge \psi) = \mathit{true}$ iff $val_{I,\eta}(\nu, \phi) = \mathit{true}$ and $val_{I,\eta}(\nu, \psi) = \mathit{true}$
3. $val_{I,\eta}(\nu, \phi \vee \psi) = \mathit{true}$ iff $val_{I,\eta}(\nu, \phi) = \mathit{true}$ or $val_{I,\eta}(\nu, \psi) = \mathit{true}$
4. $val_{I,\eta}(\nu, \neg\phi) = \mathit{true}$ iff $val_{I,\eta}(\nu, \phi) \neq \mathit{true}$
5. $val_{I,\eta}(\nu, \phi \rightarrow \psi) = \mathit{true}$ iff $val_{I,\eta}(\nu, \phi) \neq \mathit{true}$ or $val_{I,\eta}(\nu, \psi) = \mathit{true}$
6. $val_{I,\eta}(\nu, \forall x\, \phi) = \mathit{true}$ iff $val_{I,\eta[x\mapsto d]}(\nu, \phi) = \mathit{true}$ for all $d \in \mathbb{R}$
7. $val_{I,\eta}(\nu, \exists x\, \phi) = \mathit{true}$ iff $val_{I,\eta[x\mapsto d]}(\nu, \phi) = \mathit{true}$ for some $d \in \mathbb{R}$
8. $val_{I,\eta}(\nu, [\alpha]\phi) = \mathit{true}$ iff $val_{I,\eta}(\omega, \phi) = \mathit{true}$ for all states ω for which the transition relation satisfies $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$
9. $val_{I,\eta}(\nu, \langle\alpha\rangle\phi) = \mathit{true}$ iff $val_{I,\eta}(\omega, \phi) = \mathit{true}$ for some state ω for which the transition relation satisfies $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$

Following the usual notation, we also write $I, \eta, \nu \models \phi$ iff $val_{I,\eta}(\nu, \phi) = \mathit{true}$. Dually, we write $I, \eta, \nu \not\models \phi$ iff $val_{I,\eta}(\nu, \phi) \neq \mathit{true}$. Occasionally, we write just $\models \phi$ iff $I, \eta, \nu \models \phi$ holds for all I, η, ν.

2.3.3. Transition Semantics of Hybrid Programs

Now we can define the transition semantics, $\rho_{I,\eta}(\alpha)$, of a hybrid program α. The semantics of a hybrid program is captured by its hybrid state transition relation. For discrete jumps this transition relation holds for pairs of states that respect the discrete jump set. For continuous evolutions, the transition relation holds for pairs of states that can be interconnected by a continuous flow respecting the differential equations and invariant throughout the evolution.

2.7 Definition (Transition semantics of hybrid programs). The *valuation, $\rho_{I,\eta}(\alpha)$, of a hybrid program α*, is a *transition relation* on states. It specifies which state ω is reachable from a state ν by operations of the hybrid program α and is defined as follows

1. $(\nu, \omega) \in \rho_{I,\eta}(x_1 := \theta_1, .., x_n := \theta_n)$ iff $\nu[x_1 \mapsto val_{I,\eta}(\nu, \theta_1)] .. [x_n \mapsto val_{I,\eta}(\nu, \theta_n)]$ equals state ω. Particularly, the value of other variables $z \notin \{x_1, \ldots, x_n\}$ remains constant, i.e., $val_{I,\eta}(\nu, z) = val_{I,\eta}(\omega, z)$.

2. $(\nu, \omega) \in \rho_{I,\eta}(x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi)$ iff there is a *flow* f of some duration $r \geq 0$ from state ν to state ω along $x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi$, i.e., a function $f : [0, r] \to \mathrm{Sta}(\Sigma)$ such that:
   - $f(0) = \nu, f(r) = \omega$;
   - f respects the differential equations: For each x_i, $val_{I,\eta}(f(\zeta), x_i)$ is continuous in ζ on $[0, r]$ and has a derivative of value $val_{I,\eta}(f(\zeta), \theta_i)$ at each time $\zeta \in (0, r)$;
   - The value of other variables $z \notin \{x_1, \ldots, x_n\}$ remains constant, that is, $val_{I,\eta}(f(\zeta), z) = val_{I,\eta}(\nu, z)$ for all $\zeta \in [0, r]$;
   - And f respects the invariant: $val_{I,\eta}(f(\zeta), \chi) = \mathit{true}$ for each $\zeta \in [0, r]$.

3. $\rho_{I,\eta}(?\chi) = \{(\nu, \nu) \;:\; val_{I,\eta}(\nu, \chi) = \mathit{true}\}$

4. $\rho_{I,\eta}(\alpha \cup \beta) = \rho_{I,\eta}(\alpha) \cup \rho_{I,\eta}(\beta)$

5. $\rho_{I,\eta}(\alpha; \beta) = \{(\nu, \omega) : (\nu, z) \in \rho_{I,\eta}(\alpha), (z, \omega) \in \rho_{I,\eta}(\beta) \text{ for a state } z\}$

6. $(\nu, \omega) \in \rho_{I,\eta}(\alpha^*)$ iff there are an $n \in \mathbb{N}$ and states $\nu = \nu_0, \ldots, \nu_n = \omega$ such that $(\nu_i, \nu_{i+1}) \in \rho_{I,\eta}(\alpha)$ for all $0 \leq i < n$.

Note that the modifications of a discrete jump set are executed simultaneously in the sense that all terms θ_i are evaluated in the initial state ν. For simplicity, we assume the x_i to be different, and refer to previous work [BP06] for a compatible semantics and calculus handling concurrent modifications of the same x_i.

For differential equations like $x' = \theta$, Definition 2.7 characterises transitions along a continuous evolution respecting the differential equation, see Figure 2.2a. A continuous transition along $x' = \theta$ is possible from ν to ω whenever there is a continuous flow f of some duration $r \geq 0$ connecting state ν with ω such that f gives a solution of the differential equation $x' = \theta$. That is, its value is continuous on $[0, r]$ and differentiable with the value of θ as derivative on the open interval $(0, r)$. Further, only variables subject to a differential equation change during such a continuous transition. Similarly, the continuous transitions of $x' = \theta \,\&\, \chi$ with invariant region χ are those where f always resides within χ during the whole evolution, see Figure 2.2b.

For the semantics of differential equations, derivatives are well-defined on the open interval $(0, r)$ as $\mathrm{Sta}(\Sigma)$ is isomorphic to some finite-dimensional real space spanned by the variables of the differential equations (derivatives are not defined on the closed interval $[0, r]$ if $r = 0$). For the purpose of a differential equation system,

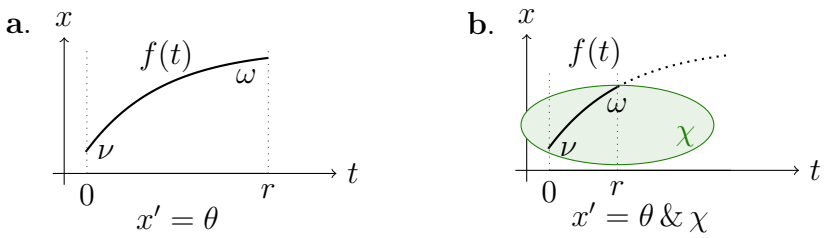


Figure 2.2.: Continuous flow along differential equation $x' = \theta$ over time t

states are fully determined by an assignment of a real value to each occurring variable, which are finitely many. Furthermore, the terms of dℒ are continuously differentiable on the open domain where divisors are non-zero, because the zero set of divisors is closed. Hence, solutions in dℒ are unique:

2.8 Lemma (Uniqueness). *Differential equations of* dℒ *have unique solutions, i.e., for each differential equation system, each state* ν *and each duration* $r \geq 0$*, there is at most one flow* $f : [0, r] \to \mathrm{Sta}(\Sigma)$ *satisfying the conditions of Case 2 of Definition 2.7.*

Proof. Let $x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi$ be a differential equation system with invariant region χ. Using simple computations in the field of rational fractions, we can assume the right-hand sides θ_i of the differential equations to be of the form p_i/q_i for polynomials p_i, q_i. The set of points in real space where $q_i = 0$ holds is closed. As a finite union of closed sets, the set where $q_1 = 0 \vee \cdots \vee q_n = 0$ holds is closed. Hence, the valuations of the θ_i are continuously differentiable on the complement of the latter set, which is open. Thus, as a consequence of Picard-Lindelöf's theorem a.k.a. Cauchy-Lipschitz theorem [Wal98, Theorem 10.VI], the solutions are unique on each connected component of this open domain. Consequently, solutions are unique when restricted to χ, which, by assumption, entails $q_1 \neq 0 \wedge \cdots \wedge q_n \neq 0$. □

For control-feedback loops α with a discrete controller regulating a continuous plant, transition structures involve all safety-critical states, hence, $\psi \to [\alpha]\phi$ is a natural rendition of the safety property that ϕ holds at all states reachable by α from initial states that satisfy ψ. Otherwise, dℒ can be augmented with temporal operators to refer to intermediate states or nonterminating traces. The corresponding calculus is compatible and reduces temporal properties to non-temporal properties at intermediate states of the hybrid program, as we illustrate in Chapter 4.

2.4. Collision Avoidance in Train Control

As a case study to illustrate how dℒ can be used for specifying and verifying hybrid systems, we examine a scenario of cooperating traffic agents in the *European*

Train Control System (ETCS) [DMO+07]. The purpose of ETCS is to ensure that trains cannot crash into other trains or pass open gates. Its secondary objective is to maximise throughput and velocity without endangering safety. To achieve these objectives, ETCS discards the static partitioning of the track into fixed segments of mutually exclusive and physically separated access by trains, which has been used traditionally. Instead, permission to move is granted dynamically by decentralised Radio Block Controllers (RBC) depending on the current track situation and movement of other traffic agents within the region of responsibility of the RBC, see Figure 2.3.

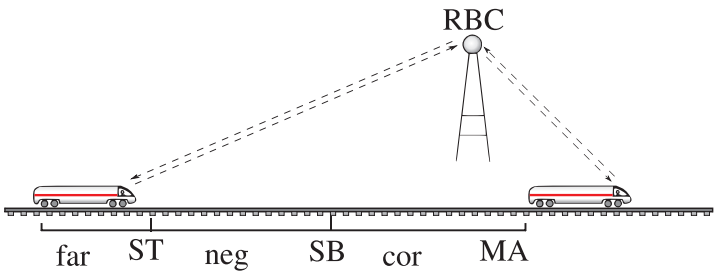


Figure 2.3.: ETCS train coordination protocol using dynamic movement authorities

Movement Authorities This moving block principle is achieved by dynamically giving a *movement authority* (MA) to each traffic agent, within which it is obliged to remain. Before a train moves into a part of the track for which it does not have MA, it asks the RBC for an MA-extension (negotiation phase *neg* of Figure 2.3). Depending on the MA that the RBC has currently given to other traffic agents or gates, the RBC will grant this extension and the train can move on. If the newly requested MA is still in possession of another train which could occupy the track, or if the MA is still consumed by an open gate, the RBC will deny the MA-extension such that the requesting train needs to reduce speed or start braking in order to safely remain within its old MA. As the negotiation process with the RBC can take time because of possibly unreliable wireless communication and negotiation of the RBC with other agents, the train initiates negotiation well before reaching the end of its MA. When the rear end of a train has safely left a part of a track, the train can give that part of its MA back to RBC control such that it can be used by other traffic agents.

In addition to increased flexibility and throughput of this moving block principle, the underlying technical concept of movement authorities can be exploited for verifying ETCS. It can be shown that a system of arbitrarily many trains, gates, and RBCs, which communicate in the aforementioned manner, safely avoids collisions if each traffic agent always resides within its MA under all circumstances, provided

that the RBCs grant MA mutually exclusive so that the MAs dynamically partition the track [DHO06]. This way, verification of a system of unboundedly many traffic agents can be reduced to an analysis of individual agents with respect to their specific MA.

Train Control Model For trains, speed supervision and automatic train protection are responsible for locally controlling the movement of a train such that it always respects its MA [DHO06]. Depending on the current driving situation, the train controller determines a point SB (for start braking) upto which driving is safe, and adjusts its acceleration a in accordance with SB. Before SB, speed can be regulated freely (to keep the desired speed and throughput of a track profile). Beyond SB (correcting phase *cor* in Figure 2.3), the train starts braking in order to make sure it remains within its MA if the RBC does not grant an extension in time.

We assume that an MA has been granted up to some track position, which we call m, and the train is located at position z, heading with initial speed v towards m. We represent the point SB as the safety distance s relative to the end m of the MA (i.e., $m - s = \text{SB}$). In this situation, dℒ can analyse the following crucial safety property of ETCS:

$$\begin{aligned} \psi &\rightarrow [(\mathit{ctrl}; \mathit{drive})^*]\, z \leq m \qquad (2.1)\\ \text{where } \mathit{ctrl} &\equiv (?m - z \leq s; a := -b) \cup (?m - z \geq s; a := A)\\ \mathit{drive} &\equiv \tau := 0; (z' = v, v' = a, \tau' = 1 \,\&\, v \geq 0 \wedge \tau \leq \varepsilon)\ . \end{aligned}$$

It expresses that a train always remains within its MA, assuming some constraint ψ for its parameters. The operational system model is a control-feedback loop of the digital controller *ctrl* and the plant *drive*. In *ctrl*, the train controller corrects its acceleration or brakes on the basis of the remaining distance $(m - z)$. As a failsafe recovery manoeuvre [DHO06], it applies brakes with force b if the remaining MA is less than s. Otherwise, speed is regulated freely. For simplicity, we assume the train uses a fixed acceleration A before having passed s. The verification is quite similar when the controller can dynamically choose any acceleration $a \leq A$ instead.

After acceleration a has been set in *ctrl*, the train continues moving in *drive*. There, the position z of the train evolves according to the system $z' = v, v' = a$ (i.e., $z'' = a$). The evolution in *drive* stops when the speed v drops below zero (or earlier). Simultaneously, clock τ measures the duration of the current *drive* phase before the controllers react to situation changes again. Clock τ is reset to zero when entering *drive*, constantly evolves along $\tau' = 1$, and is bound by the invariant region $\tau \leq \varepsilon$. The effect is that a *drive* phase is interrupted for reassessing the driving situation after at most ε seconds, and the *ctrl*; *drive* feedback loop repeats. The corresponding transition structure $\rho_{I,\eta}((\mathit{ctrl}; \mathit{drive})^*)$ is depicted in Figure 2.4a. Figure 2.4b shows possible runs of the train where speed regulation

successively decreases velocity v because MA has not been extended in time (Figure 2.4b shows 3 different runs which correspond to different choices of parameter s, where only the lowest velocity choice is safe). Finally, observe that the invariant region $v \geq 0 \wedge \tau \leq \varepsilon$ needs to be true at *all times* during continuous evolutions of *drive*, otherwise there is no corresponding transition in $\rho_{I,\eta}(drive)$. This not only restricts the maximum duration of *drive*, but also imposes a constraint on permitted initial states: The arithmetic constraint $v \geq 0$ expresses that the differential equation only applies for non-negative speed. Hence, like in a test $?v \geq 0$, program *drive* allows no transitions when v is initially less than 0. In that case, $\rho_{I,\eta}((ctrl; drive)^*)$ collapses to the trivial identity transition with zero repetitions.

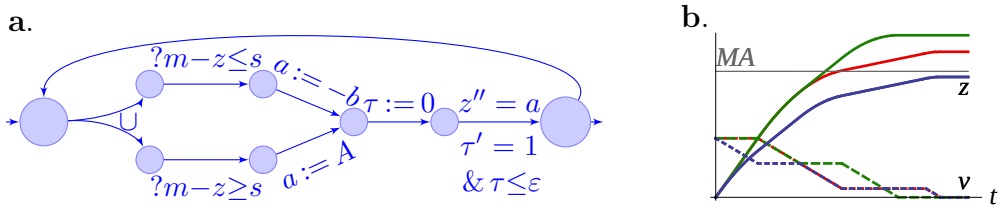


Figure 2.4.: ETCS transition structure and various choices of speed regulation for train speed control

Discussion Here, we explicitly take into account possibly delayed controller reactions to bridge the gap of continuous-time models and discrete-time control design. To get meaningful results, we need to assume a maximum reaction delay ε as safety cannot otherwise be guaranteed. Polling cycles of sensors and digital controllers as well as latencies of actuators like brakes contribute to ε. Instead of using specific estimates for ε for a particular train, we accept ε as a fully symbolic parameter. Further, instead of manually choosing specific values for the free parameters of (2.1) as in model checking approaches [DMO+07], we will use our calculus to synthesise constraints on the relationship of parameters that are required for a safe operation of train control. As they are of subordinate importance to the cooperation layer of train control [DHO06], we do not model weather conditions, slope of track, or train mass.

Because of its nonlinear behaviour and nontrivial reset relations, system (2.1) is beyond the modelling capabilities of linear hybrid automata [ACH+95, Hen96, Fre05] and beyond o-minimal automata [LPY99]. Previous approaches need linear flows [ACH+95, Hen96], do not support the coupled dynamics caused by nontrivial resets [LPY99], require polyhedral initial sets and discrete dynamics [CK03], only handle robust systems with bounded regions [Frä99], although parametric systems are not robust uniformly for all parameter choices, or they handle only bounded-time safety for systems with bounded switching [MPM05]. Finally, in addition

to general numerical limits [PC07], numerical approaches [CK03, ADG03] quickly become intractable due to the exponential impact of the number of variables.

2.5. Free Variable Calculus for Differential Dynamic Logic

In this section, we introduce a sequent calculus for verifying hybrid systems by proving corresponding dℒ formulas. The basic idea is to symbolically compute the effects of hybrid programs and successively transform them into logical formulas describing these effects by structural decomposition. The calculus consists of standard propositional rules, rules for dynamic modalities that are generalised to hybrid programs, and novel quantifier rules that integrate real quantifier elimination (or, in fact, any other quantifier elimination procedure) into the modal calculus using free variables and Skolemisation.

2.5.1. Rules of the Calculus for Differential Dynamic Logic

A *sequent* is of the form $\Gamma \vdash \Delta$, where the *antecedent* Γ and *succedent* Δ are finite sets of formulas. The semantics of $\Gamma \vdash \Delta$ is that of the formula $\bigwedge_{\phi\in\Gamma} \phi \rightarrow \bigvee_{\psi\in\Delta} \psi$. For quantifier elimination rules, we make use of this fact by considering sequent $\Gamma \vdash \Delta$ as an abbreviation for the latter formula.

The dℒ calculus uses substitutions that take effect within formulas and programs. The result of applying to ϕ the *substitution* that simultaneously replaces x_i by θ_i (for $1 \leq i \leq n$) is defined as usual; it is denoted by $\phi^{\theta_1}_{x_1}\ldots^{\theta_n}_{x_n}$. We assume α-conversion for renaming as needed. In the dℒ calculus, only admissible substitutions are applicable, which is crucial for soundness. Admissible substitutions are denotation-preserving: They ensure that symbols still denote the same values after a substitution when they did so before.

2.9 Definition (Admissible substitution). An application of a substitution σ is *admissible* if no variable x that σ replaces by $\sigma(x)$ occurs in the scope of a quantifier or modality binding x or a (logical or state) variable of the replacement $\sigma(x)$. A modality *binds* a state variable x iff it contains a discrete jump set assigning to x (like $x := \theta$) or a differential equation containing x' (like $x' = \theta$).

2.2 Example (Non-admissible substitution). For the following formula, ϕ,

$$x = z \rightarrow \langle z := z + 1\rangle(z \geq x + 1)$$

the substitution that replaces all occurrences of x by z, is not admissible. This is due to the fact that for the forming of ϕ^z_x as

$$z = z \rightarrow \langle z := z + 1\rangle(z \geq z + 1)$$

the substitution replaces x in postcondition $z \geq x+1$ by z, which is bound by modality $\langle z := z+1 \rangle$. Hence, within the scope of the modality, symbol z denotes a different value than outside the modality, thereby destroying the property of the occurrences of x, or—after the substitution—of z, to share the same value throughout the formula. Instead, a substitution of x by $y+1$ in ϕ to form ϕ_x^{y+1} is admissible for other symbols y.

When no confusion arises, we also use implicit notation for substitutions to improve readability. Let $\phi(z)$ be a formula with a free variable z. Then for any term θ, we use $\phi(\theta)$ as an abbreviation for the formula $\phi(z)_z^\theta$ that results from $\phi(z)$ by substituting θ for z.

Observe that, for soundness, the notion of bound variables can be *any* overapproximation of the set of variables that possibly change their value during a hybrid program. In vacuous identity changes like $x := x$ or $x' = 0$, variable x will not really change its value, but we still consider x as a bound variable for simplicity. For a hybrid program α, we denote by $\forall^\alpha \phi$ the *universal closure* of formula ϕ with respect to all state variables bound in α. Quantification over state variable x is definable as $\forall X\, [x := X]\Phi$ using an auxiliary logical variable X.

For handling quantifiers, we cannot use the standard rules [HS94, Fit96, FM99], because these are for uninterpreted first-order logic and (ultimately) work by instantiating quantifiers, either eagerly as in ground tableaux or lazily by unification as in free variable tableaux [HS94, Fit96, FM99]. The basis of $\mathsf{d}\mathcal{L}$, instead, is first-order logic interpreted over the reals or in the theory of real-closed fields [Tar51]. A formula like $\exists a\, \forall x\, (x^2 + a > 0)$ cannot be proven by instantiation-based quantifier rules but is valid in the theory of real-closed fields. Unfortunately, quantifier elimination (QE) over the reals [CH91, Tar51], which is the standard decision procedure for real arithmetic, cannot be applied to formulas with modalities either. Hence, we introduce novel quantifier rules that integrate quantifier elimination in a way that is compatible with dynamic modalities (as we illustrate in Section 2.5.2).

2.10 Definition (Quantifier elimination). A first-order theory admits *quantifier elimination* if, to each formula ϕ, a quantifier-free formula $\mathrm{QE}(\phi)$ can be associated effectively that is equivalent (i.e., $\phi \leftrightarrow \mathrm{QE}(\phi)$ is valid) and has no additional free variables or function symbols. The operation QE is further assumed to evaluate ground formulas (i.e., without variables), yielding a decision procedure for closed formulas of this theory.

As usual in sequent calculus rules—although the direction of entailment is from *premisses* (above rule bar) to *conclusion* (below)—the order of reasoning is *goal-directed*: Rules are applied in tableau-style, i.e., starting from the desired conclusion at the bottom (*goal*) to the resulting premisses (*sub-goals*). To highlight the logical essence of the $\mathsf{d}\mathcal{L}$ calculus, Figure 2.5 provides *rule schemata* to which the following definition associates the calculus rules that are applicable in $\mathsf{d}\mathcal{L}$ proofs. The calculus

consists of propositional rules (P-rules: P1–P10), first-order quantifier rules (F-rules: F1–F6), rules for dynamic modalities (D-rules: D1–D12), and global rules (G-rules: G1–G4).

2.11 Definition (Rules). The *rule schemata* in Figure 2.5 induce *calculus rules* by:

1. If
$$\frac{\Phi_1 \vdash \Psi_1 \quad \ldots \quad \Phi_n \vdash \Psi_n}{\Phi_0 \vdash \Psi_0}$$
is an instance of a P, G, or F1–F5 rule schema in Figure 2.5, then
$$\frac{\Gamma, \langle\mathcal{J}\rangle\Phi_1 \vdash \langle\mathcal{J}\rangle\Psi_1, \Delta \quad \ldots \quad \Gamma, \langle\mathcal{J}\rangle\Phi_n \vdash \langle\mathcal{J}\rangle\Psi_n, \Delta}{\Gamma, \langle\mathcal{J}\rangle\Phi_0 \vdash \langle\mathcal{J}\rangle\Psi_0, \Delta}$$
can be applied as a proof rule of the $\mathsf{d}\mathcal{L}$ calculus, where Γ, Δ are arbitrary finite sets of additional context formulas (including empty sets) and $\mathcal{J}$ is a discrete jump set (including the empty set). Hence, the rule context Γ, Δ and prefix $\langle\mathcal{J}\rangle$ remain unchanged during rule applications.

2. Symmetric schemata can be applied on either side of the sequent: If
$$\frac{\phi_1}{\phi_0}$$
is an instance of one of the symmetric rule schemata (D-rules) in Figure 2.5, then
$$\frac{\Gamma \vdash \langle\mathcal{J}\rangle\phi_1, \Delta}{\Gamma \vdash \langle\mathcal{J}\rangle\phi_0, \Delta} \qquad \text{and} \qquad \frac{\Gamma, \langle\mathcal{J}\rangle\phi_1 \vdash \Delta}{\Gamma, \langle\mathcal{J}\rangle\phi_0 \vdash \Delta}$$
can both be applied as proof rules of the $\mathsf{d}\mathcal{L}$ calculus, where Γ, Δ are arbitrary finite sets of context formulas and $\mathcal{J}$ is a discrete jump set (including empty sets). In particular, symmetric schemata yield equivalence transformations, because the same rule applies in the antecedent as in the succedent.

3. Schema F6 applies to *all* goals containing X: If $\Phi_1 \vdash \Psi_1, \ldots, \Phi_n \vdash \Psi_n$ is the list of all open goals of the proof that contain free variable X, then an instance
$$\frac{\vdash \mathrm{QE}(\exists X \bigwedge_i (\Phi_i \vdash \Psi_i))}{\Phi_1 \vdash \Psi_1 \quad \ldots \quad \Phi_n \vdash \Psi_n}$$
of rule schema F6 can be applied as a proof rule of the $\mathsf{d}\mathcal{L}$ calculus.

$$\text{(P1)}\ \frac{\phi \vdash}{\vdash \neg\phi} \qquad \text{(P3)}\ \frac{\vdash \phi, \psi}{\vdash \phi \vee \psi} \qquad \text{(P5)}\ \frac{\vdash \phi \quad \vdash \psi}{\vdash \phi \wedge \psi} \qquad \text{(P7)}\ \frac{\phi \vdash \psi}{\vdash \phi \to \psi}$$

$$\text{(P2)}\ \frac{\vdash \phi}{\neg\phi \vdash} \qquad \text{(P4)}\ \frac{\phi \vdash \quad \psi \vdash}{\phi \vee \psi \vdash} \qquad \text{(P6)}\ \frac{\phi, \psi \vdash}{\phi \wedge \psi \vdash} \qquad \text{(P8)}\ \frac{\vdash \phi \quad \psi \vdash}{\phi \to \psi \vdash}$$

$$\text{(P9)}\ \frac{}{\phi \vdash \phi} \qquad \text{(P10)}\ \frac{\vdash \phi \quad \phi \vdash}{\vdash}$$

$$\text{(D1)}\ \frac{\langle\alpha\rangle\langle\beta\rangle\phi}{\langle\alpha;\beta\rangle\phi} \qquad \text{(D5)}\ \frac{\phi \vee \langle\alpha\rangle\langle\alpha^*\rangle\phi}{\langle\alpha^*\rangle\phi} \qquad \text{(D9)}\ \frac{\phi^{\theta_1}_{x_1}\ldots^{\theta_n}_{x_n}}{\langle x_1 := \theta_1, .., x_n := \theta_n\rangle\phi}$$

$$\text{(D2)}\ \frac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi} \qquad \text{(D6)}\ \frac{\phi \wedge [\alpha][\alpha^*]\phi}{[\alpha^*]\phi} \qquad \text{(D10)}\ \frac{\langle x_1 := \theta_1, .., x_n := \theta_n\rangle\phi}{[x_1 := \theta_1, .., x_n := \theta_n]\phi}$$

$$\text{(D3)}\ \frac{\langle\alpha\rangle\phi \vee \langle\beta\rangle\phi}{\langle\alpha \cup \beta\rangle\phi} \qquad \text{(D7)}\ \frac{\chi \wedge \psi}{\langle ?\chi\rangle\psi} \qquad \text{(D11)}\ \frac{\exists t{\geq}0\, \big((\forall 0{\leq}\tilde{t}{\leq}t\, \langle\mathcal{S}_{\tilde{t}}\rangle\chi) \wedge \langle\mathcal{S}_t\rangle\phi\big)}{\langle x_1' = \theta_1, .., x_n' = \theta_n \,\&\, \chi\rangle\phi}$$

$$\text{(D4)}\ \frac{[\alpha]\phi \wedge [\beta]\phi}{[\alpha \cup \beta]\phi} \qquad \text{(D8)}\ \frac{\chi \to \psi}{[?\chi]\psi} \qquad \text{(D12)}\ \frac{\forall t{\geq}0\, \big((\forall 0{\leq}\tilde{t}{\leq}t\, \langle\mathcal{S}_{\tilde{t}}\rangle\chi) \to \langle\mathcal{S}_t\rangle\phi\big)}{[x_1' = \theta_1, .., x_n' = \theta_n \,\&\, \chi]\phi}$$

$$\text{(F1)}\ \frac{\vdash \phi(s(X_1, .., X_n))}{\vdash \forall x\, \phi(x)} \qquad \text{(F4)}\ \frac{\vdash \phi(X)}{\vdash \exists x\, \phi(x)}$$

$$\text{(F2)}\ \frac{\phi(s(X_1, .., X_n)) \vdash}{\exists x\, \phi(x) \vdash} \qquad \text{(F5)}\ \frac{\phi(X) \vdash}{\forall x\, \phi(x) \vdash}$$

$$\text{(F3)}\ \frac{\vdash \mathrm{QE}(\forall X\, (\Phi(X) \vdash \Psi(X)))}{\Phi(s(X_1, .., X_n)) \vdash \Psi(s(X_1, .., X_n))} \qquad \text{(F6)}\ \frac{\vdash \mathrm{QE}(\exists X\, \bigwedge_i (\Phi_i \vdash \Psi_i))}{\Phi_1 \vdash \Psi_1 \quad \ldots \quad \Phi_n \vdash \Psi_n}$$

$$\text{(G1)}\ \frac{\vdash \forall^\alpha(\phi \to \psi)}{[\alpha]\phi \vdash [\alpha]\psi} \qquad \text{(G2)}\ \frac{\vdash \forall^\alpha(\phi \to \psi)}{\langle\alpha\rangle\phi \vdash \langle\alpha\rangle\psi} \qquad \text{(G3)}\ \frac{\vdash \forall^\alpha(\phi \to [\alpha]\phi)}{\phi \vdash [\alpha^*]\phi}$$

$$\text{(G4)}\ \frac{\vdash \forall^\alpha \forall v{>}0\, (\varphi(v) \to \langle\alpha\rangle\varphi(v-1))}{\exists v\, \varphi(v) \vdash \langle\alpha^*\rangle\exists v{\leq}0\, \varphi(v)}$$

All substitutions need to be admissible, including the substitution that inserts $s(X_1, .., X_n)$ into $\phi(s(X_1, .., X_n))$. In D11–D12, t and $\tilde{t}$ are fresh logical variables and $\langle\mathcal{S}_t\rangle$ is the jump set $\langle x_1 := y_1(t), .., x_n := y_n(t)\rangle$ with simultaneous solutions $y_1, .., y_n$ of the respective differential equations with constant symbols x_i as symbolic initial values. In G4, logical variable v does not occur in α. In F1 and F2, s is a new Skolem function and $X_1, .., X_n$ are all free logical variables of $\forall x\, \phi(x)$. In F3–F5, X is a new logical variable. In F6, among all open branches, the free logical variable X only occurs in the branches $\Phi_i \vdash \Psi_i$. Finally, QE needs to be defined for the formulas in F3 and F6. Especially, no Skolem dependencies on X occur in F6.

Figure 2.5.: Rule schemata of the free variable calculus for differential dynamic logic

P-Rules For propositional logic, standard rules P1–P9 with cut P10 are listed in Figure 2.5. They decompose the propositional structure of formulas. Rules P1 and P2 use simple dualities caused by the implicative semantics of sequents. P3 uses that formulas are combined disjunctively in succedents, P6 that they are conjunctive in antecedents. P4 and P5 split the proof into two cases, because conjuncts in the succedent can be proven separately (P5) and, dually, disjuncts of the antecedent can be assumed separately (P4). P7 and P8 can be derived from the equivalence of $\phi \to \psi$ and $\neg\phi \vee \psi$. The axiom rule P9 closes a goal (there are no further sub-goals), because assumption ϕ in the antecedent trivially entails ϕ in the succedent. Rule P10 is the *cut* rule that can be used for case distinctions: The right sub-goal assumes any additional formula ϕ in the antecedent that the left sub-goal shows in the succedent. We only use cuts in an orderly fashion to derive simple rule dualities and to simplify metaproofs. In practical applications, cuts are not usually needed and we conjecture that this is no coincidence.

F-Rules The quantifier rules F1 and F2 correspond to the liberalised δ^+-rule of Hähnle and Schmitt [HS94]. F4 and F5 resemble the usual γ-rule but, unlike in [Fit96, FM99, HS94, Gie01], they cannot be applied twice because the original formula is removed ($\exists x\, \phi(x)$ in F4). The calculus still has a complete handling of quantifiers due to F3 and F6, which can reconstruct and eliminate quantifiers once QE is applicable as the remaining constraints are first-order in the respective variables. In the premiss of F3 and F6, we again consider sequents $\Phi \vdash \Psi$ as abbreviations for formulas. For closed formulas, we do not need other arithmetic rules. We defer illustrations and further discussion of F-rules to Section 2.5.2.

D-Rules The dynamic modality rules transform a hybrid program into simpler logical formulas. Rules D1–D8 are as in discrete dynamic logic [HKT00, BP06]. Sequential compositions are proven using nested modalities (D1–D2), and non-deterministic choices split into their alternatives (D3–D4). D5 and D6 are the usual iteration rules, which partially unwind loops. Tests are proven by showing (D7) or assuming (D8) that the test succeeds, because $?\chi$ can only make a transition when χ holds true (Definition 2.7).

D9 uses simultaneous substitutions for handling discrete jump sets. To show that ϕ is true after a discrete jump, D9 shows that ϕ has already been true before, when replacing the x_i by their new values θ_i in ϕ by an admissible substitution. Instead, the discrete jump set can remain an unchanged prefix ($\mathcal{J}$ in Definition 2.11) for other $\mathsf{d}\mathcal{L}$ rules applied to ϕ, until the substitution for D9 is admissible. D10 uses that discrete jump sets characterise a unique deterministic transition, hence, its premiss and conclusion are equivalent. Assuming the presence of vacuous identity jumps $a := a$ for variables a that do not otherwise change (vacuous identity jumps can be added as they do not change state), we can further use D9 to *merge* sub-

sequent discrete jumps into a single discrete jump set (see previous results [BP06] for a compatible calculus detailing jump set merging, which works without the need to add vacuous identity jumps $a := a$):

$$\dfrac{\dfrac{\dfrac{\vdash \langle z := -\frac{b}{2}t^2 + Vt, v := V+1, a := -b\rangle[\beta]\phi}{\text{D9}\ \vdash \langle a := -b, v := V\rangle\langle z := \frac{a}{2}t^2 + vt, v := v+1, a := a\rangle[\beta]\phi}}{\text{D10}\ \vdash \langle a := -b, v := V\rangle[z := \frac{a}{2}t^2 + vt, v := v+1, a := a][\beta]\phi}}{\text{D2}\ \vdash \langle a := -b, v := V\rangle[z := \frac{a}{2}t^2 + vt, v := v+1, a := a;\beta]\phi}$$

More generally, $\langle x_1 := \theta_1, \ldots, x_n := \theta_n\rangle\langle x_1 := \vartheta_1, \ldots, x_n := \vartheta_n\rangle\phi$ can be merged by D9 to $\langle x_1 := \vartheta_1{}^{\theta_1}_{x_1}\ldots{}^{\theta_n}_{x_n}, \ldots, x_n := \vartheta_n{}^{\theta_1}_{x_1}\ldots{}^{\theta_n}_{x_n}\rangle\phi$.

Given first-order definable flows for their differential equations, D11–D12 handle continuous evolutions (see [AW01, LPY99, PC07] for flow approximation and solution techniques). These flows are combined in the jump set $\mathcal{S}_t$. Given a solution for the differential equation system with symbolic initial values $x_1, \ldots, x_n$, continuous evolution along differential equations can be replaced by a discrete jump $\langle \mathcal{S}_t\rangle$ with an additional quantifier for the evolution time t. The effect of the constraint on χ is to restrict the continuous evolution such that its solution $\mathcal{S}_{\tilde{t}}$ remains in the invariant region χ at all intermediate times $\tilde{t} \leq t$. This constraint simplifies to *true* if χ is *true*. Similar simplifications can be made for convex invariant conditions (Section 2.9).

G-Rules The G-rules are *global rules*. They depend on the truth of their premisses in all states reachable by α, which is ensured by the universal closure $\forall^\alpha$ with respect to all bound state variables (Definition 2.9) of the respective hybrid program α. This universal closure is required for soundness in the presence of contexts Γ, Δ (Definition 2.11) or of free variables. The G-rules are given in a form that best displays their underlying logical principles. The general pattern for applying G-rules to prove that the succedent of their conclusion holds is to prove that both their premiss and the antecedent of their conclusion hold.

G1–G2 are generalisation rules and can be used to strengthen postconditions: antecedent $[\alpha]\phi$ is sufficient for proving succedent $[\alpha]\psi$ when postcondition ϕ entails ψ in all relevant states reachable by α, which are overapproximated by the universal closure $\forall^\alpha$ with respect to the bound variables of α. G3 is an induction schema with *inductive invariant* ϕ. Similarly, G4 is a generalisation of Harel's convergence rule [HKT00] to the hybrid case with decreasing *variant* φ. Both rules are given in a form that best displays their underlying logical principles and similarity. G3 says that ϕ holds after any number of repetitions of α, if it holds initially (antecedent) and, for all reachable states (as overapproximated by $\forall^\alpha$), invariant ϕ remains true after one iteration of α (premiss). G4 expresses that the variant $\varphi(v)$ holds for some real number $v \leq 0$ after repeating α sufficiently often, if $\varphi(v)$ holds for some real number at all (antecedent) and, by premiss, decreases after every execution of α by 1 (or at least any other positive real constant).

For practical verification, rules G3 or G4 can be combined with generalisation (G1–G2) to prove a postcondition ψ of a loop α^* by showing that (a) the antecedent of the respective goals of G3 and G4 holds initially, that (b) their sub-goals hold, which represent the induction step, and that (c) finally, the postcondition of the succedent in their goals entails ψ. The corresponding variants of G3 and G4 are derived rules:

$$\text{(G3')}\quad \frac{\vdash \phi \quad \vdash \forall^\alpha(\phi \to [\alpha]\phi) \quad \vdash \forall^\alpha(\phi \to \psi)}{\vdash [\alpha^*]\psi}$$

$$\text{(G4')}\quad \frac{\vdash \exists v\, \varphi(v) \quad \vdash \forall^\alpha \forall v{>}0\, (\varphi(v) \to \langle\alpha\rangle\varphi(v-1)) \quad \vdash \forall^\alpha(\exists v{\leq}0\, \varphi(v) \to \psi)}{\vdash \langle\alpha^*\rangle\psi}$$

For instance, using a cut with $\phi \to [\alpha^*]\phi$, rule G3' can be derived from G3 and G1:

$$\dfrac{\dfrac{\dfrac{\vdash \forall^\alpha(\phi \to [\alpha]\phi)}{\phi \vdash [\alpha^*]\phi}\,\text{G3}}{\vdash \phi \to [\alpha^*]\phi}\,\text{P7} \qquad \dfrac{\vdash \phi \qquad \dfrac{\vdash \forall^\alpha(\phi \to \psi)}{[\alpha^*]\phi \vdash [\alpha^*]\psi}\,\text{G1}}{\phi \to [\alpha^*]\phi \vdash [\alpha^*]\psi}\,\text{P8}}{\vdash [\alpha^*]\psi}\,\text{P10}$$

The notions of derivations and proofs are standard, except that F6 produces multiple conclusions. Hence, we define derivations as finite acyclic graphs instead of trees:

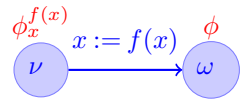
2.12 Definition (Provability). A *derivation* is a finite acyclic graph labelled with sequents such that, for every node, the (set of) labels of its children must be the (set of) premisses of an instance of one of the calculus rules (Definition 2.11) and the (set of) labels of the parents of these children must be the (set of) conclusions of that rule instance. A formula ψ is *provable* from a set Φ of formulas, denoted by $\Phi \vdash_{\mathsf{d}\mathcal{L}} \psi$, iff there is a finite subset $\Phi_0 \subseteq \Phi$ for which the sequent $\Phi_0 \vdash \psi$ is derivable, i.e., there is a derivation with a single root (i.e., node without parents) labelled $\Phi_0 \vdash \psi$.

See Figure 2.6 for an illustration of the correpondence of a representative set of proof rules for dynamic modalities to the transition semantics of hybrid programs (Definition 2.7).

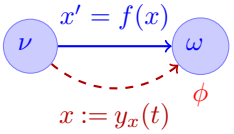
2.5.2. Deduction Modulo with Invertible Quantifiers and Real Quantifier Elimination

The F-rules lift quantifier elimination to $\mathsf{d}\mathcal{L}$ by following a generalised deduction modulo approach. They integrate decision procedures, e.g., for real quantifier elimination as a background prover [Bec99] into the deductive proof system. Yet, unlike in the approaches of Dowek et al. [DHK03] and Tinelli [Tin03], the information

$$\frac{\phi_x^{f(x)}}{[x := f(x)]\phi}$$

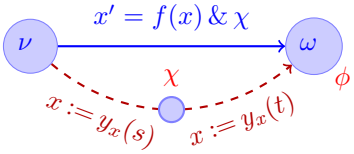


$$\frac{\exists t{\geq}0\, \langle x := y_x(t)\rangle\phi}{\langle x' = f(x)\rangle\phi}$$

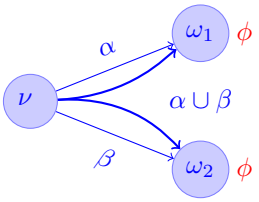


$$\frac{\exists t{\geq}0\, (\bar{\chi} \wedge \langle x := y_x(t)\rangle\phi)}{\langle x' = f(x)\,\&\,\chi\rangle\phi}$$

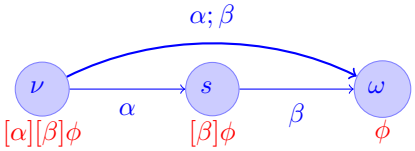
$$\bar{\chi} \equiv \forall 0{\leq}s{\leq}t\, \langle x := y_x(s)\rangle\chi$$



$$\frac{[\alpha]\phi \wedge [\beta]\phi}{[\alpha \cup \beta]\phi}$$



$$\frac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi}$$



$$\frac{\vdash \forall^{\alpha}(\phi \to [\alpha]\phi)}{\phi \vdash [\alpha^*]\phi}$$

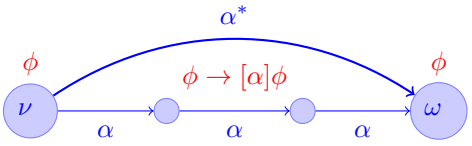


Figure 2.6.: Correspondence of dynamic proof rules and transition semantics

given to the background prover is not restricted to ground formulas [Tin03] or atomic formulas [DHK03]. Further, real quantifier elimination is quite different from uninterpreted logic [HS94, Fit96, Gie01] in that the resulting formulas are not obtained by instantiation but by intricate arithmetic recombination. The F-rules can use any theory that admits quantifier elimination (see Definition 2.10) and has a decidable ground theory, for instance, the first-order theory of real arithmetic (i.e., the theory of real-closed fields [Tar51, CH91]). A *formula of real arithmetic* is a first-order formula with $+, -, \cdot, /, =, \leq, <, \geq, >$ as the only function or predicate symbols besides constant symbols of Σ and logical variables of V.

Integrating quantifier elimination to deal with statements about real quantities is quite challenging in the presence of modalities that influence the value of flexible symbols. In principle, quantifier elimination can be used to handle quantified constraints as arising for continuous evolutions. In dℒ, however, real quantifiers interact with modalities containing further discrete or continuous transitions, which is an effect that is inherent in the interacting nature of hybrid systems. A hybrid formula like $\exists z \langle z'' = -b; ?m - z \geq s; z'' = 0\rangle\, m - z < s$ is not first-order, hence quantifier elimination cannot be applied. Even more so, the effect of a modality depends on the solutions of the differential equations contained therein. For instance, it is hard to know in advance, which first-order constraints need to be solved by QE for the above formula. To find out how z evolves from $\exists z$ to $m - z < s$, the system dynamics needs to be taken into account (similar for repetitions). Hence, our calculus first unwraps the first-order structure before applying QE to the resulting arithmetic formulas.

Lifting Quantifier Elimination by Invertible Quantifier Rules

The purpose of the F-rules is to postpone QE until the actual arithmetic constraints become apparent. The idea is that F1,F2,F4, and F5 temporarily remove quantifiers by introducing new auxiliary symbols for quantified variables such that the proof can be continued beyond the occurrence of the quantifier to further analyse the modalities contained therein. Later, when the actual first-order constraints for the auxiliary symbol have been discovered, the corresponding quantifier can be reintroduced (F3, F6) and quantifier elimination QE is applied to reduce the sequents equivalently to a simpler formula with less (distinct) symbols. In F4–F6, the respective auxiliary symbols are free logical variables. In F1–F3, Skolem function terms are used instead for reasons that are crucial for soundness and will be illustrated in the sequel. In this context, we think of *free* logical variables as being introduced by γ-rules (F4 and F5), hence implicitly existentially quantified.

To illustrate how quantifier and dynamic rules of dℒ interact to combine arithmetic with dynamic reasoning in hybrid systems, we analyse the braking behaviour in train control. The proof in Figure 2.7 can be used to analyse whether a train can violate its MA although it is braking. As the proof reveals, the answer depends on

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{v \geq 0, z < m \vdash v^2 > 2b(m-z)}{\vdash v \geq 0 \land z < m \to v^2 > 2b(m-z)}\,\text{P7,P6} \quad \dfrac{v \geq 0, z < m \vdash -\frac{b}{2}T^2 + vT + z > m}{v \geq 0, z < m \vdash \langle z := -\frac{b}{2}T^2 + vT + z \rangle z > m}\,\text{D9}}{v \geq 0, z < m \vdash T \geq 0 \land \langle z := -\frac{b}{2}T^2 + vT + z \rangle z > m}\,\text{F6}}{v \geq 0, z < m \vdash \exists t{\geq}0\, \langle z := -\frac{b}{2}t^2 + vt + z \rangle z > m}\,\text{P5}}{v \geq 0, z < m \vdash \langle z' = v, v' = -b \rangle z > m}\,\text{F4}}{\vdash v \geq 0 \land z < m \to \langle z' = v, v' = -b \rangle z > m}\,\text{D11}\,\text{P7,P6}$$

Figure 2.7.: Deduction modulo for analysis of MA-violation in braking mode

the initial velocity v. For notational convenience, we use the simplified D11 rule, as the differential equation is not restricted to an invariant region. Rule F4 introduces a new free variable T for the quantified variable t to postpone QE. Later, when F6 is applied in Figure 2.7, the conjunction of its two goals can be handled by QE and simplification, yielding the resulting sub-goal:

$$\text{QE}\left(\exists T\left((v \geq 0 \land z < m \to T \geq 0) \land (v \geq 0 \land z < m \to -\frac{b}{2}T^2 + vT + z > m)\right)\right)$$
$$\equiv v \geq 0 \land z < m \;\to\; v^2 > 2b(m-z)\;.$$

The open branch with this formula reveals the speed limit and can be used to synthesise a corresponding parameter constraint. When $v^2 > 2b(m-z)$ holds initially, m can be violated even in braking mode, as the velocity exceeds the braking power. Similarly, $v^2 \leq 2b(m-z)$ guarantees that m can be respected by appropriate braking. The constraint so discovered thus forms a *controllability constraint* of ETCS, i.e., a constraint that characterises from which states control choices exist that guarantee safety. It is essentially equivalent to $[z'' = -b]z \leq m$ and to $\exists a\,(-b \leq a \leq A \land [z'' = a]z \leq m)$. The controllable region of the state space of ETCS is illustrated in Figure 2.8.

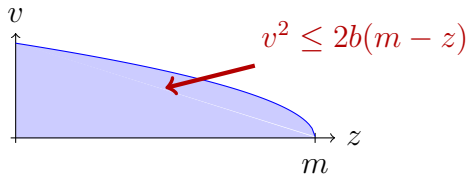


Figure 2.8.: Controllable region of ETCS dynamics

Admissibility in Invertible Quantifier Rules

The requirement that substitutions in F3 are admissible implies that no occurrence of $s(X_1, \ldots, X_n)$ is within the scope of a quantifier for any of these X_i.

This prevents F3 from rearranging the order of quantifiers from $\exists X_i \forall s$ to the weaker $\forall s \exists X_i$, which would be unsound, because it is not sufficient to show the weak sub-goal $\forall s \exists X_i$ in order to prove the strong statement $\exists X_i \forall s$ saying that the same X_i works for all s.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{F3 is not applicable}}{\vdash \mathrm{QE}(\exists X\,(2X+1 < s(X)))}}{\vdash 2X+1 < s(X)}\,\text{F6}}{\vdash \langle x := 2X+1\rangle(x < s(X))}\,\text{D9}}{\vdash \forall y\,\langle x := 2X+1\rangle(x < y)}\,\text{F1}}{\vdash \exists x \forall y\,\langle x := 2x+1\rangle(x < y)}\,\text{F4}$$

a. Wrong rearrangement attempt

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overbrace{\vdash \mathrm{QE}\,(\exists X\ \mathrm{QE}(\forall s\,(2X+1 < s)))}^{\text{false}}}{\vdash \mathrm{QE}(\forall s\,(2X+1 < s))}\,\text{F6}}{\vdash 2X+1 < s(X)}\,\text{F3}}{\vdash \langle x := 2X+1\rangle(x < s(X))}\,\text{D9}}{\vdash \forall y\,\langle x := 2X+1\rangle(x < y)}\,\text{F1}}{\vdash \exists x \forall y\,\langle x := 2x+1\rangle(x < y)}\,\text{F4}$$

b. Correct reintroduction order

Figure 2.9.: Deduction modulo with invertible quantifiers

For the moment, suppose the rules did not contain QE. The requirement for admissible substitutions (Definition 2.9) ensures that the proof attempt of an invalid formula in Figure 2.9a cannot close in the $\mathsf{d}\mathcal{L}$ calculus. At the indicated position, F3, which would unsoundly invert the quantifier order to $\forall S \exists X$, cannot be applied: In F3, the substitution inserting $s(X)$ gives $\exists Y\,(2Y+1 < s(X))$ by α-renaming, instead of $\exists X\,(2X+1 < s(X))$. Thus, F3 is not applicable, because the quantified formula is not of the form $\Psi(s(X))$

Now, we consider what happens in the presence of QE. The purpose of QE is to (equivalently) remove quantifiers like $\exists X$. Thus it is no longer obvious that the admissibility argument applies, because the blocking variable X would have disappeared after successful quantifier elimination. However, quantifier elimination over the reals is defined in the first-order theory of real arithmetic [Tar51, CH91]. Yet, when eliminating X in Figure 2.9a, the Skolem term $s(X)$ is no term of real arithmetic, as, unlike that of $+$, the interpretation of s is arbitrary. The truth value of $\exists X\,(2X+1 < s(X))$ depends on the interpretation of s. If $I(s)$ is a constant function, the formula is true, if $I(s)(a) = 2a$, it is false. In general, such cases cannot be distinguished without quantifiers. Thus, in the presence of uninterpreted function terms, real arithmetic does not generally admit quantifier elimination. Consequently, F6 and F3 are only applicable if QE is defined. Yet, QE can be lifted to formulas with Skolem functions when these are instances of real arithmetic formulas:

2.13 Lemma (Quantifier elimination lifting). *Quantifier elimination can be lifted to instances of formulas of first-order theories that admit quantifier elimination, i.e., to formulas that result from the base theory by substitution.*

Proof. Let formula ϕ be an instance of ψ, with ψ being a formula of the base theory, i.e., ϕ is $\psi_{z_1}^{\theta_1}\ldots_{z_n}^{\theta_n}$ for some variables z_i and arbitrary terms θ_i. As QE is defined for the base theory, let $\mathrm{QE}(\psi)$ be the quantifier-free formula belonging to ψ according to Definition 2.10. Then $\mathrm{QE}\,(\psi)_{z_1}^{\theta_1}\ldots_{z_n}^{\theta_n}$ satisfies the requirements of Definition 2.10 for ϕ, because $\vDash \psi_{z_1}^{\theta_1}\ldots_{z_n}^{\theta_n} \leftrightarrow \mathrm{QE}\,(\psi)_{z_1}^{\theta_1}\ldots_{z_n}^{\theta_n}$: For F defined as $\psi \leftrightarrow \mathrm{QE}(\psi)$, we have that $\vDash F$ implies $\vDash F_{z_1}^{\theta_1}\ldots_{z_n}^{\theta_n}$ by a standard consequence of the substitution lemma. □

By Lemma 2.13, QE is defined in the presence of Skolem terms that do not depend on quantified variables, e.g., for $\exists X\,(2X+1 < t(Y,Z))$ which is an instance of the form $(\exists X\,(2X+1<z))_z^{t(Y,Z)}$. However, QE is *not defined* in the premiss of F6 when Skolem-dependencies on X occur. In Figure 2.9a, $\exists X\,(2X+1<s(X))$ is no instance of first-order real arithmetic, because, by α-renaming, $(\exists X\,(2X+1<z))_z^{s(X)}$ yields a different formula $\exists Y\,(2Y+1<s(X))$. An occurrence of $s(X)$, which corresponds to a quantifier nesting of $\exists X\,\forall s$, thus requires $s(X)$ to be eliminated by F3 before F6 can eliminate X, see Figure 2.9b. Hence, inner universal quantifiers are handled first and unsound quantifier rearrangements are prevented even in the presence of QE.

Finally observe that F3 and F6 do not require quantifiers to be eliminated in the same order in which they occurred in the original formula. The elimination order within homogeneous quantifier blocks like $\forall x_1\,\forall x_2$ is not restricted as there are no Skolem dependencies among the corresponding auxiliary Skolem terms. Yet, eliminating such a quantifier block is sound in any order (accordingly for $\exists x_1\,\exists x_2$). Similarly, F6 and F3 could interchange the order of $\forall x\,\exists y$ to the stronger $\exists y\,\forall x$, because the resulting Skolem term s for x in the former formula does not depend on y. In this direction, however, the interchange is sound, as it amounts to proving a stronger statement.

Quantifier Elimination and Modalities

Quantifier elimination over the first-order theory of reals cannot handle modal formulas. Hence, the $\mathsf{d}\mathcal{L}$ calculus first reduces modalities to first-order constraints before applying QE. Yet, this is not necessary for all modalities. The modal subformula in the following example does not impose any constraints on X but its truth value only determines which first-order constraints are imposed on X:

$$\mathrm{QE}(\exists X\,\big(X<0\wedge\big((\langle y:=2y+1\rangle y>0)\rightarrow X>y\big)\big)) \;\equiv\; (\langle y:=2y+1\rangle y>0)\rightarrow y<0$$

Modal formulas not containing elimination variable X can be handled by propositional abstraction in QE and remain unchanged. Syntactically, the reason for this is that $\mathsf{d}\mathcal{L}$ rule applications on modal formulas that do not contain X will never produce formulas which do. The semantical reason for the same fact is a generalisation of the coincidence lemma to $\mathsf{d}\mathcal{L}$, which says that values of variables that do

not occur will neither affect the transition structure of a hybrid program nor the truth value of formulas.

2.14 Lemma (Coincidence). *If the interpretations (and assignments and states, respectively) I, η, ν and J, ε, ω agree on all symbols that occur free in the formula ϕ, then $val_{I,\eta}(\nu, \phi) = val_{J,\varepsilon}(\omega, \phi)$.*

Proof. The proof is by a simple structural induction using the definition of valuation $val_{I,\eta}(\nu, \cdot)$ and $\rho_{I,\eta}(\cdot)$ in Definition 2.5–2.7. □

Global Invertible Quantifier Rules

Rules F3 and F6 display an asymmetry. While F3 works locally on a branch, F6 needs to respect all branches that contain X. The reason for this is that branches are implicitly combined conjunctively in sequent calculus, as all branches have to close simultaneously for a proof to succeed (Definition 2.12). Universal quantifiers can be handled separately for conjunctions by $\forall x\,(\phi \wedge \psi) \equiv \forall x\,\phi \wedge \forall x\,\psi$. Existential quantifiers, however, can only be dealt with separately for disjunctions but not for conjunctions. In calculi with a disjunctive proof structure, the roles of F3 and F6 would be interchanged but the phenomenon remains.

Rule F6 can be applied to the full proof (i.e., all open goals) like a global closing substitution in the tableau calculus [Fit96]. By Lemma 2.14 it only needs to consider the set of all open goals $\Phi_i \vdash \Psi_i$ that actually contain X. F6 resembles global closing substitutions in uninterpreted free variable tableaux [Gie01]. Both avoid the backtracking over closing substitutions that local closing substitutions require. Unlike closing substitutions, however, F6 uses the fixed semantics of function and predicate symbols of real arithmetic such that variables can already be eliminated equivalently by QE before the proof completes. Applying F3 or F6 early does not necessarily close the proof. Instead, equivalent constraints on the remaining variables will be revealed, which can simplify the proof or help deriving parametric constraints or invariants.

2.6. Soundness

In this section, we prove that the dℒ calculus is a sound axiomatisation of the transition behaviour of hybrid systems.

We prove that a successful deduction in the dℒ calculus always produces correct verification results about hybrid systems: The dℒ calculus is sound, i.e., all provable (closed) formulas are valid in all states of all interpretations. To reflect the interaction of free variables and Skolem terms, we adapt the notion of soundness for the liberalised δ^+-rule in free variable tableau calculi [HS94] to sequent calculus.

A formula ϕ is *satisfiable* [HS94] (or has a model) if there is an interpretation I and a state ν such that *for all* variable assignments η we have $I, \eta, \nu \models \phi$. Closed

tableaux prove the unsatisfiability of the negated goal. Sequent calculi work dually and show validity of the proof obligation. Consequently, we use the dual notion and say that ψ is a *consequence* of ϕ iff, for every I, ν *there is* an assignment η such that $I, \eta, \nu \models \psi$, provided that, for every I, ν *there is* an assignment η such that $I, \eta, \nu \models \phi$. A calculus rule that concludes Ψ from the premisses Φ is *sound* if Ψ is a consequence of Φ. As usual, multiple branches in Ψ or Φ are combined conjunctively.

In this context, we think of free logical variables as being introduced by γ-rules, i.e., F4 and F5 (hence the implicit existential quantification of free logical variables by η). For closed formulas (without free logical variables), validity corresponds to being a consequence from an empty set of open goals. Hence, closed formulas that are provable with a sound deduction are *valid* (true in all states of all interpretations).

2.15 Theorem (Soundness). *The* $\mathsf{d}\mathcal{L}$ *calculus is sound.*

Proof. The calculus is sound if each rule instance is sound. All rules of the $\mathsf{d}\mathcal{L}$ calculus except F1,F2 and F6 are even *locally sound*, i.e., their conclusion is true at I, η, ν if all its premisses are true in I, η, ν, which implies soundness. It is also easy to show that locally sound rules remain sound when adding contexts $\Gamma, \Delta, \langle\mathcal{J}\rangle$ as in Definition 2.11, since a discrete jump set $\langle\mathcal{J}\rangle$ characterises a unique state transition. Local soundness proofs of D1–D8 and propositional rules are as usual. Note that, for symmetric rules, local soundness implies that the premiss and conclusion are *equivalent*, i.e., true in the same states.

D9 The rule D9 is locally sound. Assume that the premiss holds in I, η, ν, i.e., $I, \eta, \nu \models \phi^{\theta_1}_{x_1} \ldots ^{\theta_n}_{x_n}$. We have to show that $I, \eta, \nu \models \langle x_1 := \theta_1, \ldots, x_n := \theta_n\rangle\phi$, i.e., $I, \eta, \omega \models \phi$ for a state ω with $(\nu, \omega) \in \rho_{I,\eta}(x_1 := \theta_1, \ldots, x_n := \theta_n)$. This follows directly from the substitution lemma, which generalises to dynamic logic for admissible substitutions (Definition 2.9). Rule D10 uses that discrete jumps are deterministic.

D11 The rule D11 is locally sound. Let $y_1, \ldots, y_n$ be a solution for the differential equation system $x_1' = \theta_1, \ldots, x_n' = \theta_n$ with symbolic initial values $x_1, \ldots, x_n$. Let further $\langle\mathcal{S}_t\rangle$ be the jump set $\langle x_1 := y_1(t), \ldots, x_n := y_n(t)\rangle$. Assume I, η, ν are such that the premiss is true: $I, \eta, \nu \models \exists t{\geq}0\, (\bar{\chi} \wedge \langle\mathcal{S}_t\rangle\phi)$ with $\forall 0{\leq}\tilde{t}{\leq}t\, \langle\mathcal{S}_{\tilde{t}}\rangle\chi$ abbreviated as $\bar{\chi}$. For any $\zeta \in \mathbb{R}$, we denote by η^ζ the assignment that agrees with η except that it assigns ζ to t. Then, by assumption, there is a real value $r \geq 0$ such that $I, \eta^r, \nu \models \bar{\chi} \wedge \langle\mathcal{S}_t\rangle\phi$. Abbreviate $x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi$ by $\mathcal{D}$. We have to show that $I, \eta, \nu \models \langle\mathcal{D}\rangle\phi$. Equivalently, by Lemma 2.14, we show $I, \eta^r, \nu \models \langle\mathcal{D}\rangle\phi$, because t is a fresh variable that does not occur in $\mathcal{D}$ or ϕ. Let function $f : [0, r] \to \mathrm{Sta}(\Sigma)$ be defined such that $(\nu, f(\zeta)) \in \rho_{I,\eta^\zeta}(\mathcal{S}_t)$ for all $\zeta \in [0, r]$. By premiss, $f(0)$ is identical to ν and ϕ holds at $f(r)$.

Thus it only remains to show that f respects the constraints of Definition 2.7 for $\mathcal{D}$. In fact, f obeys the continuity and differentiability properties of Definition 2.7 by the corresponding properties of the y_i. Moreover, $val_{I,\eta^r}(f(\zeta), x_i) = val_{I,\eta^r}(\nu, y_i(t))$ has a derivative of value $val_{I,\eta^r}(f(\zeta), \theta_i)$, because y_i is a solution of the differential equation $x_i' = \theta_i$ with corresponding initial value $\nu(x_i)$. Further, it can be shown that the evolution invariant region χ is respected along f as follows: By premiss, $I, \eta^r, \nu \models \bar{\chi}$ holds for the initial state ν, thus $val_{I,\eta^r}(f(\zeta), \chi) = \mathit{true}$ for all $\zeta \in [0, r]$. Combining these results, we can conclude that f is a witness for $I, \eta, \nu \models \langle\mathcal{D}\rangle\phi$. The converse direction can be shown accordingly to prove the dual rule D12 using Lemma 2.8.

F1 The proof is a sequent calculus adaptation of that in [HS94]. By contraposition, assume that there are I, ν such that for all η it is the case that $I, \eta, \nu \not\models \forall x\, \phi(x)$, hence $I, \eta, \nu \models \exists x\, \neg\phi(x)$. We construct an interpretation I' that agrees with I except for the new function symbol s. Let $b_1, \ldots, b_n \in \mathbb{R}$ be arbitrary elements and let η^b assign b_i to the respective X_i for $1 \leq i \leq n$. As $I, \eta, \nu \models \exists x\, \neg\phi(x)$ holds for all η, we pick a witness d for $I, \eta^b, \nu \models \exists x\, \neg\phi(x)$ and choose $I'(s)(b_1, \ldots, b_n) = d$. For this interpretation I' and state ν we have $I', \eta, \nu \not\models \phi(s(X_1, \ldots, X_n))$ for all assignments η by Lemma 2.14, as $X_1, \ldots, X_n$ are all free variables determining the truth value of $\phi(s(X_1, \ldots, X_n))$. To see that the contexts Γ, Δ of Definition 2.11 can be added to instantiate this rule, consider the following. Since s is new and does not occur in the context Γ, Δ, the latter do not change their truth value by passing from I to I'. Likewise, s is rigid so that it does not change its value by adding jump prefix $\langle\mathcal{J}\rangle$ which concludes the proof. The proof of F2 is dual.

F3 F3 is locally sound. Assume that $I, \eta, \nu \models \mathrm{QE}(\forall X\, (\Phi(X) \vdash \Psi(X)))$. Since QE yields an equivalence, we can conclude $I, \eta, \nu \models \forall X\, (\Phi(X) \vdash \Psi(X))$. Then if the antecedent of the conclusion is true, i.e., $I, \eta, \nu \models \Phi(s(X_1, \ldots, X_n))$, we can conclude $I, \eta, \nu \models \Psi(s(X_1, \ldots, X_n))$ by choosing $val_{I,\eta}(\nu, s(X_1, \ldots, X_n))$ for X in the premiss. By admissibility of substitutions, variables $X_1, \ldots, X_n$ are free at all occurrences of $s(X_1, \ldots, X_n)$, hence their value is the same in all occurrences.

F4 F4 is locally sound by a simplified version of the proof in [HS94]. For any I, η, ν with $I, \eta, \nu \models \phi(X)$ we can conclude $I, \eta, \nu \models \exists x\, \phi(x)$ according to the witness $\eta(X)$. The proof of F5 is dual.

F6 For any I, ν let η be such that $I, \eta, \nu \models \mathrm{QE}(\exists X\, \bigwedge_i(\Phi_i \vdash \Psi_i))$. Again, this implies $I, \eta, \nu \models \exists X\, \bigwedge_i(\Phi_i \vdash \Psi_i)$, because quantifier elimination yields an equivalence. We pick a witness $d \in \mathbb{R}$ for this existential quantifier. As X does not occur anywhere else in the proof, it disappears from all open premisses of

the proof by applying F6. Hence, by the coincidence lemma 2.14, the value of X does not change the truth value of the premise of F6. Consequently, η can be extended to η' by changing the interpretation of X to the witness d such that $I, \eta', \nu \models \bigwedge_i (\Phi_i \vdash \Psi_i)$. Thus, η' extends I, η, ν to a simultaneous model of all conclusions.

G2 Rules G1–G4 are locally sound by a variation of the usual proofs [HKT00] using universal closures for local soundness. G1–G2 are simple refinements of Lemma 2.14 using that the universal closure $\forall^\alpha$ comprises all variables that change in α. Let $I, \eta, \nu \models \langle\alpha\rangle\phi$, i.e., let $(\nu, \nu') \in \rho_{I,\eta}(\alpha)$ with $I, \eta, \nu' \models \phi$. As α can only change its bound variables, which are quantified universally in the universal closure $\forall^\alpha$, the premiss implies $I, \eta, \nu' \models \phi \to \psi$, thus $I, \eta, \nu' \models \psi$ and $I, \eta, \nu \models \langle\alpha\rangle\psi$. The proof of G1 is accordingly.

G3 For any I, η, ν with $I, \eta, \nu \models \forall^\alpha(\phi \to [\alpha]\phi)$, we conclude $I, \eta, \nu' \models \phi \to [\alpha]\phi$ for all ν' with $(\nu, \nu') \in \rho_{I,\eta}(\alpha)$. As these share the same η, we can further conclude $I, \eta, \nu \models \phi \to [\alpha^*]\phi$ by induction along the series of states ν' reached from ν by repeating α. The universal closure is necessary as, otherwise, the premiss may yield different η in different states ν'.

G4 Assume that the antecedent and premiss hold in I, η, ν. By premiss, we have $I, \eta[v \mapsto d], \nu' \models v > 0 \land \varphi(v) \to \langle\alpha\rangle\varphi(v-1)$ for all $d \in \mathbb{R}$ and all states ν' that are reachable by α^* from ν, because $\forall^\alpha$ comprises all variables that are bound by α, which are the same as those bound by α^*. By antecedent, there is a $d \in \mathbb{R}$ such that $I, \eta[v \mapsto d], \nu \models \varphi(v)$. Now, the proof is a well-founded induction on d. If $d \leq 0$, we directly have $I, \eta, \nu \models \langle\alpha^*\rangle\exists v{\leq}0\, \varphi(v)$ for zero repetitions. Otherwise, if $d > 0$, we have, by premiss, that

$$I, \eta[v \mapsto d], \nu \models v > 0 \land \varphi(v) \to \langle\alpha\rangle\varphi(v-1)$$

As $v > 0 \land \varphi(v)$ holds true, we have for some ν' with $(\nu, \nu') \in \rho_{I,\eta[v\mapsto d]}(\alpha)$ that $I, \eta[v \mapsto d], \nu' \models \varphi(v-1)$. Thus, $I, \eta[v \mapsto d-1], \nu' \models \varphi(v)$ satisfies the induction hypothesis for a smaller d and a reachable ν', because $(\nu, \nu') \in \rho_{I,\eta}(\alpha)$ as v does not occur in α. The induction is well-founded, because d decreases by 1 up to the base case $d \leq 0$. □

2.7. Completeness

In this section, we prove that the $\mathsf{d}\mathcal{L}$ calculus is a sound and complete axiomatisation of the transition behaviour of hybrid systems relative to differential equations.

2.7.1. Incompleteness

Theorem 2.15 shows that all provable closed dℒ formulas are valid. The converse question is whether the dℒ calculus is *complete*, i.e., all valid dℒ formulas are provable. Combining completeness for first-order logic [HS94] and decidability of real-arithmetic [CH91], it is easy to see that our calculus is complete for closed formulas of first-order real arithmetic by chaining the quantifier rules F1,F2,F4,F5 with the respective inverse rules F3,F6, using P-rules as needed to unfold the propositional structure. In the presence of modalities, however, dℒ is not axiomatisable and, unlike its basis of first-order *real* arithmetic, dℒ is undecidable. Both unbounded repetition in the discrete fragment and unbounded evolution in the continuous fragment cause incompleteness. Beyond hybrid dynamics, where reachability is known to be undecidable [Hen96], we show that even the purely discrete and purely continuous parts of dℒ are not effectively axiomatisable. Hence, valid dℒ formulas are not always provable.

2.16 Theorem (Incompleteness). *Both the discrete fragment and the continuous fragment of* dℒ *are* not effectively axiomatisable*, i.e., they have no sound and complete effective calculus, because natural numbers are definable in both fragments.*

Proof. We prove that natural numbers are definable among the real numbers of dℒ interpretations in both fragments. Then these fragments extend first-order *integer* arithmetic such that the incompleteness theorem of Gödel [Göd31] applies. Natural numbers are definable in the discrete fragment without continuous evolutions using repetitive additions:

$$nat(n) \;\leftrightarrow\; \langle x := 0; (x := x + 1)^* \rangle\, x = n \quad .$$

In the continuous fragment, an isomorphic copy of the natural numbers is definable using linear differential equations:

$$nat(n) \;\leftrightarrow\; \exists s\, \exists c\, \exists \tau\, (s = 0 \wedge c = 1 \wedge \tau = 0 \wedge \langle s' = c, c' = -s, \tau' = 1 \rangle (s = 0 \wedge \tau = n)) \quad .$$

These differential equations characterise sin and cos as unique solutions for s and c,

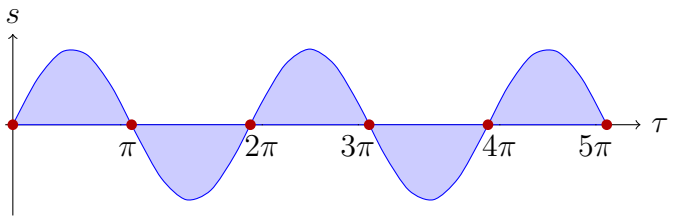


Figure 2.10.: Characterisation of ℕ as zeros of solutions of differential equations

respectively. Their zeros, as detected by τ, correspond to an isomorphic copy of

natural numbers, scaled by π, i.e., $nat(n)$ holds iff n is of the form $k\pi$ for a $k \in \mathbb{N}$, see Figure 2.10. The initial values for s and c prevent the trivial solution identical to 0. □

In this context, note that hybrid programs contain a computationally complete sublanguage and that reachability of hybrid systems is undecidable [Hen96].

2.7.2. Relative Completeness

The standard approach for showing adequacy of a calculus when its logic is not effectively axiomatisable is to analyse the deductive power of the calculus relative to a base logic or relative to an ineffective oracle rule for the base logic [Coo78, Har79, HKT00]. In calculi for discrete programs, completeness is proven relative to the handling of data [Coo78, Har79, HKT00]. For hybrid systems, this is inadequate: By Theorem 2.16, no sound calculus for $\mathsf{d}\mathcal{L}$ can be complete relative to its data (the reals), because its basis, first-order real arithmetic, is a perfectly decidable and axiomatisable theory [Tar51].

According to Theorem 2.16, continuous evolutions, repetitive discrete transitions, and their interaction cause non-axiomatisability of $\mathsf{d}\mathcal{L}$. Discrete transitions and repetition do not supersede the complexity of continuous transitions. Even relative to an oracle for handling properties of discrete jumps and repetition, the $\mathsf{d}\mathcal{L}$ calculus is not complete, simply because not all differential equations have solutions that are definable in first-order arithmetic so that D12 can be used. For instance, the solutions of $s' = c, c' = -s$ are trigonometric functions (like sin and cos), which are not first-order definable. The question is whether the converse is true, i.e., whether hybrid programs can be verified given that all required differential equations can be handled.

To calibrate the deductive power of the $\mathsf{d}\mathcal{L}$ calculus in light of its inherent incompleteness, we analyse the quotient of reasoning about hybrid systems modulo differential equation handling. Using generalisations of the usual notions of relative completeness for discrete systems [Coo78, Har79, HKT00] to the hybrid case, we show that the $\mathsf{d}\mathcal{L}$ calculus completely axiomatises $\mathsf{d}\mathcal{L}$ relative to one single additional axiom about valid first-order properties of differential equations. Essentially, we drop the effectiveness requirement for one oracle axiom and show that the resulting $\mathsf{d}\mathcal{L}$ calculus is sound and complete.

As a basis, we define FOD as the *first-order logic of differential equations*, i.e., first-order real arithmetic augmented with formulas expressing properties of differential equations, that is, $\mathsf{d}\mathcal{L}$ formulas of the form $[x_1' = \theta_1, \ldots, x_n' = \theta_n]F$ with a first-order formula F. Dually, the diamond formula $\langle x_1' = \theta_1, \ldots, x_n' = \theta_n\rangle F$ is expressible as $\neg[x_1' = \theta_1, \ldots, x_n' = \theta_n]\neg F$.

2.17 Theorem (Relative completeness). *The* $\mathsf{d}\mathcal{L}$ *calculus is* complete relative to *FOD, i.e., every valid* $\mathsf{d}\mathcal{L}$ *formula can be derived from FOD-tautologies.*

Proof Outline. The (constructive) proof, which, in full, is contained in the remainder of Section 2.7, adapts the techniques of Cook [Coo78] and Harel [Har79, HKT00] to the hybrid case. The decisive step is to show that every valid property of a repetition α^* can be proven by G3 or G4, respectively, with a sufficiently strong invariant or variant that is expressible in dℒ. For this, we show that dℒ formulas can be expressed equivalently in FOD, and that valid dℒ formulas can be derived from corresponding FOD axioms in the dℒ calculus. In turn, the crucial step is to construct a finite FOD formula that characterises the effect of unboundedly many repetitive hybrid transitions and just uses finitely many real variables. □

This main result completely aligns hybrid and continuous verification proof-theoretically. It gives a formal justification that reasoning about hybrid systems is possible to *exactly* the same extent to which it is possible to show properties of solutions of differential equations. Theorem 2.17 shows that superpositions of discrete jumps, continuous evolutions, and repetitions of hybrid processes, can be verified when corresponding (intermediate) properties of differential equations are provable. Moreover, in a proof-theoretical sense, our calculus completely lifts all verification techniques for dynamical systems to hybrid systems.

Summarising Theorem 2.15 and 2.17, the dℒ calculus axiomatises the transition behaviour of hybrid systems completely relative to the handling of differential equations!

In the sequel, we present a fully constructive proof of Theorem 2.17, which generalises the techniques of Harel [Har79, HKT00] and Cook [Coo78] to the hybrid case. It shows that for every valid dℒ formula, there is a finite set of valid FOD-formulas from which it can be derived in the dℒ calculus. See the proof outline of Theorem 2.17 for a road map of the proof.

Natural numbers are definable in FOD by Theorem 2.16. In this section, we abbreviate quantifiers over natural numbers, e.g., $\forall x\,(nat(x) \to \phi)$ by $\forall x:\mathbb{N}\ \phi$ and $\exists x\,(nat(x) \wedge \phi)$ by $\exists x:\mathbb{N}\ \phi$. Likewise, we abbreviate quantifiers over integers, e.g., $\forall x\,((nat(x) \vee nat(-x)) \to \phi)$ by $\forall x:\mathbb{Z}\ \phi$.

2.7.3. Characterising Real Gödel Encodings

As the central device for constructing a FOD formula that captures the effect of unboundedly many repetitive hybrid transitions and just uses finitely many real variables, we prove that a real version of Gödel encoding is definable in FOD. That is, we give a FOD formula that reversibly packs finite sequences of real values into a single real number.

Observe that a single differential equation system is *not* sufficient for defining these pairing functions as their solutions are differentiable, yet, as a consequence of Morayne's theorem [Mor87], there is no differentiable surjection $\mathbb{R} \to \mathbb{R}^2$, nor to any part of $\mathbb{R}^2$ of positive measure. We show that real sequences can be encoded

nevertheless by chaining the effects of solutions of multiple differential equations and quantifiers.

2.18 Lemma (ℝ-Gödel encoding). *The formula $at(Z, n, j, z)$, which holds iff Z is a real number that represents a Gödel encoding of a sequence of n real numbers with real value z at position j (for $1 \leq j \leq m$), is definable in FOD. For a formula $\phi(z)$ we abbreviate $\exists z\,(\mathrm{at}(Z, n, j, z) \wedge \phi(z))$ by $\phi(Z_j^{(n)})$.*

$$\sum_{i=0}^{\infty} \frac{a_i}{2^i} = a_0.a_1a_2\ldots \qquad \sum_{i=0}^{\infty} \frac{b_i}{2^i} = b_0.b_1b_2\ldots \quad\rightsquigarrow\quad \sum_{i=0}^{\infty}\left(\frac{a_i}{2^{2i-1}} + \frac{b_i}{2^{2i}}\right) = a_0b_0.a_1b_1a_2b_2\ldots$$

a. Fractional encoding principle by bit interleaving

$$\begin{aligned}
\mathrm{at}(Z, n, j, z) &\leftrightarrow \forall i : \mathbb{Z}\ \ \mathrm{digit}(z, i) = \mathrm{digit}(Z, n(i-1)+j) \wedge \mathit{nat}(n) \wedge \mathit{nat}(j) \wedge n > 0\\
\mathrm{digit}(a, i) &= \mathrm{intpart}(2\,\mathrm{frac}(2^{i-1}a))\\
\mathrm{intpart}(a) &= a - \mathrm{frac}(a)\\
\mathrm{frac}(a) = z &\leftrightarrow \exists i : \mathbb{Z}\ z = a - i \wedge -1 < z \wedge z < 1 \wedge az \geq 0\\
2^i = z &\leftrightarrow i \geq 0 \wedge \exists x\,\exists t\,(x = 1 \wedge t = 0 \wedge \langle x' = x\ln 2, t' = 1\rangle(t = i \wedge x = z))\\
&\vee\ i < 0 \wedge \exists x\,\exists t\,(x = 1 \wedge t = 0 \wedge \langle x' = -x\ln 2, t' = -1\rangle(t = i \wedge x = z))\\
\ln 2 = z &\leftrightarrow \exists x\,\exists t\,(x = 1 \wedge t = 0 \wedge \langle x' = x, t' = 1\rangle(x = 2 \wedge t = z))
\end{aligned}$$

b. Definition of ℝ-Gödel encoding in FOD

Figure 2.11.: Characterising Gödel encoding of ℝ-sequences in one real number

Proof. The basic idea of the ℝ-Gödel encoding is to interleave the bits of real numbers as depicted in Figure 2.11a (for a pairing of $n = 2$ numbers a and b). For defining $\mathrm{at}(Z, n, j, z)$, we use several auxiliary functions to improve readability, see Figure 2.11b. Note that these definitions need no recursion, hence, like in the notation $\phi(Z_j^{(n)})$, we can consider occurrences of the function symbols as syntactic abbreviations for quantified variables satisfying the respective definitions.

The function symbol $\mathrm{digit}(a, i)$ gives the i-th bit of $a \in \mathbb{R}$ when represented with basis 2. For $i > 0$, $\mathrm{digit}(a, i)$ yields fractional bits, and, for $i \leq 0$, it yields bits of the integer part. For instance, $\mathrm{digit}(a, 1)$ yields the first fractional bit, $\mathrm{digit}(a, 0)$ is the least-significant bit of the integer part of a. The function $\mathrm{intpart}(a)$ represents the integer part of $a \in \mathbb{R}$. The function $\mathrm{frac}(a)$ represents the fractional part of $a \in \mathbb{R}$, which drops all integer bits. The last constraint in its definition implies that $\mathrm{frac}(a)$

keeps the sign of a (or 0). Consequently, intpart(a) and digit(a,i) also keep the sign of a (or 0). Exponentiation 2^i is definable using differential equations, using an auxiliary characterisation of the natural logarithm $\ln 2$. The definition of 2^i splits into the case of exponential growth when $i \geq 0$ and a symmetric case of exponential decay when $i < 0$. □

2.7.4. Expressibility and Rendition of Hybrid Program Semantics

In order to show that dℒ is sufficiently expressive to state the invariants and variants that are needed for proving valid statements about loops with G3 and G4, we prove an expressibility result. We give a constructive proof that the state transition relation of hybrid programs is definable in FOD, i.e., there is a FOD-formula $\mathcal{S}_\alpha(\vec{x},\vec{v})$ characterising the state transitions of hybrid program α from the state characterised by the vector $\vec{x}$ of variables to the state characterised by vector $\vec{v}$.

For this, we need to characterise hybrid processes equivalently by differential equations in FOD. Observe that the existence of such characterisations does *not* follow from results embedding Turing machines into differential equations [Bra95c, GCB07], because, unlike Turing machines, hybrid processes are not restricted to discrete values on a grid (like $\mathbb{N}^k$) but work with continuous real values. Furthermore, Turing machines only have repetitions of discrete transitions on discrete data (e.g., $\mathbb{N}$). For hybrid programs, instead, we have to characterise repetitive interactions of discrete and continuous transitions in continuous space (some $\mathbb{R}^k$).

2.19 Lemma (Program rendition). *For every hybrid program α with variables among $\vec{x} = x_1,\dots,x_k$ there is a FOD-formula $\mathcal{S}_\alpha(\vec{x},\vec{v})$ with variables among the $2k$ distinct variables $\vec{x} = x_1,\dots,x_k$ and $\vec{v} = v_1,\dots,v_k$ such that*

$$\vDash \mathcal{S}_\alpha(\vec{x},\vec{v}) \leftrightarrow \langle\alpha\rangle\vec{x} = \vec{v}$$

or, equivalently, for every I,η,ν,

$$I,\eta,\nu \vDash \mathcal{S}_\alpha(\vec{x},\vec{v}) \;\; \textit{iff} \;\; (\nu,\nu[\vec{x} \mapsto val_{I,\eta}(\nu,\vec{v})]) \in \rho_{I,\eta}(\alpha) \;\; .$$

Proof. By Lemma 2.14, interpretations of the vectors $\vec{x}$ and $\vec{v}$ characterises the input and output states, respectively, as far as α is concerned. These vectors are finite because α is finite. Vectorial equalities like $\vec{x} = \vec{v}$ or quantifiers $\exists\vec{v}$ are to be understood component-wise. The program rendition is defined inductively in Figure 2.12. To simplify the notation, we assume that all variables $x_1,\dots,x_k$ are affected in discrete jumps and differential equations by adding vacuous $x_i := x_i$ or $x_i' = 0$ if x_i does not change in the respective statement, otherwise.

$$
\begin{aligned}
\mathcal{S}_{x_1:=\theta_1,..,x_k:=\theta_k}(\vec{x},\vec{v}) &\equiv \bigwedge_{i=1}^{k}(v_i=\theta_i)\\
\mathcal{S}_{x_1'=\theta_1,..,x_k'=\theta_k}(\vec{x},\vec{v}) &\equiv \langle x_1'=\theta_1,\ldots,x_k'=\theta_k\rangle \vec{v}=\vec{x}\\
\mathcal{S}_{x_1'=\theta_1,..,x_k'=\theta_k\,\&\,\chi}(\vec{x},\vec{v}) &\equiv \exists t\,\big(t=0\wedge\langle x_1'=\theta_1,\ldots,x_k'=\theta_k,t'=1\rangle\big(\vec{v}=\vec{x}\\
&\qquad\wedge[x_1'=-\theta_1,\ldots,x_k'=-\theta_k,t'=-1](t\geq 0\to\chi)\big)\big)\\
\mathcal{S}_{?\chi}(\vec{x},\vec{v}) &\equiv \vec{v}=\vec{x}\wedge\chi\\
\mathcal{S}_{\beta\cup\gamma}(\vec{x},\vec{v}) &\equiv \mathcal{S}_\beta(\vec{x},\vec{v})\vee\mathcal{S}_\gamma(\vec{x},\vec{v})\\
\mathcal{S}_{\beta;\gamma}(\vec{x},\vec{v}) &\equiv \exists\vec{z}\,(\mathcal{S}_\beta(\vec{x},\vec{z})\wedge\mathcal{S}_\gamma(\vec{z},\vec{v}))\\
\mathcal{S}_{\beta^*}(\vec{x},\vec{v}) &\equiv \exists Z\,\exists n:\mathbb{N}\,\big(Z_1^{(n)}=\vec{x}\wedge Z_n^{(n)}=\vec{v}\\
&\qquad\wedge\forall i:\mathbb{N}\,(1\leq i<n\to\mathcal{S}_\beta(Z_i^{(n)},Z_{i+1}^{(n)}))\big)
\end{aligned}
$$

Figure 2.12.: Explicit rendition of hybrid program transition semantics in FOD

Differential equations give FOD-formulas hence no further reduction is necessary. Evolution along differential equations with invariant regions is definable by following the unique flow (Lemma 2.8) backwards. Continuous evolution is reversible, i.e., the transitions of $x_i'=-\theta$ are inverse to those of $x_i'=\theta$. Consequently, when using auxiliary variable t, all evolutions of $[x_1'=-\theta_1,\ldots,x_k'=-\theta_k,t'=-1]$ follow the same flow as $\langle x_1'=\theta_1,\ldots,x_k'=\theta_k,t'=1\rangle$ but backwards. By also reverting clock t, we ensure that, along the reverse flow, χ has been true at all times (because of the box modality) until starting time $t=0$, see Figure 2.13.

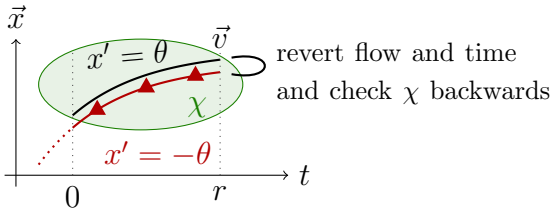


Figure 2.13.: Invariant region checks along backwards flow over time t

To show reversibility, let $(\nu,\omega)\in\rho_{I,\eta}(x_1'=\theta_1,\ldots,x_k'=\theta_k)$, that is, let $f:[0,r]\to \mathrm{Sta}(\Sigma)$ be a solution of $x_1'=\theta_1,\ldots,x_k'=\theta_k$ starting in state ν and ending in ω. Then $g:[0,r]\to\mathrm{Sta}(\Sigma)$, defined as $g(\zeta)=f(r-\zeta)$, starts in ω and ends in ν. Thus, it only remains to show that g is a solution of $x_1'=-\theta_1,\ldots,x_k'=-\theta_k$, which can be

seen for $1 \leq i \leq k$ as follows:

$$\begin{aligned}\frac{\mathsf{d}g(t)(x_i)}{\mathsf{d}t}(\zeta) =& \frac{\mathsf{d}f(r-t)(x_i)}{\mathsf{d}t}(\zeta) = \frac{\mathsf{d}f(u)(x_i)}{\mathsf{d}u}\frac{\mathsf{d}(r-t)}{\mathsf{d}t}(\zeta) = -\frac{\mathsf{d}f(u)(x_i)}{\mathsf{d}u}(\zeta)\\ =& - val_{I,\eta}(f(\zeta),\theta_i) = val_{I,\eta}(f(\zeta),-\theta_i) \ .\end{aligned}$$

Unlike all other cases, case $\mathcal{S}_{x_1'=\theta_1,..,x_k'=\theta_k \,\&\, \chi}(\vec{x},\vec{v})$ in Figure 2.12 uses nested FOD modalities. Nested modalities can be avoided in $\mathcal{S}_\alpha(\vec{x},\vec{v})$ using an equivalent FOD formula without them, see Figure 2.13:

$$\begin{aligned}\exists t\, \exists r \big(& t = 0 \wedge \langle x_1' = \theta_1, \ldots, x_k' = \theta_k, t' = 1\rangle (\vec{v} = \vec{x} \wedge r = t) \wedge\\ & \forall \vec{x}\, \forall t\, (\vec{x} = \vec{v} \wedge t = r \rightarrow [x_1' = -\theta_1, \ldots, x_k' = -\theta_k, t' = -1](t \geq 0 \rightarrow \chi))\big) \ .\end{aligned}$$

With a finite formula, the characterisation of repetition $\mathcal{S}_{\beta^*}(\vec{x},\vec{v})$ in FOD needs to capture arbitrarily long sequences of intermediate real-valued states and the correct transition between successive states of such a sequence. To achieve this with first-order quantifiers, we use the real Gödel encoding from Lemma 2.18 in Figure 2.12 to map unbounded sequences of real-valued states reversibly to a single real number Z, which can be quantified over in first-order logic. □

Using the program rendition from Lemma 2.19 to characterise modalities, we prove that every $\mathsf{d}\mathcal{L}$ formula can be expressed equivalently in FOD by structural induction.

2.20 Lemma (Expressibility). *Logic $\mathsf{d}\mathcal{L}$ is* expressible *in FOD: for all $\mathsf{d}\mathcal{L}$ formulas $\phi \in \mathrm{Fml}(\Sigma, V)$ there is a FOD-formula $\phi^\# \in \mathrm{Fml}_{FOD}(\Sigma, V)$ that is equivalent, i.e., $\vDash \phi \leftrightarrow \phi^\#$. The converse holds trivially.*

Proof. The proof follows an induction on the structure of formula ϕ for which it is imperative to find an equivalent $\phi^\#$ in FOD. Observe that the construction of $\phi^\#$ from ϕ is effective.

0. If ϕ is a first-order formula, then $\phi^\# := \phi$ already is a FOD-formula such that nothing has to be shown.

1. If ϕ is of the form $\varphi \vee \psi$, then by induction hypothesis there are FOD-formulas $\varphi^\#, \psi^\#$ such that $\vDash \varphi \leftrightarrow \varphi^\#$ and $\vDash \psi \leftrightarrow \psi^\#$, from which we can conclude by congruence that $\vDash (\varphi \vee \psi) \leftrightarrow (\varphi^\# \vee \psi^\#)$ giving $\vDash \phi \leftrightarrow \phi^\#$ by choosing $\varphi^\# \vee \psi^\#$ for $\phi^\#$. Likewise reasoning concludes the other propositional connectives or quantifiers.

2. The case where ϕ is of the form $\langle\alpha\rangle\psi$ is a consequence of the characterisation of the semantics of hybrid programs in FOD. The expressibility conjecture holds by induction hypothesis using the equivalence of explicit hybrid program renditions from Lemma 2.19:

$$\vDash \quad \langle\alpha\rangle\psi \leftrightarrow \exists \vec{v}\, (\mathcal{S}_\alpha(\vec{x},\vec{v}) \wedge \psi^{\#\vec{v}}_{\vec{x}}) \ .$$

3. The case where ϕ is $[\alpha]\psi$ is again a consequence of Lemma 2.19:

$$\vDash \quad [\alpha]\psi \leftrightarrow \forall \vec{v}\,(\mathcal{S}_\alpha(\vec{x},\vec{v}) \to \psi^{\#\vec{v}}_{\vec{x}}) \qquad \square$$

The above proofs directly carry over to rich test $\mathsf{d}\mathcal{L}$, i.e., the logic where $\mathsf{d}\mathcal{L}$ formulas are allowed in tests $?\chi$ of hybrid programs and invariant regions χ of differential equations, when using $\chi^\#$ in place of χ in Figure 2.12. Accordingly, nested modalities can be avoided in FOD by using the following formula for $\mathcal{S}_{x_1'=\theta_1,\ldots,x_k'=\theta_k\,\&\,\chi}(\vec{x},\vec{v})$:

$$\begin{aligned}\exists t\,\exists r\,\big(&t=0 \land \langle x_1'=\theta_1,\ldots,x_k'=\theta_k,t'=1\rangle(\vec{v}=\vec{x}\land r=t)\land\\ &\forall \vec{z}\,\big(\exists \vec{x}\,\exists t\,(\vec{x}=\vec{v}\land t=r\land\langle x_1'=-\theta_1,\ldots,x_k'=-\theta_k,t'=-1\rangle(t\geq 0\land \vec{z}=\vec{x}))\\ &\quad\to \chi^{\#\vec{z}}_{\vec{x}}\big)\big)\end{aligned}$$

2.7.5. Relative Completeness of First-order Assertions

As special cases of Theorem 2.17, we first prove relative completeness for first-order assertions about hybrid programs. These first-order cases constitute the basis for the general completeness proof for arbitrary formulas of differential dynamic logic.

In the sequel, we use the notation $\vdash_{\mathcal{D}} \phi$ to indicate that a $\mathsf{d}\mathcal{L}$ formula ϕ is derivable (Definition 2.12) from a set of FOD-tautologies, which is equivalent to saying that ϕ is derivable in the $\mathsf{d}\mathcal{L}$ calculus augmented with a single *oracle axiom* $\mathcal{D}$, that gives all valid FOD-instances. Likewise, we use the notation $\Gamma \vdash_{\mathcal{D}} \Delta$ to indicate that the sequent $\Gamma \vdash \Delta$ is derivable from $\mathcal{D}$.

For the completeness proof, we use several simplifications. For uniform proofs, we assume formulas to use a simplified vocabulary. A formula ϕ is valid iff it is true in *all* I,η,ν. In particular, we can assume valid ϕ to use Skolem constants (or state variables) instead of free logical variables. Existential quantifiers can be represented as modalities: $\exists x\,\phi \equiv \langle x'=1\rangle\phi \lor \langle x'=-1\rangle\phi$. For simplicity, we use cut (P10) and weakening to glue together subproofs propositionally. Weakening (i.e., from $\phi \vdash \psi$ infer $\phi_1,\phi \vdash \psi,\psi_1$) can be emulated using contexts Γ,Δ from Definition 2.11, and we use it implicitly together with P10 in the following. Derivability of sequents and corresponding formulas is equivalent by the following lemma.

2.21 Lemma (Derivability of sequents). $\vdash_{\mathcal{D}} \phi \to \psi$ *iff* $\phi \vdash_{\mathcal{D}} \psi$.

Proof. When we consider sequents as abbreviations for formulas, both sides are identical. Otherwise, let $\vdash_{\mathcal{D}} \phi \to \psi$ be derivable from $\mathcal{D}$. Using P10 (and weakening) with $\phi \to \psi$, this derivation can be extended to one of $\phi \vdash_{\mathcal{D}} \psi$:

$$\dfrac{\dfrac{*}{\phi \vdash \phi \to \psi,\psi} \qquad \dfrac{\dfrac{*}{\phi \vdash \phi,\psi}\,\text{P9} \qquad \dfrac{*}{\psi,\phi \vdash \psi}\,\text{P9}}{\phi,\phi \to \psi \vdash \psi}\,\text{P8}}{\phi \vdash \psi}\,\text{P10}$$

The converse direction is by an application of P7. □

2.22 Lemma (Generalisation). *If $\vdash_{\mathcal{D}} \phi$ is provable without free logical variables, then so are $\vdash_{\mathcal{D}} \forall x\, \phi$ and $\vdash_{\mathcal{D}} \langle x_1 := \theta_1, \dots x_n := \theta_n \rangle \phi$.*

Proof. For the second conjecture, let $\langle \mathcal{I} \rangle$ abbreviate $\langle x_1 := \theta_1, \dots x_n := \theta_n \rangle$. We prefix each formula in the proof of ϕ with $\langle \mathcal{I} \rangle$ and show that this gives a proof of $\langle \mathcal{I} \rangle \phi$. F6 is not needed in the proof due to the absence of free logical variables. As an intermediate step, we first show that prefixing with $\langle \mathcal{I} \rangle$ gives an (extended) proof with rule applications generalised to allowing for nested jump prefixes $\langle \mathcal{I} \rangle \langle \mathcal{J} \rangle$: By the argument in Theorem 4.7, it is easy to see for discrete jump sets $\mathcal{I}$ and $\mathcal{J}$ that the dℒ rules remain sound with nested jump prefix $\langle \mathcal{I} \rangle \langle \mathcal{J} \rangle$ in place of only a single prefix $\langle \mathcal{J} \rangle$ from Definition 2.11. Applicability conditions of rules do not depend on jump prefixes, as Definition 2.11 allows adding *any* jump prefix. Thus, we obtain a sound (extended) proof of $\langle \mathcal{I} \rangle \phi$ when replacing—with arbitrary unchanged context $\Gamma, \Delta, \langle \mathcal{J} \rangle$—every rule application of the form

$$\frac{\Gamma, \langle \mathcal{J} \rangle \Phi_1 \vdash \langle \mathcal{J} \rangle \Psi_1, \Delta \quad \dots \quad \Gamma, \langle \mathcal{J} \rangle \Phi_n \vdash \langle \mathcal{J} \rangle \Psi_n, \Delta}{\Gamma, \langle \mathcal{J} \rangle \Phi_0 \vdash \langle \mathcal{J} \rangle \Psi_0, \Delta}$$

in the proof of ϕ by a rule application with additional unchanged prefix $\langle \mathcal{I} \rangle$ for corresponding $\Gamma, \Delta, \langle \mathcal{J} \rangle$:

$$\frac{\Gamma, \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_1 \vdash \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Psi_1, \Delta \quad \dots \quad \Gamma, \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_n \vdash \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Psi_n, \Delta}{\Gamma, \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_0 \vdash \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Psi_0, \Delta} \tag{2.2}$$

Next, we show that these nested jump prefixes can be reduced to a single jump prefix as Definition 2.11 allows for: Let $\langle \mathcal{I}\mathcal{J} \rangle$ denote the discrete jump set obtained by merging $\langle \mathcal{I} \rangle$ and $\langle \mathcal{J} \rangle$ using D9 as in Section 2.5.1. We replace each rule application (with nested prefixes) of the form (2.2) by the following derivation with only a single prefix (assuming $n = 1$ for notational convenience):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_1 \vdash \langle \mathcal{I}\mathcal{J} \rangle \Psi_1, \Delta}\dots \quad \dfrac{\dfrac{*}{\Gamma, \langle \mathcal{I}\mathcal{J} \rangle \Phi_1 \vdash \langle \mathcal{I}\mathcal{J} \rangle \Phi_1, \Delta}\text{P9}}{\Gamma, \langle \mathcal{I}\mathcal{J} \rangle \Phi_1 \vdash \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_1, \Delta}\text{D9}}{\Gamma, \langle \mathcal{I}\mathcal{J} \rangle \Phi_1 \vdash \langle \mathcal{I}\mathcal{J} \rangle \Psi_1, \Delta}\text{P10}}{\Gamma, \langle \mathcal{I}\mathcal{J} \rangle \Phi_0 \vdash \langle \mathcal{I}\mathcal{J} \rangle \Psi_0, \Delta}}{\Gamma, \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_0 \vdash \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Psi_0, \Delta}\text{D9,D9}$$

The bottom-most D9 applications merge $\langle \mathcal{I} \rangle$ into $\langle \mathcal{J} \rangle$ in the antecedent and succedent, respectively. The unmarked rule applies the same rule that has been used in (2.2), which is applicable on $\Phi_0 \vdash \Psi_0$ for *any* context by Definition 2.11, including $\Gamma, \Delta, \langle \mathcal{I}\mathcal{J} \rangle$. The subsequent cut with $\langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_1$ restores the form of the premiss in (2.2). The left branch continues using a dual argument to turn succedent $\langle \mathcal{I}\mathcal{J} \rangle \Psi_1$

into $\langle\mathcal{I}\rangle\langle\mathcal{J}\rangle\Psi_1$, thereby yielding a set of non-extended rule applications with the same conclusions and premisses as the extended rule application (2.2):

$$\text{P10}\dfrac{\Gamma, \langle\mathcal{I}\rangle\langle\mathcal{J}\rangle\Phi_1 \vdash \langle\mathcal{I}\rangle\langle\mathcal{J}\rangle\Psi_1, \Delta \qquad \text{D9}\dfrac{\text{P9}\dfrac{*}{\Gamma, \langle\mathcal{I}\mathcal{J}\rangle\Psi_1 \vdash \langle\mathcal{I}\mathcal{J}\rangle\Psi_1, \Delta}}{\Gamma, \langle\mathcal{I}\rangle\langle\mathcal{J}\rangle\Psi_1 \vdash \langle\mathcal{I}\mathcal{J}\rangle\Psi_1, \Delta}}{\Gamma, \langle\mathcal{I}\rangle\langle\mathcal{J}\rangle\Phi_1 \vdash \langle\mathcal{I}\mathcal{J}\rangle\Psi_1, \Delta}$$

For reducing the first conjecture of this lemma to the second, let s be a Skolem constant for state variable x. By the above proof, we derive $\vdash_{\mathcal{D}} \langle x := s\rangle\phi$. Using F1, we continue this derivation to a proof of $\forall X\, \langle x := X\rangle\phi$, which we abbreviate as $\forall x\, \phi$ (see text below Definition 2.9). Rule F1 is applicable for Skolem constant s as no free logical variables occur in the proof. □

2.23 Proposition (Relative completeness of first-order safety). *For every hybrid program $\alpha \in \mathrm{HP}(\Sigma, V)$ and each $F, G \in \mathrm{Fml}_{FOL}(\Sigma, V)$ of first-order logic*

$$\vDash F \to [\alpha]G \text{ implies } \vdash_{\mathcal{D}} F \to [\alpha]G \text{ (and } F \vdash_{\mathcal{D}} [\alpha]G \text{ by Lemma 2.21)} .$$

Proof. We generalise the relative completeness proof by Cook [Coo78] to $\mathsf{d}\mathcal{L}$ and follow an induction on the structure of program α. In the following, *IH* is short for the induction hypothesis.

1. The cases where α is of the form $x_1 := \theta_1, \ldots, x_n := \theta_n$, $?\chi$, $\beta \cup \gamma$, or $\beta; \gamma$ are consequences of the soundness of the symmetric rules D2, D4, and D8–D10. Since these rules are symmetric, they perform equivalent transformations. Consequently, whenever their conclusion is valid, their premiss is valid and of smaller complexity (the programs get simpler), hence derivable by IH. Thus, we can derive $F \to [\alpha]G$ by applying the respective rule. We explicitly show the proof for $\beta; \gamma$ as it contains an extra twist.

2. $\vDash F \to [\beta;\gamma]G$, which implies $\vDash F \to [\beta][\gamma]G$. By Lemma 2.20, there is a FOD-formula $G^{\#}$ such that $\vDash G^{\#} \leftrightarrow [\gamma]G$. From the validity of $\vDash F \to [\beta]G^{\#}$, we can conclude by IH that $F \vdash_{\mathcal{D}} [\beta]G^{\#}$ is derivable. Similarly, because of $\vDash G^{\#} \to [\gamma]G$, we conclude $\vdash_{\mathcal{D}} G^{\#} \to [\gamma]G$ by IH. Using Lemma 2.22, we conclude $\vdash_{\mathcal{D}} \forall^{\beta}(G^{\#} \to [\gamma]G)$. With an application of G1, the latter derivation can be extended to a derivation of $[\beta]G^{\#} \vdash_{\mathcal{D}} [\beta][\gamma]G$. Combining the above derivations propositionally by a cut with $[\beta]G^{\#}$, we can derive $F \vdash_{\mathcal{D}} [\beta][\gamma]G$, from which D2 yields $F \vdash_{\mathcal{D}} [\beta;\gamma]G$ as desired (and Lemma 2.21 or P7 yield $\vdash_{\mathcal{D}} F \to [\beta;\gamma]G$).

3. $\vDash F \to [x_1' = \theta_1, \ldots, x_n' = \theta_n]G$ is a FOD-formula and hence derivable as a $\mathcal{D}$ axiom. Continuous evolution $x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi$ with invariant regions is definable in FOD by Lemma 2.19, which we consider as an abbreviation in this proof.

4. $\vDash F \to [\beta^*]G$ can be derived by induction. For this, we define the invariant as a FOD encoding of the statement that all potential poststates of β^* satisfy G according to Lemma 2.20:

$$\phi \equiv ([\beta^*]G)^{\#} \equiv \forall \vec{v}\,(\mathcal{S}_{\beta^*}(\vec{x},\vec{v}) \to G^{\vec{v}}_{\vec{x}}) \ .$$

Since $F \to \phi$ and $\phi \to G$ are valid FOD-formulas, they are derivable by $\mathcal{D}$; so is $F \vdash_{\mathcal{D}} \phi$ by Lemma 2.21. By Lemma 2.22 and G1, $[\beta^*]\phi \vdash_{\mathcal{D}} [\beta^*]G$ is derivable. Likewise, $\phi \to [\beta]\phi$ is valid according to the semantics of repetition, thus derivable by IH, since β is less complex. Using Lemma 2.22, we can derive $\vdash_{\mathcal{D}} \forall^{\beta}(\phi \to [\beta]\phi)$, from which G3 yields $\phi \vdash_{\mathcal{D}} [\beta^*]\phi$. Combining the above derivations propositionally by a cut with $[\beta^*]\phi$ and ϕ yields $F \vdash_{\mathcal{D}} [\beta^*]G$. □

2.24 Proposition (Relative completeness of first-order liveness). *For each hybrid program $\alpha \in \mathrm{HP}(\Sigma, V)$ and each $F, G \in \mathrm{Fml}_{FOL}(\Sigma, V)$ of first-order logic*

$$\vDash F \to \langle\alpha\rangle G \text{ implies } \vdash_{\mathcal{D}} F \to \langle\alpha\rangle G \text{ (and } F \vdash_{\mathcal{D}} \langle\alpha\rangle G \text{ by Lemma 2.21)} \ .$$

Proof. We generalise the arithmetic completeness proof by Harel [Har79] to the hybrid case. Most cases of the proof are simple adaptations of the corresponding cases in Proposition 2.23. What remains to be shown is the case of repetitions. Assume that $\vDash F \to \langle\beta^*\rangle G$. To derive this formula by G4, we use a FOD-formula $\varphi(n)$ as a variant expressing that, after n iterations, β can lead to a state satisfying G. This formula is obtained from Lemma 2.19-2.20 as $(\langle\beta^*\rangle G)^{\#} \equiv \exists \vec{v}\,(\mathcal{S}_{\beta^*}(\vec{x},\vec{v}) \wedge G^{\vec{v}}_{\vec{x}})$, *except* that the quantifier on the repetition count n is removed such that n becomes a free variable (plus index shifting to count repetitions):

$$\varphi(n-1) \ \equiv \ \exists \vec{v} \exists Z \left(Z^{(n)}_1 = \vec{x} \wedge Z^{(n)}_n = \vec{v} \wedge \forall i : \mathbb{N}\ (1 \leq i < n \to \mathcal{S}_{\beta}(Z^{(n)}_i, Z^{(n)}_{i+1})) \wedge G^{\vec{v}}_{\vec{x}}\right) \ .$$

By Lemma 2.18, $\varphi(n)$ can only hold true if n is a natural number.

According to the loop semantics, $\vDash n > 0 \wedge \varphi(n) \to \langle\beta\rangle\varphi(n-1)$ is valid by construction: If $n > 0$ is a natural number then so is $n-1$, and if β reaches G after n repetitions, then, after executing β once, $n-1$ repetitions of β reach G. By IH, this formula is derivable, since β contains less loops. By Lemma 2.22, we extend this derivation to $\vdash_{\mathcal{D}} \forall^{\beta} \forall n{>}0\,(\varphi(n) \to \langle\beta\rangle\varphi(n-1))$. Thus $\exists v\,\varphi(v) \vdash_{\mathcal{D}} \langle\beta^*\rangle \exists v{\leq}0\,\varphi(v)$ by G4. It only remains to show that the antecedent is derivable from F and $\langle\beta^*\rangle G$ is derivable from the succedent. From our assumption, we conclude that the following are valid FOD-formulas, hence $\mathcal{D}$-axioms:

- $\vDash F \to \exists v\,\varphi(v)$, because $\vDash F \to \langle\beta^*\rangle G$, and
- $\vDash (\exists v{\leq}0\,\varphi(v)) \to G$, because $v{\leq}0$ and the fact, that, by Lemma 2.18, $\varphi(v)$ only holds true for natural numbers, imply $\varphi(0)$. Further, $\varphi(0)$ entails G, because zero repetitions of β have no effect.

From the latter we derive $\vdash_{\mathcal{D}} \forall^{\beta}(\exists v{\leq}0\, \varphi(v) \rightarrow G)$ by Lemma 2.22 and extend the derivation to $\langle\beta^*\rangle\exists v{\leq}0\, \varphi(v) \vdash_{\mathcal{D}} \langle\beta^*\rangle G$ by G2. From $\vdash_{\mathcal{D}} F \rightarrow \exists v\, \varphi(v)$ we conclude $F \vdash_{\mathcal{D}} \exists v\, \varphi(v)$ by Lemma 2.21. Now, the above derivations can be combined propositionally by a cut with $\langle\beta^*\rangle\exists v{\leq}0\, \varphi(v)$ and with $\exists v\, \varphi(v)$ to yield $F \vdash_{\mathcal{D}} \langle\beta^*\rangle G$. □

2.7.6. Relative Completeness of the Differential Logic Calculus

Having succeeded with the proofs of the above statements we can finish the proof of the Theorem 2.17, which is the central result of this work.

Proof of Theorem 2.17. The proof follows a basic structure analogous to that of Harel's proof for the discrete case [Har79, Theorem 3.1]. We have to show that every valid $\mathsf{d}\mathcal{L}$ formula ϕ can be proven from FOD axioms within the $\mathsf{d}\mathcal{L}$ calculus: from $\vDash \phi$ we have to prove $\vdash_{\mathcal{D}} \phi$. The proof proceeds as follows: By propositional recombination, we inductively identify fragments of ϕ that correspond to $\phi_1 \rightarrow [\alpha]\phi_2$ or $\phi_1 \rightarrow \langle\alpha\rangle\phi_2$ logically. Next, we express subformulas ϕ_i equivalently in FOD by Lemma 2.20, and use Proposition 2.23 and 2.24 to resolve these first-order safety or liveness assertions. Finally, we prove that the original $\mathsf{d}\mathcal{L}$ formula can be re-derived from the subproofs.

We can assume ϕ to be given in conjunctive normal form by appropriate propositional reasoning. In particular, we assume that negations are pushed inside over modalities using the dualities $\neg[\alpha]\phi \equiv \langle\alpha\rangle\neg\phi$ and $\neg\langle\alpha\rangle\phi \equiv [\alpha]\neg\phi$. The remainder of the proof follows an induction on a measure $|\phi|$ defined as the number of modalities in ϕ. For a simple and uniform proof, we assume quantifiers to be abbreviations for modal formulas: $\exists x\, \phi \equiv \langle x' = 1\rangle\phi \vee \langle x' = -1\rangle\phi$ and $\forall x\, \phi \equiv [x' = 1]\phi \wedge [x' = -1]\phi$.

0. $|\phi| = 0$ then ϕ is a first-order formula, hence derivable by $\mathcal{D}$.

1. ϕ is of the form $\neg\phi_1$, then ϕ_1 is first-order, as we assumed negations to be pushed inside. Hence, $|\phi| = 0$ and Case 0 applies.

2. ϕ is of the form $\phi_1 \wedge \phi_2$, then individually deduce the simpler proofs for $\vdash_{\mathcal{D}} \phi_1$ and $\vdash_{\mathcal{D}} \phi_2$ by IH, which can be combined by P5.

3. ϕ is a disjunction and—without loss of generality—has one of the following forms (otherwise use associativity and commutativity to select a different order for the disjunction):

$$\begin{array}{lcl} \phi_1 & \vee & [\alpha]\phi_2 \\ \phi_1 & \vee & \langle\alpha\rangle\phi_2 \end{array}$$

   As a unified notation for those cases we use $\phi_1 \vee \langle\!\langle\alpha\rangle\!\rangle\phi_2$. Then, $|\phi_2| < |\phi|$, since ϕ_2 has less modalities. Likewise, $|\phi_1| < |\phi|$ because $\langle\!\langle\alpha\rangle\!\rangle\phi_2$ contributes one modality to $|\phi|$ that is not part of ϕ_1.

According to Lemma 2.20 there are equivalent FOD-formulas $\phi_1^\#, \phi_2^\#$ with $\vDash \phi_i \leftrightarrow \phi_i^\#$ for $i = 1, 2$. By congruence, the validity $\vDash \phi$ yields $\vDash \phi_1^\# \vee \langle\!\langle\alpha\rangle\!\rangle \phi_2^\#$, which directly implies $\vDash \neg\phi_1^\# \rightarrow \langle\!\langle\alpha\rangle\!\rangle \phi_2^\#$. Then by Proposition 2.23 or 2.24, respectively, we can derive

$$\neg\phi_1^\# \vdash_{\mathcal{D}} \langle\!\langle\alpha\rangle\!\rangle \phi_2^\# \quad . \tag{2.3}$$

Further $\vDash \phi_1 \leftrightarrow \phi_1^\#$ implies $\vDash \neg\phi_1 \rightarrow \neg\phi_1^\#$, which is derivable by IH, because $|\phi_1| < |\phi|$. By Lemma 2.21, we obtain $\neg\phi_1 \vdash_{\mathcal{D}} \neg\phi_1^\#$, which we combine with (2.3) by a cut with $\neg\phi_1^\#$ to

$$\neg\phi_1 \vdash_{\mathcal{D}} \langle\!\langle\alpha\rangle\!\rangle \phi_2^\# \quad . \tag{2.4}$$

Likewise $\vDash \phi_2 \leftrightarrow \phi_2^\#$ implies $\vDash \phi_2^\# \rightarrow \phi_2$, which is derivable by IH, as $|\phi_2| < |\phi|$. We can extend the derivation of $\vdash_{\mathcal{D}} \phi_2^\# \rightarrow \phi_2$ to one of $\vdash_{\mathcal{D}} \forall^\alpha(\phi_2^\# \rightarrow \phi_2)$ by Lemma 2.22 and conclude $\langle\!\langle\alpha\rangle\!\rangle \phi_2^\# \vdash_{\mathcal{D}} \langle\!\langle\alpha\rangle\!\rangle \phi_2$ by G1–G2. Finally we combine the latter propositionally with (2.4) by a cut with $\langle\!\langle\alpha\rangle\!\rangle \phi_2^\#$ to derive $\neg\phi_1 \vdash_{\mathcal{D}} \langle\!\langle\alpha\rangle\!\rangle \phi_2$, from which $\vdash_{\mathcal{D}} \phi_1 \vee \langle\!\langle\alpha\rangle\!\rangle \phi_2$ can be obtained, again using P10, to complete the proof. □

2.8. Relatively Semidecidable Fragments

To strengthen the completeness result from Theorem 2.17, we consider fragments of dℒ where the required FOD tautologies are sufficiently simple as differential equations have first-order definable flows and the required loop invariants (or variants) are expressible in first-order logic over the reals. In these fragments, the only difficulty is to find the required invariants and variants for the proof. Relative to an (ineffective) oracle that provides first-order invariants and variants for repetitions, the dℒ calculus can be used as a semidecision procedure. That is, when we assume the oracle to provide suitable (in)variants, validity of formulas can be proven in the dℒ calculus. If an imperfect oracle chooses inadequate (in)variants, applying the dℒ calculus rules results in goals that are not valid, which is again decidable by quantifier elimination in the dℒ calculus.

2.25 Theorem (Relatively semidecidable fragment). *Relative to an oracle generating first-order invariants and variants, the* dℒ *calculus gives a backtracking-free semidecision procedure for (closed)* dℒ *formulas with differential equations having first-order definable flows.*

Proof Outline. The (constructive) proof, which, in full, can be found in the remainder of this section, shows that there are always applicable dℒ rules that transform the formulas equivalently and that formulas in this dℒ proof descend along a well-founded order. For loops, we assume that suitable (in)variants are obtained from the oracle and we can guarantee termination when these (in)variants are first-order (or contain less loops). □

As a consequence, enumerating first-order invariants or variants gives a semidecision procedure for the fragment of Theorem 2.25. As a corollary to Theorem 2.16 and Theorem 2.25, there are valid dL formulas that need proper dL (or FOD) invariants to be provable and cannot be proven just using (in)variants of first-order real arithmetic. Similarly, the fragment with first-order definable flows and bounded loops is decidable: When loops α^* are decorated with natural numbers indicating the maximum number of repetitions of α, an effective oracle for Theorem 2.25 can be obtained by unrolling, e.g., by D5.

As an auxiliary result for proving Theorem 2.25, we show that, in dL proofs, Skolem symbols occur in a uniform way, i.e., a Skolem symbol s always occurs with the same list of arguments.

2.26 Lemma (Uniform Skolem symbols). *Let ϕ be a dL formula without Skolem symbols. In any derivation of ϕ, Skolem symbols only occur with a unique list of free logical variables as arguments, provided that the formulas in cuts (P10) obey this restriction.*

Proof. The proof is by induction on the structure of proofs in the dL calculus. For derivations of length zero, the conjecture holds, because ϕ does not contain Skolem symbols. We show that the conjectured Skolem occurrence property is preserved in all sub-goals when applying a rule to a goal that satisfies the conjecture.

F1 The symbols $s(X_1, \ldots, X_n)$ introduced by rules F1–F2 are of the required form as the X_i are precisely the free logical variables. In addition, the symbol $s(X_1, \ldots, X_n)$ does not occur nested in other Skolem terms, because, by induction hypothesis, the bound variable x does not occur in Skolem terms of the goal.

F3 Rules F3 and F6 are only applicable to instances of first-order real arithmetic (Lemma 2.13), for which the equivalence transformations of quantifier elimination preserve the Skolem occurrence property, because they never introduce quantifiers to bind free variables.

D11 Rule D11 preserves the property, as it only substitutes state variables $x_i \in \Sigma$ not logical variables $X_i \in V$.

P10 Cuts preserve the Skolem occurrence property, as we assumed the formulas that P10 introduces to adhere to the Skolem occurrence property.

- The other rules of the dL calculus preserve the property as they never replace arguments of Skolem function symbols (which are free variables by induction hypothesis). □

Proof of Theorem 2.25. The proof is by well-founded induction. We prove that there is a well-founded strict partial order $\prec$ such that:

IH: For all non-atomic formulas occurring in the sequents during a proof, there is an applicable series of dℒ rules such that all resulting sub-goals are simpler with respect to $\prec$, have no additional free variables or function symbols, and their conjunction is equivalent to the conclusion (for suitable oracle choices).

By applying these dℒ rules exhaustively, we obtain a decision procedure relative to the oracle, because the sub-goals descend along the well-founded order $\prec$, which has no infinite descending chain. Finally, validity of the remaining sequents with atomic formulas is decidable by evaluating ground instances (Definition 2.10), because, by IH, the resulting formulas have no free variables when the initial formula is closed (open formulas, instead, yield equivalent parameter constraints as results). We use the derived rules G3' and G4' in place of G3 and G4, see Section 2.5.1. To obtain a backtracking-free procedure, we remove rules D5–D6 and G1–G4 and P10 from the calculus: If a calculus with less rules gives a decision procedure, then so does the full calculus.

We define the order $\prec$ as the lexicographical order of, respectively, the numbers of: loops, differential equations, sequential compositions, choices, modalities, quantifiers, number of different variables and Skolem function symbols, and the number of logical connectives. As a lexicographical order of natural numbers, $\prec$ is well-founded [DM79]. It lifts to sequents in rule applications (Definition 2.11) when all sub-goals of all rule schemata are simpler than their goals with respect to $\prec$, which can be shown to retain well-foundedness as a multiset ordering [DM79].

Now the proof of IH is by induction along $\prec$. Let ϕ be a non-atomic formula of a sequent in an open branch of the proof. We assume ϕ to occur in the succedent; the respective proofs for the antecedent are dual. Hence, we consider the sequent to be of the form $\Gamma \vdash \phi, \Delta$.

1. If ϕ is of the form $\psi_1 \wedge \psi_2$, then P5 is applicable, yielding smaller sequents (with less logical connectives) that are equivalent. Other logical connectives are handled likewise using P1–P7, respectively.

2. If ϕ is of the form $[\alpha]\psi$ or $\langle\alpha\rangle\psi$ and α is of the form $?\chi$, $\beta;\gamma$, or $\beta \cup \gamma$ the corresponding rule D1–D4 or D7–D8 is applicable, yielding a simpler yet equivalent formula.

3. If ϕ is of the form $[x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi]\psi$, then D12 is applicable, as we assumed differential equations to have first-order definable flows. The resulting formula is equivalent and simpler, because it contains less differential equations. It involves additional bound variables but not free variables. Case $\langle x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi\rangle\psi$ is similar, by D11.

4. If ϕ is of the form $[\alpha^*]\psi$, then G3' is applicable with a first-order invariant F obtained from the oracle. The resulting sub-goals are simpler according to $\prec$,

because they contain less loops (F does not contain loops). The resulting sub-goals do not have additional free variables as all bound variables of α^* remain bound by the universal closure $\forall^\alpha$ in the respective premisses. Finally, we assume the oracle to give an invariant such that the conjunction of the resulting sub-goals is equivalent to the goal (otherwise we have nothing to show for inadequate choices by the oracle). The case $\langle\alpha^*\rangle\psi$ is similar, using G4' instead.

5. If ϕ is of the form $\langle x_1 := \theta_1, \ldots, x_n := \theta_n\rangle\psi$, there are two cases. If D9 is applicable, it yields equivalent simpler sequents. Otherwise, we have

$$\psi \prec \langle x_1 := \theta_1, \ldots, x_n := \theta_n\rangle\psi$$

Thus, by IH, there is a finite sequence of rule applications on ψ yielding equivalent sequents with atomic formulas. Prefixing the resulting proof with $\langle x_1 := \theta_1, \ldots, x_n := \theta_n\rangle$ yields a corresponding proof for deriving $\Gamma \vdash \phi, \Delta$ by Lemma 2.22. The formulas of the open branches of this proof resulting from ϕ are of the form $\langle x_1 := \theta_1, \ldots, x_n := \theta_n\rangle G$ for atomic formulas G, where, at the latest, D9 is applicable, as substitutions are admissible on atomic formulas. Case $[x_1 := \theta_1, \ldots, x_n := \theta_n]\psi$ is similar, using D10 first.

6. If ϕ is of the form $\forall x\, \psi(x)$, we can apply F1 giving $\psi(s(X_1, \ldots, X_n))$. Now, we have $\psi(s(X_1, \ldots, X_n)) \prec \forall x\, \psi(x)$, hence, by IH, $\psi(s(X_1, \ldots, X_n))$ can be transformed equivalently to a set of sequents of the form

$$\Phi_i(s(X_1, \ldots, X_n)) \vdash \Psi_i(s(X_1, \ldots, X_n))$$

with atomic formulas (without loss of generality, we can assume $s(X_1, \ldots, X_n)$ to occur in all branches). Hence, QE is defined for these atomic formulas and F3 can be applied on each branch, yielding $\mathrm{QE}(\forall s\, (\Phi_i(s) \vdash \Psi_i(s)))$. Consequently, the original sequent $\Gamma \vdash \forall x\, \psi(x), \Delta$ is equivalent to

$$\bigwedge_i \mathrm{QE}(\forall s\, (\Phi_i(s) \vdash \Psi_i(s)))$$

for the following reason: $\Gamma \vdash \psi(s(X_1, \ldots, X_n)), \Delta$ is equivalent to

$$\bigwedge_i (\Phi_i(s(X_1, \ldots, X_n)) \vdash \Psi_i(s(X_1, \ldots, X_n)))$$

by IH, using the equivalence $\mathrm{QE}(\forall s\, (F \wedge G)) \equiv \mathrm{QE}(\forall s\, F) \wedge \mathrm{QE}(\forall s\, G)$ and that s does not occur in Γ, Δ. After applying F3, the result has no additional free symbols, although intermediate formulas do.

7. If ϕ is of the form $\exists x\, \psi(x)$, then F4 is applicable giving $\psi(X)$ for a fresh logical variable X. Then $\psi(X) \prec \exists x\, \psi(x)$, hence, by IH, $\psi(X)$ can be transformed equivalently to a set of sequents $\Phi_i \vdash \Psi_i$ with atomic formulas. If no Skolem dependency on X occurs in $\Phi_i \vdash \Psi_i$, then QE is defined and F6 applicable, giving $\text{QE}(\exists X \bigwedge_i(\Phi_i \vdash \Psi_i))$, which is equivalent to $\exists X \bigwedge_i(\Phi_i \vdash \Psi_i)$. By IH, this is equivalent to $\Gamma \vdash \exists X\, \psi(X), \Delta$, because X does not occur in Γ, Δ. Otherwise, if a Skolem term $s(X_1, \ldots, X, \ldots, X_n)$ occurs in a $\Phi_i \vdash \Psi_i$, then, by IH, the Skolem function s already occurred in $\psi(X)$. By Lemma 2.26, the Skolem term $s(X_1, \ldots, X, \ldots, X_n)$ itself must already have occurred in $\psi(X)$, which contradicts the fact that X is fresh and that bound variable x does not occur in Skolem terms of $\exists x\, \psi(x)$, again by Lemma 2.26. After applying F6 the additional free variable X disappears. □

2.9. Train Control Verification

In this section, we verify collision avoidance of the train control system presented in Section 2.4.

2.9.1. Finding Inductive Candidates

We want to prove safety statement (2.1) of the European Train Control System from Section 2.4. Using parametric extraction techniques, we identify both the requirement ψ for safe driving and the induction hypothesis ϕ that is required for the proof. Dually to the proof in Figure 2.7, an unwinding of the loop in (2.1) by D6 can be used to extract a candidate for a parametric inductive hypothesis. It expresses that there is sufficient braking distance at current speed v, which basically corresponds to the controllability constraint for ETCS:

$$\phi \;\equiv v^2 \leq 2b(m-z) \wedge b > 0 \wedge A \geq 0 \;. \tag{2.5}$$

2.9.2. Inductive Verification

Using G3 to prove (2.1) by induction, we show that (a) invariant ϕ holds initially, i.e., $\psi \vdash \phi$ (implying antecedent of the conclusion of G3), that (b) the invariant is sustained after each execution of $\mathit{ctrl}; \mathit{drive}$, and that (c) invariant ϕ implies postcondition $z \leq m$. Case (c) holds by QE, as $0 \leq v^2 \leq 2b(m-z)$ and $b > 0$. The induction start (a) will be examined after the full proof, since we want to identify the prerequisite ψ for safe driving by proof analysis. In the proof of the induction step $\phi \rightarrow [\mathit{ctrl}; \mathit{drive}]\phi$, we omit condition $m - z \leq s$ from ctrl, because it is not used in the proof (braking remains safe with respect to $z \leq m$). The induction is provable in dℒ as follows (for notational convenience, we assume F1 to

call the Skolem constant for m again m etc., as there are no free logical variables):

$$
\dfrac{\dfrac{\dots}{\phi \vdash \langle a := -b\rangle[\mathit{drive}]\phi} \qquad \dfrac{\dfrac{\dots}{\phi, m - z \geq s \vdash \langle a := A\rangle[\mathit{drive}]\phi}}{\phi \vdash [?m - z \geq s; a := A][\mathit{drive}]\phi}\,\scriptstyle\text{D8,P7}}{\phi \vdash [\mathit{ctrl}][\mathit{drive}]\phi}\,\scriptstyle\text{D4,P5}
$$

$$
\begin{array}{ll}
 & \phi \vdash [\mathit{ctrl}][\mathit{drive}]\phi \\
\hline
\scriptstyle\text{D2} & \phi \vdash [\mathit{ctrl}; \mathit{drive}]\phi \\
\hline
\scriptstyle\text{P7} & \vdash \phi \rightarrow [\mathit{ctrl}; \mathit{drive}]\phi \\
\hline
\scriptstyle\text{F1} & \vdash \forall^{\alpha}(\phi \rightarrow [\mathit{ctrl}; \mathit{drive}]\phi) \\
\hline
\scriptstyle\text{G3} & \phi \vdash [(\mathit{ctrl}; \mathit{drive})^{*}]\phi
\end{array}
$$

The differential equation system in *drive* is linear with a constant coefficient matrix M. Its solution can be obtained by symbolically computing the exponential series $e^{Mt}\eta$ with symbolic initial value $\eta = (z, v)$ and similar symbolic integration of the inhomogeneous part [Wal98, §18.VI]. We abbreviate the solution $\langle z := -\frac{b}{2}t^2 + vt + z, v := -bt + v\rangle$ thus obtained by $\langle \mathcal{S}_t\rangle$. In this example, the invariant evolution conditions are convex, hence the constraint $\forall 0 {\leq} \tilde{t} {\leq} t\, \langle \mathcal{S}_{\tilde{t}}\rangle\chi$ of D12 can be simplified to $\langle \mathcal{S}_t\rangle\chi$ to save space. Further, we leave out conditions which are unnecessary for closing the above proof. In the left branch, the constrained evolution of τ is irrelevant and will be left out. The left branch closes (marked $*$):

$$
\begin{array}{ll}
 & * \\
\hline
\scriptstyle\text{D9,F3} & \phi, t \geq 0, -bt + v \geq 0 \vdash \langle \mathcal{S}_t\rangle\phi \\
\hline
\scriptstyle\text{D9} & \phi, t \geq 0, \langle v := -bt + v\rangle v \geq 0 \vdash \langle \mathcal{S}_t\rangle\phi \\
\hline
\scriptstyle\text{P7,P7} & \phi \vdash t \geq 0 \rightarrow (\langle v := -bt + v\rangle v \geq 0 \rightarrow \langle \mathcal{S}_t\rangle\phi) \\
\hline
\scriptstyle\text{F1} & \phi \vdash \forall t{\geq}0\, (\langle v := -bt + v\rangle v \geq 0 \rightarrow \langle \mathcal{S}_t\rangle\phi) \\
\hline
\scriptstyle\text{D12} & \phi \vdash [z' = v, v' = -b \,\&\, v \geq 0]\phi \\
\hline
\scriptstyle\text{D9} & \phi \vdash \langle a := -b\rangle[\mathit{drive}]\phi \\
\hline
\scriptstyle\text{D10} & \phi \vdash [a := -b][\mathit{drive}]\phi
\end{array}
$$

The right branch does not need $v \geq 0$, because v does not decrease. To abbreviate solution $\langle z := \frac{A}{2}t^2 + vt + z, v := At + v\rangle$, we again use $\langle \mathcal{S}_t\rangle$.

$$
\begin{array}{ll}
 & \dots \\
\hline
 & \phi, m - z \geq s \vdash s \geq \frac{v^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v\right) \\
\hline
\scriptstyle\text{D9,F3} & \phi, m - z \geq s, 0 \leq t \leq \varepsilon \vdash \langle \mathcal{S}_t\rangle\phi \\
\hline
\scriptstyle\text{P7,D9} & \phi, m - z \geq s \vdash t \geq 0 \rightarrow (\langle \tau := t\rangle\tau \leq \varepsilon \rightarrow \langle \mathcal{S}_t\rangle\phi) \\
\hline
\scriptstyle\text{F1} & \phi, m - z \geq s \vdash \forall t{\geq}0\, (\langle \tau := t\rangle\tau \leq \varepsilon \rightarrow \langle \mathcal{S}_t\rangle\phi) \\
\hline
\scriptstyle\text{D9} & \phi, m - z \geq s \vdash \langle \tau := 0\rangle\forall t{\geq}0\, (\langle \tau := t + \tau\rangle\tau \leq \varepsilon \rightarrow \langle \mathcal{S}_t\rangle\phi) \\
\hline
\scriptstyle\text{D12} & \phi, m - z \geq s \vdash \langle \tau := 0\rangle[z' = v, v' = A, \tau' = 1 \,\&\, \tau \leq \varepsilon]\phi \\
\hline
\scriptstyle\text{D10} & \phi, m - z \geq s \vdash [\tau := 0][z' = v, v' = A, \tau' = 1 \,\&\, \tau \leq \varepsilon]\phi \\
\hline
\scriptstyle\text{D9} & \phi, m - z \geq s \vdash \langle a := A\rangle[\tau := 0][z' = v, v' = a, \tau' = 1 \,\&\, \tau \leq \varepsilon]\phi \\
\hline
\scriptstyle\text{D2} & \phi, m - z \geq s \vdash \langle a := A\rangle[\mathit{drive}]\phi \\
\hline
\scriptstyle\text{D10} & \phi, m - z \geq s \vdash [a := A][\mathit{drive}]\phi
\end{array}
$$

2.9.3. Parameter Constraint Discovery

The right branch only closes when the succedent of its open goal is guaranteed. That formula expresses that there will still be sufficient braking distance even after accelerating by $\leq A$ for up to ε seconds:

$$s \geq \frac{v^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v\right) \quad . \tag{2.6}$$

This constraint can be discovered automatically in the above proof by the indicated application of F3 using quantifier elimination with some simplifications. Constraint (2.6) is required to make sure invariant (2.5) still holds after accelerating. In fact, augmenting the case study with (2.6) makes the argument inductive, and the whole proof of the safety statement (2.1) closes when ψ is chosen identical to ϕ. Here, the conditions of ψ cannot be removed without leaving the proof open due to a counterexample, as the invariant (2.5) is a controllability constraint, see Section 2.5.2.

Quite unlike in the acceleration-free case [Pla07b], constraint (2.6) needs to be enforced dynamically as the affected variables change over time. That is, at the beginning of each *ctrl*-cycle, s needs to be updated in accordance with (2.6), which admits complex behaviour like in Figure 2.4b. Further, this constraint can be used to find out how dense a track can be packed with trains in order to maximise ETCS throughput without endangering safety. Using the dℒ calculus, similar constraints can be derived (Section 4.8) to find out how early a train needs to start negotiation in order to minimise the risk of having to reduce speed when the MA is not extendable in time, which is the ST parameter of Figure 2.3.

For the resulting ETCS system, liveness can be proven in the dℒ calculus by showing that the train can pass every point p by an appropriate choice of m by the RBC:

$$z = z_0 \wedge v = v_0 > 0 \wedge \varepsilon > 0 \wedge b > 0 \wedge A \geq 0 \rightarrow \forall p\, \exists m\, \langle (\mathit{ctrl}; \mathit{drive})^* \rangle\, z \geq p \tag{2.7}$$

The proof of property (2.7) uses the variant $z + n\varepsilon v_0 \geq p \wedge v = v_0$ for G4, which expresses that the speed does not decrease (until $n < 0$) and that the remaining distance from z to target p can be covered after at most n iteration cycles. This directly proves the property even when $A = 0$ for appropriate acceleration choices. For $A \geq 0$, the following variant proves property (2.7):

$$\varphi(n) \equiv ((z + n\varepsilon v_0 \geq p \wedge z_0 \leq z \wedge v^2 \leq v_0^2 + 2A(z - z_0) \wedge v \geq v_0 \wedge z \leq p) \vee z \geq p) \wedge v \geq 0$$

It expresses that, when $z \leq p$, the remaining distance can be covered after at most n iterations, while the train position and velocity increase, yet the velocity is bounded depending on the initial velocity v_0, acceleration A, and distance $z - z_0$. The

appropriate choice of m for property (2.7) is

$$m \geq p + \frac{v_o^2 + 2A(p - z_0)}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2\right) + \varepsilon\sqrt{v_0^2 + 2A(p - z_0)}$$

which can be obtained by overapproximating braking condition (2.6) with the speed limit $v^2 \leq v_0^2 + 2A(z - z_o)$ from the variant. We will examine ETCS in more detail in Chapter 7.

In this example, we can see the effect of the $\mathsf{d}\mathcal{L}$ calculus. It takes a specification of a hybrid system and successively identifies constraints on the parameters which are needed for correctness. These constraints can then be handled in a purely modular way by F3 and F6. As a typical characteristics of hybrid systems, further observe that intermediate formulas are significantly more complex than the original proof obligation, which can be expressed succinctly in $\mathsf{d}\mathcal{L}$. This reflects the fact that the actual complexity of hybrid systems originates from hybrid interaction, not from a single transition. Still, using appropriate proof strategies (Chapter 5) for the $\mathsf{d}\mathcal{L}$ calculus, the safety statement (2.1) with invariant (2.5) can be verified automatically in a theorem prover that invokes Mathematica for D11–D12, F3, and F6.

2.10. Summary

We have introduced a first-order dynamic logic for hybrid programs, which are uniform operational models for hybrid systems with interacting discrete jumps and continuous evolutions along differential equations. For this differential dynamic logic, $\mathsf{d}\mathcal{L}$, we have presented a concise generalised free variable proof calculus over the reals.

Our sequent calculus for $\mathsf{d}\mathcal{L}$ is a generalisation of classical calculi for discrete dynamic logic [BHS07, BP06, HKT00, Har79] to the hybrid case. It is a compositional verification calculus for verifying properties of hybrid programs by decomposing them into properties of their parts. In order to handle interacting hybrid dynamics, we lift real quantifier elimination to the deductive calculus in a new modular way that is suitable for automation, using real-valued free variables, Skolem terms, and invertible quantifier rules over the reals.

As a fundamental result aligning hybrid and continuous reasoning proof-theoretically, we have proven our calculus to axiomatise the transition behaviour of hybrid systems completely relative to the handling of differential equations. Moreover, we have demonstrated that our calculus is well-suited for practical automatic verification in a realistic case study of a fully parametric version of the European Train Control System.

Dynamic logic can be augmented [BP06] to support reasoning about dynamically reconfiguring system structures, which we want to extend to hybrid systems in

future work. While the $\mathsf{d}\mathcal{L}$ calculus is complete relative to the continuous fragment, it is a subtle open problem whether a converse calculus can exist that is complete relative to various discrete fragments.

Chapter 3.

Differential-Algebraic Dynamic Logic DAL

Contents

Synopsis

We generalise dynamic logic to a logic for differential-algebraic programs, i.e., discrete programs augmented with first-order differential-algebraic formulas as continuous evolution constraints in addition to first-order discrete jump formulas. These programs characterise interacting discrete and continuous dynamics of hybrid systems elegantly and uniformly. For our logic, we introduce a calculus over real arithmetic with discrete induction and a new *differential induction* with which differential-algebraic programs can be verified by exploiting their differential constraints algebraically without having to solve them. We develop the theory of differential induction and differential refinement and analyse their deductive power. As a case study, we present parametric tangential roundabout maneuvers in air traffic control and prove collision avoidance in our calculus.

3.1. Introduction

Verification of Hybrid Systems

Flight maneuvers in air traffic control [TPS98, LLL00, MF01, DMC05, DPR05, PC07, GMAR07, HKT07] give hybrid systems with challenging dynamics. There the continuous dynamics results from continuous movement of aircraft in space, and the discrete dynamics is caused by the instantaneous switching of maneuvering modes or by discrete aircraft controllers that decide when and how to initiate flight maneuvers. Proper functioning of these systems is highly safety-critical with respect to spatial separation of aircraft during all flight maneuvers, especially collision

avoidance maneuvers. Their analysis, however, is challenging due to the superposition of involved continuous flight dynamics with nontrivial discrete control, causing hybrid systems like these to be neither amenable to mere continuous reasoning nor to verification techniques for purely discrete systems. Since, especially in the presence of parameters, hybrid systems cannot be verified numerically [PC07, CL05], we present a purely symbolic approach using combined deductive and algebraic verification techniques.

In practice, correctness of hybrid systems further depends on the choice of parameters that naturally arise from the degrees of freedom of how a part of the system can be instantiated or how a controller can respond to input [TPS98, DN00, DMO+07, PC07, HKT07]. For instance, correct angular velocities, proper timing, and compatible maneuver points are equally required for safe air traffic control [TPS98, PC07]. Additionally, relevant correctness properties for hybrid systems include safety, liveness, and mixed properties like reactivity (see Chapter 7), all of which can possibly involve (alternating) quantifiers or free variables for parameters. As a uniform approach for specifying and verifying these heterogeneous properties of hybrid systems with symbolic parameters, we introduce an extension of first-order logic and dynamic logic [HKT00] for handling correctness statements about hybrid systems in the presence of (quantified) parameters. These combinations can even be used to discover constraints on the free parameters that are required for system correctness.

Logic for Hybrid Systems

The aim of this chapter is to present logic-based techniques with which general hybrid systems with interacting discrete and continuous dynamics can be specified and verified in a coherent logical framework. To this end, we introduce the *differential-algebraic dynamic logic* (DA-logic or DAL for short) as the logic of general hybrid change. As an elegant and uniform operational model for hybrid systems in DAL, we introduce *differential-algebraic programs* (DA-programs). These programs combine *first-order discrete jump constraints* (DJ-constraints) to characterise discrete transitions with support for *first-order differential-algebraic constraints* (DA-constraints) to characterise continuous transitions. DA-constraints provide a convenient way for expressing continuous system evolution constraints and give a uniform semantics to differential evolutions, systems of differential equations [Wal98], switched systems [Bra95a], invariant constraints [Hen96, DN00], triggers [Bra95a], and differential-algebraic equations [Gea88]. In DJ-constraints and DA-constraints, first-order quantifiers further give a natural and semantically well-founded way of expressing unbounded discrete or continuous nondeterminism in the dynamics, including nondeterminism resulting from internal choices or external disturbances. In interaction with appropriate control structure, DJ-constraints and DA-constraints can be combined to form DA-programs as uniform operational mod-

els for hybrid systems. With this, DA-programs are a generalised program notation for the standard notation of hybrid systems as hybrid automata [Hen96].

As a specification and verification logic for hybrid systems given as DA-programs, we design the first-order dynamic logic DAL. In particular, we generalise discrete dynamic logic [HKT00] to hybrid control and support DA-programs as actions of a first-order multi-modal logic [FM99], such that its modalities can be used to specify and verify correctness properties of hybrid systems. For instance, the DAL formula $[\alpha]\phi$ expresses that all traces of DA-program α lead to states satisfying the DAL formula ϕ. Likewise, $\langle\alpha\rangle\phi$ says that there is at least one state reachable by α which satisfies ϕ. Similarly, $\exists p\,[\alpha]\langle\beta\rangle\phi$ says that there is a choice of parameter p such that for all possible behaviour of DA-program α there is a reaction of DA-program β that ensures ϕ.

Deductive Verification and Differential Induction

As a means for verifying hybrid systems by proving corresponding DAL formulas, we introduce a sequent calculus. It uses side deductions [Pla07b] as a simple and concise, yet constructive, modular technique to integrate real quantifier elimination with calculus rules for modalities. For handling discrete transitions, we present a first-order generalisation of standard calculus rules [HKT00, BP06]. Interacting continuous transitions are more involved. Formulas with very simple differential equations can be verified by using their solutions [Frä99, PAM+05, AW01]: Linear differential equations with nilpotent constant coefficients (i.e., $x' = Ax$ for a matrix A with $A^n = 0$ for some n) have polynomial solutions so that arithmetic formulas about these solutions can be verified by quantifier elimination [CH91]. This approach, however, does not scale to hybrid systems with more sophisticated differential constraints because their solutions do not support quantifier elimination (e.g., when they involve transcendental functions), cannot be given in closed form [Wal98], are not computable [PER79], or do not even exist [Wal98, Kol72]. Solutions of differential equations are much more complicated than the original equations and can become transcendental even for simple linear differential equations like $x' = -y, y' = x$, where the solutions will be trigonometric functions.

Instead, as a logic-based technique for verifying DA-programs with more general differential-algebraic constraints, we introduce *first-order differential induction* as a fully algebraic form of proving logical statements about DA-constraints using their differential-algebraic constraints in a differential induction step instead of using their solutions in a reachability computation. Unlike in discrete induction, the invariant is a *differential invariant*, i.e., a property that is closed under total differentiation with respect to the differential constraints. There, the basic idea for showing invariance of a property F is to show that F holds initially and its total derivative F' holds always along the dynamics (with generalisations of total differentials to logical formulas and corresponding generalisations for quantified

DA-constraints). This analysis considers all non-Zeno executions, i.e., where the system cannot switch its mode infinitely often in finite time. In addition, we introduce *differential strengthening* as a technique for refining the system dynamics by differential invariants until the property becomes provable for the refined dynamics, which we show to be crucial in practical applications.

Comparison

In Chapter 2 we have introduced a logic and calculus for verifying *hybrid programs*, which is the quantifier-free subclass of DA-programs without propositional connectives (see Table 3.1 for examples). Further, we have proven this calculus to be complete *relative to* the handling of differential equations (Theorem 2.17). Complementary, in this chapter, we address the question how sophisticated differential constraints themselves can be specified and verified in a way that lifts to hybrid systems, and how these techniques can be integrated seamlessly into a logic.

To this end, we design differential-algebraic programs as the *first-order completion* of hybrid programs, and we augment both the logic and the calculus with means for handling DA-constraints. In particular, we extend our logic $\mathsf{d}\mathcal{L}$ to the logic DAL with general first-order differential constraints plus first-order jump formulas and introduce differential induction for verifying differential-algebraic programs. Specifically, the continuous evolutions which can be handled by differential induction are strictly more expressive than those that previous calculi [ZRH92, RRS03, DN00] or the $\mathsf{d}\mathcal{L}$ calculus are able to handle. DAL even supports differential-algebraic equations [Gea88]. Consequently, the DAL calculus can verify much more general scenarios, including the dynamics of aircraft maneuvers, which were out of scope for approaches that require polynomial solutions [Frä99, PAM+05]. Table 3.1 summarises the differences in syntactic expressiveness, discrete and continuous verification technology, arithmetic quantifier integration approach, and overall scope of applicability. The DAL extensions presented in this chapter are both complementary to and compatible with our $\mathsf{d}\mathcal{L}$ calculus extensions for integrating arithmetic as presented in Chapter 2.

Contributions

The first contribution of this chapter is the generalised differential-algebraic dynamic logic DAL for differential-algebraic programs as the first-order completion of hybrid programs. DAL provides a uniform semantics and a concise language for specifying and verifying correctness properties of general hybrid systems with sophisticated (possibly quantified) first-order dynamics. The main contribution is a verification calculus for DAL including uniform proof rules for differential induction along first-order differential-algebraic constraints with differential invariants, differential variants, and differential strengthening. Our main theoretical contribution is

Table 3.1.: Comparison of DAL with DA-programs versus dL with hybrid programs

	dL/hybrid programs	DAL/DA-programs
expressive power	single assignments $x := 1$ differential equations $x_1' = d_1, x_2' = d_2$	propositional/quantified DJ-constraints $x > 0 \rightarrow \exists a\,(a < 5 \wedge x := a^2 + 1)$ propositional/quantified DA-constraints $\exists \omega \leq 1\,(d_1' = -\omega d_2 \wedge d_2' = \omega d_1) \vee d_1' \leq d_2' \leq 2d_1$
verification technology	substitutions *polynomial* solutions	quantifier elimination and substitutions first-order differential induction
quantifier integration	real-valued free variables, Skolemisation	side deductions
scope of applications	nilpotent dynamics, e.g., trains in $\mathbb{R}^1$	algebraic dynamics and polynomial differential constraints, e.g., curved aircraft flight

our analysis of the deductive power of differential induction for classes of differential invariants. As an applied contribution, we introduce a generalised tangential roundabout maneuver in air traffic control and we demonstrate the capabilities of our approach by verifying collision avoidance in the DAL calculus. To the best of our knowledge, this is the first formal proof for (unbounded) safety of the hybrid dynamics of an aircraft maneuver with curved flight dynamics and the first sound verification result for collision avoidance with curved aircraft dynamics.

3.1.1. Related Work

Most verification approaches for hybrid systems follow the model checking paradigm for hybrid automata and use approximations or abstraction refinement, e.g., [Hen96, CFH+03, ADG03], because reachability is undecidable for hybrid automata [Hen96]. We have shown in previous work [PC07] that even reachability problems for fairly restricted classes of single continuous transitions are not decidable using numerical computations. Thus, we follow a purely symbolic approach in this thesis. Moreover, we introduce the logic DAL, which gives a more expressive specification and verification language than reachability in model checking. In addition, using quantifiers, DAL is capable of handling quantified parametric properties.

Invariants of Hybrid Systems Several authors [SSM04, RCT05, PJ04, PJP07] argue that invariant techniques scale to more general dynamics than explicit reach-set computations or techniques that require solutions of the differential equations [Frä99, PAM+05, Pla07b, Pla07e]. Among them, there are model checking approaches [SSM04, RCT05] that use equational polynomial invariants based on

Gröbner basis computations. Still, the approach of Rodríguez-Carbonell and Tiwari [RCT05] requires closed-form solutions and is restricted to linear dynamics. The major limitation of these approaches [SSM04, RCT05], however, is that they only work for equational invariants of fully equation-definable hybrid systems, including equational initial sets and switching surfaces. Yet, this assumes highly regular systems without tolerances and only works for null sets. In practice, the set of initial states usually does not have measure zero, though. A thorough analysis of collision avoidance maneuvers, for instance, should consider all initial flight paths in free flight instead of just a single restricted position corridor.

Prajna et al. [PJ04, PJP07] have generalised Lyapunov functions to barrier certificates, i.e., a function B decreasing along the dynamics whose zero set separates initial from unsafe states. Further, they focus on stochastic extensions. DAL provides barrier certificates as a special case using $B \leq 0$ as a differential invariant. In a similar vein, criticality functions [DMO$^+$07] generalise Lyapunov-functions from stability to safety, which DAL provides as a special case of differential invariants.

We generalise purely equational invariants [SSM04, RCT05] and single polynomial expressions [SSM04, PJ04, PJP07, DMO$^+$07] to general differential induction with real arithmetic formulas. In practice, such more general differential invariants are needed for verifying sophisticated hybrid systems including aircraft maneuvers. Further, unlike other approaches [SSM04, RCT05, PJ04, PJP07], DAL leverages the full deductive power of logic, combining differential induction with discrete induction to lift these proof techniques uniformly to hybrid systems. In addition, dynamic logic can be used to prove sophisticated statements involving quantifier and modality alternations for parametric verification [Pla07b]. Finally, the DAL calculus supports combinations with differential variants for liveness properties or combinations with differential strengthening, which we show to be crucial in verifying realistic aircraft maneuvers.

Air Traffic Control Verification In air traffic control, Tomlin et al. [TPS98] analyse competitive aircraft maneuvers game-theoretically using Hamilton-Jacobi-Isaacs partial differential equations. They derive saddle solutions for purely angular or purely linear control actions. They propose roundabout maneuvers and give bounded-time verification results for trapezoidal straight-line approximations. Our symbolic techniques avoid exponential state space discretisations that are required for complicated PDEs and are thus more scalable for automation. Further, we handle fully parametric cases, even for more complicated curved flight dynamics.

Hwang et al. [HKT07] have presented a straight-line aircraft conflict avoidance maneuver that involves optimisation over complicated trigonometric computations, and validate it on random numerical simulation. They show examples where the decisions of the maneuver change only slightly for small perturbations. Hwang et

al. do not, however, prove that their proposed maneuver is safe with respect to actual hybrid flight dynamics.

Dowek et al. [DMC05] and Galdino et al. [GMAR07] consider straight-line maneuvers and formalise geometrical proofs in PVS. Like in the work of Hwang et al. [HKT07], they do not, however, consider curved flight paths nor verify actual hybrid dynamics but work with geometrical meta-level reasoning, instead.

In all these approaches [DMC05, GMAR07, HKT07], it remains to be proven separately that the geometrical meta-level considerations actually fit to the hybrid dynamics and flight equations. In contrast, our approach directly works for the hybrid flight dynamics and we verify roundabout maneuvers with curves instead of straight-line maneuvers with non-flyable instant turns only. A few approaches [DPR05, MF01] have been undertaken to modelcheck discretisations of roundabout maneuvers, which indicate avoidance of orthogonal collisions. However, the counterexamples found by our model checker in previous work [PC07] show for these maneuvers that collision avoidance does not extend to other initial flight paths.

Structure of this Chapter

In Section 3.2 and Section 3.3, we introduce syntax and semantics of the differential-algebraic logic DAL. In Section 3.4, we introduce tangential roundabout maneuvers in air traffic control as a case study and running example. Further, we introduce a sequent calculus with differential induction for DAL in Section 3.5 and prove soundness in Section 3.6. We show extensions of differential induction techniques in Section 3.7. We exploit differential induction techniques for differential monotonicity relaxations in Section 3.8. We prove relative completeness of the DAL calculus in Section 3.9 and compare the deductive strength of differential invariants in Section 3.10. Using the DAL calculus, we prove, in Section 3.11, safety of the tangential roundabout maneuver in air traffic control. Finally, we draw conclusions and discuss future work in Section 3.12.

3.2. Syntax of Differential-Algebraic Logic

In this section, we introduce the *differential-algebraic logic* (DAL) as a specification and verification logic for *differential-algebraic programs* (DA-programs). DA-programs constitute an elegant and uniform model for hybrid systems. We start with an informal introduction that motivates the definitions to come. DA-programs have three basic characteristics:

Discrete jump constraints Discrete transitions, which can possibly lead to discontinuous change, are represented as discrete jump constraints (*DJ-constraints*),

i.e., first-order formulas with instantaneous assignments of values to state variables as additional atomic formulas. DJ-constraints specify what new values the respective state variables assume by an instant change. For instance, $d_1 := -d_2$ specifies that the value of variable d_1 is changed to the value of $-d_2$. Multiple discrete changes can be combined conjunctively with simultaneous effect, for instance $d_1 := -d_2 \wedge d_2 := d_1$, which assigns the previous value of $-d_2$ to d_1 and, simultaneously, the previous value of d_1 to d_2. This operation instantly rotates the vector $d = (d_1, d_2)$ by $\pi/2$ to the left. Using $d := d^\perp$ as a short vectorial notation for this jump, the DJ-constraint $(d_1 > 0 \to d := d^\perp) \wedge (d_1 \leq 0 \to d := -d^\perp)$ specifies that the direction of the rotation depends on the initial value of d_1. Finally, the DJ-constraint $\exists a\,(\omega := a^2 \wedge a < 5)$ assigns the square of some number less than 5 to ω.

Differential-algebraic constraints Continuous dynamics is represented with differential-algebraic constraints (*DA-constraints*) as evolution constraints, i.e., first-order formulas with differential symbols x', e.g., in differential equations or inequalities. DA-constraints specify how state variables change continuously over time. For instance, $x_1' = d_1 \wedge x_2' = d_2$ says that the system continuously evolves by moving the vector $x = (x_1, x_2)$ into direction $d = (d_1, d_2)$ along the differential equation system $(x_1' = d_1, x_2' = d_2)$. Likewise, $d_1' = -\omega d_2 \wedge d_2' = \omega d_1 \wedge d_1 \geq 0$ specifies that the vector d is continuously rotating with angular velocity ω, so that (in conjunction with $x_1' = d_1 \wedge x_2' = d_2$), the direction where point x is heading to changes over time. By adding $d_1 \geq 0$ conjunctively to the DA-constraint, we express that the curving will only be able to continue while $d_1 \geq 0$. This evolution will have to stop before $d_1 < 0$. The evolution is impossible if $d_1 \geq 0$ already fails to hold initially. The DA-constraint $\exists\omega\,(d_1' = -\omega d_2 \wedge d_2' = \omega d_1 \wedge -1 \leq \omega \leq 1)$ characterises rotation with *some* angular velocity $-1 \leq \omega \leq 1$, which may even change over time, in contrast to $d_1' = -\omega d_2 \wedge d_2' = \omega d_1 \wedge -1 \leq \omega \leq 1 \wedge \omega' = 0$ or DA-constraint $d_1' = -\omega d_2 \wedge d_2' = \omega$ where ω is not allowed to change.

Differential-algebraic programs As an operational model for hybrid systems, DJ-constraints and DA-constraints, which represent general discrete and continuous transitions, respectively, can be combined to form a DA-program using regular expression operators $(\cup, ^*, ;)$ of regular discrete dynamic logic [HKT00] as control structure. For example, $\omega := 1 \cup \omega := -1$ describes a controller that can either choose to set angular velocity ω to a left or right curve, by a nondeterministic choice $(\cup)$. Similarly, sequential composition $\omega := \omega + 1;\ d_1' = -\omega d_2 \wedge d_2' = \omega d_1$ says that the system first increases its angular velocity by a discrete transition and then switches to a mode in which it follows a continuous rotation with this angular velocity.

Discussion Not all constraints involving $x := \theta$ or x' qualify as reasonable ways of characterising elementary system transitions. Unlike positive occurrences, negative occurrences of assignments like in $\neg(x := 5)$ are pointless, because they impose no meaningful transition constraints on which new value x actually assumes (but only on which value it is not assigned to). Likewise, negative occurrences of differential constraints as in $\neg(x' = 5)$ would be pointless as they do not constrain the overall evolution but allow arbitrary transitions.

Further, we disallow duplicate constraints that constrain the same variable in incompatible ways at the same time as, e.g., in $x := 2 \wedge x := 3$ or $x' = 2 \wedge x' = 3$. At any state during a system evolution, variable x can only assume one value at a time, not both 2 and 3 at once. Similarly, variables cannot evolve with contradictory slopes at the same time for any positive duration.

Finally, $\forall a\, x := a$ would be equivalent to false, because it is impossible to assign all possible choices for a (hence all reals) simultaneously to x, which can only assume one value at a time. Likewise $\forall a\, x' = a$ would be equivalent to false, because x' can only equal one real value at a time. Dually, $\exists a\, a := \theta$ is equivalent to true, because the DJ-constraint imposes no constraints nor has any visible effects (the scope of the quantified a ends with the DJ-constraint). The situation with $\exists a\, a' = \theta$ is similar.

Even though a semantics and proof rules for these cases can be defined, the respective transitions are degenerate and their technical handling is not very illuminating. Hence, in the sequel, we define DJ-constraints and DA-constraints to avoid these insignificant cases altogether. Note that the syntactical restrictions are non-essential but simplify the presentation by allowing us to focus on the interesting cases.

3.2.1. Terms

To simplify the presentation, we use side deduction rules [Pla07b] for quantifiers in this chapter (the free variable calculus rules from Chapter 2 are still compatible with the findings in this chapter). Consequently, we do not need to distinguish between free logical variables from V and free state variables from Σ. Thus, we do not distinguish Σ and V here.

The formulas of DAL are built over a signature Σ of real-valued function and predicate symbols. The signature Σ contains the usual function and predicate symbols for real arithmetic: $+, -, \cdot, /, =, \leq, <, \geq, >$ and number symbols such as $0, 1$. State variables are represented as real-valued function symbols of arity zero (constants) in Σ. These state variables are *flexible* [BP06], i.e., their interpretation can change from state to state while following the transitions of a DA-program. Observe that there is no need to distinguish between discrete and continuous variables in DAL. The set $\mathrm{Term}(\Sigma, V)$ of *terms* is defined as in classical first-order logic, yielding ra-

tional expressions over the reals. The set of formulas of first-order logic is defined as common, giving first-order real arithmetic.

Although we are primarily interested in polynomial cases, our techniques generalise to the presence of division. Yet to avoid partiality in the semantics, we only allow to use p/q when $q \neq 0$ is present or ensured. Any formula or constraint ϕ containing a term of the form p/q is taken to mean $\phi \wedge \neg(q = 0)$. Note that, in a certain sense, divisions cause less difficulties for the calculus than for the semantics. Particularly, our calculus uses indirect means of differential induction to conclude properties of solutions of DA-constraints, thereby avoiding the need to handle singularities in these solutions explicitly as caused by divisions by zero.

3.2.2. Differential-Algebraic Programs

DA-programs consist of first-order discrete jump formulas and first-order differential-algebraic formulas as primitive operations, which interact using regular control structure.

Reflecting the discussion before Section 3.2, we characterise reasonable occurrences for changes like $x := \theta$ or x' as follows. We call a formula G an *affirmative subformula* of a first-order formula F iff:

1. G is a positive subformula of F, i.e., it occurs with an even number of negations, and

2. no variable y that occurs in G is in the scope of a universal quantifier $\forall y$ of a positive subformula of F (or $\exists y$ of a negative subformula of F).

3.1 Definition (Discrete jump constraint). A *discrete jump constraint (DJ-constraint)* is a formula $\mathcal{J}$ of first-order real arithmetic over Σ with additional atomic formulas of the form $x := \theta$ where $x \in \Sigma$, $\theta \in \mathrm{Term}(\Sigma, V)$. The latter are called assignments and are only allowed in affirmative subformulas of DJ-constraints that are not in the scope of a quantifier for x of $\mathcal{J}$. A DJ-constraint without assignments is called *jump-free*. A variable x is (possibly) *changed* in $\mathcal{J}$ iff an assignment of the form $x := \theta$ occurs in $\mathcal{J}$.

The effect of $(x_1 := \theta_1 \wedge \ldots \wedge x_n := \theta_n \wedge x_1 > 0) \vee (x_1 := \vartheta_1 \wedge \ldots \wedge x_n := \vartheta_n \wedge x_1 < 0)$ is to simultaneously change the interpretations of the variables x_i to the respective θ_i if $x_1 > 0$, and to change the x_i to ϑ_i if, instead, $x_1 < 0$. If neither case applies ($x_1 = 0$), the DJ-constraint evaluates to false as no disjunct applies so that no jump is possible at all, which will prevent the system from continuing any further. In particular, a jump-free DJ-constraint like $x \geq y$ corresponds to a test. It completes without changing the state if, in fact, $x \geq y$ holds true in the current state, and it aborts system evolution otherwise (deadlock). Especially, unlike the assignment $x := \theta$, which changes the value of x to that of θ, the test $x = \theta$ fails by

aborting the system evolution if x does not already happen to have the value θ. If cases overlap, as in $(x := x - 1 \wedge x \geq 0) \vee x := 0$, either disjunct can be chosen to take effect by a nondeterministic choice.

Quantifiers within DJ-constraints express *unbounded discrete nondeterministic choices*. For instance, the following quantified DJ-constraint assigns *some* vector $u \in \mathbb{R}^2$ to e such that the rays spanned by $d = (d_1, d_2)$ and $u = (u_1, u_2)$ intersect:

$$\exists u_1 \exists u_2 \; (e_1 := u_1 \wedge e_2 := u_2 \wedge \exists \lambda{>}0 \, \exists \mu{>}0 \, (\lambda d_1 = \mu u_1 \wedge \lambda d_2 = \mu u_2)) \;\; .$$

We informally use *vectorial notation* when no confusion arises. Using vectorial quantifiers, equations, arithmetic, and assignments, the latter DJ-constraint simplifies to:

$$\exists u \, (e := u \wedge \exists \lambda{>}0 \, \exists \mu{>}0 \, \lambda d = \mu u) \;\; .$$

3.2 Definition (Differential-algebraic constraints). A *differential-algebraic constraint (DA-constraint)* is a formula $\mathcal{D}$ of first-order real arithmetic over $\Sigma \cup \Sigma'$, in which symbols of Σ' only occur in affirmative subformulas that are not in the scope of a quantifier of $\mathcal{D}$ for that symbol. Here Σ' is the set of all *differential symbols* $x^{(n)}$ with $n \in \mathbb{N}$ for state variables $x \in \Sigma$. A DA-constraint without differential symbols is called *non-differential*. A variable x is (possibly) *changed* in $\mathcal{D}$ iff $x^{(n)}$ occurs in $\mathcal{D}$ for an $n \geq 1$.

Syntactically, $x^{(n)}$ is like an ordinary function symbol of arity 0 but only allowed to occur within DA-constraints not in any other formula. The intended semantics of a differential symbol $x^{(n)}$ is to denote the n-th time-derivative of x, which is used to form differential equations (or differential inequalities). We write x' for $x^{(1)}$ and x'' for $x^{(2)}$ and, sometimes, $x^{(0)}$ for the non-differential symbol x. The (partial) *order* $\mathrm{ord}_x \mathcal{D}$ of a DA-constraint $\mathcal{D}$ *in* x is the highest order $n \in \mathbb{N}$ of a differential symbol $x^{(n)}$ occurring in $\mathcal{D}$, or is not defined if no such $x^{(n)}$ occurs. The notion of order is accordingly for terms instead of DA-constraint.

The effect of a DA-constraint $\mathcal{D}$ is an ongoing continuous evolution respecting the differential and non-differential constraints of $\mathcal{D}$ during the whole evolution. For instance, the effect of $(x' = \theta \wedge x > 0) \vee (x' = -x^2 \wedge x < 0)$ is that the system evolves along $x' = \theta$ while $x > 0$, and evolves along $x' = -x^2$ when $x < 0$. This evolution can stop at any time but is never allowed to enter the region where neither case applies anymore ($x = 0$).

More generally, the differential constraints of $\mathcal{D}$ describe how the valuations of the respective state variables change continuously over time while following $\mathcal{D}$. The non-differential constraints of $\mathcal{D}$ can be understood to express domain restrictions or invariant regions of these evolutions for which the differential equations apply or within which the evolution resides. For instance, in the DA-constraint $d_1' = -\omega d_2 \wedge d_2' = \omega d_1 \wedge d_1 \geq 0$, the differential equations $d_1' = -\omega d_2$ and $d_2' = \omega d_1$

describe the change and $d_1 \geq 0$ the invariance region or maximal domain of evolution. Overlapping cases are resolved like in DJ-constraints, i.e., by nondeterministic choice. Likewise, a DA-constraint where no case applies abort the system evolution as it does not satisfy the DA-constraint. Hence, non-differential DA-constraints and jump-free DJ-constraints are both equivalent to pure tests [HKT00]. Except for such tests, we need to distinguish DA-constraints from DJ-constraints: Only DA-constraints can have evolutions of non-zero duration and only DJ-constraints can lead to discontinuous changes.

Quantifiers within DA-constraints express *continuous nondeterministic choices*. For example, constraint $\exists u\,(d_1' = -(\omega + u)d_2 \wedge d_2' = (\omega + u)d_1 \wedge -0.1 \leq u \leq 0.1)$ expresses that the system follows a continuous evolution in which, at each time, the differential equations are respected for *some* choice of u in $-0.1 \leq u \leq 0.1$. In particular, the choice of u can be different at each time so that u amounts to a bounded nondeterministic disturbance during the rotation in the above DA-constraint.

When using constraint formulas to characterise system transitions, we face the usual frame problem: Typically, one does not expect variables to change their values unless the respective constraint explicitly specifies how. In this chapter, we indicate constant variables explicitly so that no confusion arises. In practical applications, however, it can be quite cumbersome to have to specify $z := z$ or $z' = 0$ explicitly for all variables z that are not supposed to change. To account for that, we will define the DAL semantics so that variables that are not changed by a DJ-constraint or DA-constraint keep their value. Since free nondeterministic change of variable y is expressible using $\exists a\, y := a$ or $\exists a\, y' = a$, respectively, we expect the changes of all changed variables to be specified explicitly in all cases of the constraints to improve readability:

3.3 Definition (Homogeneous constraints). A DA-constraint or DJ-constraint $\mathcal{C}$ is called *homogeneous* iff, in each of the disjuncts of a disjunctive normal form of $\mathcal{C}$, every changed variable of $\mathcal{C}$ is changed exactly once.

Note that Lemma 3.17 from Section 3.5.1 will show that DA-constraints are equivalent to their disjunctive normal forms. Throughout the chapter, we assume that all DA-constraints and DJ-constraints are homogeneous, thereby ensuring that all changed variables receive a new value in all cases of the respective constraint (or stay constant because they are changed nowhere in the constraint) and that no change conflicts occur.

Hence, variable y does not change during the DA-constraint $x' = -x \wedge x \geq y$ but works as a constant lower bound for the evolution of x, because no differential symbol $y^{(n)}$ with $n \geq 1$ occurs so that $y' = 0$ is assumed. If, instead, y is intended to vary, yet its variation is not specified by a differential equation but y varies according to some algebraic relation with x, then quantified DA-constraints can be used to represent such *differential-algebraic equations* [Gea88]. For instance,

the differential-algebraic equation $x' = -x, y^2 = x$, in which $y^2 = x$ is an algebraic variational constraint specifying how y changes over time, is expressible as the DA-constraint $x' = -x \wedge \exists u\,(y' = u \wedge y^2 = x)$. There, the quantified differential constraint on y essentially says that y can change arbitrarily (with arbitrary disturbance u) but only so that it always respects the relation $y^2 = x$.

Now we can define DA-programs as regular combinations of DJ-constraints and DA-constraints.

3.4 Definition (Differential-algebraic programs). The set DA-program(Σ, V) of *differential-algebraic programs*, with typical elements α, β, is inductively defined as the smallest set such that:

- If $\mathcal{J}$ is a DJ-constraint over Σ, then $\mathcal{J} \in$ DA-program(Σ, V).
- If $\mathcal{D}$ is a DA-constraint over $\Sigma \cup \Sigma'$, then $\mathcal{D} \in$ DA-program(Σ, V).
- If $\alpha, \beta \in$ DA-program(Σ, V) then $(\alpha \cup \beta) \in$ DA-program(Σ, V).
- If $\alpha, \beta \in$ DA-program(Σ, V) then $(\alpha; \beta) \in$ DA-program(Σ, V).
- If $\alpha \in$ DA-program(Σ, V) then $(\alpha^*) \in$ DA-program(Σ, V).

Choices $\alpha \cup \beta$ are used to express behavioural alternatives between α and β, i.e., the system either follows α or it follows β. In particular, the difference between the DA-constraint $\mathcal{D} \vee \mathcal{E}$ and the DA-program $\mathcal{D} \cup \mathcal{E}$ is that the system has to commit to one choice of $\mathcal{D}$ or $\mathcal{E}$ in $\mathcal{D} \cup \mathcal{E}$, but it can switch back and forth multiple times between $\mathcal{D}$ and $\mathcal{E}$ in $\mathcal{D} \vee \mathcal{E}$. The sequential composition $\alpha; \beta$ says that DA-program β starts executing after α has finished (β never starts if α does not terminate, e.g., due to a failed test in α). Observe that, like repetitions, continuous evolutions within α can take longer or shorter. This nondeterminism is inherent in hybrid systems and as such reflected in DA-programs. Additional restrictions on the permitted duration of evolutions can simply be specified using auxiliary clocks, i.e., variables of derivative $\tau' = 1$. For instance, $\tau := 0;\ x' = -x^2 \wedge \tau' = 1 \wedge \tau \leq 5;\ ?\tau \geq 2$ specifies that the system only follows those evolutions along $x' = -x^2$ that take at most 5 but at least 2 time units. Repetition α^* is used to express that the hybrid process α repeats any number of times, including zero. With this, the repetition of hybrid automata transitions [Hen96] can be represented, see Appendix B for details.

Purely conjunctive DA-constraints correspond to continuous dynamical systems [Sib75]. DA-constraints with disjunctions correspond to switched continuous dynamical systems [Bra95a]. DA-programs without DA-constraints correspond to discrete dynamical systems or, when restricted to domain $\mathbb{N}$ (which is definable in DAL), to discrete while programs [HKT00]. Regular combinations of DJ-constraints form a complete basis of discrete programs [HKT00]. Finally, general DA-programs

correspond to (first-order generalisations of) hybrid dynamical systems [Bra95a, Hen96, DN00].

3.2.3. Formulas of Differential-Algebraic Logic

The set of *formulas* of DAL is defined as common in first-order dynamic logic [HKT00]. They are built using propositional connectives and, in addition, if α is a DA-program and ϕ is a DAL formula, then $[\alpha]\phi, \langle\alpha\rangle\phi$ are DAL formulas. The intuitive reading of $[\alpha]\phi$ is that every run of DA-program α leads to states satisfying ϕ. Dually, $\langle\alpha\rangle\phi$ expresses that there is at least one run of α leading to such a state.

3.5 Definition (DAL formulas). The set $\mathrm{Fml}(\Sigma, V)$ of DAL *formulas*, with typical elements ϕ, ψ, is inductively defined as the smallest set with:

- If $\theta_1, \theta_2 \in \mathrm{Term}(\Sigma, V)$ are terms, then $(\theta_1 \geq \theta_2) \in \mathrm{Fml}(\Sigma, V)$, and accordingly for $=, \leq, <, >$.
- If $\phi, \psi \in \mathrm{Fml}(\Sigma, V)$, then $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi) \in \mathrm{Fml}(\Sigma, V)$.
- If $\phi \in \mathrm{Fml}(\Sigma, V)$ and $\alpha \in \text{DA-program}(\Sigma, V)$, then $[\alpha]\phi, \langle\alpha\rangle\phi \in \mathrm{Fml}(\Sigma, V)$.

Quantifiers in DAL formulas are definable in terms of DA-constraints or quantified DJ-constraints. We consider quantifiers as abbreviations:

$$\begin{aligned} \forall x\, \phi &\equiv [\exists a\, x := a]\phi \equiv [x' = 1 \vee x' = -1]\phi \\ \exists x\, \phi &\equiv \langle \exists a\, x := a\rangle\phi \equiv \langle x' = 1 \vee x' = -1\rangle\phi \ . \end{aligned}$$

The DAL formula $[\exists a\, x := a]\phi$ considers *all* possibilities of assigning *some* value a to x, which amounts to universal quantification. Likewise, $\langle\exists a\, x := a\rangle\phi$ considers some such choice, which is existential quantification. Similarly, the indeterminate continuous evolution $x' = 1 \vee x' = -1$ reaches all values, which amounts to the respective quantifier when combined with the appropriate modality.

One common pattern for representing safety statements about hybrid control loops is to use DAL formulas of the form $\phi \rightarrow [(\mathit{controller}; \mathit{plant})^*]\psi$ for specifying that the system satisfies property ψ whenever the initial state satisfies ϕ. There, the system repeats a controller-plant feedback loop, with a DA-constraint *plant* describing the continuous plant dynamics and a discrete DA-program *controller* describing the control decisions. The controller plant interaction repeats as indicated by the repetition star. Still, more general forms of systems and properties can be formulated and verified in DAL as well.

3.3. Semantics of Differential-Algebraic Logic

The semantics of DAL is a Kripke semantics with possible states of a hybrid system as possible worlds, where the accessibility relation between worlds is generated by the discrete or continuous transitions of DA-programs. A potential behaviour of a hybrid system corresponds to a succession of states that contain the observable values of system variables during its hybrid evolution.

3.3.1. Transition Semantics of Differential-Algebraic Programs

Since, in this chapter, we do not distinguish free logical variables and constants, the semantics does not need to distinguish states and variable assignments.

A *state* is a map $\nu : \Sigma \to \mathbb{R}$; the set of all states is denoted by $\text{State}(\Sigma)$. The function and predicate symbols of real arithmetic are interpreted as usual.

3.6 Definition (Valuation of terms). The *valuation* $val(\nu, \cdot)$ *of terms* with respect to state ν is defined by

1. $val(\nu, x) = \nu(x)$ if $x \in \Sigma$ is a variable.
2. $val(\nu, \theta_1 + \theta_2) = val(\nu, \theta_1) + val(\nu, \theta_2)$ and accordingly for $-, \cdot$.
3. $val(\nu, \theta_1/\theta_2) = val(\nu, \theta_1)/val(\nu, \theta_2)$ if $val(\nu, \theta_2) \neq 0$.

Note that we do not need the semantics of θ_1/θ_2 for $val(\nu, \theta_2) = 0$, because we have assumed the presence of constraints ensuring $\neg(\theta_2 = 0)$ for divisions.

The interpretation of discrete jump constraints is defined as in first-order real arithmetic with the addition of an interpretation for assignment formulas.

3.7 Definition (Interpretation of discrete jump constraints). The *interpretation* of DJ-constraint $\mathcal{J}$ for the pair of states (ν, ω), denoted as $(\nu, \omega) \models \mathcal{J}$, is defined as follows, where $val(\omega, z) = val(\nu, z)$ for all variables z that are not changed in $\mathcal{J}$:

1. $(\nu, \omega) \models x := \theta$ iff $val(\omega, x) = val(\nu, \theta)$.
2. $(\nu, \omega) \models \theta_1 \geq \theta_2$ iff $val(\nu, \theta_1) \geq val(\nu, \theta_2)$, and accordingly for $=, \leq, <, >$.
3. $(\nu, \omega) \models \phi \wedge \psi$ iff $(\nu, \omega) \models \phi$ and $(\nu, \omega) \models \psi$. Accordingly for $\neg, \vee, \rightarrow$.
4. $(\nu, \omega) \models \forall x\, \phi$ iff $(\nu_x, \omega) \models \phi$ for all states ν_x that agree with ν except for the value of x.
5. $(\nu, \omega) \models \exists x\, \phi$ iff $(\nu_x, \omega) \models \phi$ for some state ν_x that agrees with ν except for the value of x.

To give a semantics to DA-constraints, differential symbols $x' \in \Sigma'$ must get a meaning. However, a DA-constraint like $d_1' = -\omega d_2 \wedge d_2' = \omega d_1$ cannot be interpreted in a single state ν, because derivatives are not defined in isolated points. Instead, DA-constraints are constraints that have to hold for an evolution of states over time. Along such a *flow* function $\varphi : [0, r] \to \mathrm{State}(\Sigma)$, DA-constraints can again be interpreted locally by assigning to the formal differential symbol d_1' the analytic time-derivative of the value of d_1 along φ at the respective points in time. As we assumed DA-constraints to avoid zero divisions, analytic derivatives are well-defined for $r > 0$ as $\mathrm{State}(\Sigma)$ is isomorphic to a finite-dimensional real space with respect to the finitely many differential symbols occurring in the DA-constraint. We give a uniform definition for all durations $r \geq 0$ and defer the discussion of the understanding for $r = 0$ until the DA-constraint semantics has been presented in full. The philosophy behind hybrid systems is to isolate discontinuities in discrete transitions. Thus we assume that state variables (and their differential symbols, if present) always vary continuously along continuous evolutions over time.

3.8 Definition (Differential state flow). A function $\varphi : [0, r] \to \mathrm{State}(\Sigma)$ is called *state flow* of duration $r \geq 0$, if φ is componentwise continuous on $[0, r]$, i.e., for all $x \in \Sigma$, $\varphi(\zeta)(x)$ is continuous in ζ. Then, the *differentially augmented state* $\bar{\varphi}(\zeta)$ of φ at $\zeta \in [0, r]$ agrees with $\varphi(\zeta)$ except that it further assigns values to some of the differential symbols $x^{(n)} \in \Sigma'$: If $\varphi(t)(x)$ is n-times continuously differentiable in t at ζ, then $\bar{\varphi}(\zeta)$ assigns the n-th time-derivative $\frac{\mathsf{d}^n \varphi(t)(x)}{\mathsf{d}t^n}(\zeta)$ of x at ζ to differential symbol $x^{(n)} \in \Sigma'$, otherwise the value of $x^{(n)} \in \Sigma'$ is not defined.

For a DA-constraint $\mathcal{D}$, a state flow φ of duration r is called *state flow of the order of* $\mathcal{D}$, iff the value of each differential symbol occurring in $\mathcal{D}$ is defined on $[0, r]$, i.e., $\varphi(\zeta)(x)$ is n-times continuously differentiable in ζ on $[0, r]$ for $n = \mathrm{ord}_x \mathcal{D}$.

3.9 Definition (Interpretation of differential-algebraic formulas). The *interpretation of DA-constraint* $\mathcal{D}$ with respect to a state flow φ of the order of $\mathcal{D}$ and duration $r \geq 0$ is defined by: $\varphi \models \mathcal{D}$ iff, for all $\zeta \in [0, r]$,

1. $\bar{\varphi}(\zeta) \models_{\mathbb{R}} \mathcal{D}$ using the standard semantics $\models_{\mathbb{R}}$ of first-order real arithmetic, and

2. $val(\bar{\varphi}(\zeta), z) = val(\bar{\varphi}(0), z)$ for all variables z that are not changed by $\mathcal{D}$.

Observe that, along the state flows for a DA-constraint $\mathcal{D}$, only those variables whose differential symbols occur in $\mathcal{D}$ have to be continuously differentiable of the appropriate order. Quantified variables can change more arbitrarily (even discontinuously) during the evolution, because the semantics does not directly relate the value of a quantified variable like u in $\exists u\, x' = u^2$ at time ζ with the values that u assumes at later times. Quantified variables may be constrained indirectly by their

relations, though: In $\exists u\, x' = u^2$, the value of u^2 (but not that of u) also varies continuously over time, because x' varies continuously.

As a consequence of Picard-Lindelöf's theorem a.k.a. Cauchy-Lipschitz theorem [Wal98, Theorem 10.VI], and using that DAL terms are continuously differentiable on the open domain where divisors are non-zero, the flows of explicit quantifier-free DA-constraints of the form $x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge \chi$ with non-differential χ are unique (as long as they exist): For each duration and initial value, there is at most one state flow φ (see Lemma 2.8). Yet, this is *not* the case for disjunctive DA-constraints, differential inequalities, quantified DA-constraints, or DA-constraints in implicit form like $x'^2 - 1 = 0$, which has solutions $x(t) = x(0) + t$ and $x(t) = x(0) - t$. Finally, a non-differential χ imposes no change but only tests whether χ holds. Hence, without differential constraints, a non-differential DA-constraint χ itself only has constant flows (if any), i.e., $\varphi(\zeta) = \varphi(0)$ for all ζ.

Restrictions of differential state flows to a prefix are again state flows. In particular, for all differential equations, the restriction to the point interval $[0, 0]$ yields a trivial flow of no effect. For such point duration $r = 0$, however, derivatives and differentiability are not defined. To admit trivial flows nevertheless, the understanding of a DA-constraint is that its differential terms take no effect for flows of zero duration. That is, for trivial flows, atomic formulas with differential symbols are defined to evaluate to *true* as they occur only positively in DA-constraints. Thus, only the non-differential constraints of $\mathcal{D}$ impose constraints for trivial flows. A state flow of duration zero satisfying $\mathcal{D}$ and starting in some state ν exists iff ν satisfies the non-differential part of $\mathcal{D}$, which acts as a test condition.

Now we can define the transition semantics, $\rho(\alpha)$, of a DA-program α. The semantics of a DA-program is captured by the discrete or continuous transitions that are possible by following this DA-program. For DJ-constraints this transition relation holds for pairs of states that satisfy the jump constraints. For DA-constraints, the transition relation holds for pairs of states that can be interconnected by a (continuous) state flow respecting the DA-constraint.

3.10 Definition (Transition semantics of differential-algebraic programs). The *valuation, $\rho(\alpha)$, of a DA-program* α, is a *transition relation* on states. It specifies which state ω is reachable from a state ν by operations of the hybrid system α and is defined as:

1. $(\nu, \omega) \in \rho(\mathcal{J})$ iff $(\nu, \omega) \models \mathcal{J}$ according to Definition 3.7, when $\mathcal{J}$ is a DJ-constraint.
2. $\rho(\mathcal{D}) = \{(\varphi(0), \varphi(r)) \;:\; \varphi \text{ is a state flow of the order of } \mathcal{D} \text{ and some duration } r \geq 0 \text{ such that } \varphi \models \mathcal{D}\}$, when $\mathcal{D}$ is a DA-constraint, see Definition 3.9.
3. $\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$
4. $\rho(\alpha; \beta) = \{(\nu, \omega) : (\nu, z) \in \rho(\alpha), (z, \omega) \in \rho(\beta) \text{ for some state } z\}$

5. $(\nu, \omega) \in \rho(\alpha^*)$ iff there are an $n \in \mathbb{N}$ and $\nu = \nu_0, \ldots, \nu_n = \omega$ such that $(\nu_i, \nu_{i+1}) \in \rho(\alpha)$ for all $0 \leq i < n$.

3.3.2. Valuation of Formulas

Now, the interpretation of formulas is defined as usual for first-order modal logic [FM99, HKT00], with the transition semantics, $\rho(\alpha)$, of DA-programs for modalities.

3.11 Definition (Interpretation of DAL formulas). The *interpretation* $\models$ of DAL formulas with respect to state ν is defined as

1. $\nu \models \theta_1 \geq \theta_2$ iff $val(\nu, \theta_1) \geq val(\nu, \theta_2)$, and accordingly for $=, \leq, <, >$.
2. $\nu \models \phi \wedge \psi$ iff $\nu \models \phi$ and $\nu \models \psi$. For $\neg, \vee, \rightarrow$, the definition is accordingly.
3. $\nu \models [\alpha]\phi$ iff $\omega \models \phi$ for all states ω with $(\nu, \omega) \in \rho(\alpha)$.
4. $\nu \models \langle\alpha\rangle\phi$ iff $\omega \models \phi$ for some state ω with $(\nu, \omega) \in \rho(\alpha)$.

3.3.3. Time Anomalies

Hybrid systems evolve along piecewise continuous trajectories, which consist of a sequence of continuous flows interrupted by discontinuous discrete jumps. A common phenomenon in hybrid system models is that their semantics and analysis is more controversial when discrete and continuous behaviour are allowed to interact without certain regularity assumptions [JSZL01, RRS03, DN00, Hen96]. Zeno-anomalies occur when the hybrid system is allowed to take infinitely many discrete transitions in finite time.

Consider the DA-program $(a' = -1 \wedge d \leq a;\ d := d/2)^*$ starting in a state where $a > d > 0$ and a and d progress towards goal 0. The (inverse) clock variable a decreases continuously, yet d bounds the maximum duration of each continuous evolution phase. At the latest when $a = d$, variable d decreases by a discrete transition. This Zeno system generates infinitely many transitions in finite time and it is impossible for clock a to finally reach 0, because $a \geq d > 0$ will always remain true. Yet this behaviour is, in a certain sense, counterfactual, because it fails to obey divergence of time: Real time diverges, whereas clock a converges to 0. Further, systems with Zeno-anomalies cannot be realised [JSZL01, RRS03, DN00, Hen96] so that corresponding regularity assumptions can be justified for practical purposes.

To avoid pitfalls of time anomalies, we define the DAL semantics so that it only refers to well-defined system behaviour with finitely many transitions in finite time: We restrict the semantics of DA-constraints and disallow infinite numbers of

switches between differential equations in bounded time. With DA-constraint $\mathcal{D}$ defined as $(x \geq 0 \to x'' = -1) \wedge (x < 0 \to x'' = 1) \wedge y' = 1$, the DAL formula

$$\exists e \, \langle \mathcal{D} \rangle [\mathcal{D}](y > e \to x \leq d)$$

expresses that, after some time, the system can stabilise such that it always remains within the region $x \leq d$ when $y > e$ for some choice of e. For such a stability property, we do not analyse what happens *after* there have been infinitely many switches from $x'' = 1$ to $x'' = -1$ within the first second. Instead, our semantics is such that our calculus reveals what happens for *any finite* number of switches. Accordingly, we restrict the semantics of DA-constraints to only accept non-Zeno evolutions:

3.12 Definition. A state flow φ for a DA-constraint $\mathcal{D}$ is called *non-Zeno*, if there only is a finite number of points in time where some variable needs to obey another differential constraint of $\mathcal{D}$ than before the respective point in time: Let $\mathcal{D}_1 \vee \cdots \vee \mathcal{D}_n$ be a disjunctive normal form of $\mathcal{D}$, then flow $\varphi : [0, r] \to \text{State}(\Sigma)$ is non-Zeno iff there are an $m \in \mathbb{N}$ and $0 = \zeta_0 < \zeta_1 < \cdots < \zeta_m = r$ and indices $i_1, \ldots, i_m \in \{1, \ldots, n\}$ such that φ respects $\mathcal{D}_{i_k}$ on the interval $[\zeta_{k-1}, \zeta_k]$, i.e., $\varphi|_{[\zeta_{k-1}, \zeta_k]} \models \mathcal{D}_{i_k}$ for all $k \in \{1, \ldots, m\}$.

The semantics of DA-programs entails that runs with non-Zeno state flows are non-Zeno, because α^* does not accept infinitely many switches.

3.3.4. Conservative Extension

The following result shows that $\mathsf{d}\mathcal{L}$ formulas with hybrid programs can be embedded syntactically into the extension of DAL by DA-programs without changing the meaning of the $\mathsf{d}\mathcal{L}$ formulas. That is, the semantics of DAL formulas given in Definition 3.11 and 3.10 is equivalent to the final state reachability relation semantics given in Definition 2.6 and 2.7 for the sublogic $\mathsf{d}\mathcal{L}$ of dTL, using the syntactic embedding of hybrid programs into DA-programs from Table 3.2.

Table 3.2.: Embedding hybrid programs as DA-programs

Hybrid program	DA-program
assignment / discrete jump set $x_1 := \theta_1, \ldots, x_n := \theta_n$	DA-constraint $x_1 := \theta_1 \wedge \cdots \wedge x_n := \theta_n$
differential equation system $x_1' = \theta_1, \ldots, x_n' = \theta_n \,\&\, \chi$	DA-constraint $x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge \chi$
test $?\chi$	DJ-constraint/ DA-constraint χ

3.13 Proposition (Conservative extension). *The logic* DAL *is a* conservative extension *of* $\mathsf{d}\mathcal{L}$*, i.e., the set of valid* $\mathsf{d}\mathcal{L}$*-formulas is the same with respect to transition semantics of hybrid programs (Definition 2.7) as with respect to the transition semantics of DA-programs (Definition 3.10).*

Proof. The valuation of formulas of $\mathsf{d}\mathcal{L}$ and DAL is directly compatible (Definition 3.11 and 2.6, respectively). By comparing Definition 2.7 with Definition 3.10, it is easy to see that we only need to show that differential equations generate the same transitions. Using vectorial notation, let $x' = \theta \,\&\, \chi$ be a differential equation with invariant region χ. Let $(\nu, \omega) \in \rho(x' = \theta \,\&\, \chi)$ according to a flow φ of duration r as a witness due to Definition 2.7. Then φ is a state flow of the order of $x' = \theta$ and $\varphi(0) = \nu, \varphi(r) = \omega$ and $\varphi \models \chi$ and the value of variables z other than x remains constant. Assume $r > 0$ as there is nothing else to show, otherwise. By Definition 2.7, we know that $\varphi \models x' = \theta$ holds on the interval $(0, r)$ and have to show that there is a continuation of φ so that $\varphi \models x' = \theta$ holds on $[0, r]$.

The right hand side θ of the differential equation assumes values that are defined along φ, because $\varphi \models \chi$ and χ guards against zeros of denominators. Hence, the image of φ remains in the domain of definition of θ. Further, φ is continuous on $[0, r]$, hence, as a compact image of a continuous map, its image is compact. Thus, by the continuation theorem for solutions of differential equations [Wal98, Proposition 6.VI], φ can be continued to a solution of $x' = \theta$ on $[0, r]$.

Conversely, it is easy to see that $(\nu, \omega) \in \rho(x' = \theta \wedge \chi)$ according to Definition 3.10 directly implies $(\nu, \omega) \in \rho(x' = \theta \,\&\, \chi)$ according to Definition 2.7. □

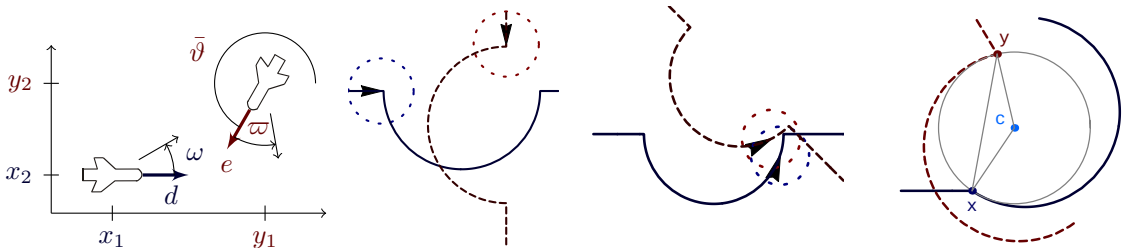
Clearly, the DAL calculus will *not* be a conservative extension of the $\mathsf{d}\mathcal{L}$ calculus, because it contains more powerful rules for verifying properties of differential equations. We will see that there are $\mathsf{d}\mathcal{L}$ formulas that can be proven in the DAL calculus but not in the $\mathsf{d}\mathcal{L}$ calculus.

3.4. Collision Avoidance in Air Traffic Control

As a case study, which will serve as a running example, we show how succinctly collision avoidance maneuvers in air traffic control can be described in DAL. In Section 3.11, we will verify such maneuvers in the DAL calculus.

3.4.1. Flight Dynamics

Assuming, for simplicity, aircraft remain at the same altitude, an aircraft is described by its planar position $x = (x_1, x_2) \in \mathbb{R}^2$ and angular orientation ϑ. The dynamics of an aircraft is determined by its linear velocity $v \in \mathbb{R}$ and angular velocity ω, see Figure 3.1a (with $\vartheta = 0$). When neglecting wind or gravitation, which is appropriate for analysing cooperation in air traffic control [TPS98, LLL00, MF01,



3.1a: Aircraft dynamics 3.1b: Roundabout 3.1c: Counterexample 3.1d: Tangential Roundabout

Figure 3.1.: Roundabout maneuvers for collision avoidance in air traffic control

DPR05, PC07], the in-flight dynamics of an aircraft at x can be described by the following differential equation system, see, e.g., [TPS98] for details:

$$x_1' = v\cos\vartheta \qquad x_2' = v\sin\vartheta \qquad \vartheta' = \omega \ . \tag{3.1}$$

3.4.2. Differential Axiomatisation

Unlike for straight-line flight ($\omega = 0$), such nonlinear dynamics is difficult to analyse [TPS98, LLL00, MF01, DPR05, PC07] for $\omega \neq 0$, especially due to the trigonometric expressions which are generally undecidable. Solving (3.1) already requires the Floquet-theory of differential equations with periodic coefficients [Wal98, Theorem 18.X] and yields mixed polynomial expressions with multiple trigonometric functions. A true challenge, however, is verifying properties of the states that the aircraft reach by following these solutions, which requires proving that complicated formulas with mixed polynomial arithmetic and trigonometric functions hold true for all values of state variables and all possible evolution durations. By Gödel's incompleteness theorem, however, the resulting first-order real arithmetic with trigonometric functions is not semidecidable, because the roots of sin characterise an isomorphic copy of natural numbers.

To obtain polynomial dynamics, we axiomatise the trigonometric functions in the dynamics differentially and reparametrise the state correspondingly. Instead of angular orientation ϑ and linear velocity v, we use the linear speed vector

$$d = (d_1, d_2) := (v\cos\vartheta, v\sin\vartheta) \in \mathbb{R}^2$$

which describes both the linear speed $\|d\| := \sqrt{d_1^2 + d_2^2} = v$ and orientation of the aircraft in space, see Figure 3.1a. Substituting this coordinate change into (3.1), we immediately have $x_1' = d_1$ and $x_2' = d_2$. With the coordinate change, we further obtain differential equations for d_1, d_2 from differential equation system (3.1) by

simple symbolic differentiation:

$$\begin{aligned} d_1' &= v' \cos\vartheta + v(-\sin\vartheta)\vartheta' = -(v\sin\vartheta)\omega = -\omega d_2 \\ d_2' &= v' \sin\vartheta + v(\cos\vartheta)\vartheta' \quad = \quad (v\cos\vartheta)\omega = \quad \omega d_1 \end{aligned}$$

The middle equality holds for constant linear velocity ($v' = 0$), which we assume, because only limited variations in linear speed are possible and cost-effective during the flight [TPS98, LLL00] so that ω is the primary control parameter in air traffic control. Hence, equations (3.1) can be restated as the DA-constraint $\mathcal{F}(\omega)$:

$$x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = -\omega d_2 \wedge d_2' = \omega d_1 \qquad (\mathcal{F}(\omega))$$
$$y_1' = e_1 \wedge y_2' = e_2 \wedge e_1' = -\varpi e_2 \wedge e_2' = \varpi e_1 \qquad (\mathcal{G}(\varpi))$$

DA-constraint $\mathcal{F}(\omega)$ expresses that position x changes according to the linear speed vector d, which in turn rotates according to ω. Simultaneous movement together with a second aircraft at $y \in \mathbb{R}^2$ having linear speed $e \in \mathbb{R}^2$ (also indicated with angle $\bar{\vartheta}$ in Figure 3.1a) and angular velocity ϖ corresponds to the DA-constraint $\mathcal{F}(\omega) \wedge \mathcal{G}(\varpi)$. Such DA-constraints capture simultaneous dynamics of multiple traffic agents succinctly using conjunction. By this differential axiomatisation, we thus obtain polynomial differential equations, even though their solutions still involve the same complicated nonlinear trigonometric expressions. Since the solutions involve trigonometric functions, previous approaches [ZRH92, Hen96, Frä99, RRS03, DN00, PAM+05] were not able to handle such dynamics.

3.4.3. Aircraft Collision Avoidance Maneuvers

Due to possible turbulence or collisions, a flight configuration is unsafe if another aircraft is within a protected zone of radius p, i.e., $\|x - y\|^2 < p^2$. Guiding aircraft by collision avoidance maneuvers to automatically resolve conflicting flight paths that would lead to possible loss of separation, is a major challenge both for air traffic control and verification [TPS98, LLL00, MF01, DMC05, DPR05, PC07, GMAR07, HKT07]. Several different classes of collision avoidance maneuvers for air traffic control have been suggested [TPS98, LLL00, MF01, DMC05, GMAR07, HKT07]. The classical traffic alert and collision avoidance system (TCAS) [LLL00] directs one aircraft on climbing routes the other on descending routes to resolve conflicts at different altitudes but keeps otherwise unmodified straight-line flight paths. While the simplistic TCAS maneuver has several benefits, it does not scale up easily to multiple aircraft or dense traffic situations nearby airports. As a more scalable alternative, Tomlin et al. [TPS98] suggested roundabout maneuvers on circular paths, see Figure 3.1b, where, even at the same altitude, several aircraft can participate in collision avoidance maneuvers. Because the continuous dynamics of curved flights with $\omega \neq 0$ is quite intricate, Tomlin et al. [TPS98] and Massink and De

Francesco [MF01] have analysed trapezoidal straight-line ($\omega = 0$) approximations of roundabouts, instead, which consist only of a series of two to five straight-line segments connected by several instant turns. Unfortunately, the discontinuities in instant turns are not flyable by aircraft.

As a more realistic model, we investigate curved roundabout maneuvers proposed by Tomlin et al. [TPS98]. Roundabouts have proper flight curves with nonzero angular velocities ω (Figure 3.1b). We have shown previously [PC07] that roundabout maneuvers with fixed turns [TPS98, LLL00, MF01, DPR05] are unsafe for non-orthogonal initial flight paths (see Figure 3.1c for a counterexample) and we have proposed a tangential roundabout maneuver [PC07] with position-dependent evasive actions to overcome these deficiencies. However, because of general limits of numerical approximation techniques [PC07, CL05], we could not actually verify the tangential roundabout maneuver numerically. In this chapter, we introduce a generalised class of tangential roundabout maneuvers with curved flight paths and formally verify this maneuver in the purely symbolic DAL calculus. Our main motivation for studying roundabouts are their curved flight paths, which constitute a substantial challenge for verification of hybrid systems with nontrivial dynamics and an important part of realistic flight maneuvers.

3.4.4. Tangential Roundabout Maneuver

In the tangential roundabout maneuver, sketched in Figure 3.1d, the idea is that the aircraft agree on some common angular velocity ω and common centre c around which both can circle safely without coming closer to each other (their linear velocities can differ, though, to compensate for different cruise speeds). Note that neither, c nor ω need to be discovered by complicated online trajectory predictions. Instead, we present in Section 3.11 a simple characterisation of safe choices for the parameters of the tangential roundabout maneuver and determine safety of the resulting flight paths using formal proofs in the DAL calculus.

In Figure 3.2, we introduce the DAL model for the tangential roundabout maneuver, which is a simplified and more uniform generalisation of our previous work [PC07]. Observe how concisely complicated aircraft maneuvers can be specified in DAL. There, safety property ψ for aircraft maneuvers expresses that protected zones are respected during the flight (specified by separation property ϕ). The flight controller (trm^*) performs collision avoidance maneuvers by tangential roundabouts and repeats these maneuvers any number of times as needed. During each *trm* phase, the aircraft first perform arbitrary free flight (*free*) by (repeatedly) independently adjusting their angular velocities ω and ϖ while they are safely separated, which is expressed by conjunct ϕ of the DA-constraint. Observe that, unlike in $\exists u\,(\omega := u);\, \mathcal{F}(\omega)$, angular velocities can be (re-)adjusted continuously during free flight in $\exists \omega\, \mathcal{F}(\omega)$, rather than just once. In particular, *free* includes piecewise constant choices as in $(\exists u\,(\omega := u) \wedge \exists u\,(\varpi := u);\, \mathcal{F}(\omega) \wedge \mathcal{G}(\varpi))^*$. Due

to invariant region ϕ of *free*, the tangential roundabout maneuver must be initiated (by a tangential initiation controller *tang*) before the flight paths become unsafe. Then, the tangential roundabout maneuver itself is carried out by the DA-constraint $\mathcal{F}(\omega) \wedge \mathcal{G}(\omega)$ according to some common angular velocity ω determined by *tang*. Finally, the collision avoidance roundabouts can be left again by repeating the loop trm^* and entering arbitrary free flight at any time. When further conflicts occur during free flight, the controller in Figure 3.2 again enters roundabout conflict resolution maneuvers.

$$\begin{aligned} \psi &\equiv \phi \rightarrow [trm^*]\phi \\ \phi &\equiv \|x-y\|^2 \geq p^2 \equiv (x_1-y_1)^2 + (x_2-y_2)^2 \geq p^2 \\ trm &\equiv free;\quad tang;\quad \mathcal{F}(\omega) \wedge \mathcal{G}(\omega) \\ free &\equiv \exists\omega\, \mathcal{F}(\omega) \wedge \exists\varpi\, \mathcal{G}(\varpi) \wedge \phi \\ tang &\equiv \text{will be derived in Section 3.11} \end{aligned}$$

Figure 3.2.: Flight control with tangential roundabout collision avoidance maneuvers

In summary, property ψ of Figure 3.2 expresses that the aircraft remain safe during the flight, especially during evasive roundabout maneuvers. For the maneuver in Figure 3.2, it is easy to see that ϕ also holds during *free*, because ϕ is specified as an invariant region of *free*. In this chapter, we do not formally investigate temporal properties like "always ϕ", but refer to Chapter 4 for appropriate extensions of our logic. In Section 3.11, we will determine a constraint on the parameter adjustment by *tang* such that the roundabout maneuver is safe, and we give a simple choice for *tang* respecting this parameter constraint.

3.5. Verification Calculus for Differential-Algebraic Logic

In this section, we introduce a sequent calculus for proving DAL formulas. The basic idea is to symbolically compute the effects of DA-programs and successively transform them into simpler logical formulas describing their effects. The calculus consists of standard propositional rules, dedicated rules for handling DA-program-modalities, including differential induction rules for sophisticated differential constraints, and side deduction rules for integrating real quantifier elimination.

For our calculus, recall the definition of substitutions: The result of applying to ϕ the substitution that replaces x by θ is defined as usual; it is denoted by ϕ_x^θ. Likewise, in a simultaneous substitution $\phi_{x_1}^{\theta_1}\ldots_{x_n}^{\theta_n}$ the x_i are replaced simultaneously by the respective θ_i.

3.5.1. Derivations and Differentiation

As a purely algebraic device for proving properties about continuous evolutions in our calculus, we define syntactic derivations of terms and show that their valuation corresponds with analytic differentiation (the *total differential*). With this, we can build proof rules for verifying DA-programs fully algebraically by a differential form of induction without the need to carry out analytic reasoning about analytic limits or similar concepts that would require higher-order logic.

3.14 Definition (Derivation). The map $D : \text{Term}(\Sigma \cup \Sigma', V) \to \text{Term}(\Sigma \cup \Sigma', V)$ that is defined as follows is called *syntactic (total) derivation*

$$D(r) = 0 \qquad \text{if } r \in \mathbb{Q} \text{ is a rational number} \tag{3.2a}$$
$$D(x^{(n)}) = x^{(n+1)} \qquad \text{if } x \in \Sigma \text{ is a state variable}, n \geq 0 \tag{3.2b}$$
$$D(a + b) = D(a) + D(b) \tag{3.2c}$$
$$D(a - b) = D(a) - D(b) \tag{3.2d}$$
$$D(a \cdot b) = D(a) \cdot b + a \cdot D(b) \tag{3.2e}$$
$$D(a/b) = (D(a) \cdot b - a \cdot D(b))/b^2 \tag{3.2f}$$

For a first-order formula F, we define the following abbreviations:

$$D(F) \equiv \bigwedge_{i=1}^{m} D(F_i) \qquad \text{where } \{F_1, \ldots, F_m\} \text{ is the set of all literals of } F$$
$$D(a \geq b) \equiv D(a) \geq D(b) \quad \text{and accordingly for } <, >, \leq, = \text{ or negative literals.}$$

To illustrate the naturalness of this definition, we briefly align it in terms of the structures from differential algebra and refer to [Kol72] for details. Case (3.2a) defines number symbols as differential constants, which do not change during continuous evolution. Equation (3.2c) and the Leibniz rule (3.2e) are defining conditions for derivation operators on rings. Equation (3.2d) is a derived rule for subtraction according to $a - b = a + (-1) \cdot b$. In addition, equation (3.2b) uniquely defines D on the differential polynomial algebra spanned by the differential indeterminates $x \in \Sigma$. Equation (3.2f) canonically extends D to the differential field of quotients. As the base field $\mathbb{R}$ has no zero divisors, the right hand side of (3.2f) is defined whenever the division a/b can be carried out, which, as we assumed, is guarded by $b \neq 0$. The resulting structure $\text{Term}(\Sigma \cup \Sigma', V)$, together with the derivation D, corresponds to the *differential field of rational fractions* with state variables as *differential indeterminates* over $\mathbb{R}$ and with rational numbers as *differential constants*.

The conjunctive definition of the formula $D(F)$ in Definition 3.14 corresponds to the joint total derivative of all atomic subformulas of F and will be an important tool for differential induction rules of our calculus.

The following central lemma, which is the differential counterpart of the substitution lemma, establishes the connection between syntactic derivation of terms and semantic differentiation as an analytic operation to obtain analytic derivatives of valuations along state flows. It will allow us to draw analytic conclusions about the behaviour of a system along differential equations from the truth of purely algebraic formulas obtained by syntactic derivation.

3.15 Lemma (Derivation lemma). *The valuation of* DAL *terms is a* differential homomorphism*: Let $\theta \in \mathrm{Term}(\Sigma, V)$ and let $\varphi : [0,r] \to \mathrm{State}(\Sigma)$ be any state flow of the order of $D(\theta)$ and duration $r > 0$ along which the value of θ is defined (as no divisions by zero occur). Then we have for all $\zeta \in [0,r]$ that*

$$\frac{\mathsf{d}\, val(\varphi(t),\theta)}{\mathsf{d}t}(\zeta) = val(\bar{\varphi}(\zeta), D(\theta)) \ .$$

In particular, $val(\varphi(t),\theta)$ is continuously differentiable (where θ is defined) and its derivative exists on $[0,r]$.

Proof. The proof is an inductive consequence of the correspondence of the semantics of differential symbols and analytic derivatives in state flows (Definition 3.8). It uses the assumption that the flow φ remains within the domain of definition of θ and is continuously differentiable in all variables of θ. In particular, all denominators are non-zero during φ.

- If θ is a variable x, the conjecture holds immediately by Definition 3.8:

  $$\frac{\mathsf{d}\, val(\varphi(t),x)}{\mathsf{d}t}(\zeta) = \frac{\mathsf{d}\, \varphi(t)(x)}{\mathsf{d}t}(\zeta) = \bar{\varphi}(\zeta)(x') = val(\bar{\varphi}(\zeta), D(x)) \ .$$

  There, the derivative exists because the state flow is of order 1 in x and, thus, (continuously) differentiable for x.

- If θ is of the form $a+b$, the desired result can be obtained by using the properties of derivatives, derivations (Definition 3.14), and valuations (Definition 3.6):

  $$\begin{aligned}
  &\frac{\mathsf{d}}{\mathsf{d}t}(val(\varphi(t), a+b))(\zeta) && \\
  &= \frac{\mathsf{d}}{\mathsf{d}t}(val(\varphi(t),a) + val(\varphi(t),b))(\zeta) && val(\nu,\cdot) \text{ homomorph for } + \\
  &= \frac{\mathsf{d}}{\mathsf{d}t}(val(\varphi(t),a))(\zeta) + \frac{\mathsf{d}}{\mathsf{d}t}(val(\varphi(t),b))(\zeta) && \frac{\mathsf{d}}{\mathsf{d}t} \text{ is a (linear) derivation} \\
  &= val(\bar{\varphi}(\zeta), D(a)) + val(\bar{\varphi}(\zeta), D(b)) && \text{by induction hypothesis} \\
  &= val(\bar{\varphi}(\zeta), D(a) + D(b)) && val(\nu,\cdot) \text{ homomorph for } + \\
  &= val(\bar{\varphi}(\zeta), D(a+b)) && D(\cdot) \text{ is a syntactic derivation}
  \end{aligned}$$

- The case where θ is of the form $a \cdot b$ or $a - b$ is accordingly, using Leibniz product rule (3.2e) or subtractiveness (3.2d) of Definition 3.14, respectively.

- The case where θ is of the form a/b uses (3.2f) of Definition 3.14 and further depends on the assumption that $b \neq 0$ along φ. This holds as the value of θ is assumed to be defined all along state flow φ.

- The values of numbers $r \in \mathbb{Q}$ do not change during a state flow (in fact, they are not affected by the state at all), hence their derivative is $D(r) = 0$. □

The *principle of substitution* [FM99] can be lifted to differential equations, i.e., differential equations can be used for equivalent substitutions along state flows respecting the corresponding differential constraints.

3.16 Lemma (Differential substitution principle). *If φ is a state flow satisfying $\varphi \models x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge \chi$, then $\varphi \models \mathcal{D} \leftrightarrow (\chi \rightarrow \mathcal{D}_{x_1'}^{\theta_1} \ldots_{x_n'}^{\theta_n})$ holds for all DA-constraints $\mathcal{D}$.*

Proof. The proof is by using the substitution lemma for first-order logic on the basis of $val(\bar{\varphi}(\zeta), x_i') = val(\bar{\varphi}(\zeta), \theta_i)$ and $\bar{\varphi}(\zeta) \models \chi$ at each time ζ in the domain of φ. □

The following lemma captures that the semantics of DA-constraints is not sensitive to how the DA-constraint is presented. It also plays its part in the soundness proof of our calculus, because it immediately makes all implicational and equivalence transformations of real-arithmetic available for DA-constraints.

3.17 Lemma (Differential transformation principle). *Let $\mathcal{D}$ and $\mathcal{E}$ be DA-constraints (with the same changed variables). If $\mathcal{D} \rightarrow \mathcal{E}$ is a tautology of (non-differential) first-order real arithmetic (that is, when considering $x^{(n)}$ as a new variable independent from x), then $\rho(\mathcal{D}) \subseteq \rho(\mathcal{E})$.*

Proof. Let the first-order formulas ϕ and ψ be obtained from $\mathcal{D}$ and $\mathcal{E}$, respectively, by replacing all x' by new variable symbols X (accordingly for higher-order differential symbols $x^{(n)}$). Using vectorial notation, we write $\phi_X^{x'}$ for the formula obtained from ϕ by substituting all variables X by x'. Thus, $\phi_X^{x'}$ is $\mathcal{D}$ and $\psi_X^{x'}$ is $\mathcal{E}$. Let $\phi \rightarrow \psi$ be valid in (non-differential) real arithmetic. Let $(\nu, \omega) \in \rho(\mathcal{D})$ according to a state flow φ. Then φ also is a state flow for $\mathcal{E}$ that justifies $(\nu, \omega) \in \rho(\mathcal{E})$: For any $\zeta \in [0, r]$, we have $\bar{\varphi}(\zeta) \models \mathcal{D}$ hence $\bar{\varphi}(\zeta) \models \mathcal{E}$, because $\bar{\varphi}(\zeta) \models \phi_X^{x'}$ immediately implies $\bar{\varphi}(\zeta) \models \psi_X^{x'}$ by validity of $\phi \rightarrow \psi$. The assumption on $\mathcal{D}$ and $\mathcal{E}$ having the same set of changed variables is only required for compatibility with condition 2 of Definition 3.9, which enforces that unchanged variables z remain constant. It can be established easily by adding constraints of the form $z' = 0$ as required. □

DA-constraints $\mathcal{D}$ and $\mathcal{E}$ are *equivalent* iff $\rho(\mathcal{D}) = \rho(\mathcal{E})$. In particular, the semantics of DA-programs is preserved when replacing a DA-constraint by another DA-constraint that is equivalent in non-differential first-order real arithmetic (similarly for DJ-constraints).

3.5.2. Differential Reduction and Differential Elimination

Using the expressive power of DA-constraints, several reductions can be performed to simplify the syntactic form of DA-constraints. With quantified DA-constraints, we can reduce differential inequalities to quantified differential equations equivalently:

3.18 Lemma (Differential inequality elimination). *DA-constraints admit* differential inequality elimination*, i.e., to each DA-constraint $\mathcal{D}$, an equivalent DA-constraint without differential inequalities can be effectively associated that has no other free variables.*

Proof. Let $\mathcal{E}$ be obtained from $\mathcal{D}$ by replacing all differential inequalities $\theta_1 \leq \theta_2$ by a quantified differential equation $\exists u\,(\theta_1 = \theta_2 - u \wedge u \geq 0)$ with a new variable u for the quantified disturbance (accordingly for $\geq, >, <$). By Lemma 3.17, the equivalence of $\mathcal{D}$ and $\mathcal{E}$ is a simple consequence of the corresponding equivalences in first-order real arithmetic. □

In the sequel, we assume this transformation has been applied such that we can focus on DA-constraints with differential equations, i.e., where differential symbols only occur in differential equations, and where inequalities do not contain differential symbols. Yet, the DA-constraint resulting from Lemma 3.18 could become inhomogeneous when multiple differential equations are produced for the same variable that result from multiple differential inequalities. For instance, $\theta_1 \leq x' \leq \theta_2$ produces $\exists u\,\exists v\,(x' = \theta_1 + u \wedge x' = \theta_2 - v \wedge u \geq 0 \wedge v \geq 0)$. To rehomogenise this DA-constraint, we use the following:

3.19 Lemma (Differential equation normalisation). *DA-constraints admit* differential equation normalisation*, i.e., to each DA-constraint $\mathcal{D}$, an equivalent DA-constraint with at most one differential equation for each differential symbol can be effectively associated that has no other free variables. Furthermore, this differential equation is* explicit*, i.e., of the form $x^{(n)} = \theta$ where $\operatorname{ord}_x \theta < n$.*

Proof. For each differential symbol $x^{(n)} \in \Sigma'$ occurring in $\mathcal{D}$, we introduce a new non-differential variable $X_n \in \Sigma$. Let $\mathcal{D}^{X_n}_{x^{(n)}}$ denote the result of substituting X_n for $x^{(n)}$ in $\mathcal{D}$. By Lemma 3.17, the equivalence of $\mathcal{D}$ and $\exists X_n\,(x^{(n)} = X_n \wedge \mathcal{D}^{X_n}_{x^{(n)}})$ is a simple consequence of the corresponding equivalence in first-order logic. Proceeding inductively for all such $x^{(n)} \in \Sigma'$ in $\mathcal{D}$ gives the desired result. □

Similarly, higher-order differential constraints reduce to first-order constraints by introducing new non-differential auxiliary variables X_n for each of the higher-order differential symbols $x^{(n)}$. For $1 \leq \operatorname{ord}_x \theta < n$, we can replace a higher-order differential equation $x^{(n)} = \theta$ by:

$$x' = X_1 \wedge X_1' = X_2 \wedge \ldots \wedge X_{n-2}' = X_{n-1} \wedge X_{n-1}' = \theta^{X_1}_{x'} \ldots {}^{X_{n-1}\ X_{n-1}'}_{x^{(n-1)}\ x^{(n)}}$$

3.5.3. Rules of the Calculus for Differential-Algebraic Logic

Sequents and substitutions are defined as in Section 2.5.1. We assume α-conversion for renaming as needed. In the DAL calculus, only admissible substitutions are applicable, which is crucial for soundness.

3.20 Definition (Admissible substitution). An application of a substitution σ is *admissible* if no replaced variable x occurs in the scope of a quantifier or modality binding x or a variable of the replacement $\sigma(x)$. A modality *binds* x if its DA-program (possibly) changes x, i.e., if it contains a DJ-constraint containing $x := \theta$ or a DA-constraint containing $x^{(n)} \in \Sigma'$ for an $n \geq 1$.

As usual in sequent calculus—although the direction of entailment is from *premisses* (above rule bar) to *conclusion* (below)—the order of reasoning and reading is *goal-directed* in practice: Rules are applied in tableau-style, that is, starting from the desired conclusion at the bottom (goal) to the resulting premisses (sub-goals). To highlight the logical essence of the DAL calculus, Figure 3.3 provides rule *schemata* to which the following definition associates the calculus rules that are applicable during a DAL proof. The calculus inherits the propositional P-rules from Figure 2.5 and further consists of first-order quantifier rules (F-rules: F1–F4), rules for dynamic modalities (D-rules: D1–D15), and global rules (G-rules: G1–G6).

The definition of rules is a simplified version of that in Definition 2.11, with side deductions in place of free variable F-rules. Further, we can simplify the presentation by avoiding update prefixes (which would also be sound here when allowing conjunctive DA-constraints as prefixes). Notice that this generally requires more complicated invariants and variants than in the $\mathsf{d}\mathcal{L}$ calculus of Chapter 2, where discrete jump set prefixes are allowed for rule applications so that more information about the prestate is retained automatically.

3.21 Definition (Rules). The *rule schemata* in Figure 3.3 induce *calculus rules* by:

1. If

$$\frac{\Phi_1 \vdash \Psi_1 \quad \ldots \quad \Phi_n \vdash \Psi_n}{\Phi_0 \vdash \Psi_0}$$

is an instance of one of the rule schemata in Figure 3.3, then

$$\frac{\Gamma, \Phi_1 \vdash \Psi_1, \Delta \quad \ldots \quad \Gamma, \Phi_n \vdash \Psi_n, \Delta}{\Gamma, \Phi_0 \vdash \Psi_0, \Delta}$$

can be applied as a proof rule of the DAL calculus, where Γ, Δ are arbitrary finite sets of context formulas (including empty sets).

2. Symmetric schemata can be applied on either side of the sequent: If

$$\frac{\phi_1}{\phi_0}$$

is an instance of one of the symmetric rule schemata D1–D12 in Figure 3.3, then

$$\frac{\Gamma \vdash \phi_1, \Delta}{\Gamma \vdash \phi_0, \Delta} \qquad \text{and} \qquad \frac{\Gamma, \phi_1 \vdash \Delta}{\Gamma, \phi_0 \vdash \Delta}$$

can both be applied as proof rules of the DAL calculus, where Γ, Δ are arbitrary finite sets of context formulas (including empty sets).

P-Rules For propositional logic, we use the standard P-rules from Figure 2.5.

F-Rules Unlike in uninterpreted first-order logic [Fit96, FM99], quantifier rules have to respect the specific semantics of real arithmetic. Thus, our rules handle real quantifiers using quantifier elimination (QE) over the reals [CH91]. Unfortunately, QE is only defined in first-order real arithmetic and cannot handle DAL-modalities, where variables evolve along hybrid trajectories over time. We establish compatibility with dynamic modalities using side deductions for the F-rules, as illustrated in Section 3.5.4. Alternatively, the F-rules can be replaced by the quantifier rules of Chapter 2, which generalise free variables, Skolemisation, and Deskolemisation to real arithmetic for integrating quantifier elimination with modal rules. Instead, here, we use side deductions that we have introduced in previous work [Pla07b] as a very intuitive and simple approach for handling real quantifiers.

D-Rules The D-rules transform a DA-program into simpler logical formulas. Rules D1–D4 are as in discrete dynamic logic [HKT00, BP06] and as in Figure 2.5. D7 and D8 normalise DJ-constraints to their disjunctive normal form such that the jump alternatives can be read off easily. Similarly, D5 and D6 lift quantified choices in DJ-constraints to DAL quantifiers, which are, in turn, handled by F-rules. Then, D9 and D10 use generalised simultaneous substitutions [BP06] for handling discrete change and check the jump-free constraint χ.

$$\text{(F1)}\ \frac{\mathrm{QE}(\forall x\ \bigwedge_i(\Gamma_i \vdash \Delta_i))}{\Gamma \vdash \Delta, \forall x\, \phi} \qquad \text{(F3)}\ \frac{\mathrm{QE}(\exists x\ \bigwedge_i(\Gamma_i \vdash \Delta_i))}{\Gamma \vdash \Delta, \exists x\, \phi}$$

$$\text{(F2)}\ \frac{\mathrm{QE}(\exists x\ \bigwedge_i(\Gamma_i \vdash \Delta_i))}{\Gamma, \forall x\, \phi \vdash \Delta} \qquad \text{(F4)}\ \frac{\mathrm{QE}(\forall x\ \bigwedge_i(\Gamma_i \vdash \Delta_i))}{\Gamma, \exists x\, \phi \vdash \Delta}$$

$$\text{(D1)}\ \frac{\langle\alpha\rangle\langle\beta\rangle\phi}{\langle\alpha;\beta\rangle\phi} \qquad \text{(D5)}\ \frac{\exists x\, \langle\mathcal{J}\rangle\phi}{\langle\exists x\, \mathcal{J}\rangle\phi} \qquad \text{(D9)}\ \frac{\chi \wedge \phi^{\theta_1 \ldots \theta_n}_{x_1 \ldots x_n}}{\langle x_1 := \theta_1 \wedge \ldots \wedge x_n := \theta_n \wedge \chi\rangle\phi}$$

$$\text{(D2)}\ \frac{[\alpha][\beta]\phi}{[\alpha;\beta]\phi} \qquad \text{(D6)}\ \frac{\forall x\, [\mathcal{J}]\phi}{[\exists x\, \mathcal{J}]\phi} \qquad \text{(D10)}\ \frac{\chi \rightarrow \phi^{\theta_1 \ldots \theta_n}_{x_1 \ldots x_n}}{[x_1 := \theta_1 \wedge \ldots \wedge x_n := \theta_n \wedge \chi]\phi}$$

$$\text{(D3)}\ \frac{\langle\alpha\rangle\phi \vee \langle\beta\rangle\phi}{\langle\alpha \cup \beta\rangle\phi} \quad \text{(D7)}\ \frac{\langle\mathcal{J}_1 \cup \ldots \cup \mathcal{J}_n\rangle\phi}{\langle\mathcal{J}\rangle\phi} \quad \text{(D11)}\ \frac{\langle(\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_n)^*\rangle\phi}{\langle\mathcal{D}\rangle\phi}$$

$$\text{(D4)}\ \frac{[\alpha]\phi \wedge [\beta]\phi}{[\alpha \cup \beta]\phi} \quad \text{(D8)}\ \frac{[\mathcal{J}_1 \cup \ldots \cup \mathcal{J}_n]\phi}{[\mathcal{J}]\phi} \quad \text{(D12)}\ \frac{[(\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_n)^*]\phi}{[\mathcal{D}]\phi}$$

$$\text{(D13)}\ \frac{\vdash [\mathcal{E}]\phi}{\vdash [\mathcal{D}]\phi} \qquad \text{(D14)}\ \frac{\vdash \langle\mathcal{D}\rangle\phi}{\vdash \langle\mathcal{E}\rangle\phi} \qquad \text{(D15)}\ \frac{\vdash [\mathcal{D}]\chi \quad \vdash [\mathcal{D} \wedge \chi]\phi}{\vdash [\mathcal{D}]\phi}$$

$$\text{(G1)}\ \frac{\vdash \forall^\alpha(\phi \rightarrow \psi)}{[\alpha]\phi \vdash [\alpha]\psi} \quad \text{(G2)}\ \frac{\vdash \forall^\alpha(\phi \rightarrow \psi)}{\langle\alpha\rangle\phi \vdash \langle\alpha\rangle\psi} \quad \text{(G3)}\ \frac{\vdash \forall^\alpha(\phi \rightarrow [\alpha]\phi)}{\phi \vdash [\alpha^*]\phi}$$

$$\text{(G4)}\ \frac{\vdash \forall^\alpha(\varphi(x) \rightarrow \langle\alpha\rangle\varphi(x-1))}{\exists v\, \varphi(v) \vdash \langle\alpha^*\rangle\exists v{\leq}0\, \varphi(v)}$$

$$\text{(G5)}\ \frac{\vdash \forall^\alpha \forall y_1 \ldots \forall y_k\, (\chi \rightarrow F'^{\theta_1 \ldots \theta_n}_{x'_1 \ldots x'_n})}{[\exists y_1 \ldots \exists y_k\, \chi]F \vdash [\exists y_1 \ldots \exists y_k\, (x'_1 = \theta_1 \wedge \ldots \wedge x'_n = \theta_n \wedge \chi)]F}$$

$$\text{(G6)}\ \frac{\vdash \exists\varepsilon{>}0\, \forall^\alpha \forall y_1 \ldots y_k\, (\neg F \wedge \chi \rightarrow (F' \geq \varepsilon)^{\theta_1 \ldots \theta_n}_{x'_1 \ldots x'_n})}{[\exists y_1 \ldots y_k\, (x'_1 = \theta_1 \wedge \ldots \wedge x'_n = \theta_n \wedge {\sim}F)]\chi \vdash \langle\exists y_1 \ldots y_k\, (x'_1 = \theta_1 \wedge \ldots \wedge x'_n = \theta_n \wedge \chi)\rangle F}$$

In all rule schemata, *all* substitutions need to be admissible. In D7–D8, $\mathcal{J}_1 \vee \cdots \vee \mathcal{J}_n$ is a disjunctive normal form of the DJ-constraint $\mathcal{J}$. In D11–D12, $\mathcal{D}_1 \vee \cdots \vee \mathcal{D}_n$ is a disjunctive normal form of the DA-constraint $\mathcal{D}$. The rules D9–D10 and G5–G6 can be applied for any reordering of the conjuncts of the DA-constraint or DJ-constraint, where χ is non-differential or jump-free, respectively. In D13 and D14, $\mathcal{D}$ implies $\mathcal{E}$, i.e., satisfies the assumptions of Lemma 3.17. In G5–G6, F is first-order without negative equalities, and F' abbreviates $D(F)$, with z' replaced by 0 for unchanged variables. In G6, F does not contain equalities and the differential equations are Lipschitz-continuous. For F-rules, the $\Gamma_i \vdash \Delta_i$ are obtained from the resulting sub-goals of a side deduction, see ($\star$) in Figure 3.4. The side deduction is started from the goal $\Gamma \vdash \Delta, \phi$ at the bottom (or $\Gamma, \phi \vdash \Delta$ for F2 and F4), where x is assumed not to occur in Γ, Δ using renaming. In the resulting sub-goals $\Gamma_i \vdash \Delta_i$, variable x is assumed to occur in first-order formulas only, as quantifier elimination (QE) is then applicable.

Figure 3.3.: Rule schemata of the DAL proof calculus

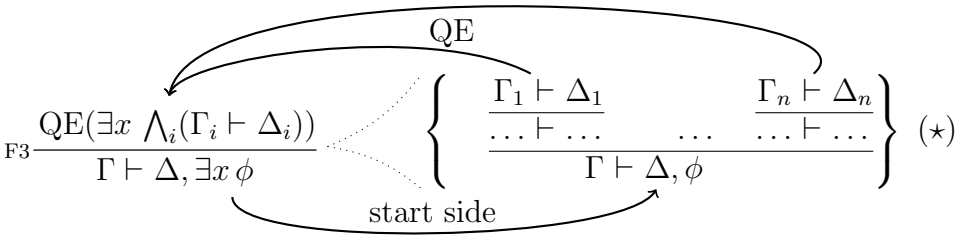


Figure 3.4.: Side deduction for quantifier elimination rules

Likewise, D11–D12 normalise DA-constraints to a form where their differential evolution alternatives are readily identifiable. Unlike for D7–D8, however, continuous evolutions take time so that the system can switch back and forth between the various cases of the DA-constraint, hence the repetition. Observe that finitely many repetitions are sufficient for non-Zeno flows (Definition 3.12), which can only switch finitely often in finite time.

Rules D13–D15 are weakening and strengthening rules for DA-constraints, respectively. In D13–D14, $\mathcal{D}$ implies $\mathcal{E}$ in real arithmetic according to Lemma 3.17, which is easy to decide by QE in practice. Note that D13–D14, are sound for *any* such combination of $\mathcal{D}$ and $\mathcal{E}$. Their primary practical purpose is to use D13 for overapproximating individual variable evolutions and D14 for refining nondeterministic variable evolutions to specific differential equations (*differential refinement*). In particular, we use D13 to project conjunctive differential constraints $\mathcal{D}$ to their non-differential constraints. As we illustrate in Section 3.11, this gives a powerful verification technique in combination with strengthening (D15), which allows to refine the system dynamics by auxiliary constraints. We address the problem of automatically determining the respective strengthenings χ in Chapter 6, where we derive automatic verification algorithms from the results presented in this chapter. Furthermore, D13–D14 make all equivalence transformations on DA-constraints from Section 3.5.2 available as proof rules, including index reduction techniques for differential-algebraic equations [Gea88].

Note that DAL does not need rules for handling negation in DA-constraints or DJ-constraints, as—possibly after applying D7–D8 or D11–D12, respectively—negations only occur in jump-free or non-differential parts, because assignments and differential symbols only occur positively by Definition 3.1 and 3.2. Similarly, no rules for universal quantifiers within DA-constraints or DJ-constraints are needed. Like other propositional operators or quantifiers, negation and universal quantifiers are allowed without restriction in non-differential or jump-free χ and are then handled by D9–D10 or G5–G6 as usual.

G-Rules The G-rules are global rules. They depend on the truth of their premisses in all states, which is ensured by the universal closure with respect to all bound variables of the respective DA-program α (see Definition 3.20). G1–G4 are as in Figure 2.5.

G5 is a rule for *differential induction*, which is a continuous form of induction along differential constraints. The induction rules G3 and G5 (or G4 and G6 respectively) differ in the way the invariant is sustained. G3 uses the inductive nature of repetition. G5, instead, uses continuity of evolution and the differential equation for a continuous induction step with the *differential invariant* F: If F holds initially (antecedent of conclusion) and its total differential F' satisfies the same relations when taking into account the differential constraints (premiss), then F itself is sustained differentially (succedent of conclusion). Formula F' abbreviates $D(F)$ with z' replaced by 0 for all variables z that are unchanged by the DA-constraint, i.e., that are distinct from $\{x_1, \ldots, x_n\}$, because these are assumed constant in the semantics. By α-renaming, the y_i do not occur in F. Rule G6 is a *differential variant* rule where the variant F is attained differentially (with some minimal progress ε), rather than sustained as in G5. Differential induction, the requirement of the differential equations for G6 to be Lipschitz-continuous, and the notations $F' \geq \varepsilon$ and $\sim F$ will be illustrated in more detail in Sections 3.5.5–3.5.6 after side deductions for quantifiers have been explained in Section 3.5.4. Finally, G-rules can be combined with generalisation (G1–G2) to strengthen postconditions as needed.

Provability can be defined as a simplified version of Definition 2.12, because all DAL rules have only one conclusion, so that a proof will be inductively defined as a tree, not an acyclic graph.

3.22 Definition (Provability). A formula ψ is *provable* from a set Φ of formulas, denoted by $\Phi \vdash_{\text{DAL}} \psi$ iff there is a finite set $\Phi_0 \subseteq \Phi$ for which the sequent $\Phi_0 \vdash \psi$ is derivable. Derivability is inductively defined so that a sequent $\Phi \vdash \Psi$ is *derivable* iff there is a proof rule of the DAL calculus (Definition 3.21) with conclusion $\Phi \vdash \Psi$ such that all premisses of the rule are derivable.

3.5.4. Deduction Modulo by Side Deduction

The F-rules constitute a purely modular interface to mathematical reasoning. They can use any theory that admits quantifier elimination and has a decidable ground theory, e.g., the theory of first-order real arithmetic, which is equivalent to the theory of real-closed fields [CH91]. Unlike in deduction modulo approaches of Dowek et al. [DHK03] and Tinelli [Tin03], the information given to the background prover is not restricted to ground formulas [Tin03] or atomic formulas [DHK03], and the effect of modalities has to be taken into account.

Integrating quantifier elimination to deal with statements about real quantities is quite challenging in the presence of modalities that influence the value of variables and terms. Real quantifier elimination cannot be applied to formulas with mixed quantifiers and modalities like $\exists x\, [x' = -x; x := 2x]x \leq 5$. To find out which first-order constraints are actually imposed on x by this DAL formula, we have to take into account how x evolves from $\exists x$ to $x \leq 5$ along the hybrid system dynamics.

Hence, our calculus first unveils the first-order constraints on x before applying QE. To achieve this in a concise and simple way, we use side deductions that we have introduced in previous work [Pla07b].

The effect of a side deduction is as follows. First, the DAL calculus discovers all relevant first-order constraints from modal formulas using a side deduction. Secondly, these constraints are reimported into the main proof and equivalently reduced using QE and the main proof continues. For instance, an application of F3 to a sequent $\Gamma \vdash \Delta, \exists x\, \phi$ starts a side deduction (marked ($\star$) in Figure 3.4) with the unquantified kernel $\Gamma \vdash \Delta, \phi$ as a goal at the bottom. This side deduction is carried out in the DAL calculus until x no longer occurs within modal formulas of the remaining open branches $\Gamma_i \vdash \Delta_i$ of ($\star$). Once all occurrences of x are in first-order formulas, the resulting sub-goals $\Gamma_i \vdash \Delta_i$ of ($\star$) are copied back to the main proof and QE is applied to eliminate x altogether (which determines the resulting subgoal of rule F3 on the upper left side of Figure 3.4). The remaining modal formulas not containing x can be considered as atoms for this purpose, as they do not impose constraints on x. Formally, this can be proven using the coincidence lemma 2.14. When several quantifiers are nested, side deductions will be nested in a cascade, as they can again spawn further side deductions. According to the applicability conditions of F-rules, inner nested side deductions need to be completed by QE before outer deductions continue. For instance, further side deductions started within ($\star$) of Figure 3.4 will be completed before ($\star$) continues and the quantifier elimination result of ($\star$) is returned to the main F3 application.

3.1 Example (Aircraft progress). To illustrate how our calculus combines arithmetic with dynamic reasoning using side deductions, we look at an aircraft example. Using the notation from Section 3.4 where $\mathcal{F}(\omega)$ denotes the flight equation with angular velocity ω, the following DAL formula expresses a simple progress property about aircraft: The aircraft at x can finally fly beyond any point $p \in \mathbb{R}^2$ by adjusting its speed vector d appropriately, using only speed vectors $d \in \mathbb{R}^2$ of bounded speed $\|d\| \leq b$, i.e., $\|d\|^2 \leq b^2 \equiv d_1^2 + d_2^2 \leq b^2$:

$$\forall p\, \exists d\, (\|d\|^2 \leq b^2 \wedge \langle \mathcal{F}(0) \rangle (x_1 \geq p_1 \wedge x_2 \geq p_2)) \ . \tag{3.3}$$

There, point p is constant during the evolution, i.e., $p_1' = p_2' = 0$ and $b' = 0$. The DAL proof in Figure 3.5 proves this property using nested side deductions for nested quantifiers and differential variant induction G6. Applying F1 in the main branch yields a side deduction for $\forall p$, which, in turn, yields another side deduction by applying F3 for the nested quantifier $\exists d$. These nested side deductions in Figure 3.5 are inlined and indicated by indenting the side deductions, with arrows pointing to the start of the respective inner side deduction and back to the continuation of the outer deduction (marked with QE as in Figure 3.4). The two branches for the side deduction for F3 recombine conjunctively and, after quantifiers are re-added, quantifier elimination yields $b > 0$, which reveals the parameter constraint on the

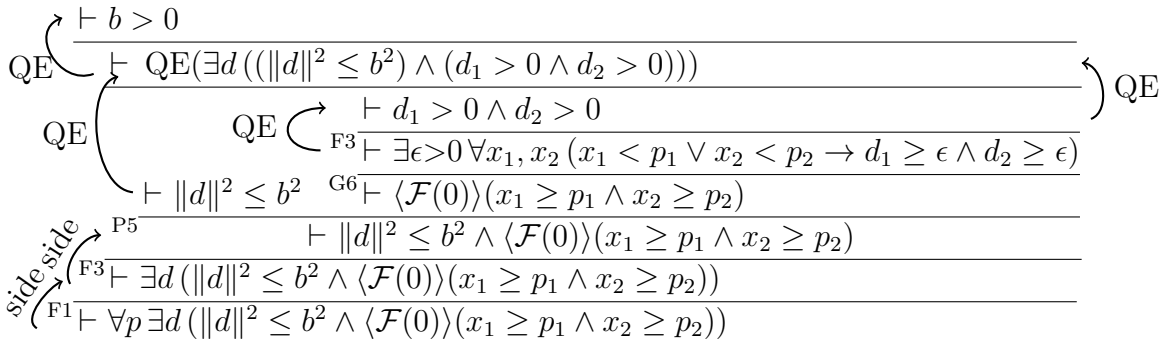


Figure 3.5.: Nested side deductions and differential variants for progress property

speed bound b. Consequently, property (3.3) holds true and the proof closes for all non-zero speed bounds. The right branch of this F3 side deduction uses differential variant induction G6, as will be illustrated in Section 3.5.6. There, the quantifiers for x_1, x_2 result from the universal closure $\forall^\alpha$ in G6. The subsequent innermost F3 side deduction can be abbreviated by directly applying QE, because the affected formula already is first-order.

Like the other aircraft examples in this chapter, formula (3.3) is provable in our theorem prover [PQ08a] within a few seconds, despite the complicated underlying aircraft dynamics.

3.5.5. Differential Induction with Differential Invariants

The purpose of G5 and G6 is to prove properties about continuous evolutions by differential induction using differential invariants or differential variants, respectively. They directly work with the differential constraints instead of complicated (possibly undecidable) arithmetic of their solutions. Unlike approaches using solutions [Frä99, PAM$^+$05, Pla07b, Pla07e, Pla08b], differential induction can even be used to verify systems with nondeterministic quantified input, which would otherwise cause quantified higher-order functions for the time-dependent input of the solutions. Further, unlike in discrete induction, these differential induction rules exploit continuity of evolution and knowledge of the differential constraints for a continuous induction step. We demonstrate the capabilities and the necessity of the requirements of differential induction rules in a series of examples and counterexamples.

Rule G5 uses differential induction to prove that F is a *differential invariant*, i.e., F is closed under total differentiation (Definition 3.14) relative to the differential constraints. For this, the premiss of G5 shows that the total differential F'—i.e., $D(F)$ with z' replaced by 0 for unchanged variables z— holds within invariant region χ, when substituting the differential equations

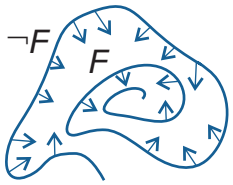


Figure 3.6.: Differential invariants

into F'. Now, if F holds initially (antecedent of conclusion), then F itself is sustained (succedent of conclusion). Intuitively, the premiss expresses that, within χ, the total derivative F' along the differential constraints is pointing inwards or transversal to F but never outwards to $\neg F$, see Figure 3.6. At this point, it is important to note that, even though meta-proofs about DAL involve analytic reasoning, proofs within the DAL calculus are fully algebraic, including the handling of differential constraints by G5. Further observe that the premiss of G5 is a well-formed DAL formula, because all differential symbols are replaced by non-differential terms when forming $F'^{\theta_1}_{x_1'}\cdots^{\theta_n}_{x_n'}$.

3.2 Example (Quartic dynamics). As a first simple example, consider the differential equation $x' = x^4$. It is not so easy to see the solution of this differential equation. Still, with implicit means of differential induction, we can establish easily that the solution always stays above $\frac{1}{4}$ whenever the dynamics initially starts above $\frac{1}{4}$:

$$\text{G5}\,\frac{\text{F1}\,\dfrac{*}{\vdash \forall x\,(x^4 \geq 0)}}{x \geq \frac{1}{4} \vdash [x' = x^4]x \geq \frac{1}{4}}$$

This deduction proves invariance of $x \geq \frac{1}{4}$ along the differential equation $x' = x^4$ by differential induction and without having to solve the differential equation. To apply the differential induction rule G5, we form the total derivative of the differential invariant $F \equiv x \geq \frac{1}{4}$, which gives the differential expression $F' \equiv D(x \geq \frac{1}{4}) \equiv x' \geq 0$. Now, the differential induction rule G5 takes into account that the derivative of state variable x along the dynamics is known (the trick, of course, is to show why this intuitive reasoning is sound, which we will prove in Section 3.6). Substituting the differential equation $x' = x^4$ into the inequality above yields $F'^{x^4}_{x'} \equiv x^4 \geq 0$, which is a valid formula and closes by quantifier elimination with F1. Observe how elegantly differential induction establishes the desired result indirectly by working with the differential equation itself in an algebraic way instead of requiring its solution.

Even more so, for the differential equations $x' = x^2 + x^4$ or $x' = x^2 - 4x + 6$, solutions are hard to obtain both symbolically and numerically. With differential induction, however, we directly establish the following result about their dynamics:

$$\text{G5}\,\frac{\text{F1}\,\dfrac{*}{\vdash \forall x\,(x^2 + x^4 \geq 0)}}{x \geq \frac{1}{4} \vdash [x' = x^2 + x^4]x \geq \frac{1}{4}}$$

3.3 Example (Linear versus angular speed). Consider the following simple proof, which shows that the speed v of an aircraft at x is maintained, even when it changes its angular velocity ω nondeterministically during the flight (as in mode

free of Figure 3.2). Again, $\mathcal{F}(\omega)$ is the flight equation with angular velocity ω.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \mathrm{QE}(\forall x_1, x_2\, \forall d_1, d_2\, \forall \omega\, (2d_1(-\omega d_2) + 2d_2\omega d_1 = 0))}\,\mathrm{QE}}{\vdash \forall x_1, x_2\, \forall d_1, d_2\, \forall \omega\, (2d_1(-\omega d_2) + 2d_2\omega d_1 = 0)}\,\mathrm{F1}}{d_1^2 + d_2^2 = v^2 \vdash [\exists \omega\, \mathcal{F}(\omega)]\, d_1^2 + d_2^2 = v^2}\,\mathrm{G5}}{\vdash d_1^2 + d_2^2 = v^2 \to [\exists \omega\, \mathcal{F}(\omega)]\, d_1^2 + d_2^2 = v^2}\,\mathrm{P7}}{\vdash \forall v\, (d_1^2 + d_2^2 = v^2 \to [\exists \omega\, \mathcal{F}(\omega)]\, d_1^2 + d_2^2 = v^2)}\,\mathrm{F1}$$

The total derivative of the property $F \equiv d_1^2 + d_2^2 = v^2$ for differential induction with G5 is $F' \equiv D(d_1^2 + d_2^2 = v^2) \equiv 2d_1d_1' + 2d_2d_2' = 2vv'$. Substituting the differential equations $\mathcal{F}(\omega)$ of flight yields $F'^{\,-\omega d_2\ \omega d_1\ 0}_{\ d_1'\ \ d_2'\ \ v'} \equiv 2d_1(-\omega d_2) + 2d_2\omega d_1 = 0$, which is valid and closes by quantifier elimination. This example shows the difference of differential continuous evolution (of d_1, d_2) and nondeterministic continuous evolution (of ω). The DA-constraint specifies how the d_i evolve along differential equations, hence d_i' is substituted in F'. For ω, instead, the DA-constraint is nondeterministic ($\exists\omega$) and does not specify how ω changes precisely. In particular, there is no equation for ω' that could be used for substition. Yet such an equation is not even needed for forming the premiss of G5, because, after α-renaming, ω cannot occur in F here, since the scope of $\exists\omega$ ends with the DA-constraint and does not extend to postcondition F. In the proof, the quantifiers for x_i and d_i result from the universal closure $\forall^\alpha$ in G5. The quantifier for ω is introduced by G6 and ensures that *all* possible evolutions of ω are taken into account as there is no specific equation for ω'. Finally note that in such cases without existential variables, side deductions can be inlined, see Chapter 2 for formal details.

3.4 Counterexample. For soundness of differential induction, it is crucial that Definition 3.14 defines $D(F \vee G)$ conjunctively as $D(F) \wedge D(G)$ instead of $D(F) \vee D(G)$. From an initial state ν which satisfies $\nu \models F$, hence $\nu \models F \vee G$, the formula $F \vee G$ only is sustained differentially if F itself is a differential invariant, not if G is. For instance, $x_1 \geq 0 \vee d_1^2 + d_2^2 = v^2$ is no differential invariant of $\exists\omega\, \mathcal{F}(\omega)$, because $x_1 \geq 0$ can be invalidated by appropriate curved flights along $\mathcal{F}(\omega)$, see formula (3.3). In practice, splitting differential induction proofs over disjunctions can be useful.

3.5 Counterexample (Restricting differential invariance). It may be tempting to suspect that, in G5, the differential invariant F only needs to be differentially inductive at the states where F actually holds true. The differential induction needs to hold in a neighbourhood, though, such that adding F (or the border of F) to the assumptions in the premiss of G5 would be unsound! Consider the following counterexample where region $x^2 \leq 0$ is actually left immediately when following $x' = 1$, which also demonstrates unsoundness of other approaches [PJ04,

GT08]:

$$\dfrac{\dfrac{*\ (\text{unsound})}{\vdash \forall x\,(x^2 \leq 0 \rightarrow 2x \leq 0)}}{x^2 \leq 0 \vdash [x' = 1]x^2 \leq 0}$$

If, however, F describes an open set (e.g., F only involves strict inequalities), then G5 is sound even when adding F to the assumptions of the premiss. Likewise F can be added to the assumptions of the premiss when strengthening F' to strict inequalities. We will prove both refinements in Section 3.7. If polynomial solutions exist, they can be used as differential invariants. Furthermore, differential strengthening (D15) can be an extraordinarily successful proof technique for successively enriching invariant regions by derived invariants until F itself becomes differentially inductive, as we illustrate in Section 3.11.

3.6 Counterexample (Negative equations). It is crucial for soundness of differential induction that F is not allowed to contain negative equations. In the following counterexample, variable x can reach $x = 0$ without its derivative ever being 0.

$$\dfrac{\dfrac{*\ (\text{unsound})}{\vdash \forall x\,(1 \neq 0)}}{x \neq 0 \vdash [x' = 1]x \neq 0}$$

If, instead, both $x < 0$ and $x > 0$ are differential invariants of a system (e.g., of $x' = x$), then $x \neq 0$ can be proven indirectly by encoding it as $x < 0 \vee x > 0$.

A useful special case of D13 is the following derived weakening rule:

3.23 Lemma (Differential weakening). *The following is a derived rule:*

$$\text{(D13')}\quad \dfrac{\vdash \forall^{\alpha} \forall y_1 \ldots \forall y_k\,(\chi \rightarrow \phi)}{\vdash [\exists y_1 \ldots \exists y_k\,(x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge \chi)]\phi}$$

Proof. D13' clearly is sound, because χ is true along all state flows of the DA-constraint and ϕ is a consequence of χ by premiss. It can be derived as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \mathrm{QE}(\forall y_1 \ldots \forall y_k \forall d_1 \ldots \forall d_n\,(\chi_{x_1 \ldots x_n}^{d_1 \ldots d_n} \rightarrow \phi_{x_1 \ldots x_n}^{d_1 \ldots d_n}))}{\vdash \chi_{x_1 \ldots x_n}^{d_1 \ldots d_n} \rightarrow \phi_{x_1 \ldots x_n}^{d_1 \ldots d_n}}\,\mathrm{QE}}{\vdash [x_1 := d_1 \wedge \cdots \wedge x_n := d_n \wedge \chi_{x_1 \ldots x_n}^{d_1 \ldots d_n}]\phi}\,\mathrm{D10}}{\vdash \forall y_1 \ldots \forall y_k \forall d_1 \ldots \forall d_n\,([x_1 := d_1 \wedge \cdots \wedge x_n := d_n \wedge \chi_{x_1 \ldots x_n}^{d_1 \ldots d_n}]\phi)}\,\mathrm{F1}}{\vdash [\exists y_1 \ldots \exists y_k \exists d_1 \ldots \exists d_n\,(x_1 := d_1 \wedge \cdots \wedge x_n := d_n \wedge \chi_{x_1 \ldots x_n}^{d_1 \ldots d_n})]\phi}\,\mathrm{D6}}{\vdash [\exists y_1 \ldots \exists y_k \exists d_1 \ldots \exists d_n\,(x_1' = d_1 \wedge \cdots \wedge x_n' = d_n \wedge \chi)]\phi}\,\mathrm{D13}}{\vdash [\exists y_1 \ldots \exists y_k\,(x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge \chi)]\phi}\,\mathrm{D13}$$

The second application of D13 uses that fully nondeterministic continuous state change is equivalent to fully nondeterministic discrete state change, as they generate the same transitions. Finally, $\chi \rightarrow \phi$ can be obtained by α-renaming. □

Differential invariants enjoy structural closure properties. They are closed under conjunction (because of the conjunctive definition in Definition 3.14) and closed under differentiation.

3.24 Lemma. *Differential invariants are closed under differentiation: The total derivative of a differential invariant is an invariant of the same DA-constraint.*

Proof. Let F be a differential invariant, i.e., satisfy G5 for some DA-constraint of the form $\exists y\,(x' = \theta \wedge \chi)$, using vectorial notation for x and y. Hence, the premiss of G5 is provable: $\forall x\,\forall y\,(\chi \to F'^{\theta}_{x'})$ where the quantifier for x results from the universal closure $\forall^{\alpha}$. We have to show that the derivative $F'^{\theta}_{x'}$ is invariant and extend the proof to a proof of $[\exists y\,(x' = \theta \wedge \chi)]F'^{\theta}_{x'}$ by weakening (Lemma 3.23):

$$\text{D13'}\,\frac{\text{F1}\,\dfrac{*}{\vdash \mathrm{QE}(\forall x\,\forall y\,(\chi \to F'^{\theta}_{x'}))}}{\vdash [\exists y\,(x' = \theta \wedge \chi)]F'^{\theta}_{x'}}$$

□

3.5.6. Differential Induction with Differential Variants

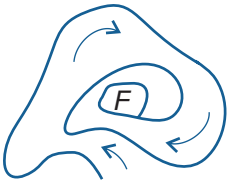


Figure 3.7.: Differential variants

Unlike the differential induction rule G5 for differential invariants, rule G6 uses differential induction to prove that F is a *differential variant*, which is attained differentially as an attractor region, rather than sustained differentially as in G5. The essential difference to G5 thus is the progress condition $F' \geq \varepsilon$ in the premiss, saying that the total differential of F along the DA-constraint is positive and at least some $\varepsilon > 0$. There, $F' \geq \varepsilon$ is a mnemonic notation for replacing all occurrences of inequalities $a \geq b$ in F' by $a \geq b + \varepsilon$ and $a > b$ by $a > b + \varepsilon$ (accordingly for $\leq, >, <$). Intuitively, the premiss expresses that, whereever χ holds but F does not yet hold, the total derivative is pointing towards F, see Figure 3.7. Especially $F' \geq \varepsilon$ guarantees a minimum progress rate of ε towards F along the dynamics. To further ensure that the continuous evolution towards F remains within χ, the antedent of the conclusion shows that χ holds *until* F is attained, which can again be proven using G5. In this context, $\sim F$ is a short hand notation for *weak negation*, i.e., the operation that behaves like $\neg$, except that $\sim(a \geq b) \equiv b \geq a$ and $\sim(a > b) \equiv a \leq b$. Unlike negation, weak negation retains the border of F, which is required in G6 as χ needs to continue to hold (including the border of F) until F is reached. Especially, for G6, invariant χ is not required to hold after F has been reached successfully. The operations $F' \geq \varepsilon$ and $\sim F$ are defined accordingly for other inequalities (in G6, we do not permit F

to contain equalities, see Counterexample 3.8 below). Again we demonstrate differential induction and the necessity of its prerequisites in a series of examples.

3.7 Example. As an example, we turn back to Figure 3.5. In the rightmost side deduction, G6 is used to prove that $F \equiv x_1 \geq p_1 \wedge x_2 \geq p_2$ is finally reached. There, the total derivative is $F' \equiv x_1' \geq 0 \wedge x_2' \geq 0$, which yields $d_1 \geq 0 \wedge d_2 \geq 0$ when substituting the flight equations $\mathcal{F}(\omega)$, because $x_1' = d_1, x_2' = d_2, p_1' = p_2' = 0$. Consequently $(F' \geq \varepsilon)^{\substack{d_1\ d_2\ -\omega d_2\ \omega d_1\ 0\ 0 \\ x_1'\ x_2'\ d_1'\ \ d_2'\ p_1'\ p_2'}}$ is identical to $(F' \geq \varepsilon)^{\substack{d_1\ d_2\ 0\ 0 \\ x_1'\ x_2'\ p_1'\ p_2'}}$, which gives $d_1 \geq \varepsilon \wedge d_2 \geq \varepsilon$. Similarly, the proof for formula (3.3) can be generalised to differential inequalities, again assuming $d_1' = d_2' = p_1' = p_2' = 0$ and $b' = 0$:

$$\forall p\, \exists d\, (\|d\|^2 \leq b^2 \wedge \langle x_1' \geq d_1 \wedge x_2' \geq d_2 \rangle (x_1 \geq p_1 \wedge x_2 \geq p_2)) \ .$$

Using Lemma 3.18, the differential inequalities, which express lower bounds on the evolution of x_1 and x_2, can be reduced to differential equations with quantified disturbance $u \in \mathbb{R}^2$:

$$\forall p\, \exists d\, .. \langle \exists u\, (x_1' = d_1 + u_1 \wedge x_2' = d_2 + u_2 \wedge u_1 \geq 0 \wedge u_2 \geq 0) \rangle (x_1 \geq p_1 \wedge x_2 \geq p_2).$$

The proof for this DAL formula is identical to Figure 3.5, except that G6 yields $\forall x\, \forall u\, ((x_1 < p_1 \vee x_2 < p_2) \wedge u_1 \geq 0 \wedge u_2 \geq 0 \ \rightarrow \ d_1 + u_1 \geq \varepsilon \wedge d_2 + u_2 \geq \varepsilon)$.

3.8 Counterexample (Equational differential variants). Rule G6 is not applicable for equations like $x = y$. Even though $x = y$ can be encoded as $F \equiv x \leq y \wedge x \geq y$, the corresponding $F' \geq \varepsilon \equiv x' + \varepsilon \leq y' \wedge x' \geq y' + \varepsilon$ is equivalent to false for $\varepsilon > 0$. Indeed, assuming $a' = b' = 0$, the validity of a formula like $\langle x' = a \wedge y' = b \rangle x = y$ depends on the relationship of the initial values of x and y and the constants a and b: It is true, iff $(x - y)(a - b) < 0$ or $x = y$ holds initially.

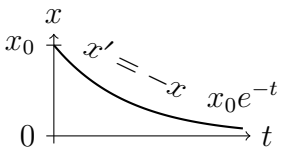
More generally, differential variants cannot (directly) verify conjunctive equations like in $\langle x' = a \wedge y' = b \rangle (x = 0 \wedge y = 0)$ because differential variants guarantee *that* a target region F will be reached, not when precisely. In particular, $x = 0$ and $y = 0$ would not necessarily be reached simultaneously. In fact, for $a, b \neq 0$, the above reachability property is only valid iff $bx = ay \wedge ax < 0$, initially.

3.9 Counterexample (Minimal progress requirement). Unlike in discrete domains, strictly monotonic sequences can converge in $\mathbb{R}$. Thus, the premiss $F' \geq \varepsilon$ for an $\varepsilon > 0$ of G6 cannot be weakened to $F' > 0$ as the counterexample in Figure 3.8a shows, in which x converges monotonically to 0 along the dynamics shown in Figure 3.8b. Moreover, this example demonstrates that, in the presence of convergent dynamics, a property like $x \geq 0$ can be invariant, even though it is not differentially invariant, see Figure 3.8c.

3.10 Counterexample (Lipschitz-continuity requirement). As the counterexample in Figure 3.9a shows, Lipschitz-continuity (or at least the existence of a solution of sufficient duration) is, in fact, a necessary prerequisite for G6. For $x = y = 0$ initially,

$$\dfrac{*\text{ (unsound)}}{\dfrac{\vdash \forall x\,(x > 0 \rightarrow -x < 0)}{\vdash \langle x' = -x\rangle x \leq 0}}$$

3.8a: Counterexample



3.8b: Convergent descent

$$\dfrac{\dfrac{\mathit{false}}{\vdash \forall x\,(-x \geq 0)}\,\text{F1}}{x \geq 0 \vdash [x' = -x]x \geq 0}\,\text{G5}$$

3.8c: Non-inductive

Figure 3.8.: Monotonically decreasing convergent counterexample

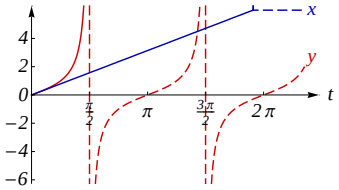
the solution of the differential equations in Figure 3.9a is $x(t) = t$ and $y(t) = \tan t$. In explosive examples like the corresponding dynamics in Figure 3.9b, where solution y grows unbounded in finite time, the duration of existence of solutions is limited so that the target region $x \geq 6$ is physically unreachable. More precisely, the dynamics is not well-posed beyond the explosive point of unbounded growth at the singularity $\frac{\pi}{2}$ and is non-physical beyond that singularity. Note that the continuous dynamics of Figure 3.9 is only locally Lipschitz-continuous and disobeys divergence of time (Section 3.3). The condition of Lipschitz-continuity is directly expressible as a formula for G6:

$$\begin{aligned}&\exists L \forall y_1 \ldots \forall y_k \forall x_1 \ldots \forall x_n \, \forall \tilde{y}_1 \ldots \forall \tilde{y}_k \forall \tilde{x}_1 \ldots \forall \tilde{x}_n \\ &\quad (\theta_1 - \tilde{\theta}_1)^2 + \cdots + (\theta_n - \tilde{\theta}_n)^2 \leq L^2((x_1 - \tilde{x}_1)^2 + \cdots + (x_n - \tilde{x}_n)^2)\end{aligned}$$

where $\tilde{\theta}_i$ denotes the result of substituting all x_j in θ_i by the corresponding $\tilde{x}_j$ and the y_j by $\tilde{y}_j$. Observe that, besides Lipschitz-continuity, any other condition can be used that ensures the existence of a solution of sufficient duration for G6.

$$\dfrac{*\text{ (unsound)}}{\dfrac{\vdash \exists \varepsilon{>}0\, \forall x \forall y\,(x < 6 \rightarrow 1 \geq \varepsilon)}{\vdash \langle x' = 1 \wedge y' = 1 + y^2\rangle x \geq 6}}$$

3.9a: Counterexample



3.9b: Explosive dynamics

Figure 3.9.: Unbounded dynamics with limited duration of solutions

3.6. Soundness

In this section we prove that verification with the DAL calculus always produces correct results about DA-programs, i.e., the DAL calculus is sound.

3.25 Theorem (Soundness). *The* DAL *calculus is* sound*, i.e., every* DAL *formula that can be derived in the* DAL *calculus is valid (true in all states).*

Proof. The calculus is sound if each rule instance is sound. The rules of the DAL calculus are even *locally sound*, i.e., their conclusion is true at ν if all its premisses are true in ν. Local soundness implies soundness. The local soundness proofs of D1–D4 and the propositional rules are as usual. Similarly, G3 and G4 are local versions of induction schemes, and the proof is as usual (Theorem 2.15, likewise for G1–G2. The local soundness of D9–D10 is a generalisation of the proofs for update rules [BP06] to first-order DJ-constraints. The proofs for D5–D8 are simple. Finally, our previous results [BP06, Pla08b] can be lifted to show that locally sound rules are closed under addition of Γ, Δ context and of conjunctive DJ-constraints in Definition 3.22. For soundness, however, conjunctive DJ-constraints are crucial here [BP06, Pla08b] as these are deterministic.

F3 Rule F3 is locally sound: Let ν be a state in which the premiss is true, i.e.,

$$\nu \models \mathrm{QE}(\exists x \bigwedge_i (\Gamma_i \vdash \Delta_i)) \ .$$

We have to show that the conclusion is true in this state. Using that quantifier elimination yields an equivalence, we see that ν also satisfies $\exists x \bigwedge_i (\Gamma_i \vdash \Delta_i)$ prior to the quantifier elimination. Hence, for some state ν_x that agrees with ν except for the value of x we obtain:

$$\nu_x \models \bigwedge_i (\Gamma_i \vdash \Delta_i) \ .$$

As side deduction ($\star$) in Figure 3.4 is inductively shown to be locally sound, we can conclude that $\nu_x \models (\Gamma \vdash \Delta, \phi)$. Therefore, $\nu \models \exists x\, (\Gamma \vdash \Delta, \phi)$. Now the conjecture can be obtained using standard reasoning with quantifiers and the absence of x in Γ, Δ by rewriting the conclusion with local equivalences:

$$\exists x\, (\Gamma \vdash \Delta, \phi) \equiv \exists x\, (\neg\Gamma \vee \Delta \vee \phi) \equiv \neg\Gamma \vee \Delta \vee \exists x\, \phi \equiv \Gamma \vdash \Delta, \exists x\, \phi \quad (3.4)$$

The soundness proof for F1 is similar since (3.4) holds for any quantifier. The proofs of F4 and F2 can be derived using duality of quantifiers.

D12 By Lemma 3.17, there is an equivalent disjunctive normal form $\mathcal{D}_1 \vee \cdots \vee \mathcal{D}_n$ of $\mathcal{D}$. Thus, it only remains to show that $\rho(\mathcal{D}) \subseteq \rho((\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_n)^*)$ as the converse inclusion is obvious. Let φ be a state flow for a transition $(\nu, \omega) \in \rho(\mathcal{D})$. We assume that φ is non-Zeno according to Definition 3.12. Thus, there is a finite number, m, of switches between the $\mathcal{D}_i$, say $\mathcal{D}_{i_1}, \mathcal{D}_{i_2}, \ldots, \mathcal{D}_{i_m}$. Then, the transition (ν, ω) belonging to φ can be simulated piecewise by m repetitions of $\mathcal{D}_1 \cup \ldots \cup \mathcal{D}_n$, where each piece selects the respective part $\mathcal{D}_{i_j}$. The proof for D11 is accordingly.

D13 Local soundness of the rules D13 and D14 is an immediate consequence of Lemma 3.17 and the respective semantics of modalities.

D15 Rule D15 can be proven locally sound using that the left premiss implies that every flow φ that satisfies $\mathcal{D}$ also satisfies χ *all along* the flow. Thus, $\varphi \models \mathcal{D}$ implies $\varphi \models \mathcal{D} \wedge \chi$ so that the right premiss entails the conclusion.

G5 Let ν satisfy the premiss and the antecedent of the conclusion as, otherwise, there is nothing to show. By Lemma 3.17, we can assume F to be in disjunctive normal form and consider any disjunct G of F that is true at ν. In order to show that F is sustained during the continuous evolution, it is sufficient to show that each conjunct of G is. We can assume these conjuncts to be of the form $c \geq 0$ (or $c > 0$ where the proof is accordingly). Finally, using vectorial notation, we write $x' = \theta$ for the differential equation system and $\exists y$ for the chain of quantifiers. Now let $\varphi : [0, r] \to \mathrm{State}(\Sigma)$ be any state flow with $\varphi \models \exists y\,(x' = \theta \wedge \chi)$ beginning in $\varphi(0) = \nu$. In particular, $\varphi \models \exists y\,\chi$, which, by antecedent, implies $\nu \models F$, i.e., $c \geq 0$ holds at ν. We assume duration $r > 0$, because the other case is immediate ($\nu \models c \geq 0$ already holds). We show that $c \geq 0$ holds all along the flow φ, i.e., $\varphi \models c \geq 0$.

Suppose there was a $\zeta \in [0, r]$ where $\varphi(\zeta) \models c < 0$, which will lead to a contradiction. Then the function $h : [0, r] \to \mathbb{R}$ defined as $h(t) = val(\varphi(t), c)$ satisfies $h(0) \geq 0 > h(\zeta)$, because $\nu \models c \geq 0$ by antecedent. Clearly, φ is of the order of $D(c)$, because: φ is of order 1 for all variables in vector x, and trivially of order ∞ for variables that do not change during the DA-constraint. Further, by α-renaming, $D(c)$ cannot contain the quantified variables y, hence, φ is not required to be of any order in y. The value of c is defined all along φ, because we have assumed χ to guard against zeros of denominators. Thus, by Lemma 3.15, h is continuous on $[0, r]$ and differentiable at every $\xi \in (0, r)$. The mean value theorem implies that there is a $\xi \in (0, \zeta)$ such that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) \cdot (\zeta - 0) = h(\zeta) - h(0) < 0$. In particular, since $\zeta \geq 0$, we can conclude that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) < 0$. Now Lemma 3.15 implies that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) = val(\bar{\varphi}(\xi), D(c)) < 0$. The latter equals[1] $val(\bar{\varphi}(\xi)^u_y, D(c)^\theta_{x'})$ by Lemma 3.16, because $\varphi \models \exists y\,(x' = \theta \wedge \chi)$ so that $\bar{\varphi}(\xi)^u_y \models x' = \theta \wedge \chi$ for some $u \in \mathbb{R}$ and because y' does not occur and $y \notin c$. This, however, is a contradiction, because the premiss implies that $\varphi \models \forall y\,(\chi \to D(c)^\theta_{x'} \geq 0)$ as $\forall^\alpha$ comprises all variables that change during the flow φ along $x' = \theta$, i.e., the vector x. In particular, as $\bar{\varphi}(\xi)^u_y \models \chi$ holds, we have $\bar{\varphi}(\xi)^u_y \models D(c)^\theta_{x'} \geq 0$.

G6 First, we consider the quantifier free case, again using vectorial notation. Let ν be any state satisfying the premiss and the antecedent of the conclusion.

[1] For $u \in \mathbb{R}$ let $\bar{\varphi}(\xi)^u_y$ denote the (augmented) state that agrees with $\bar{\varphi}(\xi)$ except that the value of y is u.

Since ν satisfies the premiss and, after α-renaming, ε is a fresh variable, we can assume ν itself to satisfy $\nu \models \forall^\alpha(\neg F \wedge \chi \to (F' \geq \varepsilon)^\theta_{x'})$. For G6, we required $x' = \theta$ to be Lipschitz-continuous so that the global Picard-Lindelöf theorem [Wal98, Propsosition 10.VII] ensures the existence of a global solution of arbitrary duration $r \geq 0$, which is all we need here. Let φ be a state flow corresponding to a solution of the differential equation $x' = \theta$ starting in ν of some duration $r \geq 0$. If there is a point in time ζ at which $\varphi(\zeta) \models F$, then by antecedent, until (and including, because $\sim F$ contains the closure of $\neg F$) the first such point, χ holds true during φ. Hence, the restriction of φ to $[0, \zeta]$ is a state flow witnessing $\nu \models \langle x' = \theta \wedge \chi\rangle F$. If, otherwise, there is no such point, then we show that extending φ by choosing a larger r will inevitably make F true. We thus have $\varphi \models \neg F \wedge \chi$ and, by premiss, $\varphi \models F'^\theta_{x'} \geq \varepsilon$, because $\forall^\alpha$ comprises the variables x that change during φ. By Definition 3.14, $F'^\theta_{x'} \geq \varepsilon$ is a conjunction. Consider one of its conjuncts, say $c'^\theta_{x'} \geq \varepsilon$ belonging to a literal $c \geq 0$ of F (the other cases are accordingly). Again, φ is of the order of $D(c)$ and the value of c is defined along φ, because $\varphi \models \chi$ and χ is assumed to guard against zeros. Hence, by mean-value theorem, Lemma 3.15, and Lemma 3.16, we conclude for each $\zeta \in [0, r]$ that

$$val(\varphi(\zeta), c) - val(\varphi(0), c) = val(\bar{\varphi}(\xi), c'^\theta_{x'})(\zeta - 0) \geq \zeta val(\varphi(0), \varepsilon)$$

for some $\xi \in (0, \zeta)$. Now as $val(\varphi(0), \varepsilon) > 0$ we have for all $\zeta > -\frac{val(\varphi(0),c)}{val(\varphi(0),\varepsilon)}$ that $\varphi(\zeta) \models c \geq 0$ and $\varphi(r) \models c \geq 0$, even $\varphi(r) \models c > 0$. By extending r sufficiently large, we have that all literals $c \geq 0$ of one conjunct of F are true, which concludes the proof, because, until F finally holds, $\varphi \models \chi$ is implied by the antecedent as shown earlier.

In the presence of quantifiers ($\exists y$ with vectorial notation), rule G6 implies a slightly stronger statement, because y is quantified universally in the premiss (and antecedent): F can be reached for *all* choices of y that respect χ (rather than just for one). By antecedent, there is a $u \in \mathbb{R}$ such that $\nu^u_y \models \chi$. Hence, ν^u_y satisfies the assumptions of the above quantifier-free case. Thus, $\nu^u_y \models \langle x' = \theta \wedge \chi\rangle F$, which entails that $\nu \models \langle \exists y\, (x' = \theta \wedge \chi)\rangle F$ using u constantly as the value for the quantified variable y during the evolution. □

3.7. Restricting Differential Invariants

While Example 3.5 shows that differential invariant F cannot generally be assumed to hold in the premiss of G5 without loosing soundness, we present two corresponding refinements of G5 that are indeed sound.

3.26 Proposition. *Using the notation of G5–G6, the following variations of dif-*

ferential induction G5 are sound (in G5', F describes an open set):

$$\text{(G5')}\quad \frac{\vdash \forall^{\alpha}\forall y_1 \ldots \forall y_k\,(F \wedge \chi \rightarrow F'^{\theta_1 \ldots \theta_n}_{x'_1 \ldots x'_n})}{[\exists y_1 \ldots \exists y_k\,\chi]F \vdash [\exists y_1 \ldots \exists y_k\,(x'_1 = \theta_1 \wedge \cdots \wedge x'_n = \theta_n \wedge \chi)]F}$$

$$\text{(G5")}\quad \frac{\vdash \forall^{\alpha}\forall y_1 \ldots \forall y_k\,(F \wedge \chi \rightarrow (F' > 0)^{\theta_1 \ldots \theta_n}_{x'_1 \ldots x'_n})}{[\exists y_1 \ldots \exists y_k\,\chi]F \vdash [\exists y_1 \ldots \exists y_k\,(x'_1 = \theta_1 \wedge \cdots \wedge x'_n = \theta_n \wedge \chi)]F}$$

Proof. The proof that the first rule is sound is similar to the proof for G5 in Theorem 3.25, except that assuming $\varphi(\zeta) \models \neg F$ only yields $h(0) \geq 0 \geq h(\zeta)$, which does not lead to a contradiction. However, by using that F is open, the distance to the border of F is positive in the initial state $\varphi(0)$, which yields the inequality $h(0) > 0 \geq h(\zeta)$, and the contradiction arises accordingly.

The soundness of the second rule needs more adaptation. Repeating the argument for G5, we can assume F to be of the form $c \geq 0$. Suppose there was a $\iota \in [0, r]$ where $\varphi(\iota) \models c < 0$, which will lead to a contradiction. Let $\zeta \in [0, r]$ be the infimum of these ι, hence, $\varphi(\zeta) \models c = 0$ by continuity. Then the function $h : [0, r] \to \mathbb{R}$ defined as $h(t) = val(\varphi(t), c)$ satisfies $h(0) \geq 0 \geq h(\zeta)$, because $\nu \models c \geq 0$ by antecedent. By repeating the argument with Lemma 3.15 like in the proof for G5, h is continuous on $[0, r]$ and differentiable at every $\xi \in (0, r)$ with a derivative of $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) = val(\bar{\varphi}(\xi), D(c))$, which in turn equals $val(\bar{\varphi}(\xi), D(c)^{\theta}_{x'})$, as $\varphi \models x' = \theta$. Now, the mean value theorem implies that there is a $\xi \in (0, \zeta)$ such that

$$\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) \cdot (\zeta - 0) = h(\zeta) - h(0) \leq 0$$

In particular, as $\zeta \geq 0$, we can conclude that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) = val(\bar{\varphi}(\xi), D(c)^{\theta}_{x'}) \leq 0$. This, however, contradicts that the premiss implies $\bar{\varphi}(\xi) \models D(c)^{\theta}_{x'} > 0$, as the flow satisfies $\varphi \models \chi$ and $\varphi(\xi) \models c \geq 0$, because $\zeta > \xi$ is the infimum of the counterexamples ι with $\varphi(\iota) \models c < 0$. □

3.11 Example. Consider the differential equation $x' = x^3$. With either rule G5' or rule G5", we can establish easily that the system stays above $\frac{1}{4}$ whenever the dynamics starts above $\frac{1}{4}$:

$$\text{G5'}\,\frac{\text{F1}\,\dfrac{*}{\vdash \forall x\,(x > \frac{1}{4} \rightarrow x^3 > 0)}}{x > \frac{1}{4} \vdash [x' = x^3]x > \frac{1}{4}}$$

3.8. Differential Monotonicity Relaxations

Invariant regions of DA-constraints are helpful for differential induction rule G5, because they provide stronger assumptions for the premiss. In fact, the whole

purpose of the differential strengthening rule D15 is to enrich invariant regions of DA-constraints in order to weaken the sub-goals of G5.

In contrast, invariant regions are more demanding for differential variant induction rule G6, because the antecedent of its goal requires that invariant region χ is shown to remain true throughout the evolution (after all, invariant regions χ of DA-constraints $\mathcal{D}$ are domain restrictions that can be used as assumptions for $[\mathcal{D} \wedge \chi]\phi$ but have to be shown to hold true for $\langle \mathcal{D} \wedge \chi \rangle \phi$). Similarly, for evolution rules that are based on solutions of differential equations—rules D11–D12 from Figure 2.5 on page 35—invariant regions make the sub-goal formulas much more complex (even though they even lead to weaker sub-goals for rule D12). In particular, invariant regions increase the number of quantifier alternations in D11–D12, which have the predominant influence on the complexity of quantifier elimination [DH88].

For simplifying non-differential invariant region χ from $[\mathcal{D} \wedge \chi]\phi$ with a DA-constraint $\mathcal{D}$, we can simply use the differential weakening rule D13 to drop χ:

$$\text{D13}\frac{\vdash [\mathcal{D}]\phi}{\vdash [\mathcal{D} \wedge \chi]\phi}$$

Slightly less conservatively, we can approximate the assumption $\forall 0 \leq \tilde{t} \leq t \, \langle \mathcal{S}_{\tilde{t}} \rangle \chi$ on the solution $\mathcal{S}_t$ in the sub-goal of D12 from Figure 2.5 by $\langle \mathcal{S}_t \rangle \chi$ (or by $\chi \wedge \langle \mathcal{S}_t \rangle \chi$) in a sound yet incomplete way, because if every evolution of the solution $\mathcal{S}_t$ satisfying χ at the end satisfies ϕ, then every evolution along $\mathcal{D}$ satisfying χ all the time must satisfy ϕ even more so. Thus, the following variant of D12 is sound where $\mathcal{S}_t$ is the solution of the differential equation like for D12:

$$\text{(D12')} \quad \frac{\forall t \geq 0 \, \big(\chi \rightarrow \langle \mathcal{S}_t \rangle (\chi \rightarrow \phi)\big)}{[x_1' = \theta_1 \wedge \ldots \wedge x_n' = \theta_n \wedge \chi]\phi}$$

For simplifying non-differential invariant region χ from $\langle \mathcal{D} \wedge \chi \rangle \phi$, we can use the dual of the differential strengthening rule D15 and show that χ remains true throughout the evolution along $\mathcal{D}$ anyhow (left sub-goal which can be handled using G5) so that only some evolution along $\mathcal{D}$ remains to be found that actually reaches ϕ (right sub-goal):

$$\text{D15}\frac{\vdash [\mathcal{D}]\chi \quad \vdash \langle \mathcal{D} \rangle \phi}{\vdash \langle \mathcal{D} \wedge \chi \rangle \phi}$$

When using G5 to prove the left sub-goal $[\mathcal{D}]\chi$ by showing validity of a formula of the form $\forall^{\mathcal{D}} {\chi'}^{\theta}_{x'}$, we actually show invariance of χ along $\mathcal{D}$. More generally, we can use differential-algebraic techniques similar to differential induction to prove the weaker property of monotonicity instead of invariance of χ.

3.27 Definition (Monotonicity derivation). Let α be a DA-program. For a first-order formula F, the following formula is called *monotonicity derivation* of F,

where the syntactic derivative $D(a)$ is defined according to Definition 3.14:

$$M^\alpha(F) \equiv \bigwedge_{i=1}^{m} M^\alpha(F_i) \quad \text{where } \{F_1, \ldots, F_m\} \text{ is the set of all literals of } F$$
$$M^\alpha(a \sim b) \equiv \forall^\alpha(D(a) \geq D(b)) \vee \forall^\alpha(D(a) \leq D(b)) \quad \text{where } \sim \in \{\leq, \geq, <, >, =\}$$

3.28 Proposition (Differential monotonicity). *Let $\langle \mathcal{S}_t \rangle$ be the DJ-constraint for the solution at time t of the symbolic initial-value problem for the differential equation $\mathcal{D}$ defined as $x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n$ like in rule D11 of Figure 2.5. Let the non-differential constraint χ be a conjunction of atomic formulas without negative equalities, then the following is a sound proof rule:*

$$\text{(D11')} \quad \frac{\vdash \exists t{\geq}0\,(\chi \wedge \langle \mathcal{S}_t \rangle(\chi \wedge \phi)) \qquad \vdash M^{\mathcal{D}}(\chi)^{\theta_1}_{x_1'} \cdots {}^{\theta_n}_{x_n'}}{\vdash \langle x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge \chi \rangle \phi}$$

Proof. Local soundness is a simple consequence of the well-known fact that, for a differentiable function f, monotonic increasing of f on an interval $[c, d]$ is equivalent to $f'(z) \geq 0$ on (c, d). With this, the right sub-goal implies that, for any conjunct $a \geq b$ of χ (likewise for $\leq, <, >, =$), the value of $a - b$ is either monotonically increasing or monotonically decreasing along the flow. Either way, if χ holds in the beginning *and* the end of a flow of some duration t (as implied by the left sub-goal), monotonicity implies that χ holds all along the flow, so that the sub-goals imply the conclusion as in case D11 of the proof of Theorem 2.15.

Formally, let φ be a state flow of an appropriate duration r following solution $\mathcal{S}_t$ according to the left sub-goal as in Theorem 2.15. By the left sub-goal we have $\varphi(r) \models \phi$ (when r is the witness for $\exists t{\geq}0$) and we only need to show that $\varphi \models \chi$. Consider a conjunct $a \geq b$ of χ and consider the case where the right sub-goal implies $\varphi(0) \models \forall^{\mathcal{D}}(D(a) \leq D(b))^{\theta}_{x'}$, using vectorial notation for x and θ. Then $\varphi \models (a' \leq b')^{\theta}_{x'}$, because the universal closure $\forall^{\mathcal{D}}$ comprises all variables that change during the flow φ along $\mathcal{D}$. Thus $\varphi \models a' \leq b'$ by Lemma 3.16. Let $h : [0, r] \to \mathbb{R}$ be the function defined as $h(t) = val(\varphi(t), a - b)$. Again, φ is of the order of $a' - b'$ (φ is of the order 1 in each x_i and of arbitrary order for other variables) and the value of $a - b$ is defined all along φ, because χ guards against zeros in χ. Thus, Lemma 3.15 is applicable and h is differentiable at every $\xi \in (0, r)$. For any $\zeta \in [0, r]$, we have to show that $\varphi(\zeta) \models \chi$. By mean value theorem, there is a $\xi \in (\zeta, r)$ such that, when using Lemma 3.15, we have

$$h(r) - h(\zeta) = h'(\xi) \cdot (r - \zeta) = val(\bar{\varphi}(\xi), a' - b') \cdot (r - \zeta) \leq 0$$

because $\bar{\varphi}(\xi) \models a' \leq b'$. Thus, we have $h(\zeta) \geq h(r) \geq 0$, since $\varphi(r) \models \chi$, which implies that $\varphi(r) \models a - b \geq 0$ and $\varphi(r) \models a \geq b$.

The other case where the right sub-goal implies $\varphi(0) \models \forall^\alpha(a' \geq b')$ is simpler using that $\varphi(0) \models \chi$ and is, in fact, a direct consequence of the proof of G5 in

Theorem 3.25. The other conjuncts of the form $a \leq b, a < b, a > b$ and $a = b$ are almost identical, because the monotonicity argument for $a - b$ carries over easily, as the respective conjunct holds before and after the continuous evolution. □

3.12 Example (Monotonic invariants in train control). Consider the differential constraints for train control (equation (2.1) on page 30 in Section 2.4). For the DA-constraint $z' = v \wedge v' = a \wedge \tau' = 1 \wedge v \geq 0 \wedge \tau \leq \varepsilon$, invariant region $v \geq 0 \wedge \tau \leq \varepsilon$ can be shown to be monotonic or convex with respect to the dynamics, that is: If it holds in the beginning and at the end of an evolution, the invariant also holds in between. Monotonicity is easy to prove with the above proof rule using the following symbolic computations for the right sub-goal (where $\forall^\alpha$ is $\forall z \forall v \forall \tau$):

$$\begin{aligned}
&\big((\forall^\alpha(v' \geq 0) \vee \forall^\alpha(v' \leq 0)) \wedge (\forall^\alpha(\tau' \geq \varepsilon') \vee \forall^\alpha(\tau' \leq \varepsilon'))\big)^{v\ a\ 1\ 0}_{z'\ v'\ \tau'\ \varepsilon'} \\
\equiv\ &(\forall^\alpha(a \geq 0) \vee \forall^\alpha(a \leq 0)) \wedge (\forall^\alpha(1 \geq 0) \vee \forall^\alpha(1 \leq 0)) \\
\equiv\ &\mathit{true}
\end{aligned}$$

Observe that the invariant domain will not be a differential invariant, here, because $v \geq 0$ is only an invariant of $z' = v \wedge v' = a$ for $a \geq 0$. For any a, however, $v \geq 0$ will either be a monotonically increasing property (if $a \geq 0$ constantly) or monotonically decreasing (if $a \leq 0$ constantly), one of which is true for every constant a. Thus, if $v \geq 0$ has been true before and after an evolution along $z' = v \wedge v' = a$, it must have been true throughout this evolution. Likewise, $\tau \leq \varepsilon$ never is a differential invariant of $\tau' = 1$, because the passing of time along $\tau' = 1$ will inevitably violate $\tau \leq \varepsilon$ sooner or later. Still, it is a monotonically decreasing property. Consequently, the monotonicity relaxation of Proposition 3.28 applies for the train control example, thereby simplifying proofs with invariant regions considerably, because the invariant only needs to be checked before and after rather than throughout the evolution. For instance, this simplifies the proof of property (2.7) on page 66.

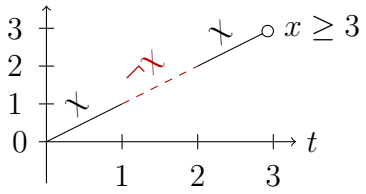
Similarly, the right sub-goal of D11' is a sufficient condition to ensure that D12' is a complete replacement for D12.

3.13 Counterexample (Disjunctive monotonicity). For soundness of differential monotonicity relaxations, it is crucial that D11' only accepts conjunctive invariant regions. As the counterexample in Figure 3.10a with the dynamics in Figure 3.10b shows, differential monotonicity relaxations do not hold for disjunctive invariant regions, because the same disjunct has to hold before and after the evolution for monotonicity arguments to be sound. Let χ abbreviate the disjunctive invariant region $x \leq 1 \vee x \geq 2$. Then the differential monotonicity criterion $\forall x\,(1 \leq 0) \vee \forall x\,(1 \geq 0)$ would be fulfilled, but a different disjunct holds at the initial state $x = 0$ than at the target $x \geq 3$ so that monotonicity neither implies that $x \leq 1$ nor that $x \geq 2$ holds in between.

3.14 Counterexample (Negative equalities). A similar counterexample shows why D11' does not allow negative equalities. Along the dynamics $x' = 1 \wedge x \neq 2$ we

$$\dfrac{\dfrac{}{\vdash \exists t{\geq}0\,\big(\chi \wedge \langle x := x + t\rangle(\chi \wedge x \geq 3)\big)}\,*\text{ (unsound)}}{x = 0 \vdash \langle x' = 1 \wedge (x \leq 1 \vee x \geq 2)\rangle x \geq 3}$$

3.10a: Counterexample



3.10b: Interrupted dynamics

Figure 3.10.: Interrupted dynamics for disjunctive monotonicity

cannot conclude from the truth of $x \neq 2$ before and after the evolution that $x \neq 2$ held true throughout the evolution, just on the basis of a condition on the derivative $x' \neq 0$. A continuous evolution from $x = 0$ to $x = 3$ still leaves $x \neq 2$ in between.

3.9. Relative Completeness

As a consequence of Theorem 2.16 and Proposition 3.13, the DAL calculus is not effectively axiomatisable (yet even pure reachability is already undecidable for hybrid systems [Hen96]).

It is easy to see that the relative completeness proof for $\mathsf{d}\mathcal{L}$ generalises to DAL with only minor modifications when using first-order logic of DA-constraints as a basis in place of FOD (again, nested modalities can be avoided when using quantifiers). The first-order logic of DA-constraints results from FOD by allowing DA-constraints in place of differential equations inside modalities.

3.29 Theorem (Relative completeness). *The* DAL *calculus is* complete relative to *DA-constraints, i.e., every valid* DAL *formula can be derived from tautologies of the first-order logic of DA-constraints.*

Proof. The proof is a simple adaptation of the proof for $\mathsf{d}\mathcal{L}$ in Section 2.7.2: In the proof of Lemma 2.19, we replace all cases for continuous evolutions along differential equations or for discrete jumps by the following cases for DA-constraints $\mathcal{D}$ or DJ-constraints $\mathcal{J}$, respectively:

$$\mathcal{S}_{\mathcal{D}}(\vec{x},\vec{v}) \equiv \langle\mathcal{D}\rangle\vec{v} = \vec{x}$$
$$\mathcal{S}_{\mathcal{J}}(\vec{x},\vec{v}) \equiv \mathcal{J}^{v_i=\theta_i}_{x_i:=\theta_i}$$

Where the first-order formula $\mathcal{J}^{v_i=\theta_i}_{x_i:=\theta_i}$ results from $\mathcal{J}$ by replacing all assignments of any form $x_i := \theta_i$ by equations $v_i = \theta_i$. The rest of the relative completeness proof generalises immediately using that the respective rules (D5–D10) for DJ-constraints are symmetric, hence equivalent, and their premisses are of smaller complexity. □

Note that, for generalising the relative completeness proof in the simple-most way, we formally need to allow update prefixes in DAL proofs as in Definition 2.11, which is easily seen to be sound for deterministic DJ-constraints.

3.10. Deductive Strength of Differential Induction

We analyse the deductive power of differential induction with respect to classes of formulas that are allowed as differential invariants. For purely equational differential invariants, the deductive power is not affected by allowing or disallowing propositional operators in differential invariants:

3.30 Proposition (Equational deductive power). *The deductive power of differential induction with atomic equations is identical to the deductive power of differential induction with propositional combinations of polynomial equations: Formulas are provable with propositional combinations of equations as differential invariants iff they are provable with only atomic equations as differential invariants.*

Proof. We show that every differential invariant that is a propositional combination ϕ of polynomial equations is expressible as a single atomic polynomial equation (the converse inclusion is obvious). We assume ϕ to be in negation normal form and reduce ϕ inductively using the following transformations:

- If ϕ is of the form $p_1 = p_2 \vee q_1 = q_2$, then ϕ is equivalent to the single equation $(p_1 - p_2)(q_1 - q_2) = 0$.
- If ϕ is of the form $p_1 = p_2 \wedge q_1 = q_2$, then ϕ is equivalent to the single equation $(p_1 - p_2)^2 + (q_1 - q_2)^2 = 0$.
- If ϕ is of the form $\neg(p_1 = p_2)$, then ϕ does not qualify as a differential invariant, because it contains a negative equality, which are disallowed for G5 according to Figure 3.3. □

Observe, however, that the required polynomial degree of atomic equations is larger than for propositional combinations, which can have computational disadvantages for quantifier elimination.

For general differential invariants, where inequalities are allowed, the situation is different: We show that, in general, the deductive power of differential induction depends on which class of formulas is allowed as differential invariants! Some DAL formulas cannot by proven by a differential induction step with only atomic formula but no propositional operators as differential invariant, while they are provable immediately using unrestricted differential invariants.

3.31 Theorem (Deductive power). *The deductive power of differential induction with arbitrary formulas exceeds the deductive power of differential induction with*

atomic formulas: All DAL *formulas that are provable using atomic differential invariants are provable using general differential invariants, but not vice versa!*

Proof. The inclusion is obvious. Conversely, we have to show that there are DAL formulas that are provable with general differential invariants but not with atomic differential invariants. Consider the following example, which is provable using rule G5', i.e., the variant of G5 for open sets (Section 3.7), with the non-atomic formula $x > 0 \wedge y > 0$ as differential invariant:

$$\text{G5'}\dfrac{\text{F1}\dfrac{*}{\vdash \forall x \forall y\,(x > 0 \wedge y > 0 \rightarrow xy > 0 \wedge xy > 0)}}{x > 0 \wedge y > 0 \vdash [x' = xy \wedge y' = xy](x > 0 \wedge y > 0)}$$

First, we show that this formula is not provable by a differential induction step with only atomic formulas as differential invariants. Suppose there was a single polynomial $p(x, y)$ in variables x, y such that $p(x, y) > 0$ is a differential invariant proving the above formula, which will lead to a contradiction. The conditions for differential invariants (G5 or G5') imply that the following formulas have to be valid:

1. $x > 0 \wedge y > 0 \rightarrow p(x, y) > 0$, as differential invariants have to hold in the prestate according to the antecedent of G5 (or G5').

2. $p(x, y) > 0 \rightarrow x > 0 \wedge y > 0$, as the differential invariant has to imply the postcondition (when using G1 to show that the differential invariant implies the postcondition).

In particular, $x > 0 \wedge y > 0 \leftrightarrow p(x, y) > 0$ is valid. Thus, p enjoys the property:

$$p(x, y) \geq 0 \text{ for } x \geq 0, y \geq 0, \text{ and, otherwise, } p(x, y) \leq 0\ . \tag{3.5}$$

Assume p has minimal total degree with property (3.5). Now, $p(x, 0)$ is a univariate polynomial in x with zeros at all $x > 0$, thus $p(x, 0) = 0$ is the zero polynomial, hence y divides $p(x, y)$. Accordingly, $p(0, y) = 0$ for all y, hence x divides $p(x, y)$. Thus, xy divides p. But by comparing the signs, we see that polynomial $\frac{-p(-x,-y)}{xy}$ also satisfies property (3.5) with a smaller total degree than p, which is a contradiction.

Similarly, there is no polynomial p such that $x > 0 \wedge y > 0 \leftrightarrow p(x, y) = 0$, because only the zero polynomial is zero on the full quadrant $(0, \infty)^2$. Finally, $x > 0 \wedge y > 0 \leftrightarrow p(x, y) \geq 0$ is impossible for continuity reasons which imply that $p(0, 0) = 0$, which is a contradiction. More generally, the same argument holds for any other sign condition that is supposed to characterise one quadrant of $\mathbb{R}^2$ uniquely.

Observe that, so far, the argument does not depend on the actual dynamics and is, thus, still valid in the presence of arbitrary differential weakening (D13).

Next, to see that the above example cannot even be proven indirectly after differential strengthening (D15), we use that, inductively, the strengthening χ itself needs to be a differential invariant: Ultimately, the left sub-goal of D15 can only be shown using differential induction. The above example, however, is built such that, as $x' = xy$ is the differential equation, $xy > 0$ is required for $x > 0$ to be a differential invariant (which thus also requires $y > 0$). Vice versa, due to $y' = xy$, formula $xy > 0$ is a prerequisite for the differential invariance of $y > 0$ (which thus also needs $x > 0$). Yet, for differential invariance of $xy > 0$, we have to prove $xy > 0 \to (y + x)xy > 0$ for G5', because $(xy)'^{xy\ xy}_{x'\ y'}$ gives $(x'y + yx')^{xy\ xy}_{x'\ y'}$, i.e., $xyy + yxy$. But $xy > 0 \to (y + x)xy > 0$ is, again, equivalent to $x \geq 0 \vee y \geq 0$, and thus to $\neg(-x > 0 \wedge -y > 0)$, which cannot be proven by atomic differential induction (or differential weakening) according to the first part of this proof. Thus, the required atomic differential invariants have circular dependencies for differential strengthenings by $x > 0$, $y > 0$, and $xy > 0$, respectively, which cannot be resolved in any proof tree without simultaneous differential induction using non-atomic differential invariants, because differential strengthenings have to be ordered totally along each proof branch. □

As a special case, this result implies that differential induction in DAL is deductively stronger than approaches using barrier certificates [PJ04, PJP07], criticality functions [DMO+07], or polynomial invariant equations [SSM04, RCT05]. On top of that, the DAL calculus adds differential strengthening and weakening techniques, which add further deductive power. The roundabout maneuver that we verify in the next section is a practical example where differential induction with mixed non-atomic formulas and successive differential strengthening turns out to be decisive.

3.11. Air Traffic Control Verification

In this section we verify that the tangential roundabout maneuver for collision avoidance in air traffic control that we presented in Section 3.4 is collision-free, i.e., directs aircraft on flight paths with global minimal distance $p > 0$, and determine a corresponding parameter constraint on the *tang* procedure. Using differential induction and differential strengthening, the flight maneuver can be verified despite the complicated hybrid flight dynamics of aircraft.

3.11.1. Characterisation of Safe Roundabout Dynamics

Property ϕ in Figure 3.2 defines *safe states* as those with separation $\|x - y\| \geq p$. This does *not*, however, characterise the states with *safe dynamics*: Several states

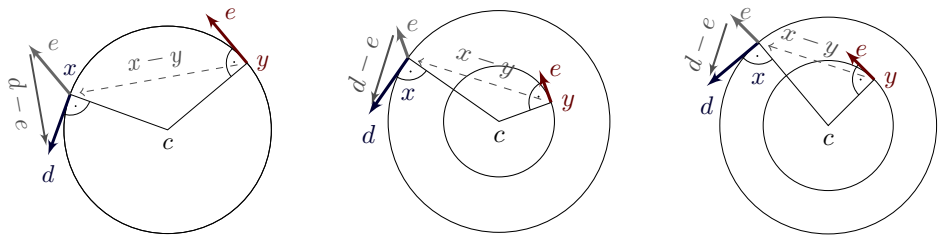
that satisfy ϕ will not remain safe when following curved roundabout flight maneuvers, see Figure 3.1c for a counterexample violating ϕ after some time. In particular, the angular velocity ω and initial speed vectors d and e must fit to the relative positioning of the aircraft x and y for the aircraft dynamics to remain safe. In order to find out the required parametric constraints for safety of the roundabout maneuver, we analyse the DAL formula ψ in the DAL calculus and identify a corresponding parameter constraint $\mathcal{T}$. For notational convenience, we inline side deductions and slightly simplify universal closure notation $\forall^\alpha$ by taking free variables as universally quantified, because the following DAL proof needs no existential variables.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{*}{\phi \vdash \forall x,y,d,e\,(\phi \to \phi)}\,\text{F1}}{\phi \vdash [free]\phi}\,\text{D13'} \qquad \dfrac{\dfrac{\dots}{\phi \vdash [tang](\phi \wedge \mathcal{T})} \qquad \dfrac{\dots}{\phi \wedge \mathcal{T} \vdash [\mathcal{F}(\omega) \wedge \mathcal{G}(\omega)]\phi}}{\phi \vdash [tang; \mathcal{F}(\omega) \wedge \mathcal{G}(\omega)]\phi}\,\text{G1}}{\phi \vdash [free][tang; \mathcal{F}(\omega) \wedge \mathcal{G}(\omega)]\phi}\,\text{G1}}{\phi \vdash [trm]\phi}\,\text{D2}}{\vdash \forall^\alpha(\phi \to [trm]\phi)}\,\text{F1,P7}}{\phi \vdash [trm^*]\phi}\,\text{G3}}{\vdash \phi \to [trm^*]\phi}\,\text{P7}$$

The left branch closes, because postcondition ϕ is the invariant region in free flight such that its DA-constraint can be weakened by Lemma 3.23. In the other branches, $\mathcal{T}$ is the parameter constraint that *tang* needs to establish in addition to ϕ (middle branch) for the roundabout dynamics to be safe (right branch). Hence condition $\mathcal{T}$ mediates among the middle and right branch. Using successive quantifier elimination, we derive the following constraint $\mathcal{T}$ as a prerequisite for ϕ to be differentially inductive. It is the decisive constraint that characterises configurations with safely controllable dynamics in curved roundabout maneuvers (using vectorial notation and orthogonal complements $d^\perp$ from Section 3.2):

$$\begin{aligned}\mathcal{T} &\equiv d - e = \omega(x-y)^\perp \quad \big(\text{or, equivalently } (d-e)^\perp = -\omega(x-y)\big) \\ &\equiv d_1 - e_1 = -\omega(x_2 - y_2) \wedge d_2 - e_2 = \omega(x_1 - y_1)\ .\end{aligned} \tag{3.6}$$

This formula expresses that the relative speed vector $d - e$ is orthogonal to the relative position $x - y$ and compatible with the angular velocity ω and tangential orientation of d and e. Figure 3.11a illustrates the symmetric case with identical linear speed $\|d\| = \|e\|$, Figure 3.11b–3.11c show asymmetric cases with distinct linear speeds $\|d\| \neq \|e\|$, which is possible as well. Condition $\mathcal{T}$ gives the decisive handle for an inductive characterisation of safe tangential roundabout configurations: For the right branch of the above proof, we need to show that the tangential configuration $\mathcal{T}$ is sufficient for ϕ to be sustained during curved evasive actions. In the following, we prove that the relative speed vector configuration $\mathcal{T}$ is itself differentially inductive (left branch) and use differential strengthening with D15 to



3.11a: Symmetric case 3.11b: Asymmetric case 3.11c: Close flight

Figure 3.11.: Tangential construction for characteristics $\mathcal{T}$ of roundabout dynamics

augment the dynamics with $\mathcal{T}$ as a derived invariant for proving that the actual safety property ϕ is sustained (right branch), again by differential induction:

$$\dfrac{\dfrac{\dfrac{*}{\vdash \forall^{\alpha}(\mathcal{T}'_{\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)})}\,\text{F1}}{\phi, \mathcal{T} \vdash [\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)]\mathcal{T}}\,\text{G5} \qquad \dfrac{\dfrac{*}{\vdash \forall^{\alpha}(\mathcal{T} \rightarrow \phi'_{\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)})}\,\text{F1}}{\phi \vdash [\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)\wedge\mathcal{T}]\phi}\,\text{G5}}{\dfrac{\phi, \mathcal{T} \vdash [\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)]\phi}{\phi\wedge\mathcal{T} \vdash [\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)]\phi}\,\text{P6}}\,\text{D15}$$

Observe that differential strengthening by D15 is crucial for the proof, because neither ϕ nor $\mathcal{T}\wedge\phi$ are differentially inductive for $\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)$! Instead, the tangential configuration $\mathcal{T}$ itself is differentially inductive relative to $\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)$ (left branch) and strong enough to make ϕ differentially inductive relative to the augmented DA-constraint $\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)\wedge\mathcal{T}$ (right branch). For readability, we use a slightly weaker rule for differential induction, with ϕ rather than $[\mathcal{T}]\phi$ in the antecedent of the conclusion. This variant can be derived easily using a cut and will again be called G5. The differential induction G5 on the left and right branch close using quantifier elimination in F1 or the following algebraic equational reasoning, respectively ($\mathcal{T}'_{\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)}$ is a short notation for substituting the differential equations from $\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)$ into $D(\mathcal{T})$, see Lemma 3.16):

$$\begin{aligned}
\mathcal{T}'_{\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)} &\equiv (d_1' - e_1' = -\omega(x_2' - y_2') \wedge d_2' - e_2' = \omega(x_1' - y_1'))_{\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)} \\
&\equiv -\omega d_2 + \omega e_2 = -\omega(d_2 - e_2) \wedge \omega d_1 - \omega e_1 = \omega(d_1 - e_1) \quad \equiv \mathit{true} \\
\phi'_{\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)} &\equiv (2(x_1 - y_1)(x_1' - y_1') + 2(x_2 - y_2)(x_2' - y_2') \geq 0)_{\mathcal{F}(\omega)\wedge\mathcal{G}(\omega)} \\
&\equiv 2(x_1 - y_1)(d_1 - e_1) + 2(x_2 - y_2)(d_2 - e_2) \geq 0 \\
(\text{using } \mathcal{T}) &\equiv 2(x_1 - y_1)(-\omega(x_2 - y_2)) + 2(x_2 - y_2)\omega(x_1 - y_1) = 0 \geq 0 \equiv \mathit{true}.
\end{aligned}$$

Altogether, we have shown that *every* tangential roundabout evasion maneuver respecting $\mathcal{T}$ is safe. Further, the middle branch of the above proof reveals the parameter constraint imposed on *tang* for safe roundabouts, which concludes the proof of the following result.

3.32 Theorem (Safety of tangential roundabout maneuver). *For every choice of the tangential entry procedure that satisfies $\phi \to [tang](\phi \wedge \mathcal{T})$, the tangential roundabout flight maneuver in Figure 3.2 safely avoids collisions, i.e., it directs aircraft on flight paths with minimal horizontal aircraft separation at least $p > 0$.*

This result can be proven in our theorem prover [PQ08a] in 2s including user interactions for G3 and D15. Its proof does not need G1, which we only used here to shorten the proof presentation. Theorem 3.32 expresses unbounded-time safety for fully parametric tangential roundabouts with arbitrary choices for the free parameters. The proof of Theorem 3.32 generalises to roundabouts entered by more than two participants when ϕ and $\mathcal{T}$ are augmented accordingly. For instance, using our automatic proof procedure from Chapter 6, our theorem prover can prove mutual collision avoidance for 5 aircraft fully automatically, see Chapter 6 and Chapter 8. Likewise, G5 and D15 can be used to prove that external separation to all other sufficiently far points is maintained during the roundabout maneuver, in particular, the maneuver only needs bounded space:

3.33 Proposition (External separation of roundabout maneuvers). *Separation of aircraft x to all external points $u \in \mathbb{R}^2$ of distance beyond the roundabout diameter $2r$ is maintained:*

$$r \geq 0 \wedge (r\omega)^2 = \|d\|^2 \to \forall u\, (\|x-u\|^2 > (2r+p)^2 \to [\mathcal{F}(\omega)](\|x-u\|^2 > p^2)) \ .$$

3.11.2. Tangential Entry Procedures

As a simple choice for the tangential initiation procedure *tang* satisfying property $\mathcal{T}$, consider the following operation which chooses an arbitrary angular velocity ω, an arbitrary centre $c \in \mathbb{R}^2$ for the roundabout maneuver, and adjusts d and e tangentially:

$$tang \ \equiv \ \exists u\, \omega := u;\ \exists c\, (d := \omega(x-c)^\perp \wedge e := \omega(y-c)^\perp) \ . \tag{3.7}$$

This formula expresses that the speed vectors d and e of both aircraft at x and y, respectively, are tangentially and of the same angular velocity ω relative to the intended centre c of the roundabout, with the same orientation (Figure 3.11). For this choice, the assumption of Theorem 3.32 can be proven after D10 substitutes the corresponding terms for d and e in $\mathcal{T}$, using F1 (or linearity of $d^\perp$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\overset{*}{\phi \vdash \phi}\,\text{P9} \qquad \overset{*}{\phi \vdash \omega(x-c)^\perp - \omega(y-c)^\perp = \omega(x-y)^\perp}}{\phi \vdash \phi \wedge \omega(x-c)^\perp - \omega(y-c)^\perp = \omega(x-y)^\perp}\,\text{P5}}{\phi \vdash [d := \omega(x-c)^\perp \wedge e := \omega(y-c)^\perp](\phi \wedge \mathcal{T})}\,\text{D10}}{\phi \vdash \forall\omega\, \forall c\, [d := \omega(x-c)^\perp \wedge e := \omega(y-c)^\perp](\phi \wedge \mathcal{T})}\,\text{F1,F1}}{\phi \vdash [tang](\phi \wedge \mathcal{T})}\,\text{D2,D6,D6}$$

It can also be shown that $\exists c\,(d = \omega(x-c)^\perp \wedge e = \omega(y-c)^\perp)$ is equivalent to $\mathcal{T}$ for nonzero ω. With choice (3.7), the tangential roundabout maneuver in Figure 3.2 is safe and has been significantly simplified and generalised in comparison to [PC07].

3.11.3. Discussion

Our tangential roundabout maneuver leaves open how and when precisely the collision avoidance maneuver is initiated or when to leave it. For instance, (3.7) does not restrict c and ω but accepts any choice including choices optimising secondary objectives like fuel consumption. Furthermore, as specified in Figure 3.2 and proven in this section, the roundabout maneuver can be left safely with arbitrary free flight by repeating the loop at any time: The roundabout maneuver will simply be initiated again during free flight when necessary. As a special case, this open policy includes free flight enabling the aircraft to leave the roundabout in their original direction. While the simple choice (3.7) is possibly discontinuous in d and e, it is comparably easy to see that there are fully curved entry and exit procedures that remain safe when the entry procedure is initiated with sufficient distance by using the separation limit of Proposition 3.33. We refine the roundabout collision avoidance maneuver and develop a corresponding entry procedure in Chapter 8. Our proof shows that the tangential roundabout maneuver is safe for every such entry procedure. In particular, the control parameters c and ω of (3.7) can also be chosen such that the resulting speed vectors d and e are in a bounded range meeting external speed requirements of the aircraft:

$$\forall v\,(\phi \to \langle tang \rangle(\phi \wedge \mathcal{T} \wedge \|d\|^2 = \|e\|^2 = v^2)) \ . \tag{3.8}$$

3.12. Summary

We have introduced a first-order dynamic logic for differential-algebraic programs with interacting first-order discrete jump constraints and first-order differential-algebraic constraints. For this differential-algebraic logic, DAL, we have presented a calculus for verifying hybrid systems given as differential-algebraic programs.

In differential-algebraic programs, both internal choices and disturbances during continuous evolutions and nondeterminism in discrete operations can be described uniformly by quantifiers. Most importantly, we have introduced first-order differential induction with differential invariants and differential variants for proving correctness statements with first-order differential-algebraic constraints purely algebraically using the differential constraints themselves instead of their solutions. In combination with successive differential strengthening for refining the system dynamics by auxiliary differential invariants, we obtain a powerful verification calculus for systems with challenging dynamics. We compare the deductive strength

for classes of differential invariants and show that the deductive power of general differential induction exceeds the deductive power of atomic differential invariants.

We have demonstrated that our calculus can be used successfully for verifying fully parametric roundabout maneuvers in air traffic control. To the best of our knowledge, this is the first formal proof for unbounded safety of hybrid aircraft dynamics in curved collision avoidance maneuvers for air traffic control. Moreover, we argue that our fully formal proof about aircraft gives more confidence in flight maneuvers than informal approaches that do not consider the actual hybrid flight dynamics [HKT07, DMC05, GMAR07] or results that only prevent orthogonal collisions in discretisations of the system [DPR05, MF01]. Our logic DAL is also more convenient, because hybrid systems like the tangential roundabout maneuver can be specified and verified uniformly within a single logic. Despite challenging flight dynamics, the DAL formulas about aircraft and roundabout maneuvers that we presented in this chapter can be proven in our theorem prover [PQ08a] within a few seconds.

While this work answers the open issues (1), (3) and (4) raised in the work of Piazza et al. [PAM+05], we are interested in extending differential-algebraic methods to address further questions about hybrid systems. In Chapter 6, we investigate algorithms for constructing differential invariants automatically on the basis of our DAL calculus presented here. Interesting future work for the aircraft case study is to find a fully curved maneuver that achieves collision avoidance by joint horizontal and vertical evasive actions.

Chapter 4.

Differential Temporal Dynamic Logic dTL

Contents

Synopsis

We combine first-order dynamic logic for reasoning about possible behaviour of hybrid systems with temporal logic for reasoning about the temporal behaviour during their operation. Our logic supports verification of hybrid programs with first-order definable flows and provides a uniform treatment of discrete and continuous evolution. For our combined logic, we generalise the semantics of dynamic modalities to refer to hybrid traces instead of final states. Further, we prove that this gives a conservative extension of dynamic logic. On this basis, we provide a modular verification calculus that reduces correctness of temporal behaviour of hybrid systems to non-temporal reasoning and prove that we obtain a complete axiomatisation relative to the non-temporal base logic. Using this calculus, we analyse safety invariants in a train control system and symbolically synthesise parametric safety constraints.

4.1. Introduction

Correctness of real-time and hybrid systems depends on a safe operation throughout *all* states of all possible trajectories, and the behaviour at intermediate states is highly relevant [DHO06].

Temporal logics (TL) use temporal operators to talk about intermediate states [Pnu77, EC82, EH86, ACD90, Sti92]. In addition to successful uses in model checking [CGP99, ACD90, HNSY92, Hen96, MPM05], temporal logics have been used in deductive approaches to prove validity of formulas in calculi [DN00, DCMM04]. Among other shortcommings and difficulties discussed in Chapter 1, the major drawback of TL calculi for our purpose is that TL formulas cannot generally characterise the operations of a specific hybrid system.

Like model checking, *dynamic logic* (DL) [HKT00] can directly analyse the behaviour of actual system models. However, DL only considers the behaviour at final states, which is insufficient for verifying safety invariants that have to hold all the time.

We close this gap of expressivity by combining first-order dynamic logic [HKT00] with temporal logic [Pnu77, EC82, EH86]. We use the generalisation of operational system models and semantics to hybrid systems from Chapter 2. In this chapter, we introduce a temporal dynamic logic dTL, which provides modalities for quantifying over traces of hybrid systems based on differential dynamic logic. We equip dTL with temporal operators to state what is true all along a trace or at some point during a trace. In this chapter, we modify the semantics of the dynamic modality $[\alpha]$ to refer to all *traces* of α instead of all final states reachable with α (similarly

for $\langle\alpha\rangle$). For instance, the formula $[\alpha]\Box\phi$ expresses that ϕ is true at each state during all traces of the hybrid system α. With this, dTL can also be used to verify temporal statements about the behaviour of α at intermediate states during system runs. As in our non-temporal dynamic logic $\mathsf{d}\mathcal{L}$, we use hybrid programs as an operational model for hybrid systems, since they admit a uniform compositional treatment of interacting discrete and continuous evolution in logic.

As a semantical foundation for combined temporal dynamic formulas, we introduce a hybrid trace semantics for dTL. We prove that dTL is a conservative extension of $\mathsf{d}\mathcal{L}$, that is, for non-temporal specifications, trace semantics is equivalent to the non-temporal transition semantics of $\mathsf{d}\mathcal{L}$ from Chapter 2.

As a means for verification, we introduce a sequent calculus for dTL that successively reduces temporal statements about traces of hybrid programs to non-temporal $\mathsf{d}\mathcal{L}$ formulas. In this way, we make the intuition formally precise that temporal safety invariants can be checked by augmenting proofs with appropriate assertions about intermediate states. Like in Chapter 2, our calculus works compositionally: It structurally decomposes correctness statements about hybrid programs into corresponding statements about its parts by symbolic transformation. Observe that this is challenging for hybrid systems, because even a single elementary system action of continuous evolution already exhibits temporal behaviour when it assumes several different states as time passes.

Contributions

Our approach combines the advantages of DL in reasoning about the behaviour of (multiple and parametric) operational system models with those of TL to verify temporal statements about traces. Our first contribution is the logic dTL, which provides a coherent foundation for reasoning about the temporal behaviour of operational models of hybrid systems with symbolic parameters. The main contribution is our calculus for deductively verifying temporal statements about hybrid systems, which is a complete axiomatisation relative to non-temporal $\mathsf{d}\mathcal{L}$.

4.1.1. Related Work

Based on [Pra79], Beckert and Schlager [BS01] added separate trace modalities to dynamic logic and presented a relatively complete calculus. Their approach only handles discrete state spaces. In contrast, dTL works for hybrid programs with continuous state spaces. There, a particular challenge is that invariants may already change their truth-value multiple times during a single continuous evolution, hence relevant temporal behaviour even occurs during single transitions.

Davoren and Nerode [DN00] extended the propositional modal μ-calculus with a semantics in hybrid systems and examine topological aspects. In [DCMM04], Davoren et al. gave a semantics in general flow systems for a generalisation of

CTL* [EH86]. In both cases, the authors of [DN00] and [DCMM04] provided Hilbert-style calculi to prove formulas that are valid for all systems simultaneously using abstract actions.

As discussed in Section 1.2, the strength of our logic primarily is that it is a first-order dynamic logic and handles actual hybrid programs like $x := x + 1; x' = 2y$ rather than only abstract actions of unknown effect.

Structure of this Chapter

After introducing syntax and semantics of the differential temporal dynamic logic dTL in Section 4.2 and Section 4.3, we introduce a modular sequent calculus for dTL in Section 4.5 that extends our previous calculi with temporal proof rules in a completely modular way. We prove soundness and relative completeness in Section 4.6 and Section 4.7, respectively. In Section 4.8, we use our calculus to analyse safety invariants in train control from Section 4.4. We further present extensions for quantifier alternation liveness in Section 4.9. We draw conclusions and discuss future work in Section 4.10.

4.2. Syntax of Temporal Dynamic Logic for Hybrid Systems

The temporal dynamic logic dTL extends dynamic logic [HKT00] with three concepts for verifying temporal specifications of hybrid systems:

Hybrid programs The behaviour of hybrid systems can be described by hybrid programs (Section 2.2.2), which generalise real-time programs [HNSY92] to hybrid change. The distinguishing feature of hybrid programs in this context is that they provide uniform discrete jumps and continuous evolutions along differential equations. While hybrid automata [Hen96] can be embedded, program structures are more amenable to compositional symbolic processing by calculus rules, see Chapter 2.

Modal operators Modalities of dynamic logic express statements about all possible behaviour ($[\alpha]\pi$) of a system α, or about the existence of a trace ($\langle\alpha\rangle\pi$), satisfying condition π. As in Chapter 2, the system α is described as a hybrid program. Yet, unlike in standard dynamic logic [HKT00] or in $\mathsf{d}\mathcal{L}$, π is a *trace formula* in dTL, and π is allowed to refer to all states that occur *during* a trace using temporal operators.

Temporal operators For dTL, the temporal trace formula $\Box\phi$ expresses that the formula ϕ holds all along a trace selected by $[\alpha]$ or $\langle\alpha\rangle$. For instance, the state formula $\langle\alpha\rangle\Box\phi$ says that the state formula ϕ holds at every state along at least one trace of α. Dually, the trace formula $\Diamond\phi$ expresses that ϕ holds at some point during such a trace. It can occur in a state formula $\langle\alpha\rangle\Diamond\phi$ to express that there is such a state in some trace of α, or as $[\alpha]\Diamond\phi$ to say that, along each trace, there is a state satisfying ϕ. In this chapter, the primary focus of attention is on homogeneous combinations of path and trace quantifiers like $[\alpha]\Box\phi$ or $\langle\alpha\rangle\Diamond\phi$.

4.2.1. Hybrid Programs

The formulas of dTL are built over a non-empty set V of real-valued variables and a fixed signature Σ of function and predicate symbols. For simplicity, Σ is assumed to contain exclusively the usual function and predicate symbols for real arithmetic, such as $0, 1, +, \cdot, =, \leq, <, \geq, >$. The set $\mathrm{Trm}(\Sigma, V)$ of *terms* is defined as in classical first-order logic.

The hybrid programs allowed in dynamic modalities of dTL are the same as those of $\mathsf{d}\mathcal{L}$, see Definition 2.3 in Section 2.2.2. They are built from elementary discrete jumps and continuous evolutions using a regular control structure [HKT00]. Similarly, differential-algebraic programs from Chapter 3 can be allowed when using DAL as a basis instead of $\mathsf{d}\mathcal{L}$, leading to differential-algebraic temporal dynamic logic DATL.

4.2.2. State and Trace Formulas

The formulas of dTL are defined similar to first-order dynamic logic [HKT00]. However, the modalities $[\alpha]$ and $\langle\alpha\rangle$ accept trace formulas that refer to the temporal behaviour of *all* states along a trace. Inspired by CTL and CTL* [EC82, EH86], we distinguish between state formulas, that are true or false in states, and trace formulas, that are true or false for system traces. The sets $\mathrm{Fml}(\Sigma, V)$ of state formulas, $\mathrm{Fml}_T(\Sigma, V)$ of trace formulas, and $\mathrm{HP}(\Sigma, V)$ of hybrid programs with variables in V are simultaneously inductively defined in Definition 4.1 and 2.3, respectively.

4.1 Definition (dTL formulas). The set $\mathrm{Fml}(\Sigma, V)$ of *(state) formulas* is simultaneously inductively defined as the smallest set such that:

1. If $p \in \Sigma$ is a predicate of arity $n \geq 0$ and $\theta_1, \ldots, \theta_n \in \mathrm{Trm}(\Sigma, V)$, then $p(\theta_1, \ldots, \theta_n) \in \mathrm{Fml}(\Sigma, V)$.
2. If $\phi, \psi \in \mathrm{Fml}(\Sigma, V)$, then $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \to \psi) \in \mathrm{Fml}(\Sigma, V)$.
3. If $\phi \in \mathrm{Fml}(\Sigma, V)$ and $x \in V$, then $\forall x\, \phi, \exists x\, \phi \in \mathrm{Fml}(\Sigma, V)$.

4. If $\pi \in \mathrm{Fml}_T(\Sigma, V)$ and $\alpha \in \mathrm{HP}(\Sigma, V)$, then $[\alpha]\pi, \langle\alpha\rangle\pi \in \mathrm{Fml}(\Sigma, V)$.

The set $\mathrm{Fml}_T(\Sigma, V)$ of *trace formulas* is the smallest set with:

1. If $\phi \in \mathrm{Fml}(\Sigma, V)$, then $\phi \in \mathrm{Fml}_T(\Sigma, V)$.
2. If $\phi \in \mathrm{Fml}(\Sigma, V)$, then $\Box\phi, \Diamond\phi \in \mathrm{Fml}_T(\Sigma, V)$.

Formulas without $\Box$ and $\Diamond$, i.e., without Case 2 of the trace formulas, are *non-temporal* $\mathsf{d}\mathcal{L}$ *formulas* (Chapter 2). Unlike in CTL, state formulas are true on a trace (Case 1) if they hold for the *last* state of a trace, not for the first. Thus, $[\alpha]\phi$ expresses that ϕ is true at the end of each trace of α. In contrast, $[\alpha]\Box\phi$ expresses that ϕ is true all along all states of every trace of α. This combination gives a smooth embedding of non-temporal $\mathsf{d}\mathcal{L}$ into dTL and makes it possible to define a compositional calculus. Like CTL, dTL allows nesting with a branching time semantics [EC82], e.g., $[\alpha]\Box(x \geq 2 \rightarrow \langle\beta\rangle\Diamond x \leq 0)$.

Insipred by CTL* [EH86], syntactic and semantic extensions from dTL to dTL* are straightforward and amount to allowing propositional combinations of trace formulas. Finding appropriate proof calculi, however, is much more difficult, even for CTL* [PK02, Rey05].

4.3. Semantics

In standard dynamic logic [HKT00], the logic $\mathsf{d}\mathcal{L}$ from Chapter 2, and DAL from Chapter 3, modalities only refer to the final states of system runs and the semantics is a reachability relation on states: State ω is reachable from state ν using α if there is a run of α which terminates in ω when started in ν. For dTL, however, formulas can refer to intermediate states of runs as well. Thus, the semantics of a hybrid system α is the set of its possible *traces*, i.e., successions of states that occur during the evolution of α.

4.3.1. Trace Semantics of Hybrid Programs

States contain values of system variables during a hybrid evolution. A *state* is a map $\nu : V \to \mathbb{R}$; the set of all states is denoted by $\mathrm{Sta}(\Sigma)$. In addition, we distinguish a state Λ to denote the failure of a system run when it is *aborted* due to a test $?\chi$ that yields *false*. In particular, Λ can only occur at the end of an aborted system run and marks that there is no further extension.

Hybrid systems evolve along piecewise continuous traces in multi-dimensional space as time passes. Continuous phases are governed by differential equations, whereas discontinuities are caused by discrete jumps in state space. Unlike in discrete cases [Pra79, BS01], traces are not just sequences of states, since hybrid systems pass through uncountably many states even in bounded time. Beyond that,

continuous changes are more involved than in pure real-time [ACD90, HNSY92], because all variables can evolve along different differential equations. Generalising the real-time traces of [HNSY92], the following definition captures hybrid behaviour by splitting the uncountable succession of states into periods σ_i that are regulated by the same control law. For discrete jumps, some periods are point flows of duration 0.

4.2 Definition (Hybrid trace). A *trace* is a (nonempty) finite or infinite sequence $\sigma = (\sigma_0, \sigma_1, \sigma_2, \dots)$ of functions $\sigma_i : [0, r_i] \to \mathrm{Sta}(\Sigma)$ with respective durations $r_i \in \mathbb{R}$ (for $i \in \mathbb{N}$). A *position* of σ is a pair (i, ζ) with $i \in \mathbb{N}$ and ζ in the interval $[0, r_i]$; the state of σ at (i, ζ) is $\sigma_i(\zeta)$. Positions of σ are ordered lexicographically by $(i, \zeta) \prec (j, \xi)$ iff either $i < j$, or $i = j$ and $\zeta < \xi$. Further, for a state $\nu \in \mathrm{Sta}(\Sigma)$, $\hat{\nu} : 0 \mapsto \nu$ is the *point flow* at ν with duration 0. A trace *terminates* if it is a finite sequence $(\sigma_0, \sigma_1, \dots, \sigma_n)$ and $\sigma_n(r_n) \neq \Lambda$. In that case, the last state $\mathrm{last}\,\sigma$ is denoted as $\sigma_n(r_n)$. The first state $\mathrm{first}\,\sigma$ is $\sigma_0(0)$.

Unlike in [ACD90, HNSY92], the definition of traces also admits finite traces of bounded duration, which is necessary for compositionality of traces in $\alpha; \gamma$. The semantics of hybrid programs α as the set $\tau(\alpha)$ of its possible traces depends on valuations $val(\nu, \cdot)$ of formulas and terms at intermediate states ν. The valuation of terms [HKT00], and interpretations of function and predicate symbols are as usual for real arithmetic. The valuation of formulas will be defined in Definition 4.4. Again, we use $\nu[x \mapsto d]$ to denote the *modification* that agrees with state ν on all variables except for the symbol x, which is changed to $d \in \mathbb{R}$.

4.3 Definition (Trace semantics of hybrid programs). The *trace semantics, $\tau(\alpha)$, of a hybrid program α*, is the set of all its possible hybrid traces and is defined as follows:

1. $\tau(x_1 := \theta_1, \dots, x_n := \theta_n) = \{(\hat{\nu}, \hat{\omega}) \ : \ \omega = \nu[x_1 \mapsto val(\nu, \theta_1)] \dots [x_n \mapsto val(\nu, \theta_n)]$ for $\nu \in \mathrm{Sta}(\Sigma)\}$
2. $\tau(x_1' = \theta_1, \dots, x_n' = \theta_n \,\&\, \chi) = \{(\varphi) \ : \ \varphi$ is a state flow of order 1 and some duration $r \geq 0$ such that $\varphi \models x_1' = \theta_1 \wedge \dots \wedge x_n' = \theta_n \wedge \chi$, see Definition 3.9$\}$
3. $\tau(?\chi) = \{(\hat{\nu}) \ : \ val(\nu, \chi) = true\} \cup \{(\hat{\nu}, \hat{\Lambda}) \ : \ val(\nu, \chi) = false\}$
4. $\tau(\alpha \cup \beta) = \tau(\alpha) \cup \tau(\beta)$
5. $\tau(\alpha; \beta) = \{\sigma \circ \varsigma \ : \ \sigma \in \tau(\alpha)\,, \varsigma \in \tau(\beta)$ when $\sigma \circ \varsigma$ is defined$\}$;
   the composition of $\sigma = (\sigma_0, \sigma_1, \sigma_2, \dots)$ and $\varsigma = (\varsigma_0, \varsigma_1, \varsigma_2, \dots)$ is

$$\sigma \circ \varsigma := \begin{cases} (\sigma_0, \dots, \sigma_n, \varsigma_0, \varsigma_1, \dots) & \text{if } \sigma \text{ terminates at } \sigma_n \text{ and } \mathrm{last}\,\sigma = \mathrm{first}\,\varsigma \\ \sigma & \text{if } \sigma \text{ does not terminate} \\ \text{not defined} & \text{otherwise} \end{cases}$$

6. $\tau(\alpha^*) = \bigcup_{n\in\mathbb{N}} \tau(\alpha^n)$, where $\alpha^{n+1} := (\alpha^n; \alpha)$ for $n \geq 1$, and $\alpha^0 := (?\mathit{true})$.

Time passes differently during discrete and continuous change. During continuous evolution, the discrete step index i of positions (i, ζ) remains constant, whereas the continuous duration ζ remains 0 during discrete point flows. This permits multiple discrete state changes to happen at the same (super-dense) continuous time, unlike in [ACD90].

4.3.2. Valuation of State and Trace Formulas

In the semantics of dTL formulas, the dynamic modalities determine the set of traces according to the trace semantics of hybrid programs, and, independently, the temporal modalities determine at which points in time, the respective postcondition needs to hold.

4.4 Definition (Valuation of dTL formulas). For state formulas, the *valuation* $val(\nu, \cdot)$ with respect to state ν is defined as follows:

1. $val(\nu, p(\theta_1, \ldots, \theta_n)) = p^\ell\big(val(\nu, \theta_1), \ldots, val(\nu, \theta_n)\big)$, where p^ℓ is the relation associated to p.

2. $val(\nu, \phi \wedge \psi)$ is defined as usual, the same holds for $\neg, \vee, \rightarrow$.

3. $val(\nu, \forall x\, \phi) = \mathit{true}$ iff $val(\nu[x \mapsto d], \phi) = \mathit{true}$ for all $d \in \mathbb{R}$

4. $val(\nu, \exists x\, \phi) = \mathit{true}$ iff $val(\nu[x \mapsto d], \phi) = \mathit{true}$ for some $d \in \mathbb{R}$

5. $val(\nu, [\alpha]\pi) = \mathit{true}$ iff for each trace $\sigma \in \tau(\alpha)$ that starts in $\operatorname{first}\sigma = \nu$, if $val(\sigma, \pi)$ is defined, then $val(\sigma, \pi) = \mathit{true}$.

6. $val(\nu, \langle\alpha\rangle\pi) = \mathit{true}$ iff there is a trace $\sigma \in \tau(\alpha)$ starting in $\operatorname{first}\sigma = \nu$, such that $val(\sigma, \pi) = \mathit{true}$.

For trace formulas, the *valuation* $val(\sigma, \cdot)$ with respect to trace σ is defined as:

1. If ϕ is a state formula, then $val(\sigma, \phi) = val(\operatorname{last}\sigma, \phi)$ if σ terminates, whereas $val(\sigma, \phi)$ is *not defined* if σ does not terminate.

2. $val(\sigma, \Box\phi) = \mathit{true}$ iff $val(\sigma_i(\zeta), \phi) = \mathit{true}$ holds for all positions (i, ζ) of σ with $\sigma_i(\zeta) \neq \Lambda$.

3. $val(\sigma, \Diamond\phi) = \mathit{true}$ iff $val(\sigma_i(\zeta), \phi) = \mathit{true}$ holds for some position (i, ζ) of σ with $\sigma_i(\zeta) \neq \Lambda$.

As usual, a (state) formula is *valid* if it is true in all states. Further for (state) formula ϕ and state ν we write $\nu \models \phi$ iff $val(\nu, \phi) = true$ and we write $\nu \not\models \phi$ iff $val(\nu, \phi) = false$. Likewise, for trace formula π and trace σ we write $\sigma \models \pi$ iff $val(\sigma, \pi) = true$ and we write $\sigma \not\models \pi$ iff $val(\sigma, \pi) = false$. In particular, we only write $\sigma \models \pi$ or $\sigma \not\models \pi$ if $val(\sigma, \pi)$ is defined, which it is not if π is a state formula and σ does not terminate. The points where a dTL property ϕ has to hold for the various combinations of temporal and dynamic modalities is illustrated in Figure 4.1.

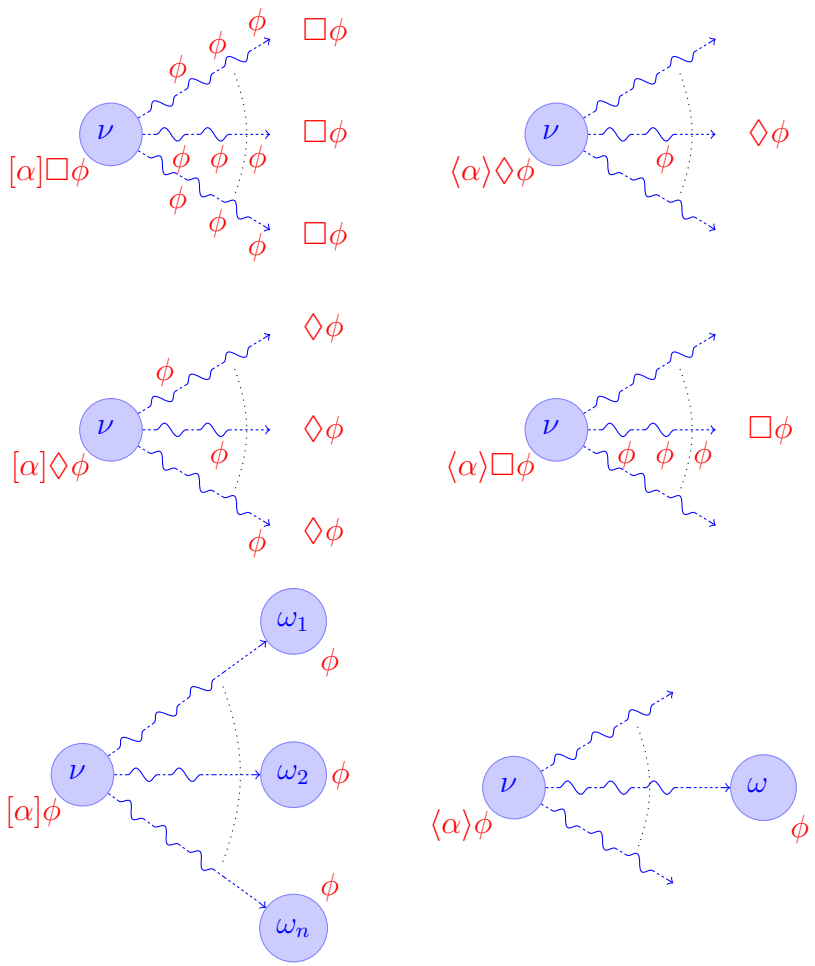


Figure 4.1.: Trace semantics of dTL formulas

4.3.3. Conservative Temporal Extension

The following result shows that the extension of dTL by temporal operators does not change the meaning of non-temporal $\mathsf{d}\mathcal{L}$ formulas. The trace semantics given in Definition 4.4 is equivalent to the final state reachability relation semantics given in Definition 2.6 for the sublogic $\mathsf{d}\mathcal{L}$ of dTL.

4.5 Proposition (Conservative temporal extension). *The logic* dTL *is a* conservative extension *of non-temporal* $\mathsf{d}\mathcal{L}$*, i.e., the set of valid* $\mathsf{d}\mathcal{L}$*-formulas is the same with respect to transition reachability semantics of* $\mathsf{d}\mathcal{L}$ *(Definition 2.6) as with respect to the trace semantics of* dTL *(Definition 4.4).*

The proof of Proposition 4.5 uses the following relationship of reachability and trace semantics of dTL programs, which agree on initial and final states.

4.6 Lemma (Trace relation). *For hybrid programs* $\alpha \in \mathrm{HP}(\Sigma, V)$*, we have*

$$\rho(\alpha) \;=\; \{(\operatorname{first}\sigma, \operatorname{last}\sigma) \;:\; \sigma \in \tau(\alpha) \text{ terminates}\} \;.$$

Proof. The proof follows an induction on the structure of α.

- The cases $x := \theta$, $x' = \theta$, and $\alpha \cup \beta$ are simple comparisons of the definitions 4.3 and 2.7.

- For $?\chi$, the reasoning splits into two directions. For the direction "$\supseteq$", assume $\sigma \in \tau(?\chi)$. We distinguish between two cases. If $val(\operatorname{first}\sigma, \chi) = \mathit{true}$, then $\sigma = (\hat{v})$ has length one, $\operatorname{last}\sigma = \operatorname{first}\sigma$, and $(\operatorname{first}\sigma, \operatorname{first}\sigma) \in \rho(\alpha)$. If, however, $val(\operatorname{first}\sigma, \chi) = \mathit{false}$, then $\sigma = (\hat{v}, \hat{\Lambda})$ does not terminate, hence, there is nothing to show. Conversely, for "$\subseteq$", assume $(v, v) \in \rho(?\chi)$, then $val(\nu, \chi) = \mathit{true}$ and $(\hat{v}) \in \tau(\alpha)$ satisfies the conditions on σ.

- For $\alpha;\beta$ and the direction "$\supseteq$", assume that $\sigma \circ \varsigma \in \tau(\alpha;\beta)$ terminates with $\sigma \in \tau(\alpha)$, $\varsigma \in \tau(\beta)$, and $\operatorname{last}\sigma = \operatorname{first}\varsigma$. Then, by induction hypothesis, we can assume that $(\operatorname{first}\sigma, \operatorname{last}\sigma) \in \rho(\alpha)$ and $(\operatorname{first}\varsigma, \operatorname{last}\varsigma) \in \rho(\beta)$. By the semantics of sequential composition, we have $(\operatorname{first}(\sigma \circ \varsigma), \operatorname{last}(\sigma \circ \varsigma)) \in \rho(\alpha;\beta)$. Conversely, for "$\subseteq$", assume that $(\nu, w) \in \rho(\alpha;\beta)$, i.e., let $(\nu, z) \in \rho(\alpha)$ and $(z, w) \in \rho(\beta)$. By induction hypothesis, there is a terminating trace $\sigma \in \tau(\alpha)$ with $\operatorname{first}\sigma = \nu$ and $\operatorname{last}\sigma = z$. Further, by induction hypothesis, there is a terminating $\varsigma \in \tau(\beta)$ with $\operatorname{first}\varsigma = z$ and $\operatorname{last}\varsigma = w$. Hence, $\sigma \circ \varsigma \in \tau(\alpha;\beta)$ terminates with $\operatorname{first}(\sigma \circ \varsigma) = \nu$ and $\operatorname{first}(\sigma \circ \varsigma) = w$.

- The case α^* is an inductive consequence of the sequential composition case. □

Proof of Proposition 4.5. The formulas of $\mathsf{d}\mathcal{L}$ are a subset of the dTL formulas. In the course of this proof, we use the notation $val_{d\mathcal{L}}(\nu, \cdot)$ to indicate that the $\mathsf{d}\mathcal{L}$ valuation from Definition 4.4 in Section 2.3 is used. For $\mathsf{d}\mathcal{L}$ formulas ψ, we show that the valuations with respect to Definition 4.4 and with respect to Definition 2.6 are the same for all states ν:

$$val(\nu, \psi) = val_{d\mathcal{L}}(\nu, \psi) \text{ for all } \nu \ .$$

We prove this by induction on the structure of ψ. The cases 1–3 of the definition of state formulas in Definition 4.1 are obvious. The other cases are proven as follows.

- If ψ has the form $[\alpha]\phi$, assume that $val(\nu, [\alpha]\phi) = \mathit{false}$. Then there is some terminating trace $\sigma \in \tau(\alpha)$ with $\text{first}\,\sigma = \nu$ such that $val(\text{last}\,\sigma, \phi) = \mathit{false}$. By induction hypothesis, this implies that $val_{d\mathcal{L}}(\text{last}\,\sigma, \phi) = \mathit{false}$. According to Lemma 4.6, $(\nu, \text{last}\,\sigma) \in \rho(\alpha)$ holds, which implies $val_{d\mathcal{L}}(\nu, [\alpha]\phi) = \mathit{false}$. For the converse direction, assume that $val_{d\mathcal{L}}(\nu, [\alpha]\phi) = \mathit{false}$. Then there is a $(\nu, w) \in \rho(\alpha)$ with $val_{d\mathcal{L}}(w, \phi) = \mathit{false}$. By Lemma 4.6, there is a terminating trace $\sigma \in \tau(\alpha)$ with $\text{first}\,\sigma = \nu$ and $\text{last}\,\sigma = w$. By induction hypothesis, $val(\text{last}\,\sigma, \phi) = \mathit{false}$. Thus, we can conclude that both $val(\sigma, \phi) = \mathit{false}$ and $val(\nu, [\alpha]\phi) = \mathit{false}$.

- The case $\psi = \langle\alpha\rangle\phi$ is proven accordingly. □

4.4. Safety Invariants in Train Control

In the European Train Control System (ETCS), trains are coordinated by decentralised Radio Block Centres (RBC), which grant or deny movement authorities (MA) dynamically to the individual trains by wireless communication. In emergencies, trains always have to stop within the MA issued by the RBC, see Figure 4.2. Following the reasoning pattern for traffic agents in [DHO03], each train negotiates with the RBC to extend its MA when approaching the end of its current MA. Since wireless communication takes time, this negotiation is initiated in due time before reaching m. To simplify the presentation, we adopt the assumption of Damm et al. [DHO03] that trains keep their desired speed (or at least their maximum speed limit) during negotiation. Before entering negotiation at some point ST (for start talking), the train still has sufficient distance to MA (it is in *far* mode) and can regulate its speed freely within the track limits.

As a model for train movements, we use the ideal-world model from Section 2.4. For a safe operation of multiple traffic agents, it is crucial that the MA is respected at *every* point in time during this protocol, not only at its end. Hence, we need to consider temporal safety invariants. For instance, when the train has entered the

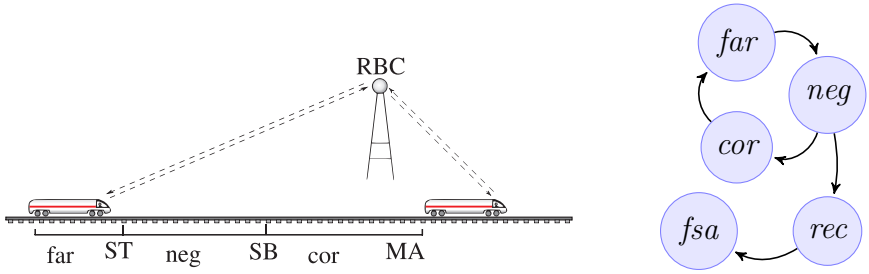


Figure 4.2.: ETCS train coordination protocol phases

negotiation phase at its current position z, dTL can analyse the following safety invariant of a part of the protocol cycle of the train controller:

$$\psi \rightarrow [neg; cor; drive]\Box(\ell \leq L \rightarrow z < m) \tag{4.1}$$

$$\text{where } neg \equiv z' = v, \ell' = 1$$

$$cor \equiv (?m - z < s; a := -b) \cup (?m - z \geq s; a := \dots) \tag{4.2}$$

$$drive \equiv z' = v, v' = a \ .$$

It expresses that—under a sanity condition ψ for parameters—a train will *always* remain within its MA m, as long as the accumulated RBC negotiation latency ℓ is at most L. We refer to the work of Faber and Meyer [FM06] for details on what kind of message passing contributes to ℓ. Like in [DHO03], we model the train to first negotiate while keeping a constant speed ($z' = v$) in *neg*. Thereafter, in *cor*, the train corrects its acceleration or brakes with force b (as a failsafe recovery manoeuvre) on the basis of the remaining distance ($m - z$). Finally, the train continues moving according to the differential equation system *drive* or, equivalently, $z'' = a$. Instead of manually choosing specific values for the free parameters of (4.1) as in [DHO03, FM06], we will use the techniques developed in this thesis to automatically synthesise constraints on the relationship of parameters that are required for a safe operation of cooperative train control.

4.5. A Verification Calculus for Safety Invariants

In this section, we introduce a sequent calculus for verifying temporal specifications of hybrid systems in dTL. With the basic idea being to perform a symbolic decomposition, hybrid programs are successively transformed into simpler logical formulas describing their effects. There, statements about the temporal behaviour of a hybrid program are successively reduced to corresponding non-temporal statements about the intermediate states.

The dTL calculus is presented in Figure 4.3 and inherits the propositional, first-order, and dynamic rules from $\mathsf{d}\mathcal{L}$. That is, it includes the propositional P-rules

from Figure 2.5 and either the free variable quantifier F-rules from Figure 2.5 or the simpler quantifier F-rules from Figure 3.3 that are based on side deductions. The D-rules and G-rules for handling non-temporal dynamic modalities are also inherited from Figure 2.5, except that D3–D4 are generalised to apply for formulas of the form $[\alpha \cup \beta]\pi$ where π is an arbitrary trace formula not just a state formula as in $\mathsf{d}\mathcal{L}$. Thus, π may begin with $\Box$ or $\Diamond$, which is why the rules are repeated in this generalised form as T8 and T1 in Figure 4.3.

The new T-rules in Figure 4.3 successively transform temporal specifications of hybrid programs into non-temporal $\mathsf{d}\mathcal{L}$ formulas. The idea underlying this transformation is to decompose hybrid programs and recursively augment intermediate state transitions with appropriate specifications.

$$\text{(T1)}\ \frac{[\alpha]\pi \land [\beta]\pi}{[\alpha \cup \beta]\pi} \qquad \text{(T8)}\ \frac{\langle\alpha\rangle\pi \lor \langle\beta\rangle\pi}{\langle\alpha \cup \beta\rangle\pi}$$

$$\text{(T2)}\ \frac{[\alpha]\Box\phi \land [\alpha][\beta]\Box\phi}{[\alpha;\beta]\Box\phi} \qquad \text{(T9)}\ \frac{\langle\alpha\rangle\Diamond\phi \lor \langle\alpha\rangle\langle\beta\rangle\Diamond\phi}{\langle\alpha;\beta\rangle\Diamond\phi}$$

$$\text{(T3)}\ \frac{\phi}{[?\chi]\Box\phi} \qquad \text{(T10)}\ \frac{\phi}{\langle?\chi\rangle\Diamond\phi}$$

$$\text{(T4)}\ \frac{\phi \land [x := \theta]\phi}{[x := \theta]\Box\phi} \qquad \text{(T11)}\ \frac{\phi \lor \langle x := \theta\rangle\phi}{\langle x := \theta\rangle\Diamond\phi}$$

$$\text{(T5)}\ \frac{[x' = \theta]\phi}{[x' = \theta]\Box\phi} \qquad \text{(T12)}\ \frac{\langle x' = \theta\rangle\phi}{\langle x' = \theta\rangle\Diamond\phi}$$

$$\text{(T6)}\ \frac{[\alpha;\alpha^*]\Box\phi}{[\alpha^*]\Box\phi} \qquad \text{(T13)}\ \frac{\langle\alpha;\alpha^*\rangle\Diamond\phi}{\langle\alpha^*\rangle\Diamond\phi}$$

$$\text{(T7)}\ \frac{[\alpha^*][\alpha]\Box\phi}{[\alpha^*]\Box\phi} \qquad \text{(T14)}\ \frac{\langle\alpha^*\rangle\langle\alpha\rangle\Diamond\phi}{\langle\alpha^*\rangle\Diamond\phi}$$

In these rules, ϕ and ψ are (state) formulas, whereas π is a trace formula. Unlike ϕ and ψ, the trace formula π may thus begin with temporal modalities $\Box$ or $\Diamond$.

Figure 4.3.: Rule schemata of the temporal dynamic dTL verification calculus

Rule T2 decomposes invariants of $\alpha;\beta$ into an invariant of α and an invariant of β that holds when β is started in *any* final state of α. The difference to D2 is that T2 also checks safety invariant ϕ at the symbolic state in between the execution of α and β, and recursively so because of the temporal modality $\Box$. T4 expresses that invariants of assignments need to hold before and after the discrete change (similarly for T3, except that tests do not lead to a state change). T5 can directly reduce invariants of continuous evolutions to non-temporal formulas as restrictions of solutions of differential equations are themselves solutions of different duration

and thus already included in the evolutions of $x' = \theta$. In particular, observe that the handling of differential equations within hybrid systems is fully encapsulated within the D-rule fragment. The rules T5, T12, T4, and T11 directly generalise to discrete jump sets and systems of differential equations or even DAF.

The (optional) iteration rule T6 can partially unwind loops. It relies on T2 and is simpler than D6, because the other rules will inductively produce a premiss that ϕ holds in the current state. The dual rules T9–T13 work similarly.

In Chapter 2 and Chapter 3, the primary means for handling loops are the invariant induction (G3) and variant convergence (G4) rules. Here, we take a different, completely modular approach for verifying temporal properties of loops based on the $\mathsf{d}\mathcal{L}$ capabilities for verifying non-temporal properties of loops. Rule T7 and T14 actually *define* temporal properties of loops inductively. Rule T7 expresses that ϕ holds at all times during repetitions of α, iff, after repeating α any number of times, ϕ holds at all times during one execution of α. Dually, T14 expresses that α holds at some time during repetitions of α, iff, after some number of repetitions of α, formula ϕ holds at some point during one execution of α. In this context, the non-temporal modality $\langle\alpha^*\rangle$ can be thought of as skipping over to the iteration of α during which ϕ actually occurs, as expressed by the nested dTL-formula $\langle\alpha\rangle\Diamond\phi$. The inductive definition rules T7 and T14 completely reduce temporal properties of loops to dTL-properties of standard non-temporal $\mathsf{d}\mathcal{L}$-modalities such that standard induction (G3) or convergence rules (G4) can be used for the outer non-temporal modality of the loop. Hence, after applying the inductive loop definition rules T7 and T14, the standard $\mathsf{d}\mathcal{L}$ loop invariant and variant rules can be used for verifying temporal properties of loops without change, except that the postcondition contains temporal modalities.

Rules for handling $[\alpha]\Diamond\phi$ and $\langle\alpha\rangle\Box\phi$ are discussed in Section 4.9. Finally, provability in the dTL calculus is denoted by $\Phi \vdash_{\text{dTL}} \psi$, and defined according to Definition 2.12.

4.6. Soundness

The following result shows that verification with the dTL calculus always produces correct results about safety of hybrid systems, i.e., the dTL calculus is sound.

4.7 Theorem (Soundness). *The* dTL *calculus is* sound*, i.e., derivable (state) formulas are valid.*

Proof. We show that all rules of the dTL calculus are *locally sound*, i.e., for all states ν, the conclusion of a rule is true in state ν when all premisses are true in ν. Let ν be any state. For each rule we have to show that the conclusion is true in ν assuming the premisses are true in ν. The P-rules are locally sound by Theorem 2.15. Inductively, the soundness of the D-rules follows from Proposition 4.5

and local soundness of the corresponding rules in $\mathsf{d}\mathcal{L}$. The proof for the generalisation in D4 and D3 to path formulas π is a straightforward extension. The F-rules are sound by Theorem 2.15 or Theorem 3.25, respectively.

T2 Assuming $\nu \models [\alpha]\Box\phi$ and $\nu \models [\alpha][\beta]\Box\phi$. Let $\sigma \in \tau(\alpha;\beta)$, i.e., $\sigma = \varrho \circ \varsigma$ with $\text{first}\,\sigma = \nu$, $\varrho \in \tau(\alpha)$, and $\varsigma \in \tau(\beta)$. If ϱ does not terminate, then $\sigma = \varrho \in \tau(\alpha)$ and $\sigma \models \Box\phi$ by premise. If, instead, ϱ terminates with $\text{last}\,\varrho = \text{first}\,\varsigma$, then $\varrho \models \Box\phi$ by premise. Further, we know that $\nu \models [\alpha][\beta]\Box\phi$. In particular we have for the trace $\varrho \in \tau(\alpha)$, that $\text{last}\,\varrho \models [\beta]\Box\phi$. Thus, $\varsigma \models \Box\phi$ because $\varsigma \in \tau(\beta)$ starts at $\text{first}\,\varsigma = \text{last}\,\varrho$. By composition, $\varrho \circ \varsigma \models \Box\phi$. As $\sigma = \varrho \circ \varsigma$ was arbitrary, we can conclude $\nu \models [\alpha;\beta]\Box\phi$. The converse direction holds, as all traces of α are prefixes of traces of $\alpha;\beta$. Hence, the assumption $\nu \models [\alpha;\beta]\Box\phi$ directly implies $\nu \models [\alpha]\Box\phi$. Further, all traces of β that begin at a state reachable from ν by α are suffixes of traces of $\alpha;\beta$ starting in ν. Hence, $\nu \models [\alpha][\beta]\Box\phi$ is implied as well.

T3 Soundness of T3 is obvious, since, by premise, we can assume $\nu \models \phi$, and there is nothing to show for Λ states according to Definition 4.4. Conversely, $\hat{\nu}$ is a prefix of all traces in $\tau(?\chi)$ that start in ν.

T4 Assuming $\nu \models \phi$ and $\nu \models [x := \theta]\phi$, we have to show that $\nu \models [x := \theta]\Box\phi$. Let $\sigma \in \tau(x := \theta)$ be any trace with $\text{first}\,\sigma = \nu$, i.e., $\sigma = (\hat{\nu}, \hat{\omega})$ by Definition 4.3. Hence, the only two states we need to consider are $\sigma_0(0) = \nu$ and $\sigma_1(0) = \omega$. By premise, $\sigma_0(0) = \nu$ yields $\sigma_0(0) \models \phi$. Similarly, for the state $\sigma_1(0) = \text{last}\,\sigma = \omega$, the premise gives $\sigma_1(0) \models \phi$. The converse direction is similar.

T5 We prove that T5 is locally sound by contraposition. For this, assume that $\nu \not\models [x' = \theta]\Box\phi$, then there is a trace $\sigma = (\varphi) \in \tau(x' = \theta)$ starting in $\text{first}\,\sigma = \nu$ and $\sigma \not\models \Box\phi$. Hence, there is a position $(0, \zeta)$ of σ with $\sigma_0(\zeta) \not\models \phi$. Now φ restricted to the interval $[0, \zeta]$ also solves differential equation $x' = \theta$. Thus, $(\varphi|_{[0,\zeta]}) \not\models \phi$ as $\varphi(\zeta) \not\models \phi$, since the last state is $\varphi(\zeta)$. Consequently, $\nu \not\models [x' = \theta]\phi$. The converse direction is obvious as $\text{last}\,\sigma$ always is a state occurring during σ, hence $\nu \not\models [x' = \theta]\phi$ immediately implies $\nu \not\models [x' = \theta]\Box\phi$.

T6 By contraposition, assume that $\nu \not\models [\alpha^*]\Box\phi$, then there is an $n \in \mathbb{N}$ and a trace $\sigma \in \tau(\alpha^n)$ with $\text{first}\,\sigma = \nu$ such that $\sigma \not\models \Box\phi$. There are two cases. If $n > 0$ then $\sigma \in \tau(\alpha;\alpha^*)$, thus $\nu \not\models [\alpha;\alpha^*]\Box\phi$. If, however, $n = 0$, then $\sigma = (\hat{\nu})$ and $\nu \not\models \phi$. Hence, all traces $\varsigma \in \tau(\alpha;\alpha^*)$ with $\text{first}\,\varsigma = \nu$ satisfy $\varsigma \not\models \Box\phi$. Finally, it is easy to see that all programs have at least one such trace ς (when V is nonempty) that witnesses $\nu \not\models [\alpha;\alpha^*]\Box\phi$. The converse direction is easy as all behaviour of $\alpha;\alpha^*$ is subsumed by α^*, i.e., $\tau(\alpha;\alpha^*) \subseteq \tau(\alpha^*)$.

T7 Clearly, using that $\tau(\alpha^*) \supseteq \tau(\alpha^*;\alpha)$, the set of states along the traces of α^* at which ϕ needs to be true for the premiss is a subset of the corresponding set for the conclusion. Hence, the conclusion entails the premiss. Conversely, all states during traces of α^* are also reachable by iterating α sufficiently often to completion and then following a single trace of α. In detail: If $\nu \not\models [\alpha^*]\Box\phi$, then there is a trace $\sigma \in \tau(\alpha^*)$ on which $\neg\phi$ holds true at some state, say, at $\sigma_i(\zeta) \neq \Lambda$. Let $n \geq 0$ be the (maximum) number of complete repetitions of α along σ before discrete step index i. That is, there is some discrete step index $i_n < i$ such that the prefix $\varrho = (\sigma_0, \ldots, \sigma_{i_n}) \in \tau(\alpha^n)$ of σ consists of n complete repetitions of α and the suffix $\varsigma = (\sigma_{i_n+1}, \sigma_{i_n+2}, \ldots) \in \tau(\alpha^*)$ starts with a trace of α during which $\neg\phi$ occurs at point $\sigma_i(\zeta)$, namely at relative position $(i - (i_n + 1), \zeta)$. Let $\varsigma' \in \tau(\alpha)$ be this prefix of ς. Consequently, $\varsigma' \models \langle\alpha\rangle\Diamond\neg\phi$ and, the trace $\varrho \circ \varsigma'$ is a witness for $\nu \models \langle\alpha^*\rangle\langle\alpha\rangle\Diamond\neg\phi$.

The proofs for T9–T14 are dual, since $\langle\alpha\rangle\Diamond\phi$ is equivalent to $\neg[\alpha]\Box\neg\phi$ by duality. □

4.7. Completeness

In this section, we show that the strictly modular dTL calculus enables us to lift the relative completeness theorem 2.17 for $\mathsf{d}\mathcal{L}$ to dTL.

4.7.1. Incompleteness

The incompleteness theorem 2.16 directly generalises to temporal and non-temporal properties of dTL.

4.8 Theorem (Incompleteness). *The discrete and continuous fragments of* dTL *are non-axiomatisable for temporal safety ($[\alpha]\Box\phi$) and non-temporal ($[\alpha]\phi$) fragments of* dTL*. Hence, valid* dTL *formulas are not always derivable.*

Proof. We show that the discrete and continuous fragments of the following purely temporal and non-temporal fragments of dTL non-axiomatisable:

1. the fragment that only contains modalities of the form $[\alpha]\Box\phi$ and $\langle\alpha\rangle\Diamond\phi$
2. the fragment that only contains $[\alpha]\phi$ and $\langle\alpha\rangle\phi$ ($\mathsf{d}\mathcal{L}$ fragment).

Case 2 is a consequence of the corresponding incompleteness result Theorem 2.16 fragments of the sublogic $\mathsf{d}\mathcal{L}$, which carries over to the extension dTL by Proposition 4.5.

For Case 1, we prove that natural numbers are definable amongst the real numbers domain in both fragments, quite similar to the proof of Theorem 2.16. Then these fragments extend first-order *integer* arithmetic such that the incompleteness theorem of Gödel applies.

- Natural numbers are definable in the discrete fragment without continuous evolutions $x' = \theta$ using repetitive additions:

$$nat(n) \;\leftrightarrow\; \langle x := 0; (x := x + 1)^*\rangle\Diamond x = n \;\; .$$

- In the continuous fragment, natural numbers are definable as:

$$nat(n) \;\leftrightarrow\; \exists s\, \exists c\, (s = 0 \wedge c = 1 \wedge \langle s' = c, c' = -s, \tau' = 1\rangle\Diamond(s = 0 \wedge \tau = n)) \;\; .$$

  These ODEs have *sin* and *cos* as unique solutions for s and c, respectively. Their zeros characterise an isomorphic copy of natural numbers, scaled by π. □

4.7.2. Relative Completeness

Due to the modular construction of the dTL calculus, we can lift the major relative completeness result Theorem 2.17 from $\mathsf{d}\mathcal{L}$ to dTL. By proving dTL completeness relative to Theorem 2.17, we essentially show that dTL is complete relative to $\mathsf{d}\mathcal{L}$, which directly implies that dTL is even complete relative to FOD using Theorem 2.17 by a standard argument. Again, we restrict our attention to homogeneous combinations of path and trace quantifiers like $[\alpha]\Box\phi$ or $\langle\alpha\rangle\Diamond\phi$.

4.9 Theorem (Relative completeness). *The* dTL *calculus is* complete relative to *FOD, i.e., every valid* dTL *formula can be derived from FOD-tautologies.*

Proof Outline. The proof is a simple extension of the proof of Theorem 2.17, because the dTL calculus successively reduces temporal properties to non-temporal properties and, in particular, handles loops by inductive definition rules in terms of $\mathsf{d}\mathcal{L}$ modalities. The T-rules in Figure 4.3 transform temporal formulas to simpler formulas, i.e., where the temporal modalities occur after simpler programs (T1, T7, T8, T14) or disappear completely (T3–T5 and T10–T12). Hence, the inductive relative completeness proof in Section 2.7.2 directly generalises to dTL with the following addition: After applying T7 or T14, loops are ultimately handled by the standard $\mathsf{d}\mathcal{L}$ rules G3 and G4. To show that sufficiently strong invariants and variants exist for the temporal postcondition $[\alpha]\Box\phi$ and $\langle\alpha\rangle\Diamond\phi$, we only have to show that such temporal formulas are expressible in FOD, i.e., Lemma 2.20 generalises to dTL. □

This result, which we prove formally in the remainder of Section 4.7.2, gives a formal justification that the dTL calculus reduces temporal properties to non-temporal $\mathsf{d}\mathcal{L}$ properties.

4.7.3. Expressibility and Rendition of Hybrid Trace Semantics

The central step for lifting the $\mathsf{d}\mathcal{L}$ completeness proof to dTL is the following: To show that, after applying T7 or T14, sufficiently strong invariants and variants for dTL postconditions can be expressed in $\mathsf{d}\mathcal{L}$ for G3 or G4 to be able to prove the result, we show that the trace semantics of hybrid programs can be characterised in FOD.

4.10 Lemma (Program trace rendition). *For every hybrid program α with variables $\vec{x} = x_1, \ldots, x_k$ there is a FOD-formula $\mathcal{T}_\alpha(\vec{x}, \vec{v})$ with variables among the $2k$ distinct variables $\vec{x} = x_1, \ldots, x_k$ and $\vec{v} = v_1, \ldots, v_k$ such that*

$$\vDash \mathcal{T}_\alpha(\vec{x}, \vec{v}) \leftrightarrow \langle\alpha\rangle\Diamond \vec{x} = \vec{v}$$

or, equivalently, for every ν, we have that $\nu \models \mathcal{T}_\alpha(\vec{x}, \vec{v})$ iff

$\sigma_i(\zeta) = \nu[\vec{x} \mapsto val(\nu, \vec{v})]$ *for a position (i, ζ) of some trace $\sigma \in \tau(\alpha)$ starting in ν .*

$$\begin{aligned}
\mathcal{T}_{x_1:=\theta_1,..,x_k:=\theta_k}(\vec{x}, \vec{v}) &\equiv \vec{x} = \vec{v} \vee \mathcal{S}_{x_1:=\theta_1,..,x_k:=\theta_k}(\vec{x}, \vec{v}) \\
\mathcal{T}_{x'_1=\theta_1,..,x'_k=\theta_k \,\&\, \chi}(\vec{x}, \vec{v}) &\equiv \mathcal{S}_{x'_1=\theta_1,..,x'_k=\theta_k \,\&\, \chi}(\vec{x}, \vec{v}) \\
\mathcal{T}_{?\chi}(\vec{x}, \vec{v}) &\equiv \mathcal{S}_{?\chi}(\vec{x}, \vec{v}) \\
\mathcal{T}_{\beta\cup\gamma}(\vec{x}, \vec{v}) &\equiv \mathcal{T}_\beta(\vec{x}, \vec{v}) \vee \mathcal{T}_\gamma(\vec{x}, \vec{v}) \\
\mathcal{T}_{\beta;\gamma}(\vec{x}, \vec{v}) &\equiv \mathcal{T}_\beta(\vec{x}, \vec{v}) \vee \exists \vec{z}\, (\mathcal{S}_\beta(\vec{x}, \vec{z}) \wedge \mathcal{T}_\gamma(\vec{z}, \vec{v})) \\
\mathcal{T}_{\beta^*}(\vec{x}, \vec{v}) &\equiv \exists \vec{z}\, (\mathcal{S}_{\beta^*}(\vec{x}, \vec{z}) \wedge \mathcal{T}_\beta(\vec{z}, \vec{v}))
\end{aligned}$$

Figure 4.4.: Explicit rendition of hybrid program trace semantics in FOD

Proof. The proof is similar to that of Lemma 2.19, yet using the definition in Figure 4.4. We recurse on corresponding characterisations from Lemma 2.19, which simplifies the characterisation of $\mathcal{T}_\alpha(\vec{x}, \vec{v})$, because we only have to augment $\mathcal{S}_\alpha(\vec{x}, \vec{v})$ by appropriate disjunctions for intermediate states, which can again be defined in terms of $\mathcal{S}_\alpha(\vec{x}, \vec{v})$, recursively.

For instance, $\mathcal{T}_{\beta^*}(\vec{x}, \vec{v})$ characterises the states reachable during traces of β^* as the states reachable during traces of β that start after running β^* to completion for some number of iterations. □

Using this program rendition to characterise temporal trace modalities, Lemma 2.20 generalises immediately to dTL as follows:

4.11 Lemma (Expressibility). *Logic* dTL *is* expressible *in FOD: for all* dTL *formulas* $\phi \in \mathrm{Fml}(\Sigma, V)$ *there is a FOD-formula* $\phi^{\#} \in \mathrm{Fml}_{FOD}(\Sigma, V)$ *that is equivalent, i.e.,* $\vDash \phi \leftrightarrow \phi^{\#}$. *The converse holds trivially.*

Proof. The proof is by a structural induction identical to that in the proof of Lemma 2.20 with the following additions:

1. The case where ϕ is of the form $\langle\alpha\rangle\Diamond\psi$ is a consequence of Lemma 4.10:

$$\vDash \quad \langle\alpha\rangle\Diamond\psi \leftrightarrow \exists\vec{v}\,(\mathcal{T}_\alpha(\vec{x},\vec{v}) \wedge \psi^{\#^{\vec{v}}_{\vec{x}}}) \ .$$

2. The case where ϕ is $[\alpha]\Box\psi$ is again a consequence of Lemma 4.10:

$$\vDash \quad [\alpha]\Box\psi \leftrightarrow \forall\vec{v}\,(\mathcal{T}_\alpha(\vec{x},\vec{v}) \rightarrow \psi^{\#^{\vec{v}}_{\vec{x}}}) \ . \qquad \Box$$

4.7.4. Modular Relative Completeness Proof for the Differential Temporal Dynamic Logic Calculus

Now we assemble the proof of Theorem 4.9 from the previous results following a simplified form of the proof of Theorem 2.17, since we can apply Theorem 2.17 for $\mathsf{d}\mathcal{L}$ formulas.

Proof of Theorem 4.9. The proof is a simple extension of the relative completeness theorem 2.17 for $\mathsf{d}\mathcal{L}$. Unlike for the rules of the $\mathsf{d}\mathcal{L}$ calculus, all new T-rules are symmetric, hence perform equivalent transformations. Consequently, whenever their conclusion is valid, their premiss is valid and of smaller complexity (temporal modalities occur after simpler programs), hence derivable by induction hypothesis.

For instance, in analogy to the induction step for loops in Proposition 2.23, let $\vDash F \rightarrow [\beta^*]\Box G$, then $\vDash F \rightarrow [\beta^*][\beta]\Box G$. By Lemma 4.11, there is a $\mathsf{d}\mathcal{L}$ formula or even FOD formula $([\beta]\Box G)^{\#}$ that characterises the temporal postcondition equivalently, i.e., such that $\vDash ([\beta]\Box G)^{\#} \leftrightarrow [\beta]\Box G$. By induction hypothesis, we can derive the simpler formula $\vdash_{\mathcal{D}} ([\beta]\Box G)^{\#} \rightarrow [\beta]\Box G$. Using Lemma 2.22, we conclude $\vdash_{\mathcal{D}} \forall^{\beta}(([\beta]\Box G)^{\#} \rightarrow [\beta]\Box G)$, thus $[\beta^*]([\beta]\Box G)^{\#} \vdash_{\mathcal{D}} [\beta^*][\beta]\Box G$ is derivable by G1. Furthermore, $\vDash F \rightarrow [\beta^*]([\beta]\Box G)^{\#}$ is a valid $\mathsf{d}\mathcal{L}$ formula and, thus, $F \vdash_{\mathcal{D}} [\beta^*]([\beta]\Box G)^{\#}$ is derivable by Theorem 2.17. Combining these derivations by a cut with $[\beta^*]([\beta]\Box G)^{\#}$, we derive $F \vdash_{\mathcal{D}} [\beta^*]\Box G$. Since the T-rules perform a modular reduction to non-temporal D-rules, the case $\vDash F \rightarrow \langle\beta^*\rangle\Diamond G$ is almost identical here, because the differences between variant and invariant rules have already been captured in the proof of Theorem 2.17. $\Box$

4.8. Verification of Train Control Safety Invariants

Continuing the ETCS study from Section 4.4, we consider a slightly simplified version of equation (4.1) that gives a more concise proof. By a safe abstraction (provable in dTL), we simplify *cor* to permit braking even when $m - z \geq s$, since braking remains safe with respect to $z < m$. We use the following abbreviations in addition to (4.1):

$$\begin{aligned} \psi &\equiv z < m \land v > 0 \land \ell = 0 \land L \geq 0 \\ \phi &\equiv \ell \leq L \rightarrow z < m \\ cor &\equiv a := -b \cup (?m - z \geq s; a := \dots) \ . \end{aligned}$$

Within the following proof, $\langle\!|\rangle$ brackets are used instead of modalities to visually identify the update prefix (Definition 2.12). The dTL proof of the safety invariant in (4.1) splits into two cases that correspond to the respective protocol phases:

$$\dfrac{\dfrac{\dots}{\psi \vdash [neg]\Box\phi} \qquad \dfrac{\dots}{\psi \vdash [neg][cor; drive]\Box\phi}}{\psi \vdash [neg; cor; drive]\Box\phi}\,\text{T2}}{\vdash \psi \rightarrow [neg; cor; drive]\Box\phi}\,\text{P7}$$

There, the left branch proves that ϕ holds while negotiating and is as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\psi \vdash Lv + z < m}{\psi \vdash \forall l{\geq}0\,(l \leq L \rightarrow lv + z < m)}\,\text{F1,F3}}{\psi \vdash \forall l{\geq}0\,\langle\!|z := lv + z, \ell := l|\!\rangle\phi}\,\text{D10,D9}}{\psi \vdash [neg]\phi}\,\text{D12}}{\psi \vdash [neg]\Box\phi}\,\text{T5}$$

The right branch shows that ϕ continues to hold after negotiation has completed when continuing with an adjusted acceleration a in *cor*; *drive*:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\psi, \ell{\geq}0 \vdash v^2 < 2b(m - Lv - z) \land Lv + z < m}{\psi, \ell{\geq}0 \vdash \langle\!|z := \ell v{+}z, a := -b|\!\rangle\forall t{\geq}0\,(\ell{\leq}L \rightarrow \frac{a}{2}t^2{+}vt{+}z{<}m)}\,\text{F3}}{\psi, \ell{\geq}0 \vdash \langle\!|z := \ell v{+}z, a := -b|\!\rangle\forall t{\geq}0\,\langle\!|z := \frac{a}{2}t^2{+}vt{+}z|\!\rangle\phi}\,\text{D10,D9}}{\psi, \ell{\geq}0 \vdash \langle\!|z := \ell v{+}z, a := -b|\!\rangle[drive]\Box\phi}\,\text{T5,D12}\ \triangleright}{\psi, \ell{\geq}0 \vdash \langle\!|z := \ell v{+}z|\!\rangle[cor][drive]\Box\phi}\,\text{D4}\ \triangleright}{\psi, \ell{\geq}0 \vdash \langle\!|z := \ell v{+}z|\!\rangle[cor; drive]\Box\phi}\,\text{T2}}{\psi \vdash \ell{\geq}0 \rightarrow \langle\!|z := \ell v{+}z|\!\rangle[cor; drive]\Box\phi}\,\text{P7}}{\psi \vdash \forall \ell{\geq}0\,\langle\!|z := \ell v{+}z|\!\rangle[cor; drive]\Box\phi}\,\text{F1}}{\psi \vdash [neg][cor; drive]\Box\phi}\,\text{D12}$$

The application of T2 in this latter case spawns a third case (marked with $\triangleright$) to show that ϕ holds during *cor*. However, the reasoning in this third case is subsumed

by the cases above, since the changes on a in *cor* do not interfere with condition ϕ. Generally, this optimisation of T2 is applicable whenever the modified vocabulary is disjoint from ϕ. Here, D12 and F3 are implemented in Mathematica to handle evolutions.

The leaves of the proof branches above can even be used to automatically *synthesise parameter constraints* that are necessary to avoid MA violation. The parametric safety constraint obtained by combining the open conditions conjunctively is $Lv + z < m \land v^2 < 2b(m - Lv - z)$. It simplifies to $v^2 < 2b(m - Lv - z)$ because $b > 0$. This yields bounds for the speed limit and negotiation latency in order to guarantee safe driving and closing of the proof. Similarly, D4 leads to a branch for the case $[?m - z \geq s; a := \ldots]$, from which corresponding conditions about the safety envelope s can be derived depending on the particular speed controller, similar to what we have shown in Section 2.9.

4.9. Liveness by Quantifier Alternation

Liveness specifications of the form $[\alpha]\Diamond\phi$ or $\langle\alpha\rangle\Box\phi$ are sophisticated (Σ^1_1-hard because they can express infinite occurrence in Turing machines). Beckert and Schlager [BS01] say they failed to find sound rules for a discrete case that corresponds to $[\alpha;\beta]\Diamond\phi$.

For *finitary liveness semantics*, we accomplish this as follows. In this section, we modify the meaning of $[\alpha]\Diamond\phi$ to refer to all *terminating* traces of α. Then, the straightforward generalisation T15 in Figure 4.5 is sound, even in the hybrid case. But T15 still leads to an incomplete axiomatisation as it does not cover the case where, in some traces, ϕ becomes true at some point during α, and in other traces, ϕ only becomes true during β. To overcome this limitation, we use a program transformation approach. We instrument the hybrid program to monitor the occurrence of ϕ during all changes: In T16, $\check{\alpha}$ results from replacing all occurrences of $x := \theta$ by $x := \theta; ?\phi \to t = 1$ and $x' = \theta$ by $x' = \theta \,\&\, (\phi \to t = 1)$. The latter is a continuous evolution restricted to the region of the state space that satisfies $\phi \to t = 1$. The effect is that t detects whether ϕ has occurred during any change in α. In particular, t is guaranteed to be 1 after all runs, if ϕ occurs at least once along all traces of α. This trick directly works for first-order conditions ϕ. Using the combination presented in [Pla07g], nominals can be used as state labels to address the same issue for general ϕ.

4.12 Proposition (Local soundness). *The rules in Figure 4.5 are locally sound for finitary liveness semantics.*

Proof. Let ν be any state.

T15 Assuming that the premiss is true, we need to consider two cases corresponding to the two formulas of its succedent. If $\nu \models [\alpha]\Diamond\phi$, then obviously

$$\text{(T15)}\quad \frac{\vdash [\alpha]\Diamond\phi, [\alpha][\beta]\Diamond\phi}{\vdash [\alpha;\beta]\Diamond\phi} \qquad \text{(T16)}\quad \frac{\phi \vee \forall t\,[\breve{\alpha}]t = 1}{[\alpha]\Diamond\phi}$$

Figure 4.5.: Transformation rules for alternating temporal path and trace quantifiers

$\nu \models [\alpha;\beta]\Diamond\phi$, as every trace of $\alpha;\beta$ has a trace of α as prefix, during which ϕ holds at least once. If, however, $\nu \models [\alpha][\beta]\Diamond\phi$, then ϕ occurs at least once during all traces that start in a state reachable from ν by α. Let $\varrho \circ \varsigma \in \tau(\alpha;\beta)$ with first $\varrho = \nu$, $\varrho \in \tau(\alpha)$ and $\varsigma \in \tau(\beta)$. In finitary liveness semantics, $\varrho\circ\varsigma$ can be assumed to terminate (otherwise there is nothing to show). Then, last ϱ is a state reachable from ν by α, hence $\varsigma \models \Diamond\phi$. In particular, $\varrho \circ \varsigma \models \Diamond\phi$.

T16 For the soundness of T16, first observe that the truth of $val(\nu,\phi)$ of ϕ depends on the state ν, hence it can only be affected during state changes. Further, the only actual changes of valuations happen during discrete jumps $x := \theta$ or continuous evolutions $x' = \theta$. All other system actions only cause control flow effects but no elementary state changes. Assume the premiss is true in a state ν. If $\nu \models \phi$, the conjecture is obvious. Hence, assume $\nu \models \forall t\,[\breve{\alpha}]t = 1$. Suppose $\nu \not\models [\alpha]\phi$,then there is a trace $\sigma \in \tau(\alpha)$ with $\sigma \not\models \Diamond\phi$. Then, this trace directly corresponds to a trace $\breve{\sigma}$ of $\breve{\alpha}$ in which all $\phi \to t = 1$ conditions are trivially satisfied as ϕ never holds. As there are no changes of the fresh variable t during $\breve{\alpha}$, the value of t remains constant during $\breve{\sigma}$. But then we can conclude that there is a trace, which is essentially the same as $\breve{\sigma}$, except for the constant valuation of the fresh variable t on which no conditions are imposed, hence $t = 0$ is possible. As these traces terminate in finitary liveness semantics, we can conclude $\nu \not\models \forall t\,[\breve{\alpha}]t = 1$, which is a contradiction. Conversely for equivalence of premiss and conclusion, assume $\nu \not\models \phi \vee \forall t\,[\breve{\alpha}]t = 1$. Then, the initial state ν does not satisfy ϕ and it is possible for $\breve{\alpha}$ to execute along a terminating trace σ that permits t to be $\neq 1$. Suppose there was a position (i,ζ) of σ at which $\sigma_i(\zeta) \models \phi$. Without loss of generality, we can assume (i,ζ) to be the first such position. Then, the hybrid action which regulates σ_i is accompanied by an immediate condition that $\phi \to t = 1$, hence $t = 1$ holds if σ terminates. Since the fresh variable t is rigid (is never changed during $\breve{\alpha}$) and σ terminates in finitary liveness semantics, we conclude $val(\operatorname{last}\sigma, t) = 1$, which is a contradiction. □

4.10. Summary

For reasoning about hybrid systems, we have introduced a temporal dynamic logic, dTL, with modal path quantifiers over traces and temporal quantifiers along the

traces. It combines the capabilities of dynamic logic [HKT00] to reason about possible system behaviour with the power of temporal logic [Pnu77, EC82, EH86] in reasoning about the behaviour along traces. Furthermore, we have presented a calculus for verifying temporal safety specifications of hybrid programs in dTL.

Our sequent calculus for dTL is a completely modular combination of temporal and non-temporal reasoning. Temporal formulas are handled using rules that augment intermediate state transitions with corresponding sub-specifications. Purely non-temporal $\mathsf{d}\mathcal{L}$ rules handle the effects of discrete and continuous evolution. The modular nature of the dTL calculus further enables us to lift the relative completeness result from $\mathsf{d}\mathcal{L}$ to dTL.

As an example, we demonstrate that our logic is suitable for reasoning about safety invariants in the European Train Control System. Further, we have successfully applied our calculus to automatically synthesise (nonlinear) parametric safety constraints for this system.

Future work includes extending dTL with CTL*-like [EH86] formulas of the form $[\alpha](\psi \wedge \Box\phi)$ to avoid splitting of the proof into two very similar sub-proofs for temporal parts $[\alpha]\Box\phi$ and non-temporal parts $[\alpha]\psi$ arising in T2. Our combination of temporal logic with dynamic logic is more suitable for this purpose than previous approaches for discret systems [BS01], since dTL has uniform modalities and uniform semantics for temporal and non-temporal specifications. This extension will also simplify the treatment of alternating liveness quantifiers conceptually.

Part II.

Automated Theorem Proving for Hybrid Systems

Chapter 5.

Deduction Modulo Real Algebraic and Computer Algebraic Constraints

Contents

Synopsis

We show how deductive, real algebraic, and computer algebraic methods can be combined for verifying hybrid systems in an automated theorem proving approach. In particular, we highlight the interaction of deductive and algebraic reasoning that is used for handling the joint discrete and continuous behaviour of hybrid systems. Systematically, we derive a canonical *tableau procedure modulo* from the calculus of differential dynamic logic. We delineate the nondeterminisms in the tableau procedure carefully and analyse their practical impact in the presence of computationally expensive handling of real algebraic constraints. Based on the experience with larger case

studies, we analyse proof strategies for dealing with the practical challenges for integrated algebraic and deductive verification of hybrid systems. To overcome the complexity pitfalls of integrating real arithmetic, we propose the *iterative background closure* and *iterative inflation order* strategies with which we achieve substantial computational improvements.

5.1. Introduction

While the theoretical background and formal details of our logic-based verification approach for hybrid systems can be found in Part I, here we discuss the practical aspects of combining deductive, real algebraic, and computer algebraic prover technologies. In particular, we highlight the principles how these techniques interact for verifying hybrid systems. We analyse the degrees of freedom in implementing our calculus in terms of the nondeterminisms of its canonical proof procedure. We illustrate the impact that various choices of proof strategies have on the overall performance. For hybrid system verification, we observe that the nondeterminisms in the interaction between deductive and real algebraic reasoning have considerable impact on the practical feasibility. While straightforward combinations are sufficient for verifying examples like those presented in Part I, larger case studies like those that we present in Part III are beyond the capabilities of state-of-the-art decision procedures for real arithmetic. In this chapter, we analyse and explain the causes and consequences of this effect and introduce automatic proof strategies that avoid these complexity pitfalls and work well in practice.

Here we study the modular combination in the $\mathsf{d}\mathcal{L}$ calculus (our findings generalise directly to the extensions of the DAL and dTL calculi so that we use $\mathsf{d}\mathcal{L}$ interchangably with DAL and dTL in the following). Our observations are of more general interest, though, and we conjecture that similar results hold for other prover combinations of logics with interpreted function symbols that are handled using background decision procedures for computationally expensive theories including real arithmetic, approximations of natural arithmetic, or arrays.

5.1.1. Related Work

As we have pointed out in Section 1.2, there are only a few other practical approaches [MS98, ÁMSH01] that use deduction for verifying hybrid systems and actually integrate arithmetic reasoning in STeP [MS98] or in PVS [ÁMSH01], respectively. They do not work with a genuine verification logic, however, but only generate flat mathematical verification conditions for hybrid automata with a given invariant. In contrast, the symbolic decompositions in our verification logic preserve the problem structure, which enables us to achieve good performance in practice. See Section 1.2 for a detailed comparison.

Several other approaches intend to combine deductive and arithmetic reasoning, e.g. [BJK+97, BCZ98, ADG+01]. Their focus, however, is on general mathematical reasoning in classes of higher-order logic and is not tailored to verify hybrid systems. Our work, instead, is intended to make practical verification of hybrid systems possible and we aim at automating the verification process.

Structure of this Chapter

In Section 5.2, we analyse our calculi for differential dynamic logics from a qualitative perspective and present a corresponding tableau procedure modulo solvers for handling real algebraic and computer algebraic constraints. We delineate their nondeterminisms carefully in Section 5.3 and analyse their practical impact. We present proof procedures for automated theorem proving in differential dynamic logics that navigate among the complexity pitfalls of integrating decision procedures for real arithmetic in Section 5.4 and Section 5.5. In Section 5.6, we evaluate their performance in larger case studies.

5.2. Tableau Procedures Modulo

In this section, we derive a canonical tableau procedure modulo background provers from the $\mathsf{d}\mathcal{L}$ calculus and analyse the remaining nondeterministims in the sequel.

For the purpose of this chapter, the full details of how the respective F-rules of Chapter 2–3 lift quantifier elimination to dynamic logic are not important. What is important to note, however, is that quantifier rules and rules for handling modalities need to interact because the actual constraints on quantified symbols depend on the effect of the hybrid programs within modalities (Section 2.5.2). Thus, at some point, after a number of rule applications that handle the dynamic part, F-rules will be used to discharge (or at least simplify) proof obligations over real algebraic or semialgebraic constraints by quantifier elimination [Col75, CH91, Tar51]. The remaining sub-goals will be analysed further again using dynamic rules. The F-rules constitute the modular interface that combines deduction for handling dynamic reasoning with algebraic constraint techniques for handling continuous reasoning about the reals. Here, we discuss the consequences and principles of this combination and analyse proof strategies.

The principle how the $\mathsf{d}\mathcal{L}$ calculus in Figure 2.5 combines deduction technology with methods for handling real algebraic constraints complies with the general background reasoning principles [Bec99, Tin03, DHK03]. From an abstract perspective, the $\mathsf{d}\mathcal{L}$ calculus selects a set Φ of (quantified) formulas from an open branch (Φ is called the *key*) and hands it over to the quantifier elimination procedure. The resulting formula obtained by applying QE to Φ is then returned to the main sequent prover as a result, and the main proof continues, see Figure 5.1. Similarly,

the $\mathsf{d}\mathcal{L}$ calculus triggers symbolic, computer algebraic computations for the rules for differential equations using their solutions (D11–D12 from Figure 2.5) or total differentials of differential invariants (G5) or variants (G6) from Figure 3.3.

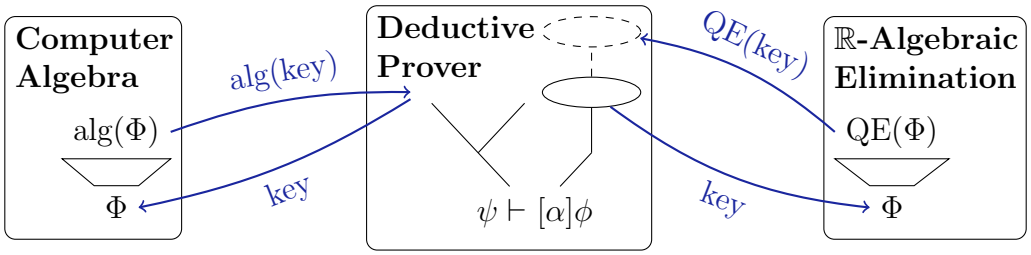


Figure 5.1.: Deductive, real algebraic, and computer algebraic prover combination

In this context, the P-rules, D-rules, and G-rules constitute the *foreground rules* in the main prover (middle box of Figure 5.1), except for evolution rules D11–D12 and differential (in)variant rules G5–G6 that represent the *computer algebraic rules* invoking a computer algebra system as a *background solver* (left box). The arithmetic F-rules (in particular F3 and F6 from Figure 2.5 and F1–F4 from Figure 3.3) form the set of rules that invoke the *background prover* (right box) for quantifier elimination. Since the primary challenges caused by the nondeterminisms in the $\mathsf{d}\mathcal{L}$ calculus originate from the interaction of deductive and real algebraic rules, we simplify the presentation in the following and only distinguish between background real arithmetic rules and foreground rules, where computer algebraic rules will be considered as foreground rules.

The canonical tableau procedure belonging to the $\mathsf{d}\mathcal{L}$ calculus is presented in Figure 5.2. Observe that the tableau procedure [Fit96] for our $\mathsf{d}\mathcal{L}$ calculus has a nonstandard set of nondeterministic steps (indicated by $\mathcal{B}$, $\mathcal{M}$, and $\mathcal{F}$, respectively in Figure 5.3):

$\mathcal{B}$: *selectBranch*, i.e., which open branch to choose for further rule applications.

$\mathcal{M}$: *selectMode*, i.e., whether to apply foreground $\mathsf{d}\mathcal{L}$ rules (P-rules, D-rules, and G-rules) or background arithmetic rules (F3, F6 from Figure 2.5 or F1–F4 from Figure 3.3).

$\mathcal{F}$: *selectFormula*, i.e., which formula(s) to select for rule applications from the current branch in the current mode.

Within the rule applications, there is an additional choice of whether to handle differential equations using their solution (D11–D12 from Figure 2.5) or by differential induction (G5–G6 from Figure 3.3). We do not follow up on this nondeterminism in the sequel but simply choose to use solutions, whenever they are first-order expressible, and fall back to differential induction (Section 3.5.5) when no such solution

```
while tableau T has open branches do
  B := selectBranch(T)          (* ℬ–nondeterminism *)
  M := selectMode(B)            (* ℳ–nondeterminism *)
  F := selectFormulas(B,M)      (* ℱ–nondeterminism *)
  if M = foreground then
    R := result of applying a P, D, or G–rule to F in B
    replace branch B by R in tableau T
  else
    send key F to background decision procedure QE
    receive result R from QE
    apply an F–rule to T with QE–result R
  end if
end while
```

Figure 5.2.: Tableau procedure for differential dynamic logics

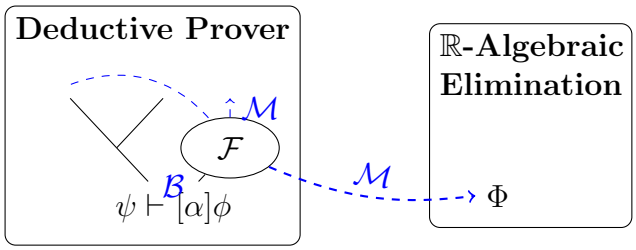


Figure 5.3.: Nondeterminisms in the tableau procedure for differential dynamic logics

can be found. The computational cost of differential induction is generally less than when working with solutions, but the corresponding differential (in)variants have to be found first, which we handle in Chapter 6. There is a further minor nondeterminism of whether to expand loops using D5,D6 or to go for an induction by G3 and G4. Yet, as unrolling (D5,D6) only handles reachability prefixes or bounded loops, our strategies prefer induction (G3 and G4) instead. The other $\mathsf{d}\mathcal{L}$ rules do not produce any conflicts once a formula has been selected as they apply to formulas of distinct syntactic structures: The top-level operators uniquely determine a calculus rule once a formula has been selected.

At this point, notice that, unlike the classical tableau procedure [Fit96], we have three rather than four points of nondeterminism, since $\mathsf{d}\mathcal{L}$ does not need closing substitutions. The reason for this is that $\mathsf{d}\mathcal{L}$ has an interpreted domain. Rather than having to try out instantiations that have been determined by unification or heuristics as in uninterpreted first-order logic [Fit96], we can make use of the structure in the interpreted case of first-order logic over the reals. In particular,

arithmetic formulas can be reduced equivalently by QE to simpler formulas in the sense that the quantified symbols no longer occur. As this transformation is an equivalence, there is no loss of information and we do not need to backtrack [Fit96] or simultaneously keep track of multiple local closing instantiations [Gie01].

Despite this, the influence of nondeterminism on the practical prover performance is remarkable. Even though the first-order theory of real arithmetic is decidable by quantifier elimination [CH91], its complexity is *doubly exponential* in the number of quantifier alternations [DH88]. While more efficient algorithms exist for linear fragments [LW93], the practical performance is an issue in nonlinear cases. Hence, the computational cost of individual rule applications is quite different from the linear complexity of applying closing substitutions in uninterpreted tableaux.

5.3. Nondeterminisms in Tableau Modulo

In principle, exhaustive fair application of background rules by the nondeterminisms $\mathcal{M}$ and $\mathcal{F}$ remains complete for appropriate fragments of $\mathsf{d}\mathcal{L}$. In practice, however, the complexity of real arithmetic quickly makes this naïve approach infeasible for larger case studies. In the remainder of this chapter, we discuss the consequences of the nondeterminisms and develop proof strategies to tackle the combination and integration challenges.

5.3.1. Nondeterminisms in Branch Selection

In classical uninterpreted tableaux, branch selection has no impact on completeness but can only have impact on the proving duration as closing substitutions can sometimes be found much earlier on one branch than on the others. In the interpreted case of $\mathsf{d}\mathcal{L}$, branch selection is even less important. As $\mathsf{d}\mathcal{L}$ has no closing substitutions, there is no direct interference among multiple branches. Branches with (explicitly or implicitly) universally quantified variables have to be closed independently, hence the branch order is not important. For instance, when x is a universally quantified variable and we denote the corresponding Skolem symbol again by x, the branches in the following proof can be handled separately (branches are implicitly combined by conjunction and universal quantifiers distribute over conjunctions):

$$\dfrac{\dfrac{\dfrac{\mathrm{QE}(\forall x\,(\ldots bx^2 \geq 0))}{\Gamma, b>0 \vdash bx^2 \geq 0}\,\mathrm{F3} \qquad \dfrac{\mathrm{QE}(\forall x\,(\ldots bx^4+x^2 \geq 0))}{\Gamma, b>0 \vdash bx^4+x^2 \geq 0}\,\mathrm{F3}}{\Gamma, b>0 \vdash bx^2 \geq 0 \wedge bx^4+x^2 \geq 0}\,\mathrm{P5}}{\Gamma, b>0 \vdash \forall x\,(bx^2 \geq 0 \wedge bx^4+x^2 \geq 0)}\,\mathrm{F1}$$

For existentially quantified variables, the situation is a bit more subtle as multiple branches interfere indirectly in the sense that a simultaneous solution needs to be found for all branches at once. In $\exists v\,(v>0 \wedge v<0)$, for instance, the two

branches resulting from the cases $v > 0$ and $v < 0$ cannot be handled separately, as the existential quantifier claims the existence of a simultaneous solution for $v > 0$ and $v < 0$, not two different solutions. Thus, when v is an existentially quantified variable and V its corresponding free existential variable, the branches in the following proof need to synchronise before quantifier elimination is applied:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{b > 2 \vdash b(V-1) > 0}\,\mathrm{QE}(\exists v\ \dots) \qquad \dfrac{}{b > 2 \vdash (V+1)^2 + b\epsilon(V+1) > 0}\,\mathrm{QE}(\exists v\ \dots)}{b > 2 \vdash [v := V-1]bv > 0}\,\mathrm{F3} \qquad \dfrac{}{b > 2 \vdash [v := V+1]v^2 + b\epsilon v > 0}\,\mathrm{F3}}{b > 2 \vdash [v := V-1]bv > 0 \wedge [v := V+1]v^2 + b\epsilon v > 0}\,\mathrm{P5}}{b > 2 \vdash \forall v\,([v := v-1]bv > 0 \wedge [v := v+1]v^2 + b\epsilon v > 0)}\,\mathrm{F1}$$

The order in which the intermediate steps at the two branches are handled has no impact on the proof. Branches like these *synchronise* on an existential free variable V in the sense that all occurrences of V need to be first-order on all branches for quantifier elimination to be applicable. Consequently, the only fairness assumption for $\mathcal{B}$ is that whenever a formula of a branch is selected that is waiting for synchronisation with another branch to become first-order, then it transfers its branch choice to the other branch. In the above case the left branch synchronises with the right branch on V. Hence, rule F6 can only be applied to $b(V-1) > 0$ on the left branch after D9 has been applied on the right branch to yield first-order occurrences of V.

Thus, the primary remaining impact of the branch nondeterminism is that closing branches by universally quantified variable reasoning simplifies all subsequent existential variable handling, because less branches remain that need to be considered simultaneously.

5.3.2. Nondeterminisms in Formula Selection

In background proving mode, it turns out that nondeterminism $\mathcal{F}$ is important for the practical performance. In practice, when a branch closes or, at least, can be simplified significantly by a quantifier elimination call, then the running time of a single decision procedure call depends strongly on the number of irrelevant formulas that are selected in addition to the relevant ones by $\mathcal{F}$.

Clearly, when Φ is a set of formulas from a sequent that yields a tautology such that applying QE closes a branch, then selecting any superset $\Psi \supseteq \Phi$ from a branch yields the same answer in the end (a sequent forms a disjunction of its formulas hence it can be closed to true when any subset closes). However, the running time until this result will be found in the larger Ψ is strongly disturbed by the presence of complicated additional but irrelevant formulas. From our experience with Mathematica, decision procedures for full real arithmetic seem to be distracted considerably by such irrelevant additional information.

5.1 Example (Computational distraction in quantifier elimination). One sequent from the proof of the ETCS kernel in Section 2.4 is depicted in Figure 5.4. It results from the right branch of the proof in Section 2.9 by exhaustive splitting. Quantifier elimination, as performed by function Reduce in Mathematica, runs more than 24h without producing a result on the formula in Figure 5.4, which has 9 symbolic variables and polynomial degree 2.

The marked constraint in Figure 5.4 corresponds to the initial state of the system, because it refers to the initial train position z and not to the current train position z_2 in the current induction step. In fact, the induction step does not depend on this part of the initial state information (it does depend on $b > 0$, though). When we remove the superfluous constraint on the initial state, the formula in Figure 5.4 suddenly becomes provable in less than one second! The $\mathsf{d}\mathcal{L}$ calculus presented in Chapter 2 already avoids this problem, because rule F3 applies quantifier elimination after reintroducing the quantifier structure that results from universal closures in global rule applications (e.g., G3).

$$
\begin{array}{l}
t_2 > 0,\ \varepsilon \geq t_2,\ v_2 \geq 0,\ A + 1/t_2 \cdot v_2 \geq 0,\ t_2 \geq 0, \\
m - z_2 \geq v_2^2/(2b) + (A/b + 1)(A/2\varepsilon^2 + \varepsilon v_2), \\
2b(m - z_2) \geq v_2^2 \\
2b(m - z) \geq v^2, \qquad\qquad \text{/* initial state */} \\
b > 0,\ A \geq 0 \\
\vdash (At_2 + v_2)^2 \leq 2b(m - 1/2(At_2^2 + 2t_2v_2 + 2z_2))
\end{array}
$$

Figure 5.4.: Computational distraction in quantifier elimination

Yet, such additional information accumulates in tableaux procedures quite naturally, because the purpose of a proof branch in $\mathsf{d}\mathcal{L}$ is to keep track of all that is known about a particular (symbolic) case of the system behaviour. Generally, not all of this knowledge finally turns out to be relevant for that case but only plays a role in other branches. Nevertheless, discarding part of this knowledge arbitrarily would, of course, endanger completeness.

For instance, the safety statement (2.1) in Section 2.4 depends on a constraint on the safety envelope s that regulates braking versus acceleration by the condition $m - z \geq s$ in *ctrl*. A maximal acceleration of A is permitted in case $m - z \geq s$, when adaptively choosing s depending on the current speed v, maximum braking force b, and maximum controller response time ϵ in accordance with constraint (2.6) as discovered in Section 2.9. This constraint is necessary for some but not for all cases of the symbolic safety analysis, though. In the case where the braking behaviour of ETCS is analysed, for instance, the constraint on s is irrelevant, because

braking is the safest operation that a train can do to avoid crashing into preceding trains. The unnecessary presence of several quite complicated constraints like, for instance, (2.6), however, can distract quantifier elimination procedures considerably.

5.3.3. Nondeterminisms in Mode Selection

In its own right, nondeterminism $\mathcal{M}$ has less impact on the prover performance than $\mathcal{F}$. Every part of a branch could be responsible for closing it. The foreground closing rule P9 of the main prover can only close branches for comparatively trivial reasons like $b > 0, \epsilon > 0 \vdash \epsilon > 0$. Hence, mode selection has to give a chance to the background procedure every once in a while, following some fair selection strategy. From the observation that some decision procedure calls can run for hours without terminating, we can see, however, that realisations of nondeterminism $\mathcal{M}$ needs to be devised with considerable care.

As the reason for closing a branch can be hidden in any part of the sequent, some expensive decision procedure calls are superfluous if the branch can be closed by continuing $\mathsf{d}\mathcal{L}$ reasoning on the other parts. For instance, if F is some complicated algebraic constraint, decision procedure calls triggered by nondeterminism $\mathcal{M}$ can lead to nontermination within any feasible time for

$$\ldots, \epsilon > 0, m - z \geq s \vdash F, [drive]\epsilon > 0, \ldots$$

Instead, if $\mathcal{M}$ chooses foreground rules, then an analysis of $[drive]\epsilon > 0$ by $\mathsf{d}\mathcal{L}$ rules will quickly discover that the maximum reaction-time ϵ remains constant while driving. Then, this part of a proof closes without the need to consider constraint F at all. For this reason, proof strategies that eagerly check for closing branches by background procedure calls are not successful in practice, see Figure 5.5. Similarly,

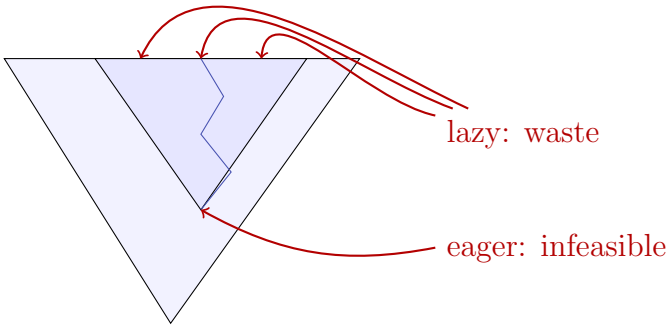


Figure 5.5.: Eager and lazy quantifier elimination in proof search space

verifying the ETCS case study that we detail in Chapter 7 requires proving pretty large subgoals of first-order real arithmetic, which cannot be proven by a quantifier

```
 state = 0,
 2 * b * (m - z) >= v ^ 2 - d ^ 2,
 v >= 0, d >= 0, v >= 0, ep > 0, b > 0, A > 0, d >= 0
==>
 v <= vdes
-> \forall R a_3;
   (   a_3 >= 0 & a_3 <= A
    -> (      m - z
           <= (A / b + 1) * ep * v
            + (v ^ 2 - d ^ 2) / (2 * b)
            + (A / b + 1) * A * ep ^ 2 / 2
        -> \forall R t0;
             (   t0 >= 0
              -> \forall R ts0;
                   (0 <= ts0 & ts0 <= t0
                       -> -b * ts0 + v >= 0 & ts0 + 0 <= ep)
              ->    2 * b * (m - 1 / 2 * (-b * t0 ^ 2 + 2 * t0 * v + 2 * z))
                   >= (-b * t0 + v) ^ 2
                    - d ^ 2
                 & -b * t0 + v >= 0
                 & d >= 0))
       & (    m - z
           >  (A / b + 1) * ep * v
            + (v ^ 2 - d ^ 2) / (2 * b)
            + (A / b + 1) * A * ep ^ 2 / 2
          -> \forall R t2;
             (   t2 >= 0
              -> \forall R ts2;
                   (0 <= ts2 & ts2 <= t2
                       -> a_3 * ts2 + v >= 0 & ts2 + 0 <= ep)
              ->    2 * b * (m - 1 / 2 * (a_3 * t2 ^ 2 + 2 * t2 * v + 2 * z))
                   >= (a_3 * t2 + v) ^ 2
                    - d ^ 2
                 & a_3 * t2 + v >= 0
                 & d >= 0)))
```

Figure 5.6.: A large subgoal of first-order real arithmetic during ETCS verification

elimination procedure within any feasible amount of time, see Figure 5.6 for an example.

Unfortunately, converse strategies with lazy checks that strongly favour foreground $\mathsf{d}\mathcal{L}$ rule applications in $\mathcal{M}$, are not appropriate either, see Figure 5.5. There, splitting rules like P5 and P4 can eagerly split the problems into multiple branches without necessarily making them any easier to solve. If this happens, slightly different but similar arithmetic problems of about the same complexity need to be solved repeatedly on multiple branches rather than just one branch, resulting in runtime blow-up.

The reason why this can happen is a substantial syntactic redundancy in the sequent encoding of formulas. For instance, the sets of sequents before and after the following rule application are equivalent:

$$\text{P5,P5}\frac{\psi \vdash v^2 \leq 2b(m-z) \quad \psi \vdash \epsilon > 0 \quad \psi \vdash (z \geq 0 \rightarrow v \leq 0)}{\psi \vdash v^2 \leq 2b(m-z) \wedge \epsilon > 0 \wedge (z \geq 0 \rightarrow v \leq 0)}$$

Yet, closing the three sequents above the bar by quantifier elimination is not necessarily easier than the single sequent below (neither conversely). Even worse, if the sequents close by applying rules to ψ, then similar reasoning has to be repeated for three branches. This threefold reasoning may not even be detected as identical when ψ is again split differently on the three resulting branches.

Further, the representational equivalence in sequents is purely syntactic, i.e., up to permutation, the representations share the same disjunctive normal form. In the uninterpreted case of first-order logic, this syntactic redundancy is exploited by the P-rules in order to transform sequents towards a canonical form with atomic formulas, where partial closing situations are more readily identifiable. In the presence of a background decision procedure, however, reduction to sequents with atomic formulas is no longer necessary as it will be undone when handing the formulas over to the background decision procedure.

Furthermore, the logical splitting along the propositional structure does not always help quantifier elimination procedures, because their working principle is not a deductive case analysis but (partial) cylindrical algebraic decomposition [Col75, CH91, CJK02]. In Section 5.4, we will see that the deductive analysis is still an extremely important factor for accelerating quantifier elimination, but has to be applied with care.

Finally, algebraic constraint handling techniques as in the Mathematica function Reduce can come up with a result that is only a restated version of the input if a selected (open) formula cannot be simplified or closed. For instance, the sequent $z < m \vdash v^2 \leq 2b(m-z)$ "reduces" to $\vdash b \geq v^2/(2m-2z) \vee m \leq z$ without any progress. Such arithmetical reformulation cannot even be detected by simple syntactical means but easily lead to infinite proof loops without progress when the outcome is split by P3 and again handled by the background procedures.

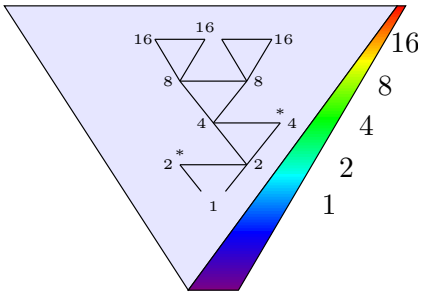
5.4. Iterative Background Closure

In the sequel, we propose strategies to solve the previously addressed computational issues caused by the nondeterminisms of the dL tableau procedure and its integration with computationally expensive background provers.

Priority-based Strategies We propose the priorities for rule applications in Figure 5.7a (with rules at the top taking precedence over rules at the bottom). In this strategy, algebraic constraints are generally left intact as opposed to being split among multiple branches, because arithmetic rules have a higher priority than propositional splitting rules on first-order constraints. Further, we only accept the result of the background procedure when the number of different variable symbols has decreased to avoid infinite proof loops. We use arithmetic background rules *either* with priority 2 *or* with priority 5.

1. non-splitting propositional rules P1–P3, P6–P7, P9
2. *arithmetic rules* if variable eliminated
3. dynamic rules D7–D10
4. splitting rules P5, P4, P8 on modalities
5. *arithmetic rules* if variable eliminated
6. (in)variant global rules G3, G4, G5, and G6
7. splitting rules P5, P4, P8 on first-order formulas

5.7a: Proof strategy priorities



5.7b: Iterative background closure

Figure 5.7.: Iterative background closure (IBC) proof strategy

The effect of using priority 2 is that branches are checked eagerly for closing conditions or variable reductions, see Figure 5.5. If reasoning about algebraic constraints does not yield any progress (no variables can be eliminated), then dL rules further analyse the system. For this choice, it is important to work with timeouts to prevent lengthy background decision procedure calls from blocking dL proof progress.

This problem is reduced significantly when using priority 5 for arithmetic rules instead. The effect of priority 5 is that formulas containing modalities are analysed and decomposed as much as possible before arithmetic reasoning is applied to algebraic constraint formulas. Then, however, the prover might consume too much time analysing the effects of programs on branches which would already close due to simple arithmetic facts like in $\epsilon > 0, \epsilon < 0 \vdash [\alpha]\phi$.

A simple compromise is to use a combination of background rules with priority 2 for quick linear arithmetic [LW93] and to fall back to expensive quantifier elimination calls for nonlinear arithmetic with priority 5.

```
1  /* prove validity of the sequent Φ ⊢ Ψ */
2  function IBC(Φ ⊢ Ψ, timeout):
3    if QE(Φ ⊢ Ψ) succeeds within timeout then
4      return QE(Φ ⊢ Ψ)
5    else
6      while foreground rule applicable and proof not split do
7        apply foreground rule to current sequent
8      end while
9      let Φ1 ⊢ Ψ1, ..., Φn ⊢ Ψn be the resulting branches
10     timeout := 2*timeout
11     return IBC(Φ1 ⊢ Ψ1, timeout) and ... and IBC(Φn ⊢ Ψn, timeout)
```

Figure 5.8.: Iterative background closure (IBC) algorithm schema

Iterative Background Closure As a more sophisticated control strategy on top of the static priorities in Figure 5.7a, we introduce *iterative background closure* (IBC). There, the idea is to periodically apply arithmetic rules with a timeout T that increases by a factor of 2 after background procedure runs have timed out, see Figure 5.7b. Thus, background rules interleave with other rule applications (triangles in Figure 5.7b), and the timeout for the sub-goals increases as indicated, until the background procedure successfully eliminated variables on a branch (marked by $*$). The effect is that the prover avoids splitting in the average case but is still able to split cases when combined handling turns out to be prohibitively expensive. As an optimisation, timed-out QE will only be invoked again after the branch has been split (or after a modal formula has disappeared from the sequent). Figure 5.8 shows a procedure that corresponds to a proof strategy that implements IBC. Unless the QE call terminated successfully within the current timeout (line 3), IBC applies foreground rules (line 7) until the proof has split. Then, the timeout increases (line 10) and IBC handles the resulting branches recursively (line 11). Our experimental results show that IBC is surprisingly decisive for handling larger case studies, see Section 5.6.

And/Or Branching More generally, the $\mathsf{d}\mathcal{L}$ calculus gives rise to theorem proving structures with combined and/or-branching. While the branches resulting from one rule application are *and-branches* (all of them have to close for the proof to succeed), the rule alternatives, especially as caused by the tableau procedure nondeterminisms, are *or-branches* (only one of the proof search alternatives has to be successful), see Figure 5.9. Most notably, induction rules give rise to or-branches, as there are several possible formulas (infinitely many) that could be used as (differential) (in)variants for G3–G6, but one single successful (in)variant is enough for closing the proof. For these or-branch alternatives, any fair parallel explora-

tion scheme or any fair sequential exploration scheme with time-interleaving can be used. Iterative background closure is one possible sequential interleaving choice that works well in practice. On parallel computers, distributed clusters, or even multi-core processors, truly parallel exploration schemes would be faster, as they exploit the natural proving parallelism in our calculi. We will make use of and/or-branching exploration based on iterative background closure in Chapter 6.

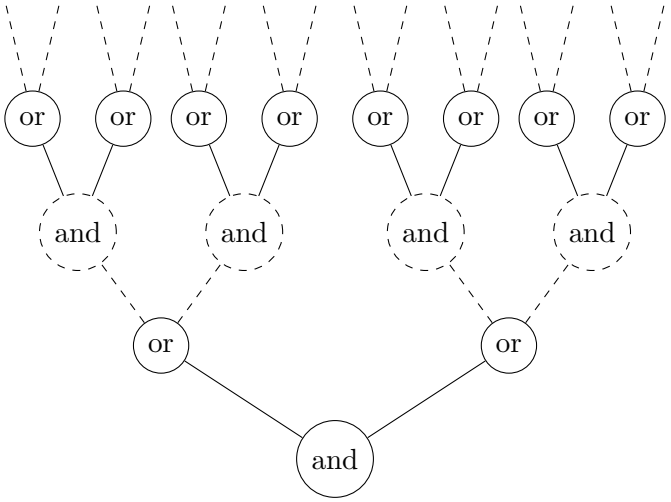


Figure 5.9.: General and/or-branching in proof strategies for differential dynamic logics

5.5. Iterative Inflation

The iterative background closure strategy already has a decisive impact on the feasibility of verifying larger case studies. Yet, there are still cases where the $\mathcal{F}$-nondeterminism has a significant computational impact that limits scalability. These difficulties are primarily caused by superfluous constraints on previous or initial states that accumulate in the sequent, see, e.g., Example 5.1.

A general possible solution for this issue is to iteratively consider more formulas of the sequent and attempt decision procedure calls with fair time-interleaving until the respective branch closes. There, only those additional formulas need to be considered that share variables with any of the other selected formulas. Further, timeouts can be used to discontinue lengthy decision procedure calls and continue along other choices of the nondeterminisms in Figure 5.2. For complicated cases with a prohibitive complexity, this heuristic process worked well on our examples.

Inflation Order As a general pattern for building iterative inflation optimisations, we propose an algorithm that selects additional formulas of a sequent successively

```
1  /* prove validity of the disjunctive set S */
2  /* of formulas of first-order real arithmetic */
3  function IIO(S):
4    timeout := 1
5    C := {false}
6    while C ≠ ∅ do
7      for φ ∈ C do
8        if (QE(φ) = true) within timeout then
9          return valid
10       else if (QE(φ) ≠ true) within timeout
11                 or FindInstance(¬φ) within timeout then
12         C := (C ∪ {φ ∨ A : A ∈ S, A ∉ φ}) \ {φ}
13     end for
14     timeout := 2*timeout
15   end while
16   return not valid
```

Figure 5.10.: Iterative inflation order (IIO) algorithm schema

while a counterexample can be found in bounded time (e.g., using the Mathematica function *FindInstance*) and terminates when a quantifier elimination call yields "true" within the current time bound (increasing timeouts as in IBC).

The *iterative inflation* algorithm schema in Figure 5.10 proves validity of a disjunction of formulas of first-order real arithmetic that is given as a set S of disjuncts. Validity of a sequent $\Phi \vdash \Psi$ can be proven by $IIO(\{\neg\phi \ : \ \phi \in \Phi\} \cup \{\psi \ : \ \psi \in \Psi\})$. Following the general scheme in Figure 5.9, the IIO algorithm explores subset candidates of S in parallel by fair time interleaving. Starting from the empty subset of formulas (line 5), the algorithm will try to prove (line 8) or disprove (line 11) each candidate subset ϕ of S. The first successful QE yielding true within the current timeout shows validity of S (line 9), because a disjunction is valid iff any subset is. When a counterexample is found (line 11) or quantifier elimination produced another result than "true" within the timebounds (line 10), candidate ϕ is dropped and any remaining formulas of S are added to ϕ to form new candidates. If no candidate could be proven (line 8) or disproven (lines 10–11) within the current time bounds, the timeout increases (line 14). The algorithm returns that S is not valid if all candidates have counterexamples (or QE yields results other than "true") and no new formulas of S can be added to produce new candidates (line 12).

Note that the IIO algorithm is a schematic algorithm. It allows for several refinements, optimisations, and caching improvements. Most importantly, line 12 can be refined to limit the number of candidates explored in parallel by adding only candidates with computationally promising properties. In general, a *reinclusion*

ordering can be used to determine in which order, new candidates are added and explored.

Canonical Reinclusion Order One simple canonical reinclusion ordering is to order formulas by precomputing the overall resulting theoretical complexity based on various complexity results for real quantifier elimination [DH88, Gri88, GV88, Ris88, Wei88, BD07, Ren92a, Ren92b, Ren92c], which depend on the number of variables and either the number of quantifiers (practical algorithms) or the number of quantifier alternations (theoretical complexity results). While this has a certain theoretically precise appeal, the disadvantage is that the theoretical complexity measures are rather coarse-grained and do not take into account the structure of the formula but only its worst-case elements.

Structural Reinclusion Order Among all possible orders for re-including formulas, total orders share the advantage that they prevent the need for parallel exploration of multiple different possibilities of adding mutually incomparable formulas, which can also turn into a disavantage once the order starts adding computationally problematic high-degree constraints. As a compromise of the canonical reinclusion order with structural information, we propose to order formulas for reinclusion according to the lexicographical order of, respectively, relative variable recency, total polynomial degree, number of new variables, and maximum term depth. Favouring more recent variables follows the rationale that formulas that mention variables that have been introduced only recently into the proof are more likely to carry relevant information about the current state than those that only refer to variables from the initial proof obligation for which more recent state variables have already been introduced during the proof. For instance, the position z from the initial state may be less relevant than the current position z_2 in the induction step. The braking power, b, however, may be as relevant as the recent z_2, because there is no updated copy of the symbolic constant b. The number of new variables that are added to the current candidate $\phi \in \mathcal{C}$ from Figure 5.10 also has an impact on the complexity. Favouring small polynomial degrees is self-explanatory and follows the observation that the polynomial degree has a substantial impact on overall performance, as we will also see in Chapter 8. The term depth is used as an indicator for the complexity of the formula. To obtain a linear, yet arbitrary, order this order can be extended easily by breaking ties by lexicographical string comparison.

Combining this structural order with more sophisticated combinations of polynomial degree and number of new variables according to the canonical orders may also be a viable alternative to improve on pure lexicographic orders.

5.6. Experimental Results

Tables 5.1–5.4 summarise verification results for our proof strategies for various case studies. Experimental results are from a 2.6GHz AMD Opteron with 4GB memory. The timeout for computations was 18000s=5h and is indicated as ∞. Memory consumption of quantifier elimination is shown, excluding the front-end. The dimension of the state space and the number of required proof steps are indicated. To isolate effects resulting from our automatic invariant discovery that we describe in Chapter 6, we conduct experiments both with (proof annotated) user interactions and in automatic mode (the number of non-automatic interactions is indicated in column "Int"). The respective case studies are based on the examples shown in Part I and will be described in full detail in Part III and IV. For the tangential roundabout maneuver from Section 3.4, the number of participating aircraft is as indicated.

Observe that the performance of the extreme strategies of eager and lazy quantifier elimination depends on the example. For ETCS and larger aircraft roundabout maneuvers (Tables 5.1 and 5.3), the lazy strategy performs faster, while the eager strategy is faster for bouncing ball and water tank examples (Tables 5.2 and 5.4), where the lazy approach splits heavily (the number of required proof steps is much higher). As an intermediate strategy, IBC generally shows intermediate performance. In our case studies, the eager strategy fails to come up with a result within the timeout fairly often. Furthermore, IBC is the only strategy that is able to prove all case studies. For the reactivity property of the ETCS case study, both lazy and eager proof strategies fail at the same time and only IBC succeeds. A result of "E" indicates that no data is available, because of current limits in the KeYmaera implementation used consistently for performance measurements.

By comparing Tables 5.1–5.2 with Tables 5.3–5.4, we also see that the performance is generally better when we do not allow partial reductions of standalone quantified subformulas of a sequent. The reason for that is that the result of standalone QE-reductions on subformulas (Table 5.1-5.2) can produce several disjunctions which split into further subbranches, such that the redundancy effects described in Section 5.3 and those illustrated in Figure 5.5 have more impact.

Finally note that, without our range of proof strategies that result from a careful analysis of the nondeterminisms in the tableau procedures for differential dynamic logics, the practical prover performance is significantly worse than all those presented in Tables 5.1–5.4. The small ETCS kernel from Chapter 2 is already provable directly when implementing the $\mathsf{d}\mathcal{L}$ calculus naïvely. Without our range of proof-strategic improvements, however, the full ETCS system that we present in Chapter 7 requires as much as 56 user interactions to be provable [Que07], while our proof strategies and the algorithms that we present in Part II can, in fact, prove ETCS completely automatically with zero user interactions.

Table 5.1.: Experimental results for proof strategies (with standalone QE) I

Case study	Int	Strategy	Time(s)	Memory(MB)	Steps	Dim
ETCS kernel	1	eager	11.1	15.1	44	9
	1	IBC	15.6	14.3	54	
	1	lazy	3.8	14.2	88	
	0	IBC	40.2	25.6	53	
	0	lazy	12.9	24.3	85	
ETCS binary safety	1	eager	9.6	11.4	144	14
	1	IBC	10.7	12.1	148	
	1	lazy	13.4	16.0	413	
	0	IBC	57.3	25.5	148	
	0	lazy	46.7	32.6	293	
ETCS safety	1	eager	∞	∞	∞	14
	1	IBC	26.8	17.6	168	
	1	lazy	25.8	29.1	423	
	0	IBC	2089.2	206.1	171	
	0	lazy	2046.5	203.3	304	
ETCS reactivity	0	eager	∞	∞	∞	14
	0	IBC	1084.1	6.1	34	
	0	lazy	∞	∞	∞	
tangential roundabout	3	eager	1.7	6.8	94	13
	3	IBC	1.5	6.8	93	
	3	lazy	1.6	6.7	139	
	0	IBC	10.3	6.9	114	
	0	lazy	10.2	6.8	197	
tangential roundabout 3	3	eager	∞	∞	∞	18
	3	IBC	75.0	24.7	165	
	3	lazy	52.3	14.9	244	
	0	IBC	1065.5	27.6	186	
	0	lazy	620.2	15.0	342	
tangential roundabout 4	3	eager	4208.2	E	E	23
	3	IBC	513.4	184.4	229	
	3	lazy	57.6	31.4	355	
	0	IBC	10998.1	184.3	256	
	0	lazy	901.7	31.4	520	
tangential roundabout 5	3	eager	7714.1	E	E	28
	3	IBC	2457.9	479.3	317	
	3	lazy	108.9	43.6	502	
	0	IBC	∞	∞	∞	
	0	lazy	3417.5	48.5	735	

Table 5.2.: Experimental results for proof strategies (with standalone QE) II

Case study	Int	Strategy	Time(s)	Memory(MB)	Steps	Dim
bouncing ball	1	eager	7.7	13.6	85	7
	1	IBC	8.0	13.6	85	
	1	lazy	∞	∞	∞	
	0	IBC	66.4	19.0	43	
	0	lazy	∞	∞	∞	
water tank	1	eager	3.8	9.1	378	3
	1	IBC	3.9	9.1	375	
	1	lazy	9.2	9.5	1604	

5.7. Summary

From the experience of using our $\mathsf{d}\mathcal{L}$ calculus for verifying parametric hybrid systems in traffic applications, we have investigated combinations of deductive, computer algebraic, and real algebraic reasoning from a practical perspective. We have analysed the principles of this prover combination, identified the nondeterminisms that remain in the canonical $\mathsf{d}\mathcal{L}$ tableau procedure, and analysed their impact.

We have proposed proof strategies that navigate among these nondeterminisms, including iterative background closure and iterative inflation order strategies. Similar to the huge importance of subsumption in resolution, background-style tableau proving requires quick techniques to rule out branches closing for simple arithmetic reasons. In our experiments with verifying cooperating traffic agents, the combination of our proof strategies reduce the number of interactions and the overall running time significantly.

Table 5.3.: Experimental results for proof strategies (no standalone QE) I

Case study	Int	Strategy	Time(s)	Memory(MB)	Steps	Dim
ETCS kernel	1	eager	11.3	15.0	42	9
	1	IBC	6.9	14.3	47	
	1	lazy	2.8	14.2	61	
	0	IBC	47.8	25.4	46	
	0	lazy	10.5	24.2	58	
ETCS binary safety	1	eager	9.8	11.3	142	14
	1	IBC	10.4	12.2	148	
	1	lazy	7.2	15.8	235	
	0	IBC	41.0	17.4	144	
	0	lazy	18.6	12.4	204	
ETCS safety	1	eager	∞	∞	∞	14
	1	IBC	26.5	23.1	168	
	1	lazy	18.9	24.2	247	
	0	IBC	2051.3	204.1	163	
	0	lazy	2031.6	203.1	216	
ETCS reactivity	0	eager	∞	∞	∞	14
	0	IBC	104.1	61.7	49	
	0	lazy	∞	∞	∞	
tangential roundabout	3	eager	1.7	6.8	94	13
	3	IBC	1.7	6.8	93	
	3	lazy	1.9	6.7	139	
	0	IBC	10.9	6.9	114	
	0	lazy	9.9	6.8	197	
tangential roundabout 3	3	eager	∞	∞	∞	18
	3	IBC	75.3	24.7	165	
	3	lazy	52.3	15.0	244	
	0	IBC	965.8	32.8	186	
	0	lazy	636.2	15.1	342	
tangential roundabout 4	3	eager	4340.6	E	E	23
	3	IBC	513	185.7	229	
	3	lazy	57	31.4	355	
	0	IBC	3323.5	44.0	247	
	0	lazy	884.9	31.4	520	
tangential roundabout 5	3	eager	7702.3	E	E	28
	3	IBC	2439.7	533.8	317	
	3	lazy	108.9	43.6	503	
	0	IBC	16303.7	827	320	
	0	lazy	3552.6	46.9	735	

Table 5.4.: Experimental results for proof strategies (no standalone QE) II

Case study	Int	Strategy	Time(s)	Memory(MB)	Steps	Dim
bouncing ball	1	eager	7.8	13.6	85	7
	1	IBC	7.8	13.6	85	
	1	lazy	20.1	13.7	166	
	0	IBC	65.0	16.9	43	
	0	lazy	78.2	26.0	92	
water tank	1	eager	3.6	9.1	387	3
	1	IBC	3.8	9.1	375	
	1	lazy	6.6	9.3	914	

Chapter 6.

Computing Differential Invariants as Fixedpoints

Contents

Synopsis

We introduce a fixedpoint algorithm for verifying safety properties of hybrid systems with differential equations whose right-hand sides are polynomials in the state variables. In order to verify nontrivial systems without

solving their differential equations and without numerical errors, we use differential induction as a continuous generalisation of induction, for which our algorithm computes the required *differential invariants*. As a means for combining local differential invariants into global system invariants in a sound way, our fixedpoint algorithm works with differential dynamic logic as a compositional verification logic for hybrid systems. To improve the verification power, we further introduce a *saturation procedure* that refines the system dynamics successively with differential invariants until safety becomes provable. By complementing our symbolic verification algorithm with a robust version of numerical falsification, we obtain a fast and sound verification procedure. We verify roundabout maneuvers in air traffic management and collision avoidance in train control.

6.1. Introduction

Reachability questions for systems with complex continuous dynamics are among the most challenging problems in verifying embedded systems, in particular for hybrid systems, which are models for these systems with interacting discrete and continuous transitions along differential equations. For simple systems whose differential equations have solutions that are polynomials in the state variables, quantifier elimination [CH91] can be used for verification as detailed in Chapter 2. Unfortunately, this symbolic approach does not scale to systems with complicated differential equations whose solutions do not support quantifier elimination (e.g., when they are transcendental functions) or cannot be given in closed form.

Numerical or approximation approaches [ADG03, PC07, DM07] can deal with more general dynamics. However, numerical or approximation errors need to be handled carefully as they easily cause unsoundness [PC07]. More specifically, we have shown previously that even single image computations of fairly restricted classes of hybrid systems are undecidable by numerical computation [PC07]. Thus, numerical approaches can be used for falsification but not (ultimately) for verification.

In this chapter, we present a verification algorithm that combines the soundness of symbolic approaches [Frä99, AW01, Pla07b, Pla08b] with support for nontrivial dynamics that is classically more dominant in numerical approaches [ADG03, PC07, DM07]. During continuous transitions, the system follows a solution of its differential equation. But for nontrivial dynamics, these solutions are much more complicated than the original equations. Solutions quickly become transcendental even if the differential equations are linear. To overcome this, we handle continuous transitions based on their vector fields, which are described by their differential equations. We use *differential induction* (Section 3.5.5), a continuous generalisation of induction that works with the differential equations themselves instead of their solutions. For the induction step, we use a condition that can be checked

easily based on *differential invariants* (Section 3.5.5), i.e., properties whose derivative holds true in the direction of the vector field of the differential equation. The derivative is a directional derivative in the direction of (the vector field generated by) the differential equation, with derivatives generalised from functions to formulas according to the findings in Chapter 3. For this to work in practice, the most crucial steps are to find sufficiently strong local differential invariants for differential equations and compatible global invariants for the hybrid system.

To this end, we introduce a *sound* verification algorithm for hybrid systems that computes the differential invariants and system invariants in a fixedpoint loop. We follow the invariants as fixedpoints paradigm [Cla79] using $\mathsf{d}\mathcal{L}$ as a verification logic that is generalised to hybrid systems accordingly. For combining multiple local differential invariants into a global invariant in a sound way, we exploit the closure properties of the underlying verification logic $\mathsf{d}\mathcal{L}$ by forming appropriate logical combinations of multiple safety statements. In addition, we introduce a *differential saturation process* that refines the hybrid dynamics successively with auxiliary differential invariants until the safety statement becomes an invariant of the refined system. Finally, each fixedpoint iteration of our algorithm can be combined with numerical falsification to accelerate the overall symbolic verification in a sound way. We validate our algorithm by verifying *aircraft roundabout maneuvers* [TPS98, PC07] and train control applications automatically, continuing the examples in Part I to the full case studies in Part III.

Contributions

The major contribution in this work is the fixedpoint algorithm for computing differential invariants coupled with a differential saturation process. We show that it can verify realistic applications that were out of scope for related invariant approaches [SSM04, RCT05, PJP07] or standard model checking approaches [Hen96, Frä99, PAM$^+$05] based on state-space exploration, both for theoretical reasons and for scalability reasons (see discussions in Section 2.4 and Section 3.1.1).

6.1.1. Related Work

Other authors [SSM04, RCT05, PJP07] already argued that invariant techniques scale to more general dynamics than explicit reach-set computations or techniques that require solutions for differential equations [Frä99, PAM$^+$05, Pla07b]. However, they cannot handle hybrid systems with inequalities in initial sets or switching surfaces [SSM04, RCT05], which occur in most real applications like aircraft maneuvers. *Barrier certificates* [PJP07] only work for inequalities, but invariants of roundabout maneuvers require mixed equations and inequalities (Section 3.11). Prajna et al. [PJP07] search for barrier certificates of a fixed degree by global optimization over the set of all proof attempts for the whole system at once,

which is infeasible: Even with degree bound 2, it already requires solving a 5848-dimensional optimization problem for ETCS (Chapter 7) and a 10005-dimensional problem for roundabouts with 5 aircraft. Similar scalability issues arise for other approaches [SSM04, GT08] that search for global invariants simultaneously for all modes of the system at once. In addition, these approaches [SSM04, GT08] require parametric templates for the respective invariants to be specified manually.

Structure of this Chapter

In Section 6.2, we introduce an automatic verification algorithm for hybrid systems that is based on the DAL calculus and computes the differential invariants and invariants as fixedpoints that are required for verification. In Section 6.3, we show how soundness of the DAL calculus inherits in a simple and elegant way to soundness of our logic-based verification procedure. We present optimisations in Section 6.4. In Section 6.5, we present experimental results for the running example of roundabout maneuvers in air traffic control and conclude in Section 6.6.

6.2. Inductive Verification by Combining Local Fixedpoints

For verifying safety properties of hybrid systems without having to solve their differential equations, we use differential induction as a continuous form of induction. In the induction step, we use a condition on directional derivatives in the direction of the vector field generated by the differential equation. The resulting properties are invariants of the differential equation (whence called *differential invariants* in Chapter 3). The crucial step for verifying discrete systems by induction is to find sufficiently strong invariants (e.g., for loops α^*). Similarly, the crucial step for verifying dynamical systems (which correspond to a single continuous mode of a hybrid system) by induction is to find sufficiently strong invariant properties of the differential equation. Consequently, for verifying hybrid systems inductively, local invariants need to be found for each differential equation and a global system invariant needs to be found that is compatible with all local invariants.

To compute the required invariants and differential invariants, we combine the invariants as fixedpoints approach from [Cla79] with the lifting of verification logics to hybrid systems from Chapter 2–3. We introduce a verification algorithm that computes invariants of a system as fixedpoints of safety constraints on subsystems. In order to obtain a local algorithm that works by decomposing global properties of hybrid programs into local properties of subsystems, we exploit the fact that hybrid programs can be decomposed into subsystems and that $\mathsf{d}\mathcal{L}$ can combine safety statements about multiple subsystems simultaneously. Note that, the algorithms

developed in this chapter apply for the logics $\mathsf{d}\mathcal{L}$, DAL, and dTL from Part I, respectively.

A simple *safety statement* corresponds to a $\mathsf{d}\mathcal{L}$ formula $\psi \to [\alpha]\phi$ with a hybrid program α, a safety property ϕ about its reachable states, and an arithmetic formula ψ that characterises the set of initial states symbolically. *Validity* of formula $\psi \vdash [\alpha]\phi$ (i.e., truth in all states) corresponds to ϕ being true in all states reachable by hybrid program α from initial states that satisfy ψ. Our verification algorithm defines the function $prove(\psi \vdash [\alpha]\phi)$ for verifying this safety statement recursively by refining the $\mathsf{d}\mathcal{L}$ and DAL calculi to an automatic verification algorithm. The discrete base cases are discussed in Section 6.2.1. In Section 6.2.2, we contrast discrete and differential induction. Section 6.2.4 shows the fixedpoint algorithm for computing differential invariants for differential equations, and Section 6.2.6 for computing loop invariants. In Section 6.2.7, we explain the interplay of local and global fixedpoint loops of our verification algorithm.

6.2.1. Verification by Symbolic Decomposition

The cases of the verification procedure *prove* where $\mathsf{d}\mathcal{L}$ enables us to verify a property of a hybrid program directly by decomposing it into a property of its parts are shown in Figure 6.1. They correspond to the cases where D-rules of Figure 3.3 can be applied depending on the top-level program operator. For a concise presentation, the case in line 1 introduces an auxiliary variable $\hat{x}$ to handle discrete assignments by substituting $\hat{x}$ for x in $\phi_x^{\hat{x}}$: E.g., $x \geq 2 \vdash [x := x-1]x \geq 0$ is shown by proving $x \geq 2 \wedge \hat{x} = x-1 \vdash \hat{x} \geq 0$. Our implementation uses rule D9 and discrete jump sets to avoid auxiliary variables according to Chapter 2. State checks $?\chi$ are shown by assuming the test succeeds, i.e., χ holds true (line 4), nondeterministic choices split into their alternatives (line 5), sequential compositions are proven using nested modalities (line 7), and random assignments by universal quantification (line 8), which is an optimisation of the handling in Chapter 3. Random updates $[x := *]\phi$ that nondeterministically assign an arbitrary real value to x are definable by $[x' = 1 \cup x' - 1]\phi$.

The base case in line 9, where ϕ is a formula of first-order real arithmetic, can be proven by real quantifier elimination [CH91]. Despite its complexity, this can remain feasible, because the formulas resulting from our algorithm do not depend on the solutions of differential equations but only their right-hand sides. In addition, we use the respective Skolemisation and free variable rules from Chapter 2 to resolve quantifiers in front of modalities.

The algorithm in Figure 6.1 recursively reduces safety of hybrid programs to properties of continuous evolutions or of repetitions, which we verify in the next sections.

```
1  function prove(ψ ⊢ [x := θ]φ):
2    return prove(ψ ∧ x̂ = θ ⊢ φ^x̂_x)
3    where x̂ is a new auxiliary variable
4  function prove(ψ ⊢ [?χ]φ): return prove(ψ ∧ χ ⊢ φ)
5  function prove(ψ ⊢ [α ∪ β]φ):
6    return prove(ψ ⊢ [α]φ) and prove(ψ ⊢ [β]φ)  /* i.e. ψ ⊢ [α]φ ∧ [β]φ */
7  function prove(ψ ⊢ [α;β]φ): return prove(ψ ⊢ [α][β]φ)
8  function prove(ψ ⊢ [x := *]φ): return prove(ψ ⊢ ∀x φ)
9  function prove(ψ ⊢ φ) where isFirstOrder(φ):
10   return QE(ψ → φ)
```

Figure 6.1.: $\mathsf{d}\mathcal{L}$-based verification by symbolic decomposition

6.2.2. Discrete and Differential Induction, Differential Invariants

In the sequel, we present algorithms for verifying loops by discrete induction (which corresponds to rule G3) and continuous evolutions by differential induction, which is a continuous form of induction and corresponds to rule G5 from Figure 3.3. In either case, we prove that an invariant F holds initially (in the states characterised symbolically by ψ, thus $\psi \to F$ is valid) and finally entails the postcondition ϕ (i.e., $F \to \phi$). The cases differ in their induction step.

6.1 Definition (Discrete induction). Formula F is a *(discrete) invariant* of $\psi \to [\alpha^*]\phi$ iff the following formulas are valid:

1. $\psi \to F$ (induction start), and
2. $F \to [\alpha]F$ (induction step).

An invariant is *sufficiently strong* if $F \to \phi$ is valid.

6.2 Definition (Continuous invariants). Let $\mathcal{D}$ be a differential equation. Formula F is a *continuous invariant* of $\psi \to [\mathcal{D} \wedge \chi]\phi$ iff the following formulas are valid:

1. $\psi \wedge \chi \to F$ (induction start), and
2. $F \to [\mathcal{D} \wedge \chi]F$ (induction step).

Again, a continuous invariant is *sufficiently strong* if $F \to \phi$ is valid.

To prove that F is a continuous invariant, it is sufficient by Theorem 3.25 and rule G5 to check a condition on the directional derivatives of all terms of the formula, which expresses that no atomic subformula of F changes its truth-value along the

dynamics of the differential equation. This condition is much easier to check than a reachability property ($F \to [\mathcal{D} \wedge \chi]F$) of a differential equation. Applications like aircraft maneuvers need invariants with mixed equations and inequalities. Thus, we use the generalisation of directional derivatives from functions to logical formulas according to Definition 3.14.

6.3 Definition (Differential induction). Let the differential equation system $\mathcal{D}$ be $x_1' = \theta_1, \ldots, x_n' = \theta_n$. Formula F is a *differential invariant* of $\psi \to [\mathcal{D} \wedge \chi]\phi$ iff the following formulas are valid:

1. $\psi \wedge \chi \to F$ (induction start), and
2. $\chi \to F'_{\mathcal{D}}$ (induction step).

Where $F'_{\mathcal{D}}$ is the result of substituting the differential equation $\mathcal{D}$ into DF, which corresponds to the conjunction of all directional derivatives of atomic formulas in F in the direction of the vector field of $\mathcal{D}$ (the partial derivative of b by x_i is $\frac{\partial b}{\partial x_i}$):

$$F'_{\mathcal{D}} \equiv \bigwedge_{(b \sim c) \in F} \left(\left(\sum_{i=1}^{n} \frac{\partial b}{\partial x_i} \theta_i \right) \sim \left(\sum_{i=1}^{n} \frac{\partial c}{\partial x_i} \theta_i \right) \right) \quad \text{for } \sim \in \{=, \geq, >, \leq, <\}.$$

These partial derivatives of terms are well-defined in the Euclidean space spanned by the variables and can be computed symbolically (Definition 3.14).

6.4 Proposition (Principle of differential induction). *All differential invariants are continuous invariants.*

Proof. This proposition is a corollary to Theorem 3.25, which shows that rule G5 is sound. □

In Sections 6.2.4–6.2.6, we present algorithms for finding differential invariants for differential equations, and for finding global invariants for repetitions.

6.2.3. Flight Dynamics in Air Traffic Control

Aircraft collision avoidance maneuvers resolve conflicting flight paths, e.g., by roundabout maneuvers [TPS98], see Figure 3.1. Their nontrivial dynamics makes safe separation of aircraft difficult to verify [TPS98, MF01, DMC05, DPR05, PC07, HKT07]. The parameters of two aircraft at (planar) position $x = (x_1, x_2) \in \mathbb{R}^2$ and $y = (y_1, y_2)$ with angular orientation ϑ and ς are illustrated in Figure 3.1a (with $\vartheta = 0$). Their dynamics is determined by their linear speeds $v, u \in \mathbb{R}$ and angular speeds $\omega, \varrho \in \mathbb{R}$, see [TPS98]:

$$x_1' = v\cos\vartheta \quad x_2' = v\sin\vartheta \quad \vartheta' = \omega \quad y_1' = u\cos\varsigma \quad y_2' = u\sin\varsigma \quad \varsigma' = \varrho \tag{6.1}$$

In safe flight configurations, aircraft are separated by at least distance p:

$$(x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2 \tag{6.2}$$

To handle the transcendental functions in (6.1), we axiomatise sin and cos by differential equations and reparametrise the system using a linear velocity vector $d = (d_1, d_2) := (v\cos\vartheta, v\sin\vartheta) \in \mathbb{R}^2$ according to Section 3.4.2:

$$\left[\begin{array}{ccccc} x_1' = d_1 & x_2' = d_2 & d_1' = -\omega d_2 & d_2' = \omega d_1 & t' = 1 \\ y_1' = e_1 & y_2' = e_2 & e_1' = -\varrho e_2 & e_2' = \varrho e_1 & s' = 1 \end{array}\right] \quad (\mathcal{F})$$

Equations $(\mathcal{F})$ and (6.1) are equivalent up to reparameterisation. Here, we add clock variables t, s that we need for synchronising collision avoidance maneuvers in Chapter 8. By a simple computation, $d_1^2 + d_2^2 \geq a^2$ is a differential invariant of $(\mathcal{F})$, thereby showing that the linear speed of aircraft does not drop below some stalling speed a during maneuvers:

$$\begin{aligned} & (d_1^2 + d_2^2 \geq a^2)'_{\mathcal{F}} \;\equiv\; (d_1^2 + d_2^2 \geq a^2)'_{(d_1' = -\omega d_2, d_2' = \omega d_1)} \\ \equiv\; & \frac{\partial(d_1^2 + d_2^2)}{\partial d_1}(-\omega d_2) + \frac{\partial(d_1^2 + d_2^2)}{\partial d_2}\omega d_1 \geq \frac{\partial a^2}{\partial d_1}(-\omega d_2) + \frac{\partial a^2}{\partial d_2}\omega d_1 \\ \equiv\; & 2d_1(-\omega d_2) + 2d_2\omega d_1 \geq 0 \;. \end{aligned}$$

6.2.4. Local Fixedpoint Computation for Differential Invariants

Figure 6.2 depicts the fixedpoint algorithm for constructing differential invariants for each continuous evolution $\mathcal{D} \wedge \chi$ with a differential equation system $\mathcal{D}$. The algorithm in Figure 6.2 (called *Differential Saturation*) successively refines the domain χ by differential invariants until saturation, i.e., χ accumulates enough information to become a strong invariant that implies postcondition ϕ (line 2). If domain χ already entails ϕ, then $\psi \vdash [\mathcal{D} \wedge \chi]\phi$ is proven (line 2). Otherwise, the algorithm considers candidates F for augmenting χ (line 3). If F is a differential invariant (line 4), then χ can soundly be refined to $\chi \wedge F$ (line 5) without affecting the states reachable by $\mathcal{D} \wedge \chi$ (Proposition 6.5 below). Then, the fixedpoint loop repeats (line 6). At each iteration of this fixedpoint loop, the previous invariant χ can be used to prove the next level of refinement $\chi \wedge F$ (line 4). The refinement of the dynamics at line 5 is correct by the following proposition, using that the conditions in line 4 imply that F is a differential invariant and, thus, a continuous invariant by Proposition 6.4.

6.5 Proposition (Differential saturation). *If F is a continuous invariant of the formula $\psi \to [\mathcal{D} \wedge \chi]\phi$, then $\psi \to [\mathcal{D} \wedge \chi]\phi$ and $\psi \to [\mathcal{D} \wedge \chi \wedge F]\phi$ are equivalent.*

```
1 function prove(ψ ⊢ [D ∧ χ]φ):
2   if prove(∀^D(χ → φ)) then return true /* property proven */
3   for each F ∈Candidates(ψ ⊢ [D ∧ χ]φ, χ) do
4     if prove(ψ ∧ χ ⊢ F) and prove(∀^D(χ → F'_D)) then
5       χ := χ ∧ F        /* refine by differential invariant */
6       goto 2;           /* repeat fixedpoint loop */
7   end for
8   return not provable using candidates
```

Figure 6.2.: Fixedpoint algorithm for differential invariants (*Differential Saturation*)

Proof. The proof is stronger version of the soundness of D15 by Theorem 3.25: Let F be a continuous invariant, which implies that $\psi \vdash [\mathcal{D} \wedge \chi]F$ is valid. Let ν be a state satisfying ψ (otherwise there is nothing to show). Then, $\nu \models [\mathcal{D} \wedge \chi]F$. Since this means that F is true all along all flows φ of $\mathcal{D} \wedge \chi$ that start in ν (Definition 3.10), the latter and $\mathcal{D} \wedge \chi \wedge F$ have the same dynamics and the same reachable states from ν, i.e., $(\nu, \omega) \in \rho(\mathcal{D} \wedge \chi)$ holds if and only if $(\nu, \omega) \in \rho(\mathcal{D} \wedge \chi \wedge F)$ (Definition 2.7). Thus, we can conclude that $\psi \vdash [\mathcal{D} \wedge \chi]\phi$ and $\psi \vdash [\mathcal{D} \wedge \chi \wedge F]\phi$ are equivalent, because their semantics uses the same transition relation. □

This progressive differential saturation turns out to be crucial in practice. For instance, the aircraft separation property (6.2) cannot be proven until ($\mathcal{F}$) has been refined by invariants for d and e, because these determine x' and y'.

Function *Candidates* determines candidates for induction (line 3) depending on transitive differential dependencies, as will be explained in Section 6.2.5. When these are insufficient for proving $\psi \vdash [\mathcal{D} \wedge \chi]\phi$, the algorithm fails (line 8, with improvements in subsequent sections). Again, $\forall^\alpha \phi$ denotes the *universal closure* of ϕ. It is required in lines 2 and 4, because the respective formulas need to hold in *all* states (that satisfy χ), which we will improve on in Section 6.4.

6.2.5. Dependency-Directed Induction Candidates

In this section, we construct likely candidates for differential induction (function *Candidates* in Figure 6.2). Later, we use the same procedure for finding global loop invariants. We construct two kinds of candidates in an order induced by differential dependencies. By following the effect of hybrid systems symbolically along their decompositions, our verification algorithm enriches precondition ψ successively with more precise information about the symbolic prestate as obtained by the symbolic decompositions and proof steps in Figure 6.1 and 6.2. To exploit this, we first look for invariant symbolic state information that accumulated in ψ and ϕ during

the iterative symbolic decomposition by selecting subformulas that are not yet contained in χ. In practice, this gives particularly good candidates for highly parametric hybrid systems.

Secondly, we generate parametric invariants. Let $V = \{x_1, \ldots, x_n\}$ be a set of relevant variables. We choose fresh names $a^{(l)}_{i_1,\ldots,i_n}$ for *formal parameters* of the invariant candidates and build polynomials $p_1, \ldots, p_k$ of degree d with variables V using formal parameters as symbolic coefficients:

$$p_l := \sum_{i_1+\cdots+i_n \leq d} a^{(l)}_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n} \quad \text{for } 1 \leq l \leq k \ .$$

We define the set of *parametric candidates* (operator $\vee$ is similarly):

$$\mathit{ParaForm}(k, d, V) \ := \ \left\{ \bigwedge_{l=1}^{i} p_l \geq 0 \wedge \bigwedge_{l=i+1}^{k} p_l = 0 \ : \ 0 \leq i \leq k \right\} \ .$$

For instance, the parametric candidate $a_{0,0} + a_{1,0}d_1 + a_{0,1}x_2 = 0$ yields a differential invariant of $(\mathcal{F})$ for the choice $a_{0,0} = 0$, $a_{1,0} = 1$, $a_{0,1} = \omega$. By simple combinatorics, *ParaForm* contains $k+1$ candidates with $k\binom{n+d}{d}$ formal parameters $a^{(l)}_{i_1,\ldots,i_n}$, which are existentially quantified. Existence of a common satisfying instantiation for these parameters can be expressed by adding $\exists a^{(l)}_{i_1,\ldots,i_n}$ to the resulting $\mathsf{d}\mathcal{L}$ formulas. For this to be feasible, the number of parameters is crucial, which we minimise by respecting (differential) dependencies.

To accelerate the differential saturation process in Section 6.2.4, it is crucial to explore candidates in a promising order from simple to complex, because the algorithm in Figure 6.2 uses successful differential invariants to refine the dynamics, thereby simplifying subsequent proofs: E.g., (6.2) is only provable after the dynamics has been refined with invariants for d and e, because x' and y' depend on the direction d and e. In fact, safety of roundabouts crucially depends on compatible directions of the aircraft, see Figure 3.1c. We construct candidates in a natural order based on variable occurrence that is consistent with the *differential dependencies* of the differential equations. For a differential equation $\mathcal{D}$, variable x depends on variable y according to the differential equation system $\mathcal{D}$ if y occurs on the right-hand side for x' (or transitively so). The resulting set $\mathit{depend}(\mathcal{D})$ of dependencies is the transitive closure of

$$\{(x, y) \ : \ (x' = \theta) \in \mathcal{D} \text{ and } y \text{ occurs in } \theta\}$$

From the differential equation system $(\mathcal{F})$, we determine the differential dependencies indicated as arrows (pointing to the dependent variables x) in Figure 6.3.

From these dependencies we determine an order on candidates. The idea is that, as the value of x_1 in $(\mathcal{F})$, depends on that of d_1, it makes sense to look for invariant expressions of d_1 first, because refinements with these help differential saturation

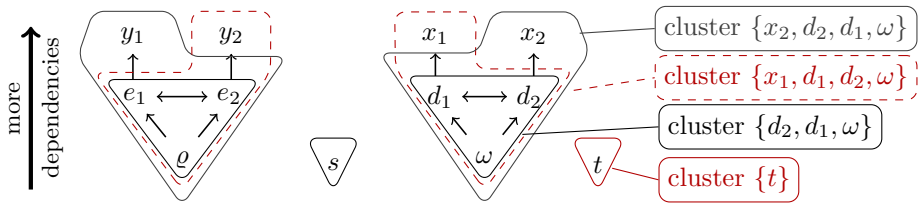


Figure 6.3.: Differential dependencies (arrows) and (triangular) variable clusters of $(\mathcal{F})$

in proving invariant expressions involving also x_1. Thus, we order variables by differential dependencies, which resembles the back substitution order in Gaussian elimination (if, in triangular form, x_1 depends on d_1 then equations for d_1 must be solved first, except that, in the differential case, variables can remain mutually dependent like, e.g., d_1 and d_2). Now we call a set V of variables a *cluster* of the differential equation $\mathcal{D}$ iff V is closed with respect to $\mathit{depend}(\mathcal{D})$, i.e., variables of V only depend on variables in V:

$$x \in V \text{ and } (x, y) \in \mathit{depend}(\mathcal{D}) \Rightarrow y \in V$$

The resulting variable clusters for system $(\mathcal{F})$ are marked as triangular shapes in Figure 6.3. Finally, we choose candidates from ψ and $\mathit{ParaForm}(k, d, V)$ starting with candidates whose variables lie in small clusters V and cover larger fractions of that cluster. Thus, the differential invariant $d_1^2 + d_2^2 \geq a^2$ of Section 6.2.3 within cluster $\{d_2, d_1, \omega\}$ can be discovered before invariants like $d_1 = -\omega x_2$ that involve x_2, because x_2 depends on d_2. Similarly, $d_1 = -\omega(x_2 - c_2)$ will be discovered before $\|x - y\|^2 \geq p^2$, as the latter lies in a larger cluster with worse coverage of that cluster. The successive differential saturation process along these dependencies further helps to keep the degrees in *ParaForm* small.

6.2.6. Global Fixedpoint Computation for Loop Invariants

With the uniform setup of $\mathsf{d}\mathcal{L}$, we can adapt the algorithm in Figure 6.2 easily to obtain a fixedpoint algorithm for loops $(\psi \vdash [\alpha^*]\phi)$ in place of continuous evolutions $(\psi \vdash [\mathcal{D} \wedge \chi]\phi)$: In line 4 of Figure 6.2, we replace the induction step from Definition 6.3 by the step for loops (Definition 6.1). As an optimisation, invariants χ of previous iterations can be exploited as refinements of the hybrid system dynamics, similar to previous differential invariants that can be used in future iterations by refining the dynamics using differential saturation:

6.6 Proposition (Loop saturation). *If χ is a discrete invariant of $\psi \to [\alpha^*]\phi$, then $\chi \wedge F$ is a discrete invariant iff $\psi \to F$ and $\chi \wedge F \to [\alpha](\chi \to F)$ are valid.*

Proof. Let χ be a discrete invariant of $\psi \to [\alpha^*]\phi$. Let, further, F be a discrete invariant of $\psi \to [\alpha^*]\phi$. Then $\psi \to F$ and $F \to [\alpha]F$ are valid by Definition 6.1.

Hence, trivially, $F \to [\alpha](\chi \to F)$ is valid, because all states that satisfy F also satisfy the weaker property $\chi \to F$. Especially, $\chi \wedge F \to [\alpha](\chi \to F)$ is valid. Finally, the validity of $\psi \to \chi \wedge F$ clearly entails $\psi \to F$.

Conversely, let, χ be a discrete invariant. Let, further, $\chi \wedge F \to [\alpha](\chi \to F)$ and $\psi \to F$ be valid. For $\chi \wedge F$ to be a discrete invariant, we have to show that F satisfies the induction step of Definition 6.1 (the induction start $\psi \to \chi \wedge F$ is an immediate combination of the validity of $\psi \to \chi$ and $\psi \to F$). Since χ is a discrete invariant, $\chi \to [\alpha]\chi$ is valid, which entails $\chi \wedge F \to [\alpha]\chi$ as a special case. Since $\chi \wedge F \to [\alpha](\chi \to F)$ is valid and $\chi \wedge F \to [\alpha]\chi$ is valid, we conclude that $\chi \wedge F \to [\alpha](\chi \wedge F)$ is valid for the following reason. Let ν be a state satisfying the initial constraints $\chi \wedge F$. Then $\nu \models [\alpha]\chi$ and $\nu \models [\alpha](\chi \to F)$. Hence, all states ω reachable from ν by α satisfy $\omega \models \chi$ and $\omega \models \chi \to F$. Thus, they satisfy $\omega \models \chi \wedge F$, essentially by modus ponens. Consequently, we have shown that $\chi \wedge F \to [\alpha](\chi \wedge F)$ is valid. and, hence, $\chi \wedge F$ is a discrete invariant of $\psi \to [\alpha^*]\phi$. □

The induction step from Proposition 6.6 can generally be proven faster, because it is a weaker property than that of Definition 6.1.

To adapt our approach from Section 6.2.5 to loops, we use discrete data-flow and control-flow dependencies of α. Dependencies can be determined immediately from the syntax of HP. There is a direct *data-flow dependency* with the value of x depending on y, if $x := \theta$ or $x' = \theta$ occurs in α with a term θ that contains y. Accordingly, there is a direct *control-flow dependency*, if, for any term θ, $x := \theta$ or $x' = \theta$ occurs in α after a $?\chi$ containing y. The respective data-flow and control-flow dependencies are the transitive closures of these relations.

```
1 function prove(ψ ⊢ [α*]φ):
2   χ := true     /* currently known invariant of ψ ⊢ [α*]φ */
3   if prove(∀α(χ → φ)) then return true /* property proven */
4   for each F ∈IndCandidates(ψ ⊢ [α*]φ, χ) do
5     if prove(ψ ∧ χ ⊢ F) and prove(∀α(χ ∧ F → [α](χ → F))) then
6       χ := χ ∧ F        /* refine by discrete invariant */
7       goto 3;           /* repeat fixedpoint loop */
8   end for
9   return not provable using candidates
```

Figure 6.4.: Fixedpoint algorithm for discrete loop invariants (loop saturation)

The algorithm in Figure 6.4 verifies loops. It is a direct adaption of that in Figure 6.2, except that it uses Proposition 6.6 as an induction step for loops. The algorithm in Figure 6.4 performs a fixedpoint computation for loops and recursively combines the local differential invariants obtained by differential saturation to form

a global invariant. It recursively uses *prove* for verifying its subtasks, which handle the discrete switching behaviour according to Figure 6.1 and infer local differential invariants according to differential saturation by the fixedpoint algorithm in Figure 6.2.

6.2.7. Interplay of Local and Global Fixedpoint Loops

The local and global fixedpoint algorithms jointly verify correctness properties of HP. Their interplay needs to be coordinated with fairness. If the local fixedpoint algorithm in Figure 6.2 does not converge, stronger invariants may need to be found by the global fixedpoint algorithm which iteratively result in stronger preconditions ψ_i for the local algorithm, see Figure 6.5.

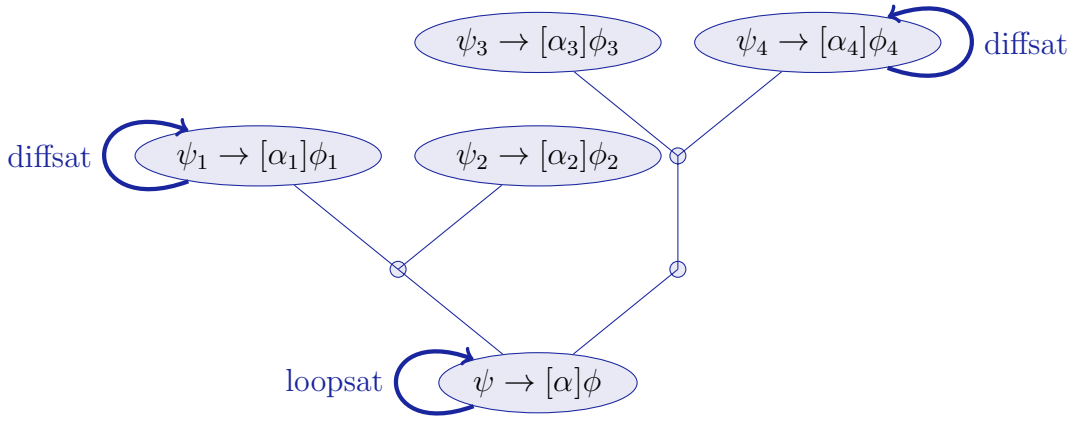


Figure 6.5.: Interplay of local (diffsat) and global (loopsat) fixedpoints verification loops during symbolic decomposition

Thus, the local fixedpoint algorithm should stop when it cannot prove its postcondition, either because of a counterexample or because it runs out of candidates for differential invariants. As in the work of Prajna [PJP07], the degrees of parametric invariants, therefore, need to be bounded and increased iteratively. As in [PJP07], there is no natural measure for how these degrees should be increased. Instead, here, we exploit the fact that the candidates of *Candidates* are independent and we explore them in parallel with fair time interleaving. There, the interleaving by iterative timeout increase is similar to iterative background closures in Chapter 5, except that it works for full subproofs instead of just one local quantifier elimination call.

In well-designed control-loop systems, global fixedpoints are easier to find than local fixedpoints of differential invariants, because the global invariant often coincides with the controllability region of the system. In these systems, the minimal

control objective is to keep the system in the controllable state region. That is, if the system is in a controllable state, i.e., there is a control choice such that the system can remain safe, suitable controllers will pick one such control option such that the system again ends up in a controllable state. Inductively, this controllability region directly corresponds to a global system invariant.

6.3. Soundness

Even though the interplay of the fixedpoint verification algorithms in this section is already quite complicated, we can exploit the fact that they are working with formula transformations in the verification logic $\mathsf{d}\mathcal{L}$ to ensure soundness. Since all formula transformations in the algorithm are justified by sound proof rules, soundness is fully captured in the logic and the algorithm can only be incomplete but not unsound:

6.7 Theorem (Soundness). *The verification algorithm in Section 6.2 is sound, i.e., whenever* `prove`$(\psi \vdash [\alpha]\phi)$ *returns "*`true`*", the* $\mathsf{d}\mathcal{L}$ *formula* $\psi \rightarrow [\alpha]\phi$ *is true in all states, i.e., all states reachable by* α *from states satisfying* ψ *satisfy* ϕ*.*

Proof. Soundness is a direct consequence of Theorem 3.25, as every statement that returns true is justified by a sound proof rule of DAL. To show this in full detail: We prove by induction on the structure of the algorithm.

- In the base case (line 9 of Figure 6.1), *prove* returns the result of quantifier elimination, which is a sound decision procedure [CH91].

- If α is of the form $x := \theta$, the algorithm in line 1 of Figure 6.1 is responsible. If it returns "true", then $\mathit{prove}(\psi \wedge \hat{x} = \theta \vdash \phi_x^{\hat{x}})$ has returned "true". Hence, by induction hypothesis, $\psi \wedge \hat{x} = \theta \vdash \phi_x^{\hat{x}}$ is valid. Now, because $\hat{x}$ was a fresh variable, the substitution lemma can be used to show that $\psi \vdash \phi_x^{\theta}$ and $\psi \vdash [x := \theta]\phi$ are valid. Hence, the result of *prove* is sound.

- If α is of the form $x := *$, the algorithm in line 8 of Figure 6.1 is responsible. Soundness can be proven directly using the fact that ϕ being true after *all* random assignments to x is equivalent to ϕ being true for all real values of x. Hence, $\psi \vdash [x := *]\phi$ is valid if and only if $\psi \vdash \forall x\, \phi$ is.

- The other cases of Figure 6.1 are similar consequences of the soundness of the DAL rules in Figure 3.3 by Theorem 3.25.

- If α is of the form $\mathcal{D} \wedge \chi$ for a differential equation system $\mathcal{D}$, the algorithm in Figure 6.2 is responsible. If it returns "true" in line 2 in the first place, then the calls to *prove* in line 2 must have resulted in "true", hence, by induction

hypothesis, χ entails ϕ. Thus, soundness of rule D13 justifies soundness as follows. Postcondition ϕ is true in a subregion of the evolution domain χ. Thus $\psi \vdash [\mathcal{D} \wedge \chi]\phi$ is valid, trivially, because all evolutions along $\mathcal{D} \wedge \chi$ always satisfy χ and, hence, ϕ. If, however, χ was changed in line 5 during the fixedpoint computation, then the calls to *prove* for the properties in line 4 must have returned "true". Thus, by induction hypothesis, the dL formulas $\psi \wedge \chi \vdash F$ and $\forall^{\mathcal{D}}(\chi \rightarrow F'_{\mathcal{D}})$ are valid, hence D15 justifies soundness as follows. Formula F is a differential invariant of $\psi \vdash [\mathcal{D} \wedge \chi]\phi$ by Definition 6.3. Consequently, by Proposition 6.4, F also is a continuous invariant (Definition 6.2). Thus, by Proposition 6.5, the dL formulas $\psi \vdash [\mathcal{D} \wedge \chi]\phi$ and $\psi \vdash [\mathcal{D} \wedge \chi \wedge F]\phi$ are equivalent, and we can (soundly) verify the former by proving the latter. Consequently, the modification of the evolution domain χ to $\chi \wedge F$ in line 5 is sound, because the algorithm will continue proving a refined but equivalent formula for a refined but equivalent system.

- If α is a loop of the form β^*, the proof is similar to the case for differential equations, except that it uses Proposition 6.6 instead of Proposition 6.4. □

Since reachability of hybrid systems is undecidable, our algorithm must be incomplete. It can fail to converge when the required invariants are not expressible in first-order logic (yet, they are always expressible in dL by Theorem 2.17).

6.4. Optimisations

6.4.1. Sound Interleaving with Numerical Simulation

During fixedpoint computations, wrong choices of candidates are time consuming. Thus, in practice, it is important to discover futile attempts quickly. For this, we use non-exhaustive numerical simulation to look for a counterexample for each candidate. To prevent rejecting good candidates due to numerical errors, we discard fragile counterexamples. We consider counterexamples with distance $<\varepsilon$ to safe states as *fragile*, because small numerical perturbations could make it safe (the right x in Figure 6.6 marks fragile examples or counterexamples). The left mark in 6.6, instead, is a *robust* counterexample, i.e., only large ($\geq \varepsilon$) perturbations could make it safe. Robust counterexamples can be ensured by replacing, e.g., $a \geq b$ by $a \geq b + \varepsilon$ in the formulas given to numerical reachability simulation for some estimate $\varepsilon \geq 0$ of the numerical error. Unlike in other approaches [ADG03, LT05, PAM+05, PJP07, PC07], numerical errors are *not* critical for soundness here, because safety is exclusively established by sound symbolic verification.

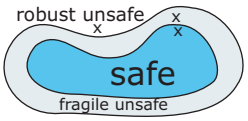


Figure 6.6.: Robustness

We can further exploit the symbolic decomposition performed by our algorithm in Section 6.2 and prefix recursive calls to $prove(\psi \vdash [\alpha]\phi)$ with a partial simulation of α. Using approximate cylindric algebraic decomposition [CH91] in the *FindInstance* function of Mathematica, we find good samples of states satisfying ψ to start the simulation of α.

6.4.2. Optimisations for the Verification Algorithm

Formulas with variables that do not change in a fragment of a HP are trivial invariants, as their truth-value is unaffected. For instance, $\omega = \varrho$ is a trivial invariant of system $(\mathcal{F})$. Hence, it can be used as an invariant without proof. A formula like $\omega^2(d_1^2 + d_2^2) > r^2$ in ψ, instead, is not trivially invariant, because d_i changes during $(\mathcal{F})$. Still, it has invariant consequences like $\omega \neq 0$. To make use of these direct and indirect trivial invariants from ψ , we (soundly) weaken all universal closures of the form $\forall^\alpha \phi$ in lines 2 and 4 of Figure 6.2 by $\psi \vdash \forall^\alpha \phi$, which directly reflects the rule context instantiations in Definition 2.11.

6.5. Experimental Results

As an example with nontrivial dynamics, we analyse aircraft roundabout maneuvers [TPS98]. Curved flight as in roundabouts is challenging for verification, because of its transcendental solutions. The maneuver in Figure 3.1b from [TPS98] and the maneuver in Figure 3.1d from [PC07, Pla08a] are not flyable, because they still involve a few instant turns. A flyable roundabout maneuver without instant turns is depicted in Figure 6.7. We verify safety properties for most (but not yet all) phases of Figure 6.7 and provide verification results in Table 6.1. Details on the case studies are presented in Part III. Finally, note that the required invariants found by our approach for the roundabout maneuver cannot even be found from Differential Gröbner Bases [Man91].

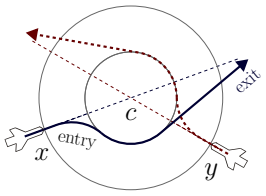


Figure 6.7.: Flyable aircraft roundabout

Verification results for roundabout aircraft maneuvers [TPS98, DPR05, PC07, Pla08a, PC08b] and the European Train Control System (ETCS) are in Table 6.1. Results are from a 2.6GHz AMD Opteron with 4GB memory. Memory consumption of quantifier elimination is shown in Table 6.1, excluding the front-end. The results are only slightly worse on a 1.7GHz Pentium M laptop with 1GB. The dimension of the continuous state space is indicated. Notice that we handle *all* these variables symbolically leading to state spaces up to $\mathbb{R}^{28}$.

Table 6.1.: Experimental results for differential invariants as fixedpoints

Case study	Time(s)	Memory(MB)	Proof steps	Dim
tangential roundabout (2 aircraft)	14	8	117	13
tangential roundabout (3 aircraft)	387	42	182	18
tangential roundabout (4 aircraft)	730	39	234	23
tangential roundabout (5 aircraft)	1964	88	317	28
ETCS kernel safety	41	28	53	9
ETCS binary safety	56	27	147	14
ETCS safety	183	87	169	14

6.6. Summary

We have presented a *sound* algorithm for verifying hybrid systems with nontrivial dynamics. It handles differential equations using differential invariants instead of requiring solutions of the differential equations, because the latter quickly yield undecidable arithmetic. We compute differential invariants as fixedpoints using $\mathsf{d}\mathcal{L}$ as a verification logic for hybrid systems. In the logic we can decompose the system for computing local invariants and we obtain sound recombinations into global invariants. Moreover, we introduce a differential saturation procedure that verifies more complicated properties by refining the system dynamics successively in a sound way. We validate our algorithm on challenging *roundabout collision avoidance maneuvers* for aircraft and on collision avoidance protocols for trains.

Our algorithm works particularly good for highly parametric hybrid systems, because their parameter constraints can be combined faster to find invariants than for systems with a single initial state, where simulation is more appropriate. Our decompositional approach exploits locality in system designs. In well-designed systems, subsystems do not generally depend on all other parts of the system but are built according to modularity and locality principles in engineering. In these cases, the $\mathsf{d}\mathcal{L}$ calculus achieves good decompositions and the required invariants for one part of the system only need very little information about other parts of the systems, so that the fixedpoint algorithm terminates faster. Our algorithm probably performs worse for systems that violate locality principles. We want to validate this in further experiments and analyse scalability.

Differential induction and the logic $\mathsf{d}\mathcal{L}$ generalise to liveness properties and to systems with disturbances (Chapter 3). In future work, we want to generalise the synthesis of corresponding differential (in)variants. Other invariant constructions for differential equations, e.g., [RCT05] can be added and lifted to hybrid systems using our uniform algorithm.

Part III.

Case Studies and Applications in Hybrid Systems Verification

Chapter 7.

Parametric European Train Control System

Contents

Synopsis

Most hybrid systems allow for several degrees of freedom so that their correct functioning depends on how their parameters are adjusted. As a case study in parametric verification of hybrid systems, we analyse a fully parametric cooperation protocol of the *European Train Control System* (ETCS). We use the calculus of our logic for hybrid systems to discover the required parameter constraints for safe train control and characterise these constraints equivalently by appropriate reachability properties of the hybrid dynamics. We formally verify controllability, safety, liveness, and reactivity properties of the ETCS cooperation protocol that entail collision freedom and show that the ETCS protocol remains correct under perturbation by disturbances in the train dynamics.

7.1. Introduction

Most hybrid systems contain substantial degrees of freedom including how specific parameters are instantiated or adjusted [DHO06, FM06, DMO+07, BBW07, ERT02]. Yet, virtually any hybrid system is only safe under certain constraints on these parameters. For instance, the European Train Control System (ETCS) has a wide range of different possible configurations of trains, track layouts, and different driving circumstances. Still, it is only safe under certain conditions on external parameters, e.g., when the speed of each train does not exceed its specific braking power given the remaining distance to the next train. Similarly, internal control design parameters for speed control and braking triggers need to be adjusted in accordance with the underlying train dynamics. Moreover, parameters must be constrained such that the system remains correct when passing from continuous models with instant reactions to sampled data discrete time controllers of hardware implementations. Finally, parameter choices must preserve correctness robustly in the presence of disturbances caused by unforeseen external forces or internal modelling inaccuracies of ideal-world dynamics. Yet, determining the range of external parameters and the choice of internal design parameters for which ETCS is safe, is not possible just by looking at the model, even less so for train dynamics that is subject to disturbance.

Likewise, it is difficult to read off the parameter constraints that are required for correctness from a failed verification attempt of model checkers [CGP99, Hen96, MPM05, Fre05], as these often exploit non-structural heuristic splits of the state space, which can lead to nonuniform parameter requirements for different states. While approaches like counterexample-guided abstraction refinement [CGJ+03, Fre05] are highly efficient in generating and learning from spurious counterexamples of a failed verification attempt of an abstract hybrid system, they stop when the counterexamples remain in the concrete case. For discovering constraints on free para-

meters, though, even concrete models will have counterexamples until all required parameter constraints have been identified. Even worse, concrete numeric values of a counterexample trace, as produced by a model checker with state splitting, cannot simply be translated into a uniform global constraint on the free parameters of the system which would prevent this kind of error.

Instead, we use symbolic decompositions for systematically exploring the design space of a hybrid system and for discovering correctness constraints on free parameters. Our approach follows a structural symbolic decomposition in our verification logic $\mathsf{d}\mathcal{L}$ [Pla07b, Pla08b] such that the discovered parametric constraints directly relate to the behaviour of the hybrid system. In this chapter, we demonstrate that parameter constraint identification by symbolic decomposition is feasible even for realistic models of train cooperation protocols in ETCS with rich specifications that include safety, controllability, reactivity, and liveness properties, which are uniformly expressible in $\mathsf{d}\mathcal{L}$. Using the parametric constraints so discovered, we verify correctness properties of the ETCS cooperation protocol that entail collision freedom. Moreover, we verify those correctness properties of the parametric ETCS case study almost fully automatically in our verification tool KeYmaera [PQ08a].

Contributions

As our central contribution we demonstrate that verification of realistic fully parametric hybrid systems at the cooperation layer of traffic agent collaboration protocols is feasible using a logic-based verification approach. We discover all relevant safety constraints on free parameters, including external system parameters and internal design parameters of controllers. Our main technical contribution is that we characterise these parameter constraints equivalently in terms of properties of the train dynamics and verify controllability, reactivity, safety, and liveness properties of ETCS.

7.1.1. Related Work

Model checkers [CGP99] for hybrid systems, e.g. HyTech [AHH96] and PHAVer [Fre05], verify by exploring the state space of the system as exhaustively as possible. In contrast to our approach they need concrete numbers for most of the parameters and cannot verify liveness or existential properties like whether and how a control parameter can be instantiated safely.

Batt et al. [BBW07] give heuristics for splitting regions by linear constraints that can be used to determine parameter constraints. Their approach is not applicable for realistic systems like ETCS which require nonlinear parameter constraints for correctness.

Peleska et al. [PGHD04] and Faber and Meyer [FM06] verify properties of trains. They do not, however, verify hybrid dynamics, i.e., they do not take into account

the actual movement of trains. Yet, the physical dynamics is crucial for faithful train models and for showing actual collision freedom, because, after all, collision freedom is a property of controlled movement.

Structure of this Chapter

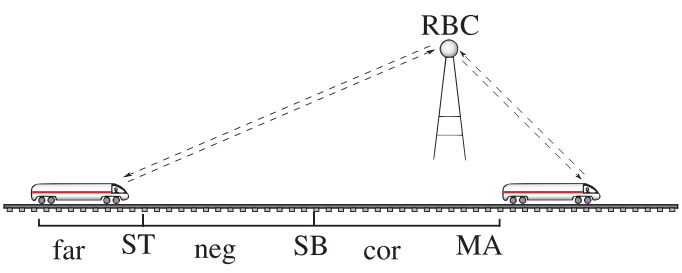
We introduce a formal model for generalised parametric ETCS in Section 7.2. Following our symbolic decomposition analysis, we systematically derive parametric correctness constraints for ETCS and verify several relevant correctness properties of the parametric ETCS model using these constraints in Section 7.3. In Section 7.4, we generalise the physical transmission model of trains to the presence of disturbances and verify ETCS with disturbances. Section 7.5 gives experimental results and a performance analysis in our verification tool KeYmaera. We summarise in Section 7.6.

7.2. Fully Parametric European Train Control System

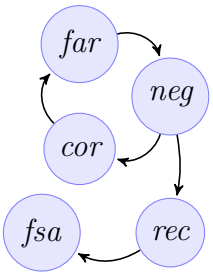
The European Train Control System (ETCS) [ERT02, FM06] is a standard to ensure safe and collision-free operation of trains and high throughput of high speed trains at speeds up to 320km/h. With 2.1 million passengers in Europe per day, correct functioning of ETCS will be highly safety-critical, because the upcoming installation of ETCS level 3 will replace all previous track-side safety measures in order to achieve its throughput objectives.

7.2.1. Overview of the ETCS Cooperation Protocol

ETCS level 3 follows the *moving block principle*, i.e., movement permissions are neither known beforehand nor fixed statically but they are determined based on the current track situation by a Radio Block Controller (RBC). Trains are only allowed to move within their current movement authority (MA), which can be updated by the RBC using wireless communication. Hence the train controller needs to regulate the movement of a train locally such that it always remains within its MA, because there can be open gates, unsafe tunnels or other trains beyond. The automatic train protection unit (*atp*) dynamically determines a safety envelope around a train τ, within which it considers driving safe, and adjusts the train acceleration $\tau.a$ accordingly. Figure 7.1a illustrates the dynamic assignment of MA with phases of controllers switching according to the protocol cycle in Figure 7.1b. When approaching towards the end of its MA the train switches from *far* mode (where speed can be regulated freely) to negotiation (*neg*), which, at the latest, happens at the point indicated by *ST* (for *start talking*). During negotiation the RBC grants or denies MA-extensions. Instead, the RBC can announce emergencies,



7.1a: Dynamic assignment of movement authorities



7.1b: Cooperation cycle

Figure 7.1.: ETCS train coordination protocol

which force train controllers to switch to the recovery mode (*rec*) by applying full emergency brakes. After the train has come to a full stop, the controller switches to a failsafe state (*fsa*) and awaits manual clearance by the train operator.

7.1 Lemma (Principle of separation by movement authorities). *If each train stays within its MA and, at any time, MAs issued by the RBC form a disjoint partitioning of the track, then trains can never collide.*

Proof. To simplify notation, we assume trains are points (the proof is a simple extension when each train has some maximal length l). Consider any point in time ζ. For some $n \in \mathbb{N}$, let $z_1, \ldots, z_n$ be the positions of all the trains 1 to n at ζ. Let M_i be the MA-range, i.e., the set of positions on the track for which train i has currently been issued MA. Suppose there was a collision at time ζ. Then $z_i = z_j$ at ζ for some $i, j \in \mathbb{N}$. However, by assumption, $z_i \in M_i$ and $z_j \in M_j$ at ζ, thus $M_i \cap M_j \neq \emptyset$, which contradicts the assumption of disjoint MA. □

Lemma 7.1 effectively reduces the verification of an unbounded number of traffic agents to a finite number. We exploit MA to decouple reasoning about global collision freedom to local cooperation of every traffic agent with its RBC. In particular, we can verify correct coordination for a train without having to consider gates, because these only communicate via RBC mediation and can be considered as special reasons for denial of MA-extensions. We only need to prove that the RBC handles all interaction between the trains by assigning or revoking MA correctly and that the trains respect their respective MA. However, to enable the RBC to guarantee disjoint partitioning of the track it has to rely on properties like appropriate safe rear end computation of the train. Additionally, safe operation of the train plant in conjunction with its environment depends on proper functioning of the gates. As these properties have a more static nature, they are much easier to show once the actual hybrid train dynamics and movements have been proven to be controlled correctly.

As trains are not allowed to drive backwards without clearance by track supervision personnel, the relevant part of the safety envelope can be reduced to the closest distance to the end of its current MA. The point SB, for *start braking*, is the latest point where the train needs to start correcting its acceleration (in mode *ctrl*) to make sure it always stays within the bounds of its MA. In Section 7.3, we derive a constraint on SB that guarantees safe driving.

We generalise the concept of MA to a vector $\mathbf{m} = (d, e, r)$ meaning that beyond point $\mathbf{m}.e$ the train must not have a velocity greater than $\mathbf{m}.d$. Additionally, the train should try not to outspeed the *recommended speed* $\mathbf{m}.r$ for the current track segment, but short periods of slightly higher speed are not considered safety-critical. Figure 7.2 shows an example of possible train behaviour in conjunction with the current value of $\mathbf{m}$ that changes over time due to RBC communication.

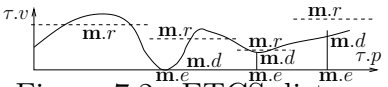


Figure 7.2.: ETCS distance profile

For a train $\tau = (p, v, a)$ at position $\tau.p$ with current velocity $\tau.v$ and acceleration $\tau.a$, we want to determine sufficient conditions ensuring safety and formally verify that $\tau.v$ is always safe with respect to its current MA, satisyfing:

$$\tau.p \geq \mathbf{m}.e \rightarrow \tau.v \leq \mathbf{m}.d \qquad (\mathcal{S})$$

Generalised movement authorities are a uniform composition of two safety-critical features. MA are crucial aspects for ensuring collision free operation in ETCS (Lemma 7.1) and they can take into account safety-critical velocity limits due to bridges, tunnels, or passing trains. For example high speed trains need to reduce their velocity while passing non-airtight or freight trains with a pressure-sensitive load within a tunnel. In our model this is done by a reduction of the speed component d of $\mathbf{m}$. Finally, two-dimensional MA permit higher throughput as we will see in Section 7.3.

7.2.2. Formal Model of Fully Parametric ETCS

For analysing the proper functioning of the ETCS system, we have developed a formal model of the ETCS control system as a hybrid program, shown in Figure 7.3. RBC and train are independent distributed components running in parallel. They interoperate by message passing over wireless communication. As the RBC is a purely digital track-side controller and has no dependent continuous dynamics, we can express parallelism equivalently by interleaving using non-deterministic choice and repetition: The decisions of the train controller only depend on the point in time where RBC messages arrive at the train, not its airtime. Thus, the nondeterministic interleaving faithfully models every possible arrival time without the need for an explicit deadtime channel model.

$$
\begin{array}{lll}
ETCS_{skel} & \equiv & (train \cup rbc)^* \\
train & \equiv & spd;\ atp;\ drive \\
spd & \equiv & (?\tau.v \leq \mathbf{m}.r;\ \tau.a := *;\ ? - b \leq \tau.a \leq A) \\
 & & \cup (?\tau.v \geq \mathbf{m}.r;\ \tau.a := *;\ ? - b \leq \tau.a \leq 0) \\
atp & \equiv & \text{if } (\mathbf{m}.e - \tau.p \leq SB \vee rbc.message = emergency) \text{ then } \tau.a := -b \text{ fi} \\
drive & \equiv & t := 0;\ (\tau.p' = \tau.v \wedge \tau.v' = \tau.a \wedge t' = 1 \wedge \tau.v \geq 0 \wedge t \leq \varepsilon) \\
rbc & \equiv & (rbc.message := emergency) \ \cup \ (\mathbf{m} := *;\ ?\mathbf{m}.r > 0)
\end{array}
$$

Figure 7.3.: Formal model of parametric ETCS cooperation protocol (skeleton)

Train Controller As a first approximation of the train PID controllers and its mechanical transmission [DMO+07], which we will refine in Section 7.4, we use a controller with a ranged choice for the effective acceleration $\tau.a$ between its lower bound $-b$ and upper bound A. For Section 7.2-7.3, we model the continuous train dynamics by the differential equation system

$$\tau.p' = \tau.v \wedge \tau.v' = \tau.a \wedge t' = 1 \wedge \tau.v \geq 0 \wedge t \leq \varepsilon \tag{7.1}$$

that formalises the ideal-world physical laws for movement, restricted to the invariant region $\tau.v \geq 0 \wedge t \leq \varepsilon$ in *drive*. Since trains do not drive backwards by braking, the system contains an invariant stating that the speed remains non-negative ($\tau.a \geq 0$). To further account for delayed effects of actuators like brakes or for delays caused by cycle times of periodic sensor polling and sampled data discrete time controllers, we permit the continuous movement of the train to continue for up to ε time units until control decisions finally take effect. This is expressed using the invariant region $t \leq \varepsilon$ on a clock t that will be used to keep track of the progress of time and advances with constant slope 1.

Speed supervision (*spd* in Figure 7.3) chooses the acceleration $\tau.a$ to keep the recommended speed $\mathbf{m}.r$ by a random assignment $\tau.a := *$, which is definable as $\tau.a' = 1 \cup \tau.a' = -1$. If the current speed $\tau.v$ exceeds $\mathbf{m}.r$, an acceleration is chosen from the interval $[-b, 0]$ otherwise the full range $[-b, A]$ is available, which, as a special case, includes controllers optimising speed and energy consumption as secondary objectives.

As a supervisory controller, the automatic train protection (*atp* in Figure 7.3) checks whether the point *SB* has been passed or a message from the RBC was received notifying of a track-side emergency situation. Both events cause immediate braking with full deceleration $-b$. Thus, *atp* decisions take precedence over *spd* speed advisory.

Radio Block Controller We model the RBC as a controller with two possible choices. It may choose to demand immediate initiation of recovery maneuvers by sending emergency messages ($rbc.message := emergency$). Alternatively, the RBC

may update the MA, either by a full exchange of $\mathbf{m}$ or a modification of any subset of its 3 components. These non-deterministic changes to $\mathbf{m}$ reflect different real-world effects like extending MA if the heading train has advanced significantly or, instead, notify of a new recommended speed for a track segment.

7.3. Parametric Verification of Train Control

Successively, we develop free parameter constraints by analysing increasingly more complex correctness properties of ETCS. We use the so discovered constraints to refine the train control model iteratively with constraints on design parameter choices and physical prerequisites on external parameters resulting from the safety requirements on the train dynamics.

7.3.1. Iterative Refinement Process

For discovering parametric constraints required for system correctness, we follow an *iterative refinement process*:

1. *Controllability discovery*: Start with uncontrolled system dynamics. Use structural decomposition in $\mathsf{d}\mathcal{L}$ until a first-order formula is obtained revealing the controllable state region, which specifies for which parameter combinations the system dynamics can be controlled safely by any control law.
2. *Control refinement*: Successively add partial control laws to the system while leaving its decision parameters (like SB or $\mathbf{m}$) free. Use structural decomposition to discover parametric constraints which preserve controllability under these control laws.
3. *Safety convergence*: Repeat step 2 until the resulting system is proven safe.
4. *Liveness check*: Prove that discovered parametric constraints do not over-constrain the system inconsistently by showing that it remains live.

In practice, variants of the controllable domain as discovered by step 1 constitute good candidates for inductive invariants, and the parameter constraints discovered by step 2 ensure that the actual control choices never leave the controllable domain. For step 4, liveness can actually be verified again by structural decomposition in $\mathsf{d}\mathcal{L}$ and no need for separate verification techniques or different models arises.

7.3.2. Controllability Discovery in Parametric ETCS

For unregulated train dynamics, our automatic structural decomposition technique discovers a controllability constraint on the external train parameters, i.e., a formula characterising the parameter combinations for which the train dynamics can

be controlled safely by any control law at all. We prove that the so discovered constraint characterises equivalently the set of states where the train dynamics still admits to respect MA by appropriate control choices (expressed by the left-hand side $\mathsf{d}\mathcal{L}$ formula):

7.2 Proposition (Controllability). *The constraint* $\tau.v^2 - \mathbf{m}.d^2 \leq 2b(\mathbf{m}.e - \tau.p)$ *is a* controllability constraint *for the train* τ *with respect to property* $(\mathcal{S})$ *on page 200, i.e., the former constraint retains the ability of the train dynamics to respect the safety property. Formally, with* $\mathbf{m}.d \geq 0 \wedge b > 0 \wedge \tau.p \leq \mathbf{m}.e \wedge \tau.v \geq 0$ *as regularity assumptions, the following equivalence is valid:*

$$\begin{aligned}&[\tau.p' = \tau.v \wedge \tau.v' = -b \wedge \tau.v \geq 0](\tau.p \geq \mathbf{m}.e \rightarrow \tau.v \leq \mathbf{m}.d)\\ \equiv\ &\tau.v^2 - \mathbf{m}.d^2 \leq 2b(\mathbf{m}.e - \tau.p)\end{aligned}$$

Observe how the above equivalence reduces a $\mathsf{d}\mathcal{L}$ statement about future controllable train dynamics to a single constraint on the current state. We will make use of this key reduction step from safe train dynamics to controllably safe states by analysing each part of the ETCS controller with respect to whether it preserves controllability.

7.3 Definition (Controllable state). A train τ is in a *controllable state*, if, by appropriate control actions, the train is always able to stay within its movement authority $\mathbf{m}$, which, by Proposition 7.2, corresponds to

$$\tau.v^2 - \mathbf{m}.d^2 \leq 2b(\mathbf{m}.e - \tau.p) \wedge \tau.v \geq 0 \wedge \mathbf{m}.d \geq 0 \wedge b > 0 \ . \qquad (\mathcal{C})$$

ETCS cannot be safe unless trains are in controllable states. Hence we pick $(\mathcal{C})$ as a minimal candidate for an inductive invariant. This invariant will be used to prove safety of the system by induction.

7.3.3. Iterative Control Refinement of ETCS Parameters

Starting from the constraints for controllable trains, we identify constraints for their various control decisions and refine the ETCS model correspondingly.

RBC Control Contraints For a safe functioning of ETCS it is important that trains always respect their current MA. Consequently, RBCs are not allowed to issue MAs that are physically impossible for the train like instantaneous full stops. Instead RBCs are only allowed to send new MAs that remain within the controllable range of the train dynamics. For technical reasons the RBC does not reliably know the train positions and velocities in its domain of responsibility to a sufficient precision, because the communication with the trains has to be performed wireless with possibly high communication delay and message loss. Thus, we give a failsafe constraint for MA updates which is reliably safe even for loss of position recording communication.

7.4 Proposition (RBC preserves controllability). *The constraint*

$$\mathbf{m}_0.d^2 - \mathbf{m}.d^2 \leq 2b(\mathbf{m}.e - \mathbf{m}_0.e) \wedge \mathbf{m}_0.d \geq 0 \wedge \mathbf{m}.d \geq 0 \qquad (\mathcal{M})$$

ensures that the RBC preserves controllability $(\mathcal{C})$ *when changing MA from* $\mathbf{m}_0$ *to* $\mathbf{m}$*, i.e., the following formula is valid:*

$$\forall\tau\Big(\mathcal{C} \to [\mathbf{m}_0 := \mathbf{m};\ rbc]\ \big(\mathcal{M} \to \mathcal{C}\big)\Big)\ . \qquad (7.2)$$

Further, RBC controllability is characterised by the following valid equivalence:

$$\mathbf{m}.d \geq 0 \wedge b > 0 \to [\mathbf{m}_0 := \mathbf{m};\ rbc]\ \Big(\mathcal{M} \leftrightarrow \forall\tau\ \big((\langle \mathbf{m} := \mathbf{m}_0\rangle\mathcal{C}) \to \mathcal{C}\big)\Big)\ . \qquad (7.3)$$

Property (7.2) expresses that, for all trains in a controllable state, every change of MA that complies with $(\mathcal{M})$ enforces that the train is still in a controllable state. Constraint $(\mathcal{M})$ says that an extension is safe if it is possible to reduce the speed by braking with deceleration b from the old target speed $\mathbf{m}_0.d$ to the new target speed $\mathbf{m}.d$ within the extension range $\mathbf{m}.e - \mathbf{m}_0.e$, regardless of the current speed of train τ. This constraint sheds light on how to layout feasible track profiles. It is characterised by the equivalence (7.3), expressing that RBC decisions ensure that all trains respecting controllability for the previous MA $\mathbf{m}_0$ remain controllable for the new MA, which holds iff $(\mathcal{M})$ is true for the RBC choice.

Train Control Constraints Now that we know constraints characterising when the cooperation of train and RBC is controllable, we need to find out under which circumstances the respective actual control choices by *spd* and *atp* retain controllability. In particular, the design parameter *SB* needs to be chosen appropriately to preserve $(\mathcal{C})$. First we show the existence of an appropriate choice for *SB*:

7.5 Proposition. *For all feasible RBC choices and all choices of speed control, there is a choice for SB that makes the train always stay within its MA, i.e., for controllable states, we can prove:*

$$[\mathbf{m}_0 := \mathbf{m};\ rbc]\big(\mathcal{M} \to [spd]\langle SB := *\rangle[atp;\ move](\tau.p \geq \mathbf{m}.e \to \tau.v \leq \mathbf{m}.d)\big)$$

To find a particular constraint on the choice of *SB*, we need to take the maximum reaction latency ε of the train controllers into account. With $\varepsilon > 0$, the point where the train needs to apply brakes to comply with $\mathbf{m}$ is not given by $(\mathcal{C})$ exactly, but needs additional safety margins to compensate for reaction delays.

7.6 Proposition (Reactivity constraint). *When the train is in a controllable state, the supervisory ETCS controller reacts appropriately in order to maintain controllability iff SB is chosen according to the following equivalence*

$$\Big(\forall\mathbf{m}.e\,\forall\tau.p\,\big(\mathbf{m}.e - \tau.p \geq SB \wedge \mathcal{C} \to [\tau.a := A;\ drive]\,\mathcal{C}\big)\Big)$$
$$\equiv SB \geq \frac{\tau.v^2 - \mathbf{m}.d^2}{2b} + \big(\frac{A}{b} + 1\big)\big(\frac{A}{2}\varepsilon^2 + \varepsilon\ \tau.v\big)\ . \qquad (\mathcal{B})$$

$$
\begin{array}{lll}
ETCS_{aug} & \equiv & (train \cup rbc_{\text{aug}})^* \\
train & \equiv & spd;\ atp_{\text{aug}};\ drive \\
atp_{\text{aug}} & \equiv & SB := \frac{\tau.v^2 - \mathbf{m}.d^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon\ \tau.v\right);\ atp \\
rbc_{\text{aug}} & \equiv & (rbc.message := emergency) \\
 & & \cup \big(\mathbf{m}_0 := \mathbf{m}; \mathbf{m} := *; \\
 & & ?\mathbf{m}.r \geq 0 \wedge \mathbf{m}.d \geq 0 \wedge \mathbf{m}_0.d^2 - \mathbf{m}.d^2 \leq 2b(\mathbf{m}.e - \mathbf{m}_0.e)\big)
\end{array}
$$

Figure 7.4.: Parametric ETCS cooperation protocol augmented with parameter constraints

Constraint ($\mathcal{B}$) on SB is derived using a projection of the train behaviour to the worst-case maximum acceleration A in a state where SB has not yet been passed. We choose this projection because the train controller needs to make sure that it can safely drive with maximum acceleration for ε time units even right before passing SB in order for an acceleration choice of A to be safe. After discovering constraint ($\mathcal{B}$), it can be explained in retrospect: It characterises the relative braking distance required to reduce speed from $\tau.v$ to target speed $\mathbf{m}.d$ with braking deceleration b—which corresponds to controllability—plus the additional distance travelled during one maximum reaction cycle of ε time units with acceleration A, including the additional distance needed to reduce the speed again down to $\tau.v$ after accelerating with A for ε time units. This extra distance results from speed changes and depends on the relation $\frac{A}{b}$ of acceleration and braking power b.

The previous propositions prove equivalences. Hence, counterexamples exist whenever the discovered parameter constraints are not met. Consequently, these constraints must be respected for correctness of *any* model of ETCS controllers, including implementation refinements. It is, thus, important to identify these safety constraints early in the overall design and verification process.

7.3.4. Safety Verification of Refined ETCS

By augmenting the system from Figure 7.3 with the parametric constraints presented in Propositions 7.2–7.6, we synthesise a safe system model completing the ETCS protocol skeleton. The refined model is presented in Figure 7.4.

7.7 Proposition (Safety). *Assuming the train starts in a controllable state, the following global and unbounded-horizon safety formula about the augmented ETCS system in Figure 7.4 is valid:*

$$\mathcal{C} \rightarrow [ETCS_{aug}](\tau.p \geq \mathbf{m}.e \rightarrow \tau.v \leq \mathbf{m}.d) \tag{7.4}$$

As an example to illustrate the proof structure for the verification of Proposition 7.7, consider the sketch in Figure 7.5. By convention, such proofs start with the conjecture at the bottom and proceed by decomposition to the leafs. We need

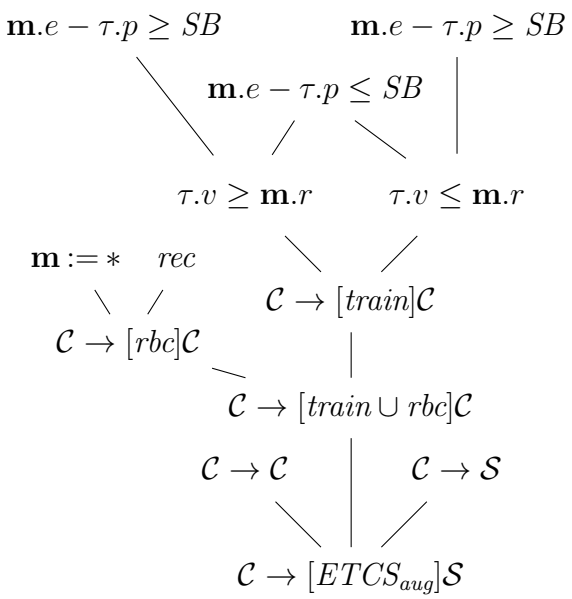


Figure 7.5.: Proof sketch for Proposition 7.7

to prove that the assumption that the train is in a controllable state expressed by $\mathcal{C}$ entails (7.4). As the system consists of a global loop, we prove that ($\mathcal{C}$) is an invariant of this loop and strong enough to imply ($\mathcal{S}$). Using KeYmaera it can be shown easily that the invariant is initially valid (left branch) and implies the postcondition (right branch). As usual, proving that the invariant is preserved by the loop body is the most challenging part of the proof (middle branch), which splits into two cases. For the left case, we have to show that the RBC preserves the invariant, which follows from Proposition 7.4. For the right case, we show that the train controller preserves the invariant. The proof splits due to the choice in the *spd* component depending on the relation of the current speed to the recommended speed. The next split on both of these branches depends on the value of SB. If the train has passed point SB (middle case), we can close this goal using Proposition 7.2, because the invariant describes a controllable state and the *spd* controller applies brakes. The outer branches, where the train has not yet passed SB, can be closed using Proposition 7.6.

7.3.5. Liveness Verification of Refined ETCS

In order to show that the discovered parameter constraints do not overconstrain the system inconsistently, we show liveness, i.e., that an ETCS train is able to reach all track positions with appropriate RBC permissions.

7.8 Proposition (Liveness). *The augmented train system is live, i.e., assuming*

the RBC can safely grant the required MAs because preceding trains are moving on, trains are able to reach any track position P by appropriate RBC choices:

$$A > 0 \wedge \varepsilon > 0 \;\rightarrow\; \forall P \, \langle ETCS_{aug} \rangle \, \tau.p \geq P$$

7.3.6. Full Correctness of ETCS

By collecting Propositions 7.2–7.8, we obtain the following main result of this chapter, which demonstrates the feasibility of $\mathsf{d}\mathcal{L}$-based parametric discovery and verification. It gives important insights in the fully parametric ETCS case study and yields conclusive and fully verified choices for the free parameters in ETCS. By virtue of the parametric formulation, this result applies to all concrete instantiations of the ETCS cooperation protocol from Section 7.2, including controllers that further optimise speed or model refinements in hardware implementations.

7.9 Theorem (Correctness of ETCS cooperation protocol). *The ETCS system augmented with constraints $(\mathcal{B})$ and $(\mathcal{M})$ is correct as given in Figure 7.4. Starting in any controllable state respecting $(\mathcal{C})$, trains remain in the controllable region at any time. They safely respect movement authorities issued by the RBC such that ETCS is collision-free. Further, trains can always react safely to all RBC decisions respecting $(\mathcal{M})$. ETCS is live: when tracks become free, trains are able to reach any track position by appropriate RBC actions. Furthermore, the augmented constraints $(\mathcal{C})$ and $(\mathcal{B})$ are necessary and sharp: Every configuration violating $(\mathcal{C})$ or $(\mathcal{B})$, respectively, gives rise to a concrete counterexample violating safety property $(\mathcal{S})$. Finally, every RBC choice violating $(\mathcal{M})$ gives rise to a counterexample in the presence of lossy wireless communication channels.*

7.4. Disturbance and the European Train Control System

In Section 7.2–7.3, we have assumed there was direct control of acceleration. In reality, acceleration results from physical transmission of corresponding forces that depend on the electrical current in the engine and are regulated by PID controllers. As a conservative overapproximation of these effects, we generalise the ETCS model to a model with differential inequalities, where we also take into account disturbances in the physical transmission of forces:

$$\tau.p' = \tau.v \wedge \tau.a - l \leq \tau.v' \leq \tau.a + u \wedge t' = 1 \wedge \tau.v \geq 0 \wedge t \leq \varepsilon \tag{7.5}$$

with a disturbance within the interval $[-l, u]$. That is, the acceleration $\tau.a$ chosen by the train controller can take effect with an error bounded by $-l$ and u, because the derivative $\tau.v'$ of the velocity will not need to be $\tau.a$ exactly but can vary arbitrarily

between $\tau.a-l$ and $\tau.a+u$ over time. We generalise the differential equation (7.1) in component *train* from Figure 7.3 by replacing it with the differential inequality (7.5) and denote the result by $train_d$.

Notice that, unlike (7.1), we cannot simply solve differential inequality (7.5), because its actual solution depends on the precise value of the disturbance, which is a quantity that changes over time. Thus, solutions will only be relative to this disturbance function and a reachability analysis would have to consider all choices of this function, which would require higher-order logic. Instead, we use differential invariants [Pla08a] as a first-order characterisation.

7.4.1. Controllability in ETCS with Disturbances

The controllability characterisation carries over to train control with disturbance when taking into account the maximum disturbance u on the maximum braking power b:

7.10 Proposition (Controllability despite disturbance). *The constraint*

$$\tau.v^2 - \mathbf{m}.d^2 \leq 2(b-u)(\mathbf{m}.e - \tau.p) \wedge \mathbf{m}.d \geq 0 \wedge b > u \geq 0 \wedge l \geq 0 \qquad (\mathcal{C}_d)$$

is a controllability constraint *with respect to property* $(\mathcal{S})$ *for the train* τ *with disturbance, i.e., it retains the ability of the train dynamics to respect the safety property. Formally, with* $\mathbf{m}.d \geq 0 \wedge b > u \geq 0 \wedge l \geq 0 \wedge \tau.p \leq \mathbf{m}.e \wedge \tau.v \geq 0$ *as regularity assumptions, the following equivalence holds:*

$$\begin{aligned} &[\tau.p' = \tau.v \wedge \tau.a - l \leq \tau.v' \leq \tau.a + u \wedge t' = 1 \wedge \tau.v \geq 0 \wedge t \leq \varepsilon]\, \mathcal{S} \\ \equiv\, &\tau.v^2 - \mathbf{m}.d^2 \leq 2(b-u)(\mathbf{m}.e - \tau.p) \end{aligned}$$

7.4.2. Iterative Control Refinement of Parameters with Disturbances

When taking into account the worst-case effect that the disturbance has on control, the reactivity constraint $(\mathcal{B})$ carries over to the presence of disturbance in train control:

7.11 Proposition (Reactivity constraint). *When the train with disturbance is in a controllable state, the supervisory ETCS controller reacts appropriately in order to maintain controllability iff SB is chosen according to the following equivalence:*

$$\begin{aligned} &\Big(\forall \mathbf{m}.e\, \forall \tau.p\, \big((\mathbf{m}.e - \tau.p \geq SB \wedge \tau.v^2 - \mathbf{m}.d^2 \leq 2(b-u)(\mathbf{m}.e - \tau.p)) \rightarrow \\ &\qquad [\tau.a := A;\ drive_d](\tau.v^2 - \mathbf{m}.d^2 \leq 2(b-u)(\mathbf{m}.e - \tau.p))\big)\Big) \\ \equiv\, &SB \geq \frac{\tau.v^2 - \mathbf{m}.d^2}{2(b-u)} + \left(\frac{A+u}{b-u} + 1\right)\left(\frac{A+u}{2}\varepsilon^2 + \varepsilon\tau.v\right) \qquad (\mathcal{B}_d) \end{aligned}$$

7.4.3. Safety Verification of ETCS with Disturbances

When we augment the ETCS model by the constraints $(\mathcal{B}_d)$ and $(\mathcal{M})$, ETCS is safe even in the presence of disturbance when starting in a state respecting $(\mathcal{C}_d)$.

7.12 Proposition (Safety despite disturbance). *Assuming the train starts in a controllable state satisfying $(\mathcal{C}_d)$, the following global and unbounded-horizon safety formula about the ETCS system with disturbance from Figure 7.6 is valid:*

$$\mathcal{C}_d \to [ETCS_d](\tau.p \geq \mathbf{m}.e \to \tau.v \leq \mathbf{m}.d)$$

$$\begin{array}{lll}
ETCS_d & \equiv & (train_d \cup rbc)^* \\
train_d & \equiv & spd;\ atp_d;\ drive_d \\
spd & \equiv & (?\tau.v \leq \mathbf{m}.r;\ \tau.a := *;\ ?-b \leq \tau.a \leq A) \\
 & & \cup(?\tau.v \geq \mathbf{m}.r;\ \tau.a := *;\ ?0 > \tau.a \geq -b) \\
atp_d & \equiv & SB := \frac{\tau.v^2 - \mathbf{m}.d^2}{2(b-u)} + \left(\frac{A+u}{b-u} + 1\right)\left(\frac{A+u}{2}\varepsilon^2 + \varepsilon\tau.v\right); \\
 & & \text{if } (\mathbf{m}.e - \tau.p \leq SB \vee rbc.message = emergency) \text{ then } \tau.a := -b \text{ fi} \\
drive_d & \equiv & t := 0;\ (\tau.p' = \tau.v \wedge \tau.a - l \leq \tau.v' \leq \tau.a + u \wedge t' = 1 \wedge \tau.v \geq 0 \wedge t \leq \varepsilon) \\
rbc & \equiv & (rbc.message := emergency) \\
 & & \cup \big(\mathbf{m}_0 := \mathbf{m}; \mathbf{m} := *; \\
 & & \quad ?\mathbf{m}.r \geq 0 \wedge \mathbf{m}.d \geq 0 \wedge \mathbf{m}_0.d^2 - \mathbf{m}.d^2 \leq 2(b-u)(\mathbf{m}.e - \mathbf{m}_0.e)\big)
\end{array}$$

Figure 7.6.: Parametric ETCS cooperation protocol with disturbances

The safety proof generalises to ETCS with disturbance, when using differential induction [Pla08a] with a time-dependent version of $(\mathcal{B}_d)$ as differential invariant for the acceleration case:

$$\mathbf{m}.e - \tau.p \geq \frac{\tau.v^2 - \mathbf{m}.d^2}{2(b-u)} + \left(\frac{A+u}{b-u} + 1\right)\left(\frac{A+u}{2}(\varepsilon - t)^2 + (\varepsilon - t)\tau.v\right) \tag{7.6}$$

The full ETCS system with disturbance, where all abbreviations are resolved, is depicted in Figure 7.7.

7.5. Experimental Results

Experimental results for verifying ETCS in our $\mathsf{d}\mathcal{L}$-based verification tool KeYmaera are presented in Table 7.1. Experimental results are from a 2.6GHz AMD Opteron with 4GB memory. All correctness properties and parameter constraints of ETCS can be verified with 94% to 100% degree automation. In the universal fragment of $\mathsf{d}\mathcal{L}$, user interactions are only needed for supplying invariants, which, in turn, can be discovered using our iterative refinement process and our invariant computation algorithm from Chapter 6. For liveness properties or substantial quantifier

$$\psi \equiv \tau.v^2 - \mathbf{m}.d^2 \leq 2(b-u)(\mathbf{m}.e - \tau.p) \wedge \mathbf{m}.d \geq 0 \wedge b > u \geq 0 \wedge l \geq 0 \rightarrow [ETCS_d](\tau.p \geq \mathbf{m}.e \rightarrow \tau.v \leq \mathbf{m}.d)$$

$$ETCS_d \equiv \Bigg(\Bigg\{ \begin{array}{ll} & \\ & \\ & \\ \textit{spd}: & \big((?\tau.v \leq \mathbf{m}.r;\ \tau.a := *;\ ?-b \leq \tau.a \leq A) \\ & \quad \cup (?\tau.v \geq \mathbf{m}.r;\ \tau.a := *;\ ?0 > \tau.a \geq -b)\big); \\ \textit{atp}_d: & SB := \frac{\tau.v^2 - \mathbf{m}.d^2}{2(b-u)} + \left(\frac{A+u}{b-u} + 1\right)\left(\frac{A+u}{2}\varepsilon^2 + \varepsilon\tau.v\right); \\ & \text{if } (\mathbf{m}.e - \tau.p \leq SB \vee \textit{rbc.message} = \textit{emergency}) \text{ then } \tau.a := -b \text{ fi}; \\ \textit{drive}_d: & t := 0;\ (\tau.p' = \tau.v \wedge \tau.a - l \leq \tau.v' \leq \tau.a + u \wedge t' = 1 \wedge \tau.v \geq 0 \wedge t \leq \varepsilon) \end{array} \Bigg) \\ \bigcup \\ \textit{rbc}: \quad \Bigg(\begin{array}{l} (\textit{rbc.message} := \textit{emergency}) \\ \quad \cup \big(\mathbf{m}_0.d := \mathbf{m}.d;\ \mathbf{m}_0.e := \mathbf{m}.e;\ \mathbf{m}_0.r := \mathbf{m}.r; \\ \qquad \mathbf{m}.d := *;\ \mathbf{m}.e := *;\ \mathbf{m}.r := *; \\ \qquad ?\mathbf{m}.r \geq 0 \wedge \mathbf{m}.d \geq 0 \wedge \mathbf{m}_0.d^2 - \mathbf{m}.d^2 \leq 2(b-u)(\mathbf{m}.e - \mathbf{m}_0.e)\big) \end{array} \Bigg) \Bigg)^*$$

Figure 7.7.: Parametric ETCS cooperation protocol with disturbances (full instantiation)

alternations beyond the capabilities of currently available decision procedures for real arithmetic, KeYmaera needs more user guidance. Yet, those properties can still be verified with KeYmaera. Further, we see that the symbolic state dimension has more impact on the computational complexity than the number of proof steps in $\mathsf{d}\mathcal{L}$ decompositions. The interactions in Proposition 7.11 and Proposition 7.12 are for the time-dependent version (7.6) of the constraint on SB.

The experimental results in Table 7.1 for Proposition 7.5 can be improved significantly. Most of the user interactions are only required to overcome the current limitations of the preliminary implementation of iterative inflation order proof strategies in KeYmaera. Finally contrast our overall experimental results with earlier implementations that had required as much as 56 user interactions even for property Proposition 7.7 to be provable [Que07], which we have managed to prove completely automatically now using our new automated theorem proving techniques for differential dynamic logics and the logic-based verification algorithms from Part II.

Table 7.1.: Experimental results for the European Train Control System

Case study	Int	Time(s)	Memory(MB)	Steps	Dim
Proposition 7.2	0	0.6	6.9	14	5
Proposition 7.4, property (7.2)	0	0.5	6.4	42	12
Proposition 7.4, property (7.3)	0	0.9	6.5	82	12
Proposition 7.5	13	279.1	98.3	265	14
Proposition 7.6	0	103.9	61.7	47	14
Proposition 7.7	0	2052.4	204.3	153	14
Proposition 7.8 (kernel)	4	35.2	92.2	62	10
Proposition 7.8 (simplified)	6	9.6	23.5	134	13
Proposition 7.10	0	2.8	8.3	26	7
Proposition 7.11	1	23.7	47.6	76	15
Proposition 7.12	1	5805.2	34	218	1 6

7.6. Summary

As a case study for parametric verification, we have verified controllability, reactivity, safety, and liveness of the fully parametric cooperation protocol of the European Train Control System, thereby demonstrating the feasibility of logic-based verification of parametric hybrid systems. We have identified parametric constraints that are both sufficient and necessary for a safe collision-free operation of ETCS. We have characterised these constraints on the free parameters of ETCS equivalently in terms of corresponding reachability properties of the underlying train dynamics. We have further shown that the ETCS system remains correct even when the

train dynamics is subject to disturbances caused, e.g., by the physical transmission, friction, or wind.

We have seen that the logic $\mathsf{d}\mathcal{L}$ can indeed express all relevant properties of train control conveniently. Our experimental results with KeYmaera show a scalable approach by combining the strengths of completely automatic verification procedures with the intuition behind user guidance to tackle even highly parametric hybrid systems and properties with substantial quantifier alternation like reactivity or liveness.

For future work, we want to generalise ETCS using probabilistic information about sensor and communication accuracy and availability.

Chapter 8.

Collision Avoidance Maneuvers in Air Traffic Control

Contents

Synopsis

Aircraft collision avoidance maneuvers are important and challenging application scenarios for verification. Flight dynamics, especially during curved

flight, exhibits nontrivial continuous dynamics. Moreover, the range of control choices during air traffic maneuvers result in hybrid systems with challenging interactions of discrete and continuous dynamics. As a case study for demonstrating the scalability in verifying hybrid systems with challenging dynamics, we analyse roundabout maneuvers in air traffic control, where appropriate curved flight, good timing, and compatible maneuver choices are crucial for guaranteeing safe spatial separation of aircraft throughout their flight. We show that our DAL-based verification algorithm is able to verify collision avoidance of the tangential roundabout maneuver automatically, even for 5 aircraft. Moreover, we introduce a fully flyable variant of the roundabout collision avoidance maneuver and verify safety properties by compositional verification in our calculus.

8.1. Introduction

As a case study, we show how safety properties of collision avoidance maneuvers in air traffic management can be verified with the DAL calculus from Chapter 3 using our verification algorithm from Chapter 6. Aircraft maneuvers are challenging for verification [TPS98, LLL00, MF01, DPR05, DMC05, PC07, GMAR07, HKT07], because of the complicated spatial/geometrical movement of aircraft. Unlike in train or car control, braking is no option for failsafe recovery for aircraft. Consequently, we cannot rely on timely failsafe breaking to maintain safe separation when, e.g., movement authority extension negotiations do not succeed in time. Instead, the full aircraft maneuver has to be coordinated in such a way that the aircraft always respect minimal and maximal speed constraints and still remain safely separated.

Technically, complexities in analysis of aircraft maneuvers manifest most prominently in difficulties with analysing hybrid systems for flight equation (3.1) on page 90 in Section 3.4 or the equivalently reparameterised differential equation system ($\mathcal{F}(\omega)$) on page 91 that we obtained from (3.1) by differential axiomatisation.

On straight lines, i.e., where the angular velocity is $\omega = 0$, the angular orientation ϑ and the value of $\sin\vartheta$ and $\cos\vartheta$ remain constant during continuous evolutions so that the solutions of (3.1) are simple (possibly piecewise) *linear functions*. For hybrid systems with linear evolution functions, there are well-known analysis techniques [Hen96, Fre05], for instance based on classical linear programming [Chv83]. Pure straight line maneuvers [TPS98, MF01, DMC05, GMAR07, HKT07] are aircraft maneuvers with piecewise linear evolutions, see, e.g., Figure 8.1. They assume instant turns for heading changes of the aircraft between multiple straight line segments. Instant turns, however, are impossible in midflight, because they are *not flyable*: Aircraft cannot suddenly change their flight direction from 0 to 45 degrees discontinuously but need to follow a smooth curve instead, in which they slowly steer towards the desired direction by adjusting the angular velocities appropriately.

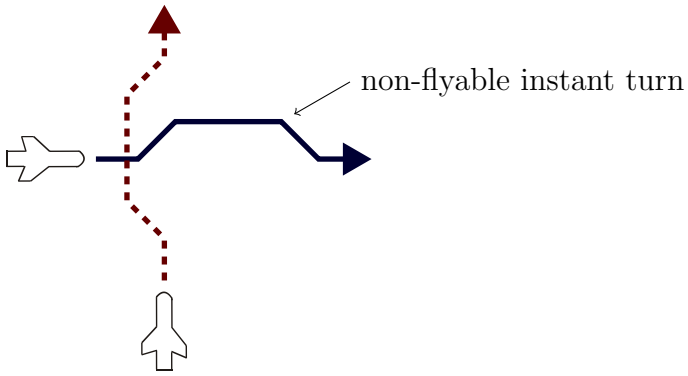


Figure 8.1.: Non-flyable straight line maneuver with instant turns

During curved flight, the angular velocity ω is non-zero, which causes the trigonometric expressions in equation (3.1) from page 90 to have a permanent non-constant and, more importantly, nonlinear effect on the dynamics of the system. For $\omega \neq 0$, the equivalent differential equation system ($\mathcal{F}(\omega)$) has transcendental solutions, so that reachability problems along these solutions fall into undecidable classes of arithmetics, as illustrated in Section 3.4.2. Consequently, maneuvers with curves, like the roundabout maneuvers in Figure 6.7 on page 190, are more realistic but also much more challenging for verification than straight line maneuvers like that in Figure 8.1, because the flight equations (3.1) and ($\mathcal{F}(\omega)$) become highly non-trivial. For instance, differential equation solving in Mathematica produces the solution depicted in Figure 8.2 for differential equation system that results from system (3.1) on page 90 by considering relative positions of two aircraft, see [TPS98] for details:

$$x_1' = -v_1 + v_2 \cos\vartheta + \omega x_2 \qquad x_2' = v_2 \sin\vartheta - \omega x_1 \qquad \vartheta' = \varrho - \omega$$

The "solution" (if it is one at all) in Figure 8.2 is not suitable for verification purposes. It involves several trigonometric functions and has an undefined singularity at $\omega = 0$. As we have illustrated in Section 3.4.2, reachability verification (e.g., with rule D12 of Figure 2.5) is not possible for trigonometric solutions like in Figure 8.2, because the resulting formulas of the form $\forall t{\geq}0\, \phi(x_1(t), x_2(t), \vartheta(t))$ involve quantified arithmetic over trigonometric functions, which is undecidable.

To verify roundabout maneuvers with curves like in Figure 3.1b on page 90 nevertheless, our calculus and verification algorithm work with differential invariants (Definition 6.3 and proof rule G5 of Figure 3.3) instead of solutions of differential equations.

Despite all complications caused by their challenging dynamics, formal verification is of tremendous importance for aircraft maneuvers. Basically, collision avoidance systems like TCAS [LLL00] are intended as last resort means for resolving air traffic conflict situations that have not been detected and prevented by pilots or flight directors of the Air Route Traffic Control Centres. Consequently, decisions

$$
\begin{aligned}
x_1(t) = \frac{1}{\varrho\omega}\big(&x_1\varrho\omega\cos(t\omega) - \omega\sin(\vartheta)v_2\cos(t\omega) + \omega\cos(t\varrho)\sin(\vartheta)v_2\cos(t\omega)\\
&+ \omega\cos(\vartheta)\sin(t\varrho)v_2\cos(t\omega) + x_2\varrho\omega\sin(t\omega) - \varrho\sin(t\omega)v_1\\
&- \omega\cos(\vartheta)\cos(t\varrho)\sin(t\omega)v_2 + \omega\sin(\vartheta)\sin(t\varrho)\sin(t\omega)v_2\\
&- \omega\sqrt{1-\sin(\vartheta)^2}\sin(t\omega)v_2\big)\\
x_2(t) = \frac{1}{\varrho\omega}\big(&\varrho v_1\cos(t\omega)^2 + x_2\varrho\omega\cos(t\omega) - \varrho v_1\cos(t\omega) - \omega\cos(\vartheta)\cos(t\varrho)v_2\cos(t\omega)\\
&+ \omega\sin(\vartheta)\sin(t\varrho)v_2\cos(t\omega) - \omega\sqrt{1-\sin(\vartheta)^2}v_2\cos(t\omega) - x_1\varrho\omega\sin(t\omega)\\
&+ \varrho\sin(t\omega)^2v_1 + \omega\sin(\vartheta)\sin(t\omega)v_2\\
&- \omega\cos(t\varrho)\sin(\vartheta)\sin(t\omega)v_2 - \omega\cos(\vartheta)\sin(t\varrho)\sin(t\omega)v_2\big)\\
\vartheta(t) = \vartheta + t(\varrho - \omega)&
\end{aligned}
$$

Figure 8.2.: Formal "solution" of flight equations produced by Mathematica

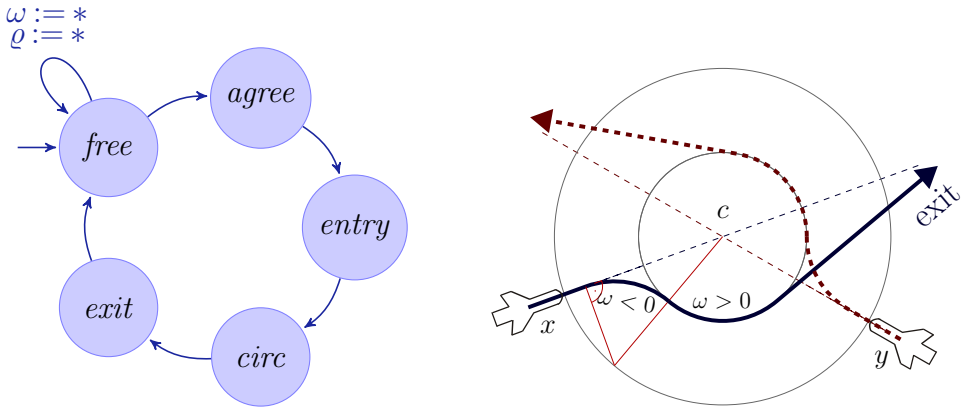
of automatic air traffic control systems should be fast and particularly reliable. In addition, the counterexample in Figure 3.1c, which has been found by our model checker AMC [PC07] for the classical roundabout maneuver with fixed parameters [TPS98, MF01, DPR05], shows that sound formal verification is of utmost importance to ensure correct functioning of air traffic control maneuvers.

Structure of this Chapter

We verify crucial safety and liveness properties of curved flight in roundabout maneuvers in Section 8.2. We analyse synchronisation properties of roundabout maneuvers for multiple aircraft in Section 8.3. In Section 8.5, we combine the previous results to a safety theorem for fully flyable tangential roundabout maneuvers following a compositional verification approach in our calculi according to Section 8.4. We present experimental results in Section 8.6 and conclude in Section 8.7.

8.2. Curved Flight in Roundabout Maneuvers

In this section, we introduce and verify the fully flyable roundabout maneuver that is depicted in Figure 6.7. It refines the tangential roundabout maneuver from Figure 3.1d, which still has discontinuities at the entry and exit points of roundabouts, to a fully flyable curved maneuver. The resulting maneuver does not contain any instant turns, but all of its curves are sufficiently smooth. The *flyable tangential roundabout maneuver* (*FTRM*) consists of the phases in the protocol cycle in Figure 8.3a which correspond to the marked flight phases in Figure 6.7.



8.3a: Collision avoidance protocol

8.3b: Maneuver construction

Figure 8.3.: Protocol cycle and construction of flyable roundabout maneuver

During free flight, the aircraft move without restrictions by repeatedly choosing arbitrary new angular velocities ω and ϱ respectively (as indicated by the self loop of phase *free* in Figure 8.3a). As in Chapter 3, continuous nondeterministic variation of ω and ϱ could be permitted safely as well by adding $\exists\omega\,\exists\varrho$, the proofs carry over immediately.

When the aircraft come too close to one another, they agree on a compatible roundabout maneuver by negotiating a common roundabout centre c (in the coordination phase *agree*). Next, the aircraft approach the actual roundabout circle by a right curve with $\omega < 0$ (*entry* mode) according to Figure 8.3b, thereby leading to a tangential configuration satisfying property (3.6) of page 122 where roundabout dynamics is safe. During the *circ* mode, the aircraft follow the circular roundabout maneuver around the agreed centre c with a left curve of common angular velocity ω (or right curve for $\omega < 0$). Finally, the aircraft leave the circular roundabout in cruise mode ($\omega = 0$) into their original direction (*exit*) and enter free flight again when they have reached sufficient distance. The collision avoidance maneuver is symmetric when exchanging left and right curves, which corresponds to using a value $\omega < 0$ in place of $\omega > 0$.

8.2.1. Compositional Verification Plan

For verifying collision avoidance and other safety properties of the flyable tangential roundabout maneuver, we pursue the following overall verification plan that modularises the proof and allows us to identify the respective safety constraints imposed by the various maneuver phases successively. We verify subsequently:

AC1 *Tangential roundabout maneuver cycle*: We prove that the protected zones of aircraft are safely separated at all times during the whole maneuver (including

repetitive collision avoidance maneuver initiation) when using a simplified *entry* operation.

AC2 *Bounded control choices for aircraft velocities*: We prove that there are realisable choices for the degrees of freedom in *entry* that require only bounded control choices for velocities. Prove that linear speeds remain bounded for the overall maneuver.

AC3 *Flyable entry procedures*: We prove that the simplified *entry* procedure can be replaced by a flyable curve meeting the requirements of the *entry* procedure identified in Section 3.11.

AC4 *Bounded entry duration*: We show that the flyable entry procedure succeeds in bounded time.

AC5 *Safe entry separation*: We prove that the protected zones of aircraft are respected during flyable entry procedure.

AC6 *Successful negotiation*: We prove that the negotiation phase (*agree*) of the aircraft succeeds with all mutual requirements of the respective aircraft for the entry phase being satisfied.

AC7 *Safe exit separation*: We show that, for its bounded duration, the exit procedure does not conflict with other flight curves and that the initial far separation is again reached as needed by the flyable entry procedure when re-initiating collision avoidance maneuvers repeatedly.

We present details on these verification tasks in the sequel and summarise the respective verification results into a joint safety property of the flyable tangential roundabout maneuver in Section 8.5. For AC4, we still rely on informal arguments.

Finally note that, to simplify notation we use square roots and norms within formulas in this chapter, because both are definable. For instance, $\|x - y\| \geq p$ is definable by $(x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2$ for $p \geq 0$. Likewise, $\|x - y\| = \sqrt{3}r$ is definable by $\|x - y\|^2 = 3r^2$ for $r \geq 0$.

8.2.2. Tangential Roundabout Maneuver Cycle

First, we analyse the roundabout maneuver with a simplified entry procedure, which also allows to simplify the exit procedure.

Modular Correctness of Tangential Roundabout Cycles For AC1, we have proven in Section 3.11 that—for arbitrary choices of the *entry* maneuver that satisfy the prerequisites of Theorem 3.32—the tangential roundabout maneuver safely avoids collisions, i.e., the aircraft always maintain a safe distance $\geq p$ during the

curved flight in the roundabout circle. In addition, these results show that arbitrary repetitions of the protocol cycle are safe at all times for a simplified choice of the entry maneuver. The model and specification for this tangential roundabout, as constructed in Section 3.4 and Section 3.11, are summarised in Figure 8.4.

$$\begin{aligned}
\psi &\equiv \phi \rightarrow [trm^*]\phi \\
\phi &\equiv \|x-y\|^2 \geq p^2 \equiv (x_1-y_1)^2+(x_2-y_2)^2 \geq p^2 \\
trm &\equiv free;\ \ agree;\ \ \mathcal{F}(\omega) \wedge \mathcal{G}(\omega) \\
free &\equiv (\omega := *;\ \ \varrho := *;\ \ \mathcal{F}(\omega) \wedge \mathcal{G}(\varrho) \wedge \phi)^* \\
agree &\equiv \omega := *;\ c := *; \\
&\quad\ d_1 := -\omega(x_2-c_2);\ d_2 := \omega(x_1-c_1); \\
&\quad\ e_1 := -\omega(y_1-c_1);\ e_2 := \omega(y_2-c_2)
\end{aligned}$$

Figure 8.4.: Flight control with tangential roundabout collision avoidance maneuvers

In summary, property ψ of Figure 8.4 expresses that the aircraft maintain a safe distance of at least the protected zone p during the flight, especially during evasive roundabout maneuvers. Our verification results for this property are already indicated in Table 6.1.

Multiple Aircraft We prove a corresponding property for up to 5 aircraft, which jointly participate in the roundabout maneuver. There, the safety property is mutual collision avoidance, i.e., each of the aircraft has a safe distance $\geq p$ to all the other aircraft, which yields a quadratic number of constraints to show. This quadratic increase in the property that actually needs to be proven for a safe roundabout of n aircraft and the increased dimension of the underlying continuous state space cause the increased verification times for more aircraft in Table 6.1. For instance, Figure 8.5 illustrates the (flyable) roundabout maneuver with multiple aircraft and Figure 8.6 contains the system and separation property specification for the 4-aircraft tangential roundabout maneuver (still with simplified entry procedure). There, property ψ expresses that the 4 aircraft at positions $x, y, z, u \in \mathbb{R}^2$, respectively, keep mutual distance $\geq p$.

8.2.3. Bounded Control Choices for Aircraft Velocities

For AC2, we show that bounded speed choices are sufficient for the choices of the entry procedure and that this bounded speed is maintained safely throughout the maneuver.

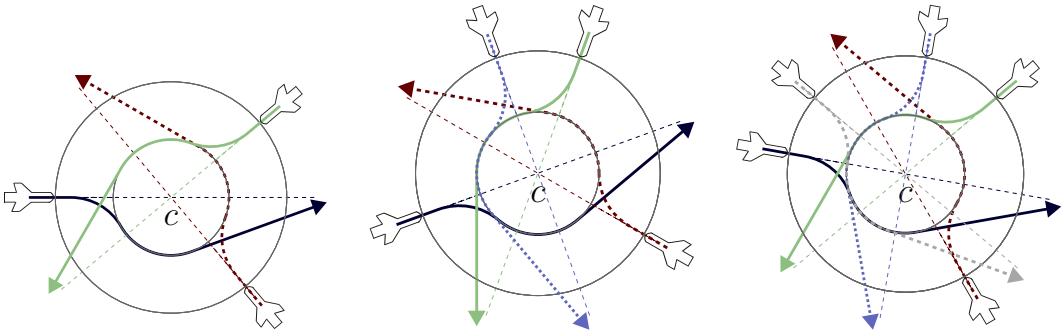


Figure 8.5.: Flyable aircraft roundabout (multiple aircraft)

$$
\begin{aligned}
\psi &\equiv \phi \rightarrow [trm^*]\phi \\
\phi &\equiv (x_1-y_1)^2+(x_2-y_2)^2 \geq p^2 \wedge (y_1-z_1)^2+(y_2-z_2)^2 \geq p^2 \\
&\quad \wedge (x_1-z_1)^2+(x_2-z_2)^2 \geq p^2 \wedge (x_1-u_1)^2+(x_2-u_2)^2 \geq p^2 \\
&\quad \wedge (y_1-u_1)^2+(y_2-u_2)^2 \geq p^2 \wedge (z_1-u_1)^2+(z_2-u_2)^2 \geq p^2 \\
trm &\equiv free;\quad agree; \\
&\quad x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = -\omega_x d_2 \wedge d_2' = \omega_x d_1 \\
&\quad \wedge y_1' = e_1 \wedge y_2' = e_2 \wedge e_1' = -\omega_y e_2 \wedge e_2' = \omega_y e_1 \\
&\quad \wedge z_1' = f_1 \wedge z_2' = f_2 \wedge f_1' = -\omega_z f_2 \wedge f_2' = \omega_z f_1 \\
&\quad \wedge u_1' = g_1 \wedge u_2' = g_2 \wedge g_1' = -\omega_u g_2 \wedge g_2' = \omega_u g_1 \\
free &\equiv (\omega_x := *;\quad \omega_y := *;\quad \omega_z := *;\quad \omega_u := *; \\
&\quad x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = -\omega_x d_2 \wedge d_2' = \omega_x d_1 \\
&\quad \wedge y_1' = e_1 \wedge y_2' = e_2 \wedge e_1' = -\omega_y e_2 \wedge e_2' = \omega_y e_1 \\
&\quad \wedge z_1' = f_1 \wedge z_2' = f_2 \wedge f_1' = -\omega_z f_2 \wedge f_2' = \omega_z f_1 \\
&\quad \wedge u_1' = g_1 \wedge u_2' = g_2 \wedge g_1' = -\omega_u g_2 \wedge g_2' = \omega_u g_1 \wedge \phi)^* \\
agree &\equiv \omega := *;\ c := *; \\
&\quad d_1 := -\omega(x_2-c_2);\ d_2 := \omega(x_1-c_1); \\
&\quad e_1 := -\omega(y_1-c_1);\ e_2 := \omega(y_2-c_2); \\
&\quad f_1 := -\omega(z_1-c_1);\ f_2 := \omega(z_2-c_2); \\
&\quad g_1 := -\omega(u_1-c_1);\ g_2 := \omega(u_2-c_2)
\end{aligned}
$$

Figure 8.6.: Tangential roundabout collision avoidance maneuver (4 aircraft)

Bounded Entry Choices The tangential roundabout maneuver in Figure 8.4 maintains collision avoidance for all its choices of centre c and angular velocity ω in *agree*. Next, we show that *there always is* a choice respecting external requirements on linear speed (aircraft flight is neither save at too high speeds nor when they are travelling too slowly, as they would stall), which corresponds to AC2 of Section 8.2.1. The fact that external speed requirements can be met is a consequence of the proof of property (3.8) on page 125 from Section 3.11. Consequently, the constraints on the entry procedure are feasible with parameter choices respecting *any* bounds for velocities that are imposed by the aircraft.

Bounded Maneuver Speed As a simple consequence of the proof in Example 3.3 on page 105, it is easy to see that external requirements on the linear speed of aircraft are maintained throughout the whole roundabout maneuver. Example 3.3 shows that the linear speed is maintained for arbitrary repeated choices of the angular velocity ω, which, as a special cases, includes the respective control choices during the roundabout maneuver.

8.2.4. Flyable Entry Procedures

In order to generalise the verification results about the tangential roundabout maneuver with simplified entry procedures (Figure 3.1) to the fully flyable tangential roundabout maneuver (Figure 6.7 and Figure 8.5), we analyse a flyable entry procedure, which replaces our simple choice of *entry* in Figure 8.4 and Figure 8.6 by flyable curves.

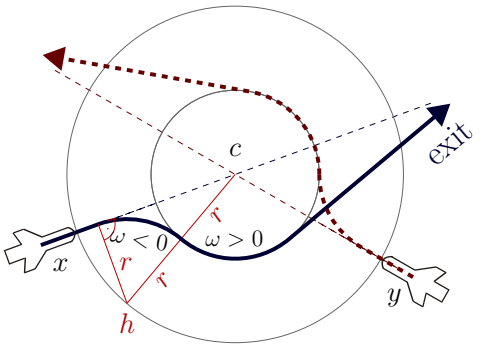


Figure 8.7.: Technical construction of flyable roundabout maneuver and entry

Flyable Entry Properties A flyable entry maneuver that follows the smooth entry curve from Figure 6.7 is given in Figure 8.8a. Its construction uses the anchor point h indicated in Figure 8.7. Anchor h is positioned relative to the roundabout centre c and the x position at the start of the entry curve (i.e., with x at the right

angle indicated in Figure 8.7). This anchor h has a distance of r to x, distance $2r$ to c, and $x - h$ is orthogonal to d, while taking into account the relative orientation of the roundabout as implied by ω in the corresponding specification of the flyable entry procedure in Figure 8.7. These declarative constraints on h are expressed in Figure 8.8 as a precondition (for notational convenience, we formulate the precondition as tests in the hybrid program rather than using an implication in the formula) to the flyable entry procedure. The property in Figure 8.8a specifies that the tangential configuration of the simple choice for *agree* in Figure 8.4 is reached by a flyable curve when waiting until aircraft x and centre c have distance r. The existence of a choice for the anchor point h satisfying the requirements in Figure 8.8a can be shown by proving the dual $\mathsf{d}\mathcal{L}$ diamond formula in Figure 8.8b.

$$
\begin{aligned}
&(r\omega)^2 = d_1^2 + d_2^2 \wedge (x_1 - c_1)^2 + (x_2 - c_2)^2 = 3r^2 \wedge \exists \lambda{\geq}0\, (x + \lambda d = c) \\
\rightarrow &[h := *; \\
&\quad ?(d_1 = \omega(x_2 - h_2) \wedge d_2 = -\omega(x_1 - h_1)); \\
&\quad ?((h_1 - c_1)^2 + (h_2 - c_2)^2 = (2r)^2); \\
&\quad x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = \omega d_2 \wedge d_2' = -\omega d_1 \,\&\, ((x_1 - c_1)^2 + (x_2 - c_2)^2 \geq r^2) \\
&]\big((x_1 - c_1)^2 + (x_2 - c_2)^2 \leq r^2 \rightarrow (d_1 = -\omega(x_2 - c_2) \wedge d_2 = \omega(x_1 - c_1))\big)
\end{aligned}
$$

8.8a: Entry procedure reaches tangential configuration

$$
\begin{aligned}
&(r\omega)^2 = d_1^2 + d_2^2 \wedge (x_1 - c_1)^2 + (x_2 - c_2)^2 = 3r^2 \wedge \exists \lambda{\geq}0\, (x + \lambda d = c) \\
\rightarrow &\langle h := *; \\
&\quad ?(d_1 = \omega(x_2 - h_2) \wedge d_2 = -\omega(x_1 - h_1)); \\
&\quad ?((h_1 - c_1)^2 + (h_2 - c_2)^2 = (2r)^2); \\
&\rangle\ \mathit{true}
\end{aligned}
$$

8.8b: Choices of entry procedure are feasible

Figure 8.8.: Flyable entry procedure

Symmetry Reduction The properties in Figure 8.8 can be verified in a simplified version. To overcome the complexity of real quantifier elimination [CH91], which is doubly exponential in the number of quantifier alternations and (at least single) exponential in the number of variables, we use *symmetry reduction* to simplify the properties in Figure 8.8 computationally. That is, we exploit symmetries in the state space to reduce its dimension by fixing some variables, concluding safety for states that result from the fixed states by symmetric transformations. Without loss of generality, we can recenter the coordinate system to have c at position 0. Further,

we can assume aircraft x to come from the left by changing the orientation of the coordinate system. Finally, we can assume, without loss of generality, the linear speed to be 1 (by rescaling units appropriately). Observe that we cannot fix a value for both the linear speed and the angular velocity, because their units are strictly interdependent. In other words, if we fix the linear speed, we need to consider all angular velocities to verify the maneuver for all possible curve radii r for the roundabout maneuver. The x position resulting from these symmetry reductions can be determined by Pythagoras theorem (i.e., $(2r)^2 = r^2 + x_1^2$ for the triangle enclosed by h, x, c) or simple trigonometry as follows, see Figure 8.7:

$$x = (2r\cos\frac{\pi}{6}, 0) = (\sqrt{(2r)^2 - r^2}, 0) = (\sqrt{3}r, 0) \ . \tag{8.1}$$

To express the square root function using polynomial terms, we can easily use a random assignment for x_1 with a test condition $x_1^2 = (\sqrt{3}r)^2 = 3r^2$. Consequently, we simplify Figure 8.8 by specialising to the following situation:

$$\begin{array}{l} d_1 := 1;\ d_2 := 0;\quad c_1 := 0;\ c_2 := 0; \\ x_2 := 0; \\ r := *;\ ?r > 0;\ \omega := 1/r; \\ x_1 := *;\ ?x_1^2 = 3r^2 \wedge x_1 \leq 0; \end{array}$$

Verification results for the resulting entry procedure after symmetry reduction, and a proof of existence of a corresponding anchor point h according to Figure 8.8b, will be shown in Table 8.2.

For proving the feasibility property in Figure 8.8b within reasonable time, it is sufficient to specialise the state by symmetry reduction to $c_1 := 0 \wedge c_2 := 0 \wedge \omega := 1$. Interestingly, this example shows another surprise of quantifier elimination complexity and the power of symmetry reduction quite impressively: Without the reduction to $\omega := 1$, QE of Mathematica runs for more than 20 days without producing a result on an Intel Xeon X5365 with 3GHz and 16GB memory (of which only 3GB are used), even when adding the state space reduction $r := 1$.

8.2.5. Bounded Entry Duration

As the first step for showing that the entry procedure finally succeeds and maintains a safe distance all the time, we show that *entry* succeeds in bounded time and cannot take arbitrarily long to succeed (AC4).

Qualitative Analysis By a simple consequence of the proof for Figure 8.8a, the entry procedure follows a circular motion around the anchor point h, which is chosen according to Figure 8.8a, see Figure 8.9. That is to say that the property $\|x - h\| = r$, with the radius r belonging to the angular velocity ω and linear

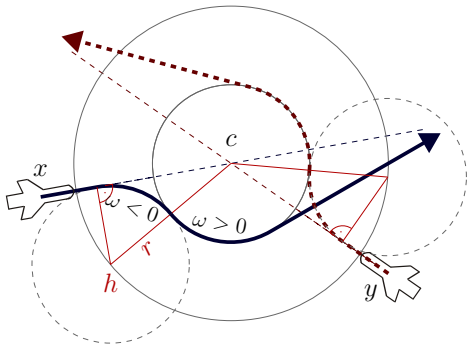


Figure 8.9.: Characteristics of flyable entry maneuver

speed $\|d\|$, is a (differential) invariant of *entry*. Indeed, the corresponding property in Figure 8.10 is provable. For the remainder of AC4, we use informal arguments. According to the proof in Example 3.3, the speed $\|d\|$ is constant during the *entry* procedure. Thus, the aircraft proceed with nonzero minimum progress rate $\|d\|$ around the circle. Consequently, the entry maneuver ends in bounded time, because the arc length of the circle around h is bounded ($2\pi r$ for radius r) and every bounded curve will be traversed in bounded time with constant velocity and orientation.

$$
\begin{array}{l}
\quad (r\omega)^2 = d_1^2 + d_2^2 \land (x_1 - c_1)^2 + (x_2 - c_2)^2 = 3r^2 \land \exists \lambda{\geq}0\, (x + \lambda d = c) \\
\rightarrow [h := *; \\
\quad ?(d_1 = \omega(x_2 - h_2) \land d_2 = -\omega(x_1 - h_1)); \\
\quad ?((h_1 - c_1)^2 + (h_2 - c_2)^2 = (2r)^2); \\
\quad x_1' = d_1 \land x_2' = d_2 \land d_1' = \omega d_2 \land d_2' = -\omega d_1 \,\&\, ((x_1 - c_1)^2 + (x_2 - c_2)^2 \geq r^2) \\
\quad](x_1 - h_1)^2 + (x_2 - h_2)^2 = r^2
\end{array}
$$

Figure 8.10.: Flyable entry procedure is circular

Quantitative Analysis To obtain a quantitative bound on the duration of the entry maneuver, consider the following. The flight duration for a full circle of radius r around h at constant linear speed $\|d\|$ is $\frac{2\pi r}{\|d\|}$, because its arc length is $2\pi r$. By using the trigonometric identities underlying (8.1), we see that the aircraft do not even have to complete the full circle, but only $\frac{1}{6}$-th of a circle, i.e., $\frac{\pi}{3} = 60°$, see Figure 8.7. Consequently, the maximum duration T of the *entry* procedure can be determined as:

$$T := \frac{1}{6} \cdot \frac{2\pi r}{\|d\|} = \frac{\pi r}{3\|d\|} \tag{8.2}$$

Using the standard relation $|\omega| = \frac{v}{r}$ between the angular velocity ω, the corresponding radius r of the circle (of evasive actions), and the linear velocity $v = \|d\|$ of the aircraft, maximum duration T can also be expressed as $T = \frac{1}{6} \cdot \frac{2\pi}{|\omega|}$.

Note that the constant π in the expression (8.2) for T is not definable in the theory of real-closed fields (the transcendental number π is not even contained in the real-closed field of algebraic numbers, thus it cannot be definable). Yet, T only has to be some upper bound on the maximum duration of the *entry* procedure. Consequently, any overapproximation of the maximum duration computed using a rational $P \geq \pi$ will do, for instance $\frac{3.15r}{3\|d\|}$ for the approximation 3.15 instead of for $\pi = 3.1415926\ldots$ in (8.2).

8.2.6. Safe Entry Separation

In Section 8.2.4, we have shown that the simplified *entry* procedure from the tangential roundabout maneuver can be replaced by a flyable entry maneuver that meets the requirements of approaching tangentially according to Theorem 3.32 for each of the aircraft. Yet, we still have to show that the respective entry maneuvers of multiple aircraft do not produce mutually conflicting flight paths, i.e., spatial separation of all aircraft is maintained during the entry maneuvers of multiple aircraft (AC5 of Section 8.2.1).

Bounded Overapproximation We show that entry separation is a consequence of the bounded speed (AC2) and bounded duration (AC4) of the flyable entry procedure when initiating the negotiation phase *agree* with sufficient distance: With bounded speed, aircraft can only come closer by a limited distance in bounded time. Let b denote the overall speed bound during FTRM according to AC2 and let T be the time bound for the duration of the entry procedure according to AC4. As an overapproximation of the actual behaviour during the *entry* phase, the following property expresses that—when the *entry* procedure is initiated with sufficient distance $p + 2bT$—the protected zone will still be respected after the 2 aircraft follow *any* curved flight (including the actual choices during *entry* and subsequent *circ*) with speed $\leq b$ up to T time units (see Figure 8.11):

$$\begin{aligned}\|d\|^2 \leq \|e\|^2 \leq b^2 \wedge \|x-y\|^2 \geq (p+2bT)^2 \wedge p \geq 0 \wedge b \geq 0 \wedge T \geq 0 \\ \rightarrow [\tau := 0; \exists\omega\, \mathcal{F}(\omega) \wedge \exists\varrho\, \mathcal{G}(\varrho) \wedge \tau' = 1 \wedge \tau \leq T]\|x-y\|^2 \geq p^2 \quad (8.3)\end{aligned}$$

This property is a consequence of the fact that—regardless of the actual angular velocity choices—aircraft only make limited progress in bounded time when starting with bounded speeds:

$$\|d\|^2 \leq b^2 \wedge b \geq 0 \wedge x = z \rightarrow [\tau := 0;\ \exists\omega\, \mathcal{F}(\omega) \wedge \tau' = 1]\|x-z\|_\infty \leq \tau b \quad (8.4)$$

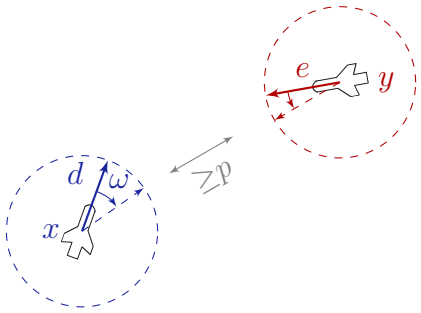


Figure 8.11.: Entry separation by bounded nondeterministic overapproximation

where the supremum norm $\|\cdot\|_\infty$ is definable as

$$\|x\|_\infty \leq c \equiv -c \leq x_1 \leq c \wedge -c \leq x_2 \leq c$$
$$\|x\|_\infty \geq c \equiv (x_1 \leq -c \vee c \leq x_1) \vee (x_2 \leq -c \vee c \leq x_2)$$

The limited progress property (8.4) is provable immediately by differential induction for the postcondition (the antecedent of G5 is provable as $x = z \vdash \|x - z\|_\infty = 0$):

$$\text{D10,D9}\,\frac{\text{G5}\,\dfrac{\text{F1}\,\dfrac{*}{\|d\|^2 \leq b^2 \wedge b \geq 0 \vdash \forall x, y, \tau\, \forall \omega\, (-b \leq d_1 \leq b \wedge -b \leq d_2 \leq b)}}{\|d\|^2 \leq b^2 \wedge b \geq 0 \wedge x = z \vdash \langle \tau := 0 \rangle [\exists \omega\, \mathcal{F}(\omega) \wedge \tau' = 1] \|x - z\|_\infty \leq \tau b}}{\|d\|^2 \leq b^2 \wedge b \geq 0 \wedge x = z \vdash [\tau := 0;\ \exists \omega\, \mathcal{F}(\omega) \wedge \tau' = 1] \|x - z\|_\infty \leq \tau b}$$

Cartesian Degree Reduction Due to the complexity of QE, the proof of the original property (8.3) does not terminate quickly enough. To overcome this issue, we simplify property (8.3) and use the (linearly definable) supremum norm $\|\cdot\|_\infty$ in place of the (quadratically definable) Euclidean 2-norm $\|\cdot\|_2$, thereby yielding the following provable variant of (8.3):

$$\|d\|^2 \leq \|e\|^2 \leq b^2 \wedge \|x - y\|_\infty \geq (p + 2bT) \wedge p \geq 0 \wedge b \geq 0 \wedge T \geq 0$$
$$\rightarrow [\tau := 0; \exists \omega\, \mathcal{F}(\omega) \wedge \exists \varrho\, \mathcal{G}(\varrho) \wedge \tau' = 1 \wedge \tau \leq T] \|x - y\|_\infty \geq p \quad (8.5)$$

It can be shown (and is provable even by QE) that the supremum norm $\|\cdot\|_\infty$ and the standard Euclidean norm $\|\cdot\|_2$ are equivalent, i.e., their values are identical up to constant factors:

$$\forall x\, (\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty)$$
$$\forall x\, (\frac{1}{\sqrt{n}}\|x\|_2 \leq \|x\|_\infty \leq \|x\|_2)$$

where n is the finite dimension of the vector space, here 2. From this equivalence of norms, we can conclude that the following variant of (8.5) with 2-norms is valid:

$$\|d\|^2 \leq \|e\|^2 \leq b^2 \wedge \|x-y\|_2 \geq \sqrt{2}(p+2bT) \wedge p \geq 0 \wedge b \geq 0 \wedge T \geq 0 \\ \rightarrow [\tau := 0; \exists\omega\, \mathcal{F}(\omega) \wedge \exists\varrho\, \mathcal{G}(\varrho) \wedge \tau' = 1 \wedge \tau \leq T]\|x-y\|_2 \geq p$$

The extra factor of $\sqrt{2}$ in the separation requirement results from the relaxation of the 2-norm to the ∞-norm. Using the bounded duration property AC4 of the entry maneuver, it is easy to see that the entry maneuver is a special case of the above nondeterministic hybrid program. Thus we conclude that the following property is valid:

$$\|d\|^2 \leq \|e\|^2 \leq b^2 \wedge \|x-y\|_2 \geq \sqrt{2}(p+2bT) \wedge p \geq 0 \wedge b \geq 0 \wedge T \geq 0 \\ \rightarrow [entry]\|x-y\|_2 \geq p \quad (8.6)$$

Far Separation By combining the estimation of the entry duration from equation (8.2) with the entry separation property (8.6), we can determine the following magnitude as the *far separation*, i.e., the distance which guarantees that protected zone p is maintained during the maneuver, including its entry phase:

$$f := \sqrt{2}(p+2bT) = \sqrt{2}\left(p + \frac{2}{3}\pi r\right) \quad (8.7)$$

Using f as an abbreviation for that term, we can abbreviate property (8.6) as follows:

$$\|d\|^2 \leq \|e\|^2 \leq b^2 \wedge \|x-y\|_2 \geq f \wedge p \geq 0 \wedge b \geq 0 \rightarrow [entry]\|x-y\|_2 \geq p \quad (8.8)$$

8.3. Synchronisation of Roundabout Maneuvers

In the previous sections, we have analysed collision freedom and separation properties of the various curved flight phases of FTRM. According to our verification plan in Section 8.2.1, we also need to show that the various actions of the respective aircraft are synchronised appropriately to ensure safety of the maneuver. We analyse the negotiation phase and compatible exit procedures next.

8.3.1. Successful Negotiation

For the negotiation phase to succeed, we have to show that there is a common choice of the roundabout centre c and angular velocity ω (or radius r) so that all participating aircraft can satisfy the requirements of their respective entry procedures simultaneously, i.e., of the property in Figure 8.8a (we thus have to show AC6 of Section 8.2.1).

Separate Success First of all, we show that there is a choice of c and r and ω during the *agree* mode such that the antecedent of the property in Figure 8.8a is satisfied, i.e., the *agree* mode is feasible:

$$\langle c := *;\ r := *;\ ?\|x - c\| = \sqrt{3}r \wedge r \geq 0; \omega := *;\ ?(r\omega)^2 = d_1^2 + d_2^2\rangle \Big(\|x - c\| = \sqrt{3}r \wedge \exists\lambda{\geq}0\,(x + \lambda d = c)\Big) \quad (8.9)$$

The dual property shows that, in fact, *all* choices of *agree* that also satisfy the (feasible) constraint $\exists\lambda{\geq}0\,(x + \lambda d = c)$ already satisfy the *entry* requirements:

$$[c := *;\ r := *;\ ?\|x{-}c\| = \sqrt{3}r{\wedge}r \geq 0;\ ?\exists\lambda{\geq}0\,(x{+}\lambda d = c);\ \omega := *;\ ?(r\omega)^2 = d_1^2{+}d_2^2] \Big(\|x - c\| = \sqrt{3}r \wedge \exists\lambda{\geq}0\,(x + \lambda d = c)\Big) \quad (8.10)$$

Joint Negotiation Success Similarly, we prove that all corresponding choices of *agree* satisfy the mutual requirements of multiple aircraft simultaneously, i.e., there are always compatible choices for c, r, ω for multiple aircraft. First, we prove that all corresponding choices satisfy mutual requirements when choosing roundabout center c as the intersection of the flight paths of the aircraft at x and y, see Figure 8.12:

$$\begin{aligned}&\|d\| = \|e\| \wedge \lambda > 0 \wedge x + \lambda d = y + \lambda e \rightarrow\\ &[c := x + \lambda d;\ r := *;\ ?\|x - c\| = \sqrt{3}r;\ ?\|y - c\| = \sqrt{3}r;\ \omega := *;\ ?(r\omega)^2 = d_1^2 + d_2^2]\\ &\big(\|x - c\| = \sqrt{3}r \wedge \lambda \geq 0 \wedge x + \lambda d = c \wedge \|y - c\| = \sqrt{3}r \wedge \lambda \geq 0 \wedge y + \lambda e = c\big)\end{aligned} \quad (8.11)$$

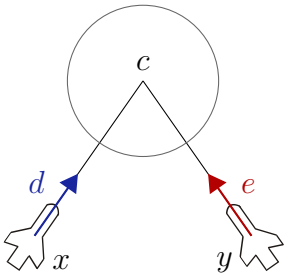


Figure 8.12.: Mutually agreeable negotiation choices for aircraft

Secondly, these choices are feasible, i.e., there always is a mutually agreeable

choice:

$$\begin{array}{l}\|d\| = \|e\| \wedge \lambda > 0 \wedge x + \lambda d = y + \lambda e \rightarrow \\ \langle c := x + \lambda d;\ r := *;\ ?\|x - c\| = \sqrt{3}r;\ ?\|y - c\| = \sqrt{3}r;\ \omega := *;\ ?(r\omega)^2 = d_1^2 + d_2^2 \rangle \\ \big(\|x - c\| = \sqrt{3}r \wedge \lambda \geq 0 \wedge x + \lambda d = c \wedge \|y - c\| = \sqrt{3}r \wedge \lambda \geq 0 \wedge y + \lambda e = c\big)\end{array} \tag{8.12}$$

This property follows from the following provable simplified variant, obtained from formula (8.12) by joining the equations $\|x - c\| = \sqrt{3}r$ and $\|y - c\| = \sqrt{3}r$:

$$\begin{array}{l}\|d\| = \|e\| \wedge \lambda > 0 \wedge x + \lambda d = y + \lambda e \rightarrow \\ \qquad \langle c := x + \lambda d;\ r := *;\ ?\|x - c\| = \|y - c\| = \sqrt{3}r \rangle \mathit{true}\end{array} \tag{8.13}$$

That $\|x - c\|$ can be written in the form $\sqrt{3}r$ for some r is easy to see just by division, similarly for the choice of ω.

Far Separation The *entry* procedure has to be initiated while the aircraft are still sufficiently far apart for safety reasons. Correspondingly, the *agree* procedure will negotiate a roundabout choice while the aircraft have far distance. Thus, the *agree* procedure will have to maintain far separation, i.e., satisfy the property

$$\|x - y\| \geq \sqrt{2}(p + \frac{2}{3}\pi r) \ \rightarrow\ [\mathit{agree}]\|x - y\| \geq \sqrt{2}(p + \frac{2}{3}\pi r) \tag{8.14}$$

This appears to be a trivial property, because *agree* models the successful completion of the negotiation, so that no time elapses during *agree*, hence the positions x and y do not even change. Observe, however, that the far separation distance according to equation (8.7) depends on the protected zone p and the radius r of evasive actions. Unlike p, radius r may change during *agree*, which allows for the flexibility of changing the flight radius r adaptively when repeating the roundabout maneuver loop at different positions. Consequently, the far separation distance $\sqrt{2}(p + \frac{2}{3}\pi r)$ is affected when changing r.

To ensure that the new radius r is chosen such that far separation is still maintained, i.e., property (8.14) is respected, we add a corresponding constraint to *agree*. Thus, changes of r are only accepted as long as they do not compromise far separation. We show that, when adding a corresponding constraint to property (8.11), all choices by *agree* maintain far separation of the aircraft at x and y according

to (8.7):

$$\begin{aligned}
&\|d\| = \|e\| \wedge \lambda > 0 \wedge x + \lambda d = y + \lambda e \rightarrow \\
&[c := x + \lambda d;\ r := *;\ ?\|x - c\| = \sqrt{3}r;\ ?\|y - c\| = \sqrt{3}r;\ ?\|x - y\| \geq \sqrt{2}(p + \frac{2}{3}\pi r); \\
&\qquad \omega := *;\ ?(r\omega)^2 = d_1^2 + d_2^2] \\
&\big(\|x - c\| = \sqrt{3}r \wedge \lambda \geq 0 \wedge x + \lambda d = c \wedge \|y - c\| = \sqrt{3}r \wedge \lambda \geq 0 \wedge y + \lambda e = c \\
&\qquad \wedge \|x - y\| \geq \sqrt{2}(p + \frac{2}{3}\pi r)\big) \quad (8.15)
\end{aligned}$$

Finally, we analyse when such choices of *agree* are feasible:

$$\begin{aligned}
&\|d\| = \|e\| \wedge \lambda > 0 \wedge x + \lambda d = y + \lambda e \rightarrow \\
&\langle c := x + \lambda d;\ r := *;\ ?\|x - c\| = \|y - c\| = \sqrt{3}r\rangle \|x - y\| \geq \sqrt{2}(p + \frac{2}{3}\pi r) \quad (8.16)
\end{aligned}$$

The corresponding distance constraints on x, y and c for *agree*, respectively, are depicted in Figure 8.13. Using standard trigonometric relations for each half of the

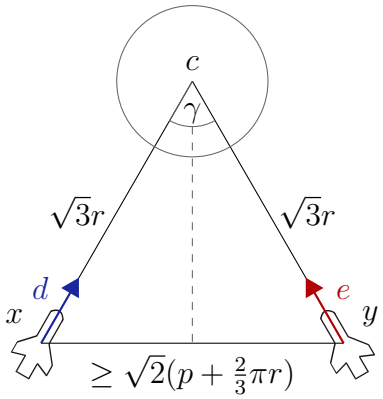


Figure 8.13.: Far separation for mutually agreeable negotiation choices

triangle, we can compute the resulting distance of x and y as $\|x - y\| = 2\sqrt{3}r \sin\frac{\gamma}{2}$. With quantifier elimination and simple evaluation for the remaining trigonometric expressions, we can determine under which circumstances property (8.16) holds true, i.e., for all protected zones p there is a radius r satisfying the distance requirements:

$$\mathrm{QE}\left(\forall p\, \exists r{\geq}0\ \left(2\sqrt{3}r \sin\frac{\gamma}{2} \geq \sqrt{2}(p + \frac{2}{3}\pi r)\right)\right) \equiv \sin\frac{\gamma}{2} > \frac{1}{3}\sqrt{\frac{2}{3}}\pi \equiv \gamma > 117.527^\circ$$

Consequently, corresponding choices are feasible for all protected zones with flight paths that do not intersect with narrow collision angles. The constraint on the

flight path intersection angle would relax to $\gamma > 74.4°$ when removing the extra factor of $\sqrt{2}$ from (8.7), which only results from our computational simplification of cartesian degree reduction from Section 8.2.6.

Despite the presence of trigonometric expressions, the above formula is a substitution instance of first-order real arithmetic and can thus be handled by QE using Lemma 2.13. Note that the primary difference to trigonometric expressions occurring in the solutions of flight equations for curved flight—which do not support quantifier elimination—is that the argument $\frac{\gamma}{2}$ of sin is not quantified over, here.

8.3.2. Safe Exit Separation

For the tangential roundabout maneuver from Chapter 3, no special exit procedure is needed for safety, because the maneuver repeats when further air traffic conflicts arise. For the FTRM, instead, we need to show that the exit procedure produces safe flight paths until the aircraft are sufficiently separated, because—when repeating the FTRM maneuver—the entry procedure requires more separation than just p for safety, see Figure 6.7.

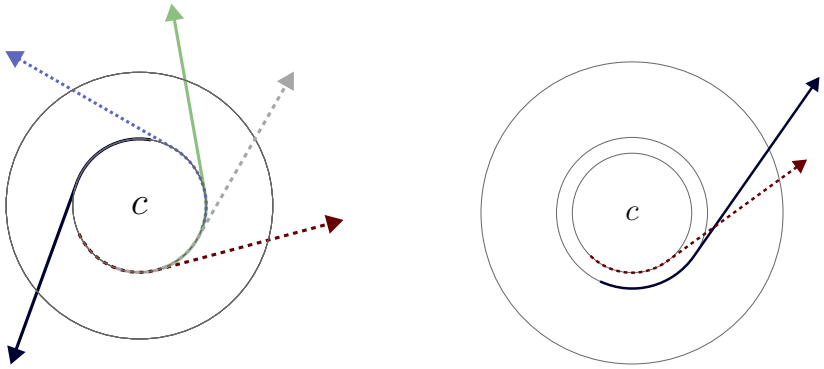
Safe Separation In order to establish the safe separation of aircraft during their *exit* procedures, we show that, for its bounded duration, the exit procedure does not conflict with other flight curves such that the initial far separation is again maintained as needed by the flyable entry procedure when re-initiating collision avoidance maneuvers repeatedly. For showing property AC7 of Section 8.2.1, we have to show the following:

$$\begin{aligned} d = \omega(x-c)^\perp \wedge e = \omega(y-c)^\perp \wedge \|x-y\|^2 \geq p^2 \\ \rightarrow [x' = d \wedge y' = e \wedge e' = e^\perp;\, x' = d \wedge y' = e]\|x-y\|^2 \geq p^2 \end{aligned} \quad (8.17)$$

This property expresses that, when x already exits on a straight line while y keeps following the roundabout for a while until both exit on straight lines, then the protected zones are respected at any point. Since, in Section 8.3.1, we have assumed simultaneous entry and identical speed, the aircraft can also be exit simultaneously. With that, property (8.17) simplifies to:

$$d = \omega(x-c)^\perp \wedge e = \omega(y-c)^\perp \wedge \|x-y\|^2 \geq p^2 \rightarrow [x' = d \wedge y' = e]\|x-y\|^2 \geq p^2 \quad (8.18)$$

This property expresses that safely separated aircraft that exit simultaneously along straight lines from tangential positions of a roundabout always remain safely separated. To reduce the arithmetical complexity, we overapproximate this property by showing that even the whole exit rays never cross when the aircraft exit the same roundabout tangentially (see Figure 8.14a; the counterexample in Figure 8.14b



8.14a: Exit ray separation

8.14b: Incompatible exit rays

Figure 8.14.: Exit procedure separation

shows that the assumption on identical radii is required for this relaxation):

$$d = \omega(x-c)^\perp \wedge e = \omega(y-c)^\perp \wedge \|x-c\|^2 = \|y-c\|^2 \wedge x \neq y \rightarrow [x' = d; y' = e]x \neq y \tag{8.19}$$

Again the computational complexity of proving this property can be simplified substantially by adding $c_1 := 0 \wedge c_2 := 0$ by symmetry reduction. From this property, the original separation property follows using the geometric fact that, for linearity reasons, rays that never cross cannot come closer than the original distance p, which can be expressed elegantly in $\mathsf{d}\mathcal{L}$ as:

$$\|x-y\|^2 \geq p^2 \wedge [x' = d \wedge y' = e]x \neq y \rightarrow [x' = d \wedge y' = e]\|x-y\|^2 \geq p^2 \tag{8.20}$$

Thus, by combining (8.19) with (8.20) and the simple fact that the sequential independent ray evolution $x' = d; y' = e$ is an overapproximation of the synchronous evolution $x' = d \wedge y' = e$, we conclude that property (8.18) is valid.

Far Separation To show that even an arbitrarily large separation is reached when following the exit procedure long enough, we prove that aircraft which enter roundabouts in different directions always remain in different directions while following the roundabout:

$$d = \omega(x-c)^\perp \wedge e = \omega(y-c)^\perp \wedge d \neq e \rightarrow [\mathcal{F}(\omega) \wedge \mathcal{G}(\omega)](d_1 - e_1)^2 + (d_2 - e_2)^2 > 0 \tag{8.21}$$

Aircraft in a simultaneous roundabout maneuver can, indeed, never enter in the same direction: Either they would already have collided (when they are entering in the same direction at the same position) or crash later on (when entering in the same direction with opposing orientations at different positions of the roundabout circle), but we have already shown that the roundabout maneuver is collision free (Theorem 3.32). Thus by (8.21), they maintain their different directions while

following the roundabout. Again, we can combine (8.21) with the geometric fact that rays into different directions which never cross will be arbitrarily far apart after sufficient time (see Figure 8.14a):

$$d \neq e \wedge [x' = d \wedge y' = e]x \neq y \rightarrow \forall a \langle x' = d \wedge y' = e \rangle \|x - y\|^2 > a^2$$

By combining this geometric fact with (8.21), we obtain the final separation property saying that—due to their different directions—the exit procedure will finally separate the aircraft arbitrarily far:

$$d = \omega(x - c)^\perp \wedge e = \omega(y - c)^\perp \wedge d \neq e \rightarrow \forall a \langle x' = d \wedge y' = e \rangle \|x - y\|^2 > a^2 \quad (8.22)$$

8.4. Compositional Verification

In the remainder of this chapter, we combine the previous results about the individual phases of aircraft flight into a full model of the flyable tangential roundabout maneuver that inherits safety by combining the individual safety properties of the respective phases. To begin with, we show systematically how verification results for large systems can be obtained compositionally in our calculus from verification lemmas about small subsystems whenever the large system has been constructed by composition from the corresponding small subsystems.

Assume we have subsystems α_i with lemmas guaranteeing $\psi_i \rightarrow [\alpha_i]\phi_i$, as is the case for the respective phases *agree*, *entry*, *circ* of FTRM. Then, in order to obtain a verification result for their sequential composition $\alpha_1; \alpha_2; \ldots; \alpha_n$, we have to show that postcondition ϕ_{i-1} of α_{i-1} implies precondition ψ_i of α_i for $i \leq n$. In fact, this is a derived proof rule of the $\mathsf{d}\mathcal{L}$ calculus and can be obtained directly from the generalisation rule G1 (with the usual cut P10 to shift the antecedent of a goal to the succedent of a new subgoal like for rule G3' in Section 2.5.1):

$$\text{G1}\dfrac{\vdash \forall^\alpha(\phi_i \rightarrow \phi) \qquad \text{D2}\dfrac{\text{G1}\dfrac{\psi \vdash [\alpha_{i-1}]\psi_i \qquad \vdash \forall^\alpha(\psi_i \rightarrow [\alpha_i]\phi_i)}{\psi \vdash [\alpha_{i-1}][\alpha_i]\phi_i}}{\psi \vdash [\alpha_{i-1}; \alpha_i]\phi_i}}{\psi \vdash [\alpha_{i-1}; \alpha_i]\phi}$$

8.5. Flyable Tangential Roundabout Maneuver

In this section, we combine the results of this chapter about the individual phases of flyable roundabouts into a full model of the flyable tangential roundabout maneuver that inherits safety modularly by joining the individual safety properties of the respective phases together. Essentially, we collect the various maneuver parts together according to the protocol cycle of Figure 8.3a and take care to ensure that

all safety prerequisites are met that we have identified for the respective phases previously. Formally, safety properties about the individual phases will be glued together in the calculus using the generalisation rule G1 following the compositional technique from Section 8.4.

The hybrid program describing the flyable tangential roundabout maneuver is depicted in Figure 8.15. The technical construction and protocol cycle of the entry

$$
\begin{aligned}
\psi &\equiv d_1^2 + d_2^2 = e_1^2 + e_2^2 \wedge r > 0 \wedge \phi(f) \rightarrow [FTRM^*]\phi(p) \\
\phi(p) &\equiv \|x-y\|^2 \geq p^2 \equiv (x_1-y_1)^2 + (x_2-y_2)^2 \geq p^2 \\
\phi(f) &\equiv \|x-y\|^2 \geq 2\left(p + \frac{2}{3}\pi r\right)^2 \\
\mathit{compat} &\equiv \|x-c\| = \sqrt{3}r \wedge \exists\lambda{\geq}0\,(x+\lambda d = c) \wedge \|y-c\| = \sqrt{3}r \wedge \exists\lambda{\geq}0\,(y+\lambda e = c) \\
\mathit{FTRM} &\equiv \mathit{free}^*;\ \mathit{agree};\ \Pi(\mathit{entry};\ \mathit{circ};\ \mathit{exit}) \\
\mathit{free} &\equiv \omega := *; \varrho := *; \mathcal{F}(\omega) \wedge \mathcal{G}(\varrho) \wedge \phi(f) \\
\mathit{agree} &\equiv c := *;\ r := *;\ ?\mathit{compat};\ ?\phi(f); \\
&\quad\ \omega := *;\ ?(r\omega)^2 = d_1^2 + d_2^2 \\
&\quad\ x_0 := x; d_0 := d; y_0 := y; e_0 := e \\
\mathit{entry} &\equiv \mathcal{F}(-\omega)\ \texttt{until}\ \|x-c\|^2 = r^2 \\
\mathit{circ} &\equiv \mathcal{F}(\omega)\ \texttt{until}\ \exists\lambda{\geq}0\,\exists\mu{>}0\,(x+\lambda d = x_0 + \mu d_0) \\
\mathit{exit} &\equiv \mathcal{F}(0);\ ?\phi(f)
\end{aligned}
$$

Figure 8.15.: Flight control with flyable tangential roundabout collision avoidance

procedure have already been illustrated in Figure 8.3. The operation $\mathcal{F}(\omega)$ `until` G expresses that the system follows differential equation $\mathcal{F}(\omega)$ until condition G is true. It is defined in terms of invariant regions and guards to make sure the system neither leaves mode $\mathcal{F}(\omega)$ later nor earlier than specified. We define $\mathcal{F}(\omega)$ `until` G as $\mathcal{F}(\omega) \wedge {\sim}G;\ ?G$, using the weak negation ${\sim}G$ from Section 3.5.6 to retain the border of G in the invariant. For instance, $\mathcal{F}(\omega)$ `until` $x_1 \geq 0$ is $\mathcal{F}(\omega) \wedge x_1 \leq 0; ?x_1 \geq 0$.

Finally, in FTRM, Π denotes the parallel product operator. Like in the work of Hwang et al. [HKT07], the FTRM maneuver is assumed to operate synchronously, i.e., all aircraft make simultaneous mode changes. Consequently, the parallel product $\Pi(\mathit{entry}; \mathit{circ}; \mathit{exit})$ simplifies to the conjunction of the respective differential equations in the various modes and can be defined easily as follows (with corresponding simplifications to resolve simultaneous tests):

$$\mathit{entry}_x \wedge \mathit{entry}_y;\ \mathit{circ}_x \wedge \mathit{circ}_y;\ \mathit{exit}_x \wedge \mathit{exit}_y$$

where entry_x is the entry procedure of the aircraft at position x etc. See Figure 8.16 for a fully instantiated version of Figure 8.15 with all abbreviations resolved.

$$\psi \equiv d_1^2 + d_2^2 = e_1^2 + e_2^2 \wedge r > 0 \wedge (x_1 - y_1)^2 + (x_2 - y_2)^2 \geq 2\left(p + \frac{2}{3}\pi r\right)^2 \rightarrow [FTRM^*]\phi(p)$$

$$FTRM \equiv \Bigg(\Big(\omega := *; \varrho := *;$$

free: $x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = -\omega d_2 \wedge d_2' = \omega d_1 \wedge y_1' = e_1 \wedge y_2' = e_2 \wedge e_1' = -\varrho e_2 \wedge e_2' = \varrho e_1 \wedge (x_1 - y_1)^2 + (x_2 - y_2)^2 \geq 2\left(p + \frac{2}{3}\pi r\right)^2 \Big)^*;$

agree: $c := *;\ r := *;\ ?(x_1 - c_1)^2 + (x_2 - c_2)^2 = 3r^2;$
$?\exists\lambda{\geq}0\,(x_1 + \lambda d_1 = c_1 \wedge x_2 + \lambda d_2 = c_2);$
$?(y_1 - c_1)^2 + (y_2 - c_2)^2 = 3r^2;$
$?\exists\lambda{\geq}0\,(y_1 + \lambda e_1 = c_1 \wedge y_2 + \lambda e_2 = c_2);$
$?(x_1 - y_1)^2 + (x_2 - y_2)^2 \geq 2\left(p + \frac{2}{3}\pi r\right)^2;$
$\omega := *;\ ?(r\omega)^2 = d_1^2 + d_2^2$
$x_1^0 := x_1;\ x_2^0 := x_2;\ d_1^0 := d_1;\ d_2^0 := d_2;$
$y_1^0 := y_1;\ y_2^0 := y_2;\ e_1^0 := e_1;\ e_2^0 := e_2;$

entry$_x \wedge$ *entry*$_y$: $x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = -(-\omega)d_2 \wedge d_2' = -\omega d_1 \wedge y_1' = e_1 \wedge y_2' = e_2 \wedge e_1' = -(-\omega)e_2 \wedge e_2' = -\omega e_1 \wedge \sim\big((x_1 - c_1)^2 + (x_2 - c_2)^2 = r^2\big);$
$?(x_1 - c_1)^2 + (x_2 - c_2)^2 = r^2;$

circ$_x \wedge$ *circ*$_y$: $x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = -\omega d_2 \wedge d_2' = \omega d_1 \wedge y_1' = e_1 \wedge y_2' = e_2 \wedge e_1' = -\omega e_2 \wedge e_2' = \omega e_1 \wedge \sim\big(\exists\lambda{\geq}0\,\exists\mu{>}0\,(x_1 + \lambda d_1 = x_1^0 + \mu d_1^0 \wedge x_2 + \lambda d_2 = x_2^0 + \mu d_2^0) \wedge \exists\lambda{\geq}0\,\exists\mu{>}0\,(y_1 + \lambda e_1 = y_1^0 + \mu e_1^0 \wedge y_2 + \lambda e_2 = y_2^0 + \mu e_2^0)\big);$
$?\big(\exists\lambda{\geq}0\,\exists\mu{>}0\,(x_1 + \lambda d_1 = x_1^0 + \mu d_1^0 \wedge x_2 + \lambda d_2 = x_2^0 + \mu d_2^0) \wedge \exists\lambda{\geq}0\,\exists\mu{>}0\,(y_1 + \lambda e_1 = y_1^0 + \mu e_1^0 \wedge y_2 + \lambda e_2 = y_2^0 + \mu e_2^0)\big);$

exit$_x \wedge$ *exit*$_y$: $x_1' = d_1 \wedge x_2' = d_2 \wedge y_1' = e_1 \wedge y_2' = e_2;$
$?(x_1 - y_1)^2 + (x_2 - y_2)^2 \geq 2\left(p + \frac{2}{3}\pi r\right)^2 \Bigg)$

Figure 8.16.: Flight control with FTRM (synchronous instantiation)

To verify this maneuver, we split the proof into the modular properties that we have already shown previously following the verification plan from Section 8.2.1. Formally, we use the generalisation rule (G1) to split the system at its sequential compositions, giving the subproperties depicted in Figure 8.17 with the mapping to previous results according to Table 8.1. The formula $\mathcal{T}$ is the characterisation of tangential configurations due to equation (3.6) from page 122.

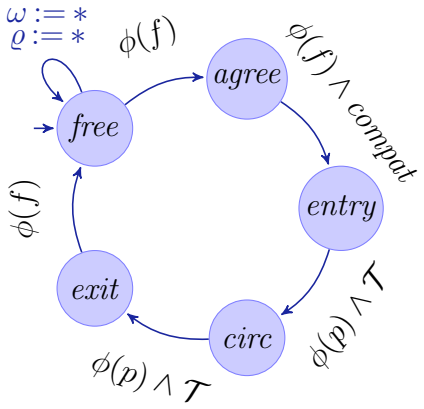


Figure 8.17.: Verification loop for flyable tangential roundabout maneuvers

Table 8.1.: Verification loop properties for flyable tangential roundabout maneuvers

Property	Consequence of
$\phi(f) \rightarrow [free]\,\phi(f)$	Theorem 3.32 or Figure 8.4
$\phi(f) \rightarrow [agree](\phi(f) \wedge compat)$	Property (8.11) and (8.15)
$compat \wedge \phi(f) \rightarrow [entry]\,\phi(p)$	Property (8.6)
$compat \wedge \phi(f) \rightarrow [entry]\,\mathcal{T}$	Figure 8.8a
$\mathcal{T} \wedge \phi(p) \rightarrow [circ](\phi(p) \wedge \mathcal{T})$	Theorem 3.32 or Figure 8.4
$\mathcal{T} \wedge \phi(p) \rightarrow [exit]\,\phi(p)$	Property (8.18)
$\mathcal{T} \wedge \phi(p) \rightarrow [exit]\,\phi(f)$	Properties (8.18) and (8.22)

By combining the results summarised in Table 8.1 about the respective flight phases of the FTRM according to the compositional verification approach obtained from generalisation rule G1 according to Section 8.4, we conclude that FTRM avoids collisions safely. In addition, we can use the modular proof structure to derive an even stronger consequence for generalised FTRM maneuvers. Using the same arguments as in Table 8.1, *any* roundabout maneuver following Figure 8.15 is collision free, even when it uses a different entry procedure that can be shown to succeed in tangential configuration within bounded time.

8.1 Theorem (Safety of flyable tangential roundabout maneuvers). *The FTRM system depicted in Figure 8.15 is collision free, i.e., the collision avoidance*

property ψ in Figure 8.15 is valid. More generally, any variation of FTRM with a modified entry procedure that safely reaches tangential configuration $\mathcal{T}$ in some bounded time T is safe, i.e., provided that the following formula is valid

$$\phi(f) \rightarrow [\tau := 0; \mathit{agree} \wedge \tau' = 1]\big((\tau \leq T \rightarrow \phi(p)) \wedge (\tau = T \rightarrow \mathcal{T})\big) \ .$$

8.6. Experimental Results

Table 8.2 summarises experimental results for the air traffic control case study of flyable tangential roundabout maneuvers. The rows marked with $*$ indicate a property where simplifications like symmetry reduction have been used to reduce the computational complexity of quantifier elimination. The results in Table 8.2 show that even aircraft maneuvers with challenging dynamics can be verified with our logic-based verification approach for hybrid systems.

Experimental results are from a 2.6GHz AMD Opteron with 4GB memory. Memory consumption of quantifier elimination is shown in Table 8.2, excluding the front-end. The dimension of the continuous state space and number of proof steps are indicated as well. For comparison, the results reported in Table 8.2 use the same settings as those in Table 5.3. Using different settings for initial timeouts for differential saturation results in faster performance for larger dimensions, particularly multiple aircraft, see Table 8.3.

The experimental results in Table 8.2 for property (8.5) can be improved. Currently, they still need as much as 29 simple user interactions to overcome preliminary simplifications in the implementation of our proof strategies in the KeYmaera tool. An improved implementation of iterative inflation order will reduce the number of required interactions to at most one that specifies the postcondition of the limited progress property (8.4) as an invariant.

8.7. Summary

For demonstrating the capabilities of our logic-based verification approach for hybrid systems, we have analysed challenging air traffic control applications. Aircraft can only follow sufficiently smooth flyable curves. Hence, mathematical maneuvers that require instant turns, especially classical straight line maneuvers, give physically impossible conflict resolution advise. As a case study for verification, we have developed a new collision avoidance maneuver that is fully flyable, i.e., it only includes sufficiently smooth curves without discontinuities. Despite its challenging dynamics and complicated maneuvering, we have verified collision avoidance in the resulting flyable tangential roundabout maneuver using a modular verification approach in our calculus. Due to the intricate spatio-temporal movement of aircraft in

Table 8.2.: Experimental results for air traffic control (initial timeout=10s)

Case study		Time(s)	Memory(MB)	Steps	Dim
tangential roundabout	(2 aircraft)	10.5	6.8	197	13
tangential roundabout	(3 aircraft)	636.1	15.1	342	18
tangential roundabout	(4 aircraft)	918.4	31.4	520	23
tangential roundabout	(5 aircraft)	3552.6	46.9	735	18
bounded speed control	(3.8)	19.6	34.4	28	12
bounded maneuver speed	Example 3.3	0.3	6.3	14	4
flyable roundabout entry*	Figure 8.8a	10.2	9.6	132	8
flyable entry feasible*	Figure 8.8b	104.4	87.9	16	10
flyable entry circular	Figure 8.10	2.9	7.6	81	5
limited progress	(8.4)	2	6.5	60	8
entry separation	(8.5), 29 int.	140.1	20.1	512	16
negotiation feasible	(8.9)	4.5	8.4	27	8
negotiation successful	(8.10)	2.7	21.8	34	8
mutual negotiation successful	(8.11)	0.9	6.4	60	12
mutual negotiation feasible	(8.13)	7.5	23.8	21	11
mutual far negotiation	(8.15)	2.4	8.1	67	14
simultaneous exit separation*	(8.19)	4.7	12.9	44	9
different exit directions	(8.21)	3	11.1	42	11

Table 8.3.: Experimental results for air traffic control (initial timeout=4s)

Case study		Time(s)	Memory(MB)	Steps	Dim
tangential roundabout	(2 aircraft)	10.4	6.8	197	13
tangential roundabout	(3 aircraft)	253.6	7.2	342	18
tangential roundabout	(4 aircraft)	382.9	10.2	520	23
tangential roundabout	(5 aircraft)	1882.9	39.1	735	18
bounded speed control	(3.8)	19.5	34.4	28	12
bounded maneuver speed	Example 3.3	0.5	6.3	14	4
flyable roundabout entry*	Figure 8.8a	10.1	9.6	132	8
flyable entry feasible*	Figure 8.8b	104.5	87.9	16	10
flyable entry circular	Figure 8.10	3.2	7.6	81	5
limited progress	(8.4)	1.9	6.5	60	8
entry separation	(8.5), 29 int.	140.1	20.1	512	16
negotiation feasible	(8.9)	4.8	8.4	27	8
negotiation successful	(8.10)	2.6	13.1	34	8
mutual negotiation successful	(8.11)	0.8	6.4	60	12
mutual negotiation feasible	(8.13)	7.5	23.8	21	11
mutual far negotiation	(8.15)	2.4	8.1	67	14
simultaneous exit separation*	(8.19)	4.3	12.9	44	9
different exit directions	(8.21)	3.1	11.1	42	11

roundabout maneuvers, some of the properties require intriguing arithmetic, which we tackled using symmetry reduction and degree-based reductions.

While the flyable roundabout maneuver is a highly nontrivial and challenging case study, there are still some modelling assumptions that can be generalised and relaxed in future work. Like in the work of Hwang et al. [HKT07], our primary analysis of the flyable roundabout maneuver still assumes synchronous conflict resolution, which implies that the collision avoidance maneuver has to be initiated from a symmetrical position, i.e., where all aircraft have the same speed and distance from the roundabout centre c. Our analysis of the tangential roundabout maneuver in Chapter 3 is more general on this respect already, concerning safety for arbitrary initial velocities and timing. Due to our simple symmetric choice of the roundabout centre c in the *agree* phase of the flyable roundabout maneuver, verification results can still be improved when aircraft approach the conflict zone with narrow angles or on almost parallel flight paths. Finally, it would be interesting future work to see if the informal robustness studies of Hwang et al. [HKT07] can be carried over to a formal proof in our calculus.

Chapter 9.

Conclusion

9.1. Summary

Hybrid systems are an equally important and challenging class of systems, whose numerous occurrences in safety-critical complex physical systems call for formal verification techniques that can be used to establish the correct functioning of these systems by rigorous mathematical analysis. The characteristic feature of the modelling idea behind hybrid systems is that they admit interacting discrete and continuous dynamics to capture the superposition of physical system dynamics with control at a natural modelling level. With these superpositions, hybrid systems can model challenging system dynamics fairly easily but also require sophisticated analysis techniques.

Application areas of hybrid systems where correctness properties play an important role range from small embedded controllers in automotive industries that regulate isolated processes like air bag inflation, over biomedical devices like glucose regulators following, e.g., model-predictive control, and large scale physical or chemical process control like injection control in combustion engines or full nuclear reactors, to complex physical traffic systems in train control or air traffic control scenarios. In other domains, hybrid effects also become increasingly important, including robotic applications, where mobile robots have to work reliably in safety-critical environments, e.g., in driverless vehicle technology. Similarly, circuit designs more often exhibit relevant hybrid effects, where increasing clock rates require to take mixed analog/digital effects into account, because larger parts of the chip remain analog, e.g., when reducing the number of extra latches that stabilise values computed by analog circuits.

Logic for Hybrid Systems As a general analysis technique for hybrid systems, we have introduced a systematic logic-based verification approach in this thesis, which is based on symbolic and mathematical logic, automated theorem proving, differ-

ential algebra, computer algebra, semialgebraic geometry, calculus, and on results from the theory of differential equations and dynamical systems, see Figure 9.1.

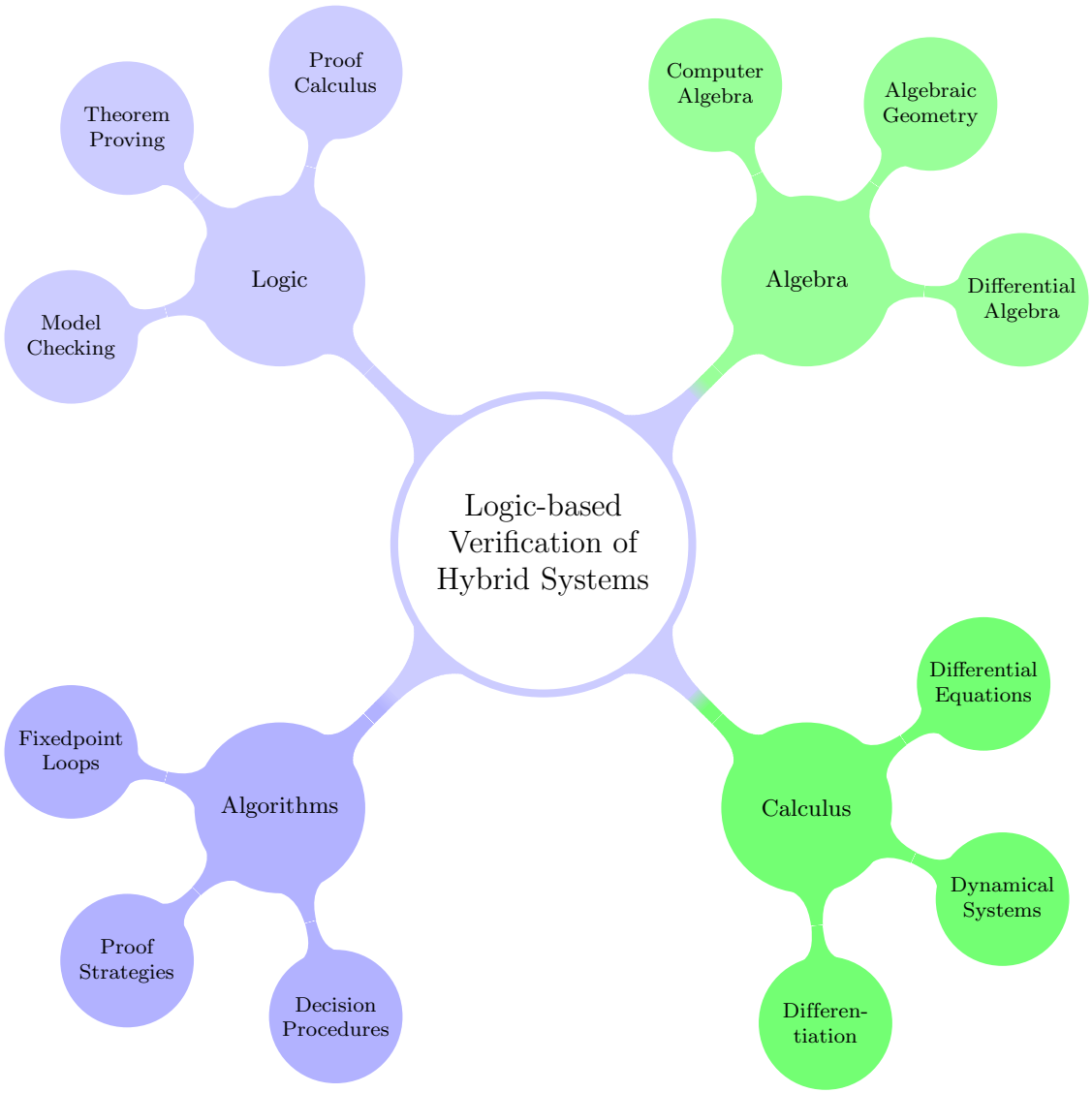


Figure 9.1.: Topics contributing to logic-based verification of hybrid systems

We introduced a series of *differential dynamic logics* (dℒ, DAL, dTL) as concise languages for specifying correctness properties of hybrid systems along with concise practical proof calculi for verifying hybrid systems. The logic dℒ is a first-order dynamic logic for hybrid programs, which extend classical discrete programs to uniform operational models for hybrid systems with interacting discrete jumps and continuous evolutions along differential equations. Its first-order completion, DAL, is a first-order dynamic logic for more general differential-algebraic programs

that allow for superpositions of first-order discrete jump constraints and first-order differential-algebraic constraints. In a further independent dimension, we augment $\mathsf{d}\mathcal{L}$ and DAL to a temporal dynamic logic, dTL, with independent modal path quantifiers over traces and temporal quantifiers along traces, thereby combining the capabilities of dynamic logic to reason about possible system behaviour with the power of temporal logic in reasoning about the temporal behaviour along traces.

Proof Calculi Our concise proof calculi for differential dynamic logics work compositionally by decomposing properties of hybrid systems into properties of their parts, which improves both tracability and scalability of results. In order to handle interacting hybrid dynamics, we lift real quantifier elimination to the deductive calculi in a new modular way that is suitable for automation, using a combination of real-valued free variables, Skolem terms, and invertible quantifier rules over the reals.

As a fundamental result aligning hybrid and continuous reasoning proof-theoretically, we have proven our calculi to axiomatise the transition behaviour of hybrid systems completely relative to the handling of differential equations. This result is based on an entirely new notion of hybrid completeness and shows that the calculi are adequate for verification purposes.

Furthermore, we have complemented discrete induction with a new first-order differential induction that uses differential invariants and differential variants for proving correctness statements about first-order differential-algebraic constraints purely algebraically based on the differential constraints themselves instead of their solutions. This is particularly relevant for verification, because solutions of differential equations quickly yield undecidable arithmetic or may not even be expressible in closed form. In combination with successive differential strengthening for refining the system dynamics by auxiliary differential invariants, we obtain a powerful verification calculus for hybrid systems with challenging dynamics.

Finally, our sequent calculus for the temporal extensino dTL of differential dynamic logic is a completely modular combination of temporal and non-temporal reasoning, where temporal formulas are handled using rules that augment intermediate state transitions with corresponding sub-specifications recursively. We have shown that this gives a complete calculus for temporal properties relative to the non-temporal base logic.

Logic-based Verification Algorithms To address practical scalability challenges for larger case studies, we have introduced proof strategies that navigate among the nondeterminisms in the proof calculi for differential dynamic logics, including iterative background closure and iterative inflation order strategies. Further, we have introduced a verification algorithm that computes differential invariants as fixed-

points in a proof loop for differential dynamic logic, thereby using the compositional properties of our calculi to exploit locality in system designs for verification.

Applications In a series of examples from practical application domains, we have demonstrated that our techniques can be used successfully for verifying safety, controllability, and liveness properties in realistic train control applications and challenging case studies for fully parametric roundabout maneuvers in air traffic control.

To put it in a nutshell, we have introduced and developed the theory, practice, and applications of differential dynamic logics, leading to a concise yet complete logic-based verification approach for hybrid systems, with which we can verify applications that were out of scope for other approaches.

9.2. Perspectives

While, broadly speaking, this thesis captures logic-based verification of hybrid systems in its full entirety, there are a number of natural extensions to an even broader range of applications.

Model Extensions As we have shown in previous work [Pla04a, BP06], dynamic logic can be augmented to support reasoning about dynamically reconfiguring system structures, which we want to extend to hybrid systems in future work by combining differential dynamic logic and dynamic logic with flexible functions [Pla04a, BP06]. This will result in a coherent verification technology for dynamic networks of hybrid systems with dynamic topology changes including dynamic appearance, e.g., of traffic agents in car platooning. Notice that—quite unlike what other approaches for hybrid systems provide—dynamic networks of hybrid systems with new appearance lead to dynamic adaptation of the *dimension* of the state space.

A further natural extension of the system modelling capabilities results from adding probabilistic behaviour on top of the hybrid behaviour, giving stochastic hybrid systems. While stochasticity reduces to mere nondeterminism in $\mathsf{d}\mathcal{L}$ for classical worst-case verification, this modelling extension will be important whenever applications call for verification results expressing that a safety or liveness property holds true with a certain minimum likelihood, though not with certainty.

Specification Logic Extensions Future work further includes extending dTL with CTL*-like [EH86] formulas of the form $[\alpha](\psi \wedge \Box\phi)$. Defining a corresponding semantics for dTL* is simple, but even the search for good deductive calculi for discrete CTL* has been a long standing challenge in branching time temporal logic [Rey01, PK02, Rey05].

Theory Extensions While the dℒ calculus is complete relative to the continuous fragment, it turns out to be a subtle open problem whether a converse calculus can exist that is complete relative to various discrete fragments. We conjecture that the answer depends in a subtle way on whether the deterministic or nondeterministic fragment is considered and further depends on whether quantifiers are allowed.

KeYmaera Implementation Concerning the tool support of our verification approach in KeYmaera, we are convinced that further significant performance improvements can be achieved by combining the different kinds of computational backends (see Section A.2) by more interleaved operations rather than independent ones. Moreover, our general ideas of and/or-branching from Section 5.4 can definitely be exploited in more detail in KeYmaera to tap the full potential of parallel proving for verification, which arises naturally in the dℒ calculus. Mere tuning of the proof strategies and KeYmaera implementation will probably also allow for a factor of 2 to 3 in improved performance when removing some preliminary simplifications in the implementation. Another area where the current implementation is still preliminary is the coupling of numerical or explorative model checking techniques with proof-based verification.

Application Domains The theoretical and practical results that we have developed in this thesis apply to several application domains other than those considered in this thesis. Inspired by the application scenario context of the AVACS project, most case studies in this thesis are from various traffic control settings in train or aircraft control. Yet, hybrid systems also occur quite naturally in completely different areas like computerized chemical or physical process control, biomedical applications, robotics, or analog-digital interaction effects on chips, which would be interesting areas for future research.

Part IV.

Appendix

Appendix A.

Implementation

Contents

Synopsis

KeYmaera is a hybrid verification tool for hybrid systems that combines deductive, real algebraic, and computer algebraic prover technologies. It is an automated and interactive theorem prover for a natural specification and verification logic for hybrid systems. KeYmaera supports *differential dynamic logic*, which is a real-valued first-order dynamic logic for hybrid programs, a program notation for hybrid automata. For automating the verification process, KeYmaera implements a generalized free-variable sequent calculus and automatic proof strategies that decompose the hybrid system specification symbolically. To overcome the complexity of real arithmetic, we integrate real quantifier elimination following our iterative background closure strategy. Our tool is particularly suitable for verifying parametric hybrid systems and has been used successfully for verifying collision avoidance in case studies from train control and air traffic control.

A.1. KeYmaera: A Hybrid Theorem Prover for Hybrid Systems

In the interest of a comprehensive presentation, we briefly characterise the implementation of our approach in our new verification tool KeYmaera [PQ08a].

KeYmaera is a hybrid theorem prover for hybrid systems that combines deductive, real algebraic, and computer algebraic prover technologies [PQ08a]. It is an automated and interactive theorem prover for differential dynamic logics for hybrid systems. It implements the calculi for differential dynamic logic $\mathsf{d}\mathcal{L}$, differential-algebraic dynamic logic DAL, and differential temporal dynamic logic dTL that we introduced in Part I.

KeYmaera has been implemented as a combination of the deductive theorem prover KeY [ABB+05, BGH+07, BHS07] with the computer algebra system Mathematica by Wolfram Research [Wol05] and the Orbital library developed by the author, with QEPCAD B [Bro03] as an quantifier elimination tool, see Figure A.1. KeY is a semi-interactive theorem prover with a user-friendly graphical interface for proving correctness properties of Java programs. We generalize KeY from discrete systems to hybrid systems by adding support for the differential dynamic logic $\mathsf{d}\mathcal{L}$ (and DAL and dTL, respectively). Figure A.2 shows a screenshot of the graphical user interface of KeYmaera.

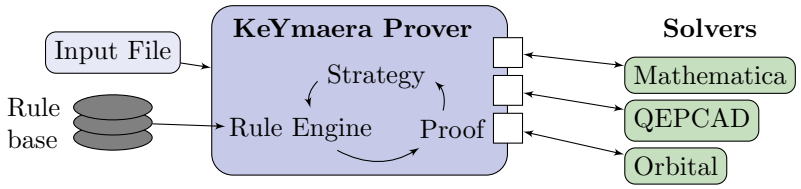


Figure A.1.: Architecture and plug-in structure of the KeYmaera prover

In discrete KeY, rule applications are comparably fast, but in KeYmaera, proof rules that use decision procedures for real arithmetic can require a substantial amount of time to produce a result. To overcome this, we have implemented new automatic proof strategies for the hybrid case that navigate among computationally expensive rule applications, following the strategies we developed in Chapter 5.

We have implemented a plug-in architecture for integrating multiple implementations of decision procedures for the different fields of arithmetic handling, cf. Figure A.1. We integrate arithmetical simplification and real quantifier elimination support by interfacing Mathematica or QEPCAD. Symbolic solutions of differential equations and symbolic total differentials for differential invariants and variants, are obtained either from Mathematica or Orbital.

To overcome the complexity pitfalls of quantifier elimination and to scale to real-world application scenarios, we implement the *iterative background closure* strategy

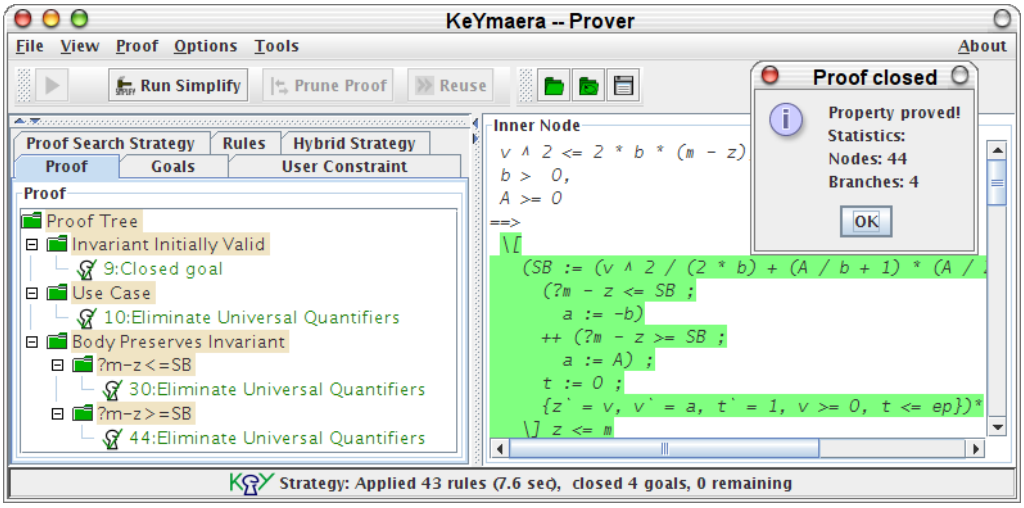


Figure A.2.: Screenshot of the KeYmaera user interface

from Section 5.4 that interleaves background solver calls with deductive $\mathsf{d}\mathcal{L}$ rules. For performance reasons, the KeYmaera implementation further optimises the order of quantifiers in universal closures $\forall^{\alpha}\phi$ according to the modification order in the hybrid program α that produced $\forall^{\alpha}\phi$ (see variable dependency orders from Section 6.2.5–6.2.6).

KeYmaera provides several options for adjusting the prover strategy, see Figure A.3. For a documentation of some of the data structures, also see the master's thesis of Jan-David Quesel [Que07] for further detail on the internal implementation of the KeYmaera tool.

Structure of this Chapter

In Section A.2, we summarise techniques that can be used as backends for handling real arithmetic in theorem provers for differential dynamic logics. We discuss the consequences of the KeYmaera architecture and implementation in Section A.3, concerning how soundness can inherit from soundness proofs for proof calculi to soundness of actual implementations. In Section A.4, we explain the setup for our experimental evaluations in this thesis.

A.2. Computational Backends for Real Arithmetic

In this section, we briefly summarise a range of techniques that we use as computational backends for handling real arithmetic in background provers for the foreground prover KeYmaera. These techniques are various alternatives for implementing the QE procedure in the F-rules of Figure 2.5 or Figure 3.3, respectively.

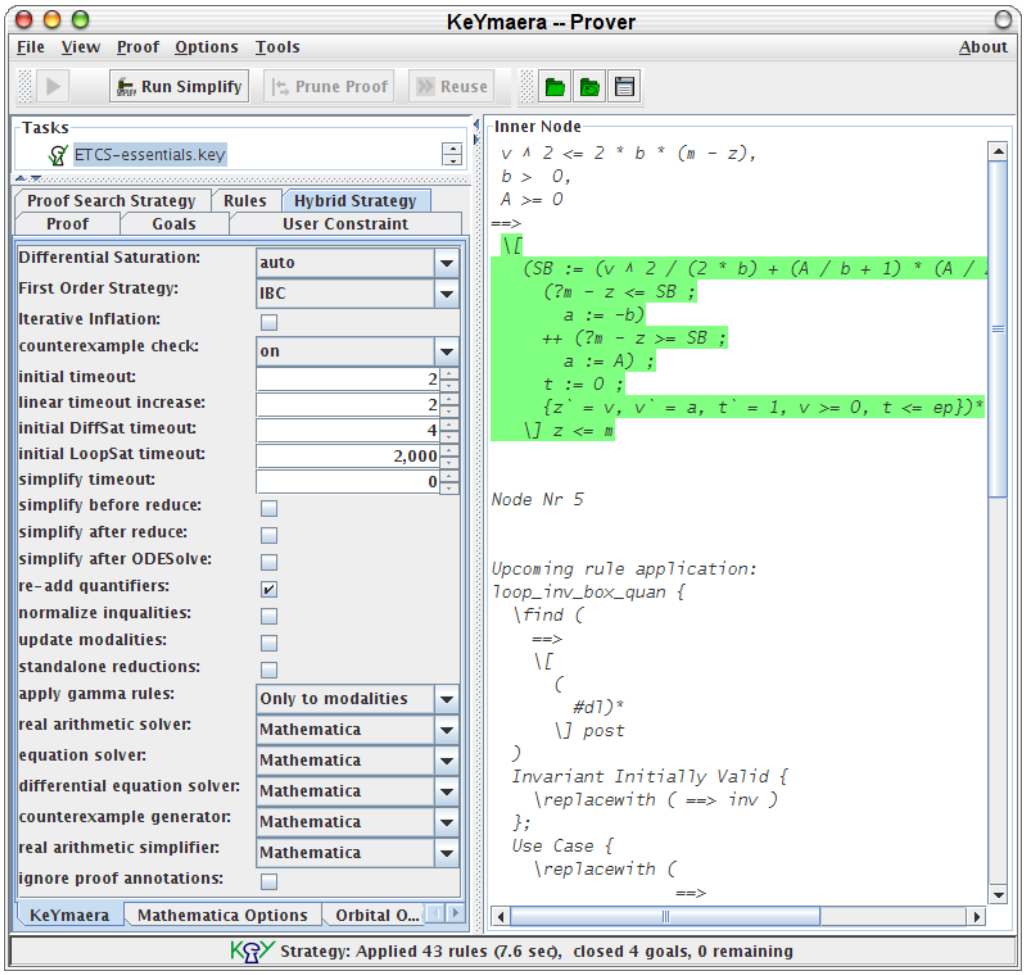


Figure A.3.: KeYmaera strategy options

A.2.1. Real-Closed Fields

Generalising classical techniques for finding the number of real roots for univariate polynomials due to Sturm [Stu35], Tarski showed that validity of closed formulas in the first-order theory of real arithmetic is equivalent to the theory of real-closed fields and decidable [Tar51] by quantifier elimination (Definition 2.10). Due to its importance, there are numerous refinements of this result that turn it into a more practical decision procedure [Sei54, Rob56, Łoj64, Coh69, KK71, DE73, Col75, ACM84a, ACM84b, CH91, CJ89, CJK90, CJK02, DH88, GV88, Wei88, CJ98, LW93, Bas99, Str06, Tiw05, BPR06, BD07], see an article by van den Dries for a historical overview [vdD88]. The complexity of quantifier elimination in real-closed fields is doubly exponential in the number of quantifier alternations and has been analysed carefully [DH88, Gri88, GV88, Ris88, Wei88, BD07, Ren92a, Ren92b, Ren92c, Ren92b, Ren92c, BPR96, Bas99]. We refer to Basu et al. [BPR06] for details.

A.1 Definition (Formally real field). A field R is a *(formally) real field* iff any of the following (equivalent) conditions holds (see [BPR06, Theorem 2.7]):

1. -1 is not a sum of squares in R.
2. For every $x_1, \ldots, x_n \in R$ we have that $\sum_{i=1}^{n} x_i^2 = 0$ implies $x_1 = \cdots = x_n = 0$.
3. R admits an ordering that makes R an ordered field.

There are several equivalent characterisations of real-closed fields [BPR06]:

A.2 Definition (Real-closed field). A field R is a *real-closed field* iff any of the following (equivalent) conditions holds (see [BPR06, Theorem 2.11]):

1. R is an ordered field where every positive element is a square and every polynomial in $R[X]$ of odd degree has a root in R (then this order is, in fact, unique).
2. R is not algebraically closed but its field extension $R[\sqrt{-1}] = R[i]/(i^2 + 1)$ is algebraically closed.
3. R is not algebraically closed but its algebraic closure is a finite extension, i.e., finitely generated over R.
4. R has the *intermediate value property*, i.e., R is an ordered field such that for any polynomial $p \in R[X]$ with $a, b \in R, a < b$ and $p(a)p(b) < 0$, there is a ζ with $a < \zeta < b$ such that $p(\zeta) = 0$.
5. R is a real field such that no proper algebraic extension is a formally real field.

A.1 Example. The following sets are real-closed fields:

- Real numbers $\mathbb{R}$.
- Real algebraic numbers $\bar{\mathbb{Q}} \cap \mathbb{R}$, that is, real numbers in the algebraic closure of $\mathbb{Q}$, i.e., real numbers that are roots of a non-zero polynomial with rational or integer coefficients
- Computable numbers [Wei05], i.e., those that can be approximated by a computable function up to any desired precision.
- Definable numbers, i.e., those real numbers $a \in \mathbb{R}$ for which there is a first-order formula φ in the language of set theory with one free variable such that a is the unique real number for which φ holds true.

According to the important Tarski-Seidenberg Principle [Tar51, Sei54] or the Transfer Principle for real-closed fields, the first-order theory of real arithmetic is equivalent to the first-order theory of real-closed fields, see [BPR06, Theorems 2.80–2.81].

A.3 Theorem (Tarski-Seidenberg principle [Tar51, Sei54]). *The first-order theory of reals is identical to the first-order theory of real-closed fields, i.e., the set of closed first-order formulas over the signature $+, \cdot, =, <, 0, 1$ that are valid over the reals $\mathbb{R}$ is the same as the corresponding set of formulas that are valid in any real-closed field.*

The technical device exploiting this result for deciding formulas of real arithmetic is quantifier elimination via cylindrical algebraic decomposition, which we examine next.

A.2.2. Semialgebraic Geometry and Cylindrical Algebraic Decomposition

From an algebraic or model-theoretic perspective, a quantifier-free formula of real arithmetic directly corresponds to the set of its satisfying assignments in $\mathbb{R}^n$. Formally, a *semialgebraic set* is a subset of $\mathbb{R}^n$ that is defined by a finite conjunction of polynomial equations and inequalities or any finite union of such sets, which, up to normalisation, is a quantifier-free formula of real-arithmetic. The most important part about projections in the following central theorem about semialgebraic sets is due to Tarski [Tar51, Sei54, Hod93, BCR98, vdD98].

A.4 Theorem (Semialgebraic sets). *Semialgebraic sets are closed under finite unions, finite intersections, complements, and projection (to linear subspaces).*

It is easy to see that projection of a semialgebraic set in $\mathbb{R}^n$ to a linear subspace (e.g., $\mathbb{R}^{n-1}$) corresponds to elimination of existential quantifiers. For instance, the projection of the set of points in $\mathbb{R}^n$ where a formula $\phi(x_1, \ldots, x_n)$ holds true to the subspace $\mathbb{R}^{n-1}$ spanned by the variables $x_1, \ldots, x_{n-1}$ are those points in $\mathbb{R}^{n-1}$ where the formula $\bar{\phi}(x_1, \ldots, x_{n-1}) \equiv \exists x_n \, \phi(x_1, \ldots, x_n)$ holds true.

Simply speaking, the basic insight of Tarski [Tar51] with subsequent simplifications by Seidenberg [Sei54], Robinson [Rob56], Łojasiewicz [Łoj64], Cohen [Coh69], and Kreisel and Krivine [KK71], that led to the development of Cylindrical Algebraic Decomposition [Col75] as a decision procedure for real-closed fields, is that the conjunction of a set of polynomial equations and inequalities partition the real space $\mathbb{R}^n$ into finitely many equivalence classes based on their sign combinations. The basic observation is that each polynomial $p \in k[X_1, \ldots, X_n]$ partitions the space into three equivalence classes based on its sign:

1. $\hat{+} := \{x \in \mathbb{R}^n \ : \ p(x) > 0\}$

2. $\hat{0} := \{x \in \mathbb{R}^n \ : \ p(x) = 0\}$

3. $\hat{-} := \{x \in \mathbb{R}^n \ : \ p(x) < 0\}$

These classes can have multiple (but only finitely many) connected components, though. Further, Tarski showed that a set of polynomials partition the real space into finitely many equivalence classes that essentially correspond to the coarsest joint refinement of all these sign relations (and possibly the relations of these polynomials on various connected components). That is, a *(natural) algebraic decomposition* of $\mathbb{R}^n$ is a partitioning of $\mathbb{R}^n$ into maximal connected regions where each of the relevant polynomials has invariant sign. Now, a satisfying assignment for a quantifier-free arithmetic formula exists if and only if the formula is true in one of those equivalence classes. Consequently, working from inside out, existential quantifiers can be replaced equivalently by a big disjunction, replacing the existentially quantified variable x by any representative point inside each of these finitely many equivalence classes. Let S be a (finite) representative system of the equivalence classes for all polynomials in the quantifier-free formula F, then:

$$\mathrm{QE}(\exists x \, F) \equiv \bigvee_{x \in S} F$$

In the following example, quantifier elimination can simply insert a range of rep-

resentative cases into the formula and evaluate the remaining arithmetic:

$$
\begin{aligned}
& \mathrm{QE}(\exists x(x > 2 \wedge x < \frac{17}{3})) \\
\equiv\ & (-\infty > 2 \wedge -\infty < \frac{17}{3}) && \text{extremal case “}x = -\infty\text{”} \\
\vee\ & (\infty > 2 \wedge \infty < \frac{17}{3}) && \text{extremal case “}x = \infty\text{”} \\
\vee\ & (2 > 2 \wedge 2 < \frac{17}{3}) && \text{border case “}x = 2\text{”} \\
\vee\ & (\frac{17}{3} > 2 \wedge \frac{17}{3} < \frac{17}{3}) && \text{border case “}x = \frac{17}{3}\text{”} \\
\vee\ & (\frac{2 + \frac{17}{3}}{2} > 2 \wedge \frac{2 + \frac{17}{3}}{2} < \frac{17}{3}) && \text{intermediate case “}x = \frac{2 + \frac{17}{3}}{2}\text{”} \\
\equiv\ & \mathit{true}
\end{aligned}
$$

Universal quantification can then be handled using duality:

$$\mathrm{QE}(\forall x\, F) \equiv \neg\, \mathrm{QE}(\exists x\, \neg F)$$

For multiple quantifiers, this procedure can be used recursively by applying quantifier elimination from inside out, i.e., starting from inner quantifiers. The final result for a closed formula gives a propositional formula without variables, which is decidable by evaluating the remaining arithmetic expressions with concrete numbers. Practical quantifier elimination procedures for real arithmetic improve on this theoretical decision procedure, e.g., by minimising the number of required equivalence classes.

KeYmaera integrates Mathematica by Wolfram Research [Wol05] and QEPCAD B [Bro03] as alternative implementations of quantifier elimination procedures.

A.2.3. Nullstellensatz and Gröbner Bases

Gröbner bases [Buc65, BW98, Mor05, CLO92] can be used as a sound but incomplete procedure for proving validity of formulas in the universal fragment of equational first-order real arithmetic. This approach is, in fact, not specific for real arithmetic but also applies for other fields. Gröbner bases have been introduced by Bruno Buchberger [Buc65] as a systematic theory and algorithm for symbolic computations in factor rings of multivariate polynomial rings. Gröbner basis algorithms can be considered as a multivariate joint generalisation of the Euclidean algorithm for computing greatest common divisors in univariate polynomial rings and of Gaussian elimination for solving linear equation systems. An analogous concept for local rings, called standard bases, was developed independently by Heisuke Hironaka in 1964.

Preliminaries First we briefly recapitulate some basic notions from algebra [Lan78, Bou89, Bou72, Eis99, Mor05]. We do not always give the most general definition of these notions but restrict our attention to what we actually need in our context. A *ring* R is an algebraic structure that is an Abelian group with respect to addition and a (commutative) semigroup with respect to multiplication where multiplication distributes over addition, i.e., for all $a, b, c \in R$:

$$\begin{aligned} a(b+c) &= ab + ac \\ (a+b)c &= ac + bc \end{aligned}$$

A *field* k is a ring such that $1 \neq 0$ and all elements $x \in k \setminus \{0\}$ have a multiplicative inverse, i.e., a $y \in k$ such that $xy = 1$. A subset $I \subseteq R$ is an *ideal* of a ring R, denoted as $I \trianglelefteq R$, iff I is a subgroup of the additive group of R and

$$rx \in I, \text{ for all } x \in I, r \in R \ .$$

The ideal *generated* by a set $G \subseteq R$ is the smallest ideal $I \trianglelefteq R$ containing G. In that case, G is called a *generating system* of I.

For a field k, the set $k[X_1, \ldots, X_n]$ of *multivariate polynomials* forms a ring and is defined as the free commutative and associative algebra over the indeterminates $X_1, \ldots, X_n$.

The notions of Gröbner bases and polynomial reductions are relative to an admissible monomial order, which also determines the *leading term* in multivariate polynomials. Here, we simply assume some fixed admissible order and refer to the literature on Gröbner bases [BW98, Mor05, CLO92, Buc65] for details on monomial orders, their admissibility conditions and their properties. As a multivariate generalisation of the Euclidean algorithm we need multivariate polynomial division:

A.5 Definition (Reduction). Let $f, g \in k[X_1, \ldots, X_n]$ be polynomials and let l be the leading term of g. If $a_{i_1,\ldots,i_n} X_1^{i_1} \cdots X_n^{i_n}$ is some (e.g., the largest) term of f which is divisible by l, then the following polynomial is called an *elementary reduction* of f by g:

$$f - \frac{a_{i_1,\ldots,i_n} X_1^{i_1} \cdots X_n^{i_n}}{l} g$$

Let $G \subset k[X_1, \ldots, X_n]$ be a set of polynomials. Polynomial f is called *reduced* with respect to G iff no elementary reduction of f with respect to some $g \in G$ is possible. Any polynomial obtained from f by repeated elementary reduction with respect to any $g \in G$ is called *reduction* (or *remainder*) of f by G if it is reduced with respect to G and is denoted by $\mathrm{red}_G f$.

The theory of Gröbner bases characterises under which circumstances reduction produces unique remainders, which is not generally so in multivariate polynomial rings.

A.6 Definition (Gröbner basis). A finite generating system G of $I \trianglelefteq k[X_1, \ldots, X_n]$, i.e., an ideal I of a polynomial ring, is called *Gröbner basis* iff any of the following (equivalent) conditions holds:

1. Reduction with respect to G gives 0 for any $p \in I$.
2. $\mathrm{red}_G\, p = 0$ iff $p \in I$.
3. Reduction with respect to G gives a unique remainder.
4. The polynomials that are reduced with respect to G form representatives of the factor ring $k[X_1, \ldots, X_n]/I$.
5. The leading ideal of I, i.e., the ideal generated by leading terms of polynomials of I, is generated by the leading terms of G.

A Gröbner basis G is *reduced* if all $g \in G$ are reduced with respect to $G \setminus \{g\}$.

We can assume Gröbner basis G to contain normalised polynomials only, i.e., where the leading coefficient of their leading terms is 1.

The most important result about Gröbner bases is that every ideal of a polynomial ring $k[X_1, \ldots, X_n]$ over a field k in finitely many variables has a unique reduced Gröbner basis (with leading coefficient 1) that can be computed effectively by the Buchberger algorithm [Buc65] or improvements thereof [BW98, Mor05, CLO92, Fau99, Fau02]. Thus, Gröbner bases give an effective version of Hilbert's basis theorem:

A.7 Theorem (Hilbert's basis theorem). *Every ideal in the ring $k[X_1, \ldots, X_n]$ of multivariate polynomials over a field k is finitely generated.*

Using Gröbner Basis Eliminations We introduce proof rules that use Gröbner basis reductions for handling fragments of equational universal arithmetic. Assume we have a sequent (using rewriting to normalise equations, inequations, and inequalities as necessary):

$$\Gamma, g_1 = \tilde{g}_1, \ldots, g_n = \tilde{g}_n \vdash f_1 = h_1, \ldots, f_e = h_e, f_{e+1} \geq h_{e+1}, \ldots, f_m \geq h_m, \Delta \quad \text{(A.1)}$$

Let G be a Gröbner basis of the ideal generated by the $g_i - \tilde{g}_i$, i.e., of the ideal

$$(g_1 - \tilde{g}_1, \ldots, g_n - \tilde{g}_n) \ . \quad \text{(A.2)}$$

If the reduction with respect to G of some f_i equals the reduction of the corresponding h_i with respect to G, i.e., $\mathrm{red}_G\, f_i = \mathrm{red}_G\, h_i$, then the sequent (A.1) is valid and can be closed, as summarised in rule A2 of Figure A.4.

Similarly, we introduce rule A1 that converts (weak) inequalities to equations by exploiting that a real number is positive iff it is a square.

$$\text{(A1)}\quad \frac{f - g = z^2 \vdash}{f \geq g \vdash}$$

$$\text{(A2)}\quad \frac{*}{g_1 = \tilde{g}_1, \ldots, g_n = \tilde{g}_n \vdash f_1 = h_1, \ldots, f_e = h_e, f_{e+1} \geq h_{e+1}, \ldots, f_m \geq h_m}$$

For A1, z is a fresh (state) variable. A2 is applicable iff, $\mathrm{red}_G f_i = \mathrm{red}_G h_i$ for some i and (the) Gröbner basis G of (A.2).

Figure A.4.: Rule schemata of Gröbner calculus rules

A.8 Proposition (Soundness of Gröbner basis rules). *The rules in Figure A.4 are sound.*

Proof. The rules are locally sound.

A1 The local soundness of A1 is a simple consequence of the following equivalence for reals: $f \geq 0 \equiv \exists z\, f = z^2$. Note that this equivalence is wrong when restricting quantifiers over the integers or rationals. Using the equivalence and the soundness of F2 from Theorem 2.15, soundness can be obtained easily from the following derivation when using the state variable z, that is implicitly quantified universally in the sequent, in place of the Skolem symbol s:

$$\cfrac{\cfrac{f - g = s^2 \vdash}{\exists z\, f - g = z^2 \vdash}\,\text{F2}}{f \geq g \vdash}$$

A2 Let $i \in \{1, \ldots, m\}$ according to the applicability condition of A2, in particular, $\mathrm{red}_G f_i = \mathrm{red}_G h_i$. Suppose the premiss was false in some state ν, then $\nu \models g_1 = \tilde{g}_1 \wedge \cdots \wedge g_n = \tilde{g}_n \wedge f_i \neq h_i$. Thus, $\nu \models g = 0$ for all $g \in G$ (using the notions of algebraic geometry, this would correspond to ν being in the algebraic variety of G). Consequently, $\nu \models g = 0$ for all polynomials g in the ideal (G) of G. As a consequence of the applicability condition, we have $\mathrm{red}_G(f_i - h_i) = 0$, which, by Definition A.6, implies that $f_i - h_i$ is in the ideal of G. In combination, we have $\nu \models v_i - h_i = 0$, hence $\nu \models f_i = h_i$, which is a contradiction. □

As a special case of A2, we assume the (invalid) equation $1 = 0$ to occur as $f_1 = h_1$ if no other equation in (nonstrict) inequation occurs in the succedent. This case directly corresponds to the weak form of Hilbert's Nullstellensatz:

A.9 Theorem (Hilbert's Nullstellensatz [Lan78]). *Let k be an algebraically closed field and $I \trianglelefteq k[X_1, \ldots, X_n]$ an ideal. Then the* algebraic variety

$$V(I) := \{x \in k^n \ :\ p(x) = 0 \text{ for all } p \in I\}$$

of I is empty iff $I = k[X_1, \ldots, X_n]$.

In particular, sequent $g_1 = \tilde{g}_1, \ldots, g_n = \tilde{g}_n \vdash$ is valid iff, $g_1 = \tilde{g}_1 \wedge \cdots \wedge g_n = \tilde{g}_n$ is unsatisfiable over $\mathbb{R}$. By Theorem A.9, the sequent is even unsatisfiable over $\mathbb{C}$ iff the Gröbner basis of (A.2) is $\{1\}$ (up to multiplication with nonzero constants).

Consequently, the Gröbner basis approach gives a sound but incomplete overapproximation. To see why Gröbner bases are incomplete for real arithmetic, consider the following. Gröbner bases are a general approach for polynomial rings over fields (or even rings). They do not take into account the special properties of the reals. For instance, the sequent $x^2 = -1 \vdash$ is valid, i.e., the formula $x^2 = -1$ is unsatisfiable over $\mathbb{R}$, but the Gröbner basis of $x^2 + 1$ is $\{x^2 + 1\}$ and, in fact, $x^2 = -1$ is satisfiable over $\mathbb{C}$ but not over $\mathbb{R}$.

KeYmaera integrates Mathematica by Wolfram Research [Wol05] and the Orbital library developed by the author as alternative implementations of Gröbner basis algorithms.

A.2.4. Positivstellensatz and Semidefinite Programming

The Positivstellensatz for real-closed fields [Ste73, BCR98], which has been introduced by Stengle along with a Nullstellensatz for real-closed-fields [Ste73], can be used as a sound procedure for proving formulas in the universal fragment of first-order real arithmetic. The Positivstellensatz has recently been exploited in combination with relaxations from semidefinite programming [Par03, Har07].

The *multiplicative monoid* generated by $H \subseteq R[X_1, \ldots, X_n]$ is the set of finite products of elements of H (including the empty product 1). The *cone* generated by a set $F \subseteq R[X_1, \ldots, X_n]$ is the subsemiring of $R[X_1, \ldots, X_n]$ generated by F and arbitrary squares, i.e., the smallest set containing F, squares p^2 of arbitrary polynomials $p \in R[X_1, \ldots, X_n]$ that is closed under addition and multiplication. For more computational representations of cones and ideals, we refer to [Par03, BCR98].

A.10 Theorem (Positivstellensatz [Ste73] for real-closed fields). *Let R be a real-closed field (e.g., $R = \mathbb{R}$) and F, G, H finite subsets of $R[X_1, \ldots, X_n]$. Let C denote the cone generated by F, let I be the ideal generated by G, and let M be the multiplicative monoid generated by H. Then*

$$\{x \in R^n \ : \ f(x) \geq 0 \text{ for all } f \in F, g(x) = 0 \text{ for all } g \in G, h(x) \neq 0 \text{ for all } h \in H\}$$

is empty iff

$$\text{there are } f \in C, g \in I, h \in M \text{ such that } f + g + h^2 = 0 \ .$$

The polynomials f, g, h are polynomial infeasibility witnesses. For bounded degree, witnesses f, g, h can be searched for using numerical semidefinite programming [Par03] by parameterising the resulting polynomials. As (theoretical) degree bounds exist for the certificate polynomials f, g, h, the Positivstellensatz yields a

decision procedure. These bounds are at least triply exponential, though [Par03]. Thus, the approach advocated by Parrilo [Par03] is to increase the bound successively and solve the existence of bounded degree witnesses due to the Positivstellensatz by semidefinite programming [BV04].

As a simple corollary to Theorem A.10 we have the following sound proof rule.

A.11 Corollary. *The rule in Figure A.5 is sound.*

$$\text{(A3)}\quad \frac{*}{f_1 \geq \tilde{f}_1, \ldots, f_m \geq \tilde{f}_m, g_1 = \tilde{g}_1, \ldots, g_n = \tilde{g}_n, h_1 \neq \tilde{h}_1, \ldots, h_l \neq \tilde{h}_l \vdash}$$

A3 is applicable iff $f + g + h^2 = 0$ for some polynomial f of the cone generated by $\{f_1 - \tilde{f}_1, \ldots, f_m - \tilde{f}_m\}$, a polynomial g of the ideal (A.2), and a polynomial h of the multiplicative magma generated by $\{h_1 - \tilde{h}_1, \ldots, h_l - \tilde{h}_l\}$.

Figure A.5.: Rule schemata of Positivstellensatz calculus rules

While both the rules in Figure A.4 and Figure A.5 assume simple rewriting to normalise the sequent, e.g., by resolving negated operators or $f \leq g$ to $g \geq f$, observe that—when using propositional rules to split branches as necessary—all quantifier-free formulas of real arithmetic can be converted equivalently into the form required by A3 but not into that required by A2 (for $\Gamma = \Delta = \emptyset$).

KeYmaera supports a preliminary integration with CSDP [Bor99] as an implementation of semidefinite programming algorithms.

A.3. Discussion

As usual, it is a nontrivial task to ensure that the soundness that has been proven for a calculus inherits to soundness of a tool implementing this calculus. Despite our simple and concise calculi from Part I, the tool KeYmaera already qualifies as a nontrivial piece of code, because of its numerous features totalling to around 22,000 SLOC[1] on top of the KeY base system with 180,000 SLOC (not all parts of the base system are used in the context of hybrid systems, though). In this section, we discuss the relationship of the soundness of our calculi and the soundness of KeYmaera. We refer to [BK06] for a general discussion of verification for verification systems.

[1]Source lines of code, not counting comments, estimated with SLOCCount from `www.dwheeler.com/sloccount/`

Deductive Kernels One advantage for deductive verification approaches is that they can have a comparably small trusted deductive kernel. The basic rationale underlying also tools like LCF [Mil72], HOL [GM93], or Isabelle/HOL [NPW02] is that programming bugs in implementations that affect completeness are much less fatal than bugs that would make the tool unsound. Essentially, when a verification tool like KeYmaera only allows to prove formulas by applying proof rules to them, it is sufficient for soundness purposes to ensure the correct implementation of these calculus rules and the proof rule application mechanism. In particular, this removes all sophisticated implementations of proof strategies from the trusted computing base. See, for instance, higher-order logic proof systems like Isabelle/HOL [NPW02] for tools that follow this approach consequently.

In KeYmaera, however, there are some additional pitfalls. Most of the rules of the calculi for our logics from Part I can be written as schematic rules (called *taclets* [BGH+04]) in the KeY system, so that the careful soundness proofs carry over from the $\mathsf{d}\mathcal{L}$/DAL/dTL calculi to the implementation KeYmaera. Yet even this already requires the (long-tested) rule application mechanism of KeY [BGH+07, BHS07] to be correct. Other rules like F3 and F6 from Figure 2.5, however, cannot be written as a KeY taclet at all but have to be implemented in Java. Taclet notations for some of the rules also require metaoperators that are implemented in Java, e.g., for obtaining the solution of differential equations in D11–D12. To improve the quality of these Java implementations, in addition to software engineering techniques like code reviews and testing, we work with runtime assertions to try to ensure that the results KeYmaera produces during the current verification run are correct. For instance, the Java rules for F3 and F6 can assert at runtime that their goal (conclusion) is actually a substitution instance of the sub-goals (premisses) that they constructed, thereby ensuring in retrospect that their formula transformation was justified by Lemma 2.13, see further explanations in Section 2.5.2. These runtime checks eliminate most of the corresponding rule implementations in KeYmaera from the trusted basis, leaving essentially only the implementation of substitutions and quantifier rearrangements in the trusted deductive kernel.

External Blackbox Procedures The soundness problem is deeper, though, because, for performance reasons, KeYmaera calls external decision procedures following the cooperation scheme in Chapter 5, which use the techniques described in Section A.2. Thus, soundness of KeYmaera generally requires the external decision procedures to be implemented correctly. We discuss the respective external procedures and means for removing them from the required trusted computing base in the sequel.

Differential Equation Handling For the implementation of differential equation handling rules like D11–D12 from Figure 2.5, it is easy to check at runtime by simple

symbolic differentiation and symbolic computations in polynomial rings whether a solution produced by a (complicated) procedure for solving differential equations actually is a solution, thereby eliminating the differential equation solver (Mathematica and Orbital library) from the trusted computing base completely. Similar observations hold for differential induction rules G5–G6 from Figure 3.3, which only require symbolic differentation and symbolic polynomial computations.

Positivstellensatz and Semidefinite Programming A pleasant property of the Positivstellensatz Theorem A.10 and the approach in Section A.2.4 is that it produces a witness (the polynomials f, g, h) for the validity of a formula (which corresponds to the closing of the branch by rule A3). Once the witness has been found, it is checkable by simple computations in the polynomial ring to determine whether $f + g + h^2 = 0$ by comparing the coefficients. Thus, complicated numerical semidefinite programming tools [BV04] do not need to be part of the trusted computing base concerning soundness.

Semialgebraic Geometry and Cylindrical Algebraic Decomposition Most quantifier elimination procedures follow cylindrical algebraic decomposition [Col75] or partial cylindrical algebraic decomposition [CH91], which are efficient but quite intricate procedures. Unlike approaches using the Positivstellensatz, general quantifier elimination, unfortunately, does not produce simple checkable certificates.

In order to remove the implementations of quantifier elimination procedures in Mathematica from the trusted computing base, diversification (i.e., crossvalidating results by multiple background solvers) only is a partial answer, because the solvers might still agree on the same wrong result. Further, crossvalidation limits the overall performance to the worst-of-breed rather than best-of-breed running times. As a more fundamental approach, we can use verified implementations, e.g., the verified quantifier elimination procedure for linear real arithmetic by Nipkow [Nip08] in an executable fragment of Isabelle/HOL. Another approach is to use nonverified but proof-producing implementation of general quantifier elimination by McLaughlin and Harrison [MH05].

Unfortunately, the practical performance achieved by those prototype implementations is not yet sufficient for larger case studies. A good compromise is to use the paradigm of reverification: The proof search procedures IBC and IIO generate several calls to the quantifier elimination procedure to find a proof, but only those in the final proof are soundness-critical, see Chapter 5 for details. Thus, for soundness, it is sufficient to use a fast and possibly untrusted implementation of QE during the proof search and to reverify the final proof in a proof checker with a verified or proof-producing QE implementation [MH05, Nip08]. For this purpose, the iterative background closure strategy from Chapter 5 is especially useful, be-

cause it iteratively identifies the sweetspot for quantifier elimination during the proof search.

Nullstellensatz and Gröbner Bases While Gröbner basis approaches do not have as simple witnesses as Positivstellensatz approaches, their working principle is strictly based on appropriate symbolic computations in the ring of polynomials. Consequently, these polynomial reductions can be carried out from a small set of rewrite rules within a logic. For the case of integer arithmetic, this has already been implemented in KeY [Rüm07], which can be generalised to real arithmetic.

Further, for performance reasons, the actual polynomial reductions that are performed during the construction of the Gröbner basis (Definition A.6) and the reductions with respect to that basis during the application of A2 can be tracked within a fast Gröbner basis implementation and replayed within the logic to produce a reliable proof script that only requires limited symbolic computation power within the trusted prover kernel. Typically, only a few of the polynomial reductions during the Gröbner basis constructions and reductions are finally required for proving an equality.

Simplification KeYmaera can call simplification procedures that simplify mathematical expressions to improve readability. For instance, an external simplification procedure that simplifies x/x to 1 would be unsound, however, for the compactified domain $\mathbb{R} \cup \{-\infty, +\infty\}$, because ∞/∞ is undefined and not identical to 1. Likewise, this simplification could be unsound depending on whether x is allowed to assume 0. In our case, the above simplification is in fact sound for the domain of reals ($\mathbb{R}$) when adopting our convention from Sections 2.2.3 and 3.2.1 that any formula containing x/x is understood to mean that $x \neq 0$ holds in addition. Generally, however, arbitrary mathematical simplification algorithms might perform non-equivalent transformations and are thus disabled in KeYmaera by default.

Summary The level of formal ensurance of the correct functioning of the implementation of KeYmaera is, unfortunately, not quite comparable to that in systems like LCF [Mil72], HOL [GM93], or Isabelle/HOL [NPW02]. The built-in proof rule language in KeY allows to state most proof rules of the $\mathsf{d}\mathcal{L}$ calculus primarily as taclets. Ensuring a correct implementation of the remaining code is not quite trivial but simplified by using a deductive kernel with a smaller trusted basis and by using runtime assertions that check for coherence of the current proof rule application with its formal prerequisites. Still, the simplicity of the proof rules of our calculi should make it possible to come up with a correct implementation.

The most tricky part, however, is that, for performance reasons, KeYmaera relies on tuned external decision procedures to ensure scalability to our larger case studies. These procedures can be eliminated from the trusted verification base by

checking witnesses when working with solutions of differential equations, the Positivstellensatz and Göbner basis rules. For full quantifier elimination over real-closed fields, verified or proof-generating implementations can be helpful.

A.4. Performance Measurements

Unless otherwise indicated, the measurements in this thesis have been performed on a Dual-Core AMD Opteron™ 1218 Processor (F2 stepping) running at 2.6GHz clock speed with 64kb instruction cache, 64kb first level data cache, and 1MB second level cache per core sharing 4GB of DDR2-667 main memory with CAS latency 5. KeYmaera has been run on a Java™ SE Virtual Machine, build 1.6.0_06-b02, with a HotSpot™ engine in mixed mode on a Gentoo Linux with a 2.6.25-g SMP kernel for x86/64bit. Measurements are further based on the Mathematica 6.0.2 kernel.

Due to several effects including JAVA dynamic class loading and caching effects, the timing measurements are not necessarily always statistically significant. Rather, the timing information is intended to give a feeling for magnitudal differences. The measurements have been repeated to ensure reproducibility of results, though. While the JAVA Virtual Machine itself already contributes to nondeterministic effects that result from dynamic class loading, garbage collection and caching, the implementation of the KeYmaera system itself is subject to nondeterminism as well, including the proof strategies, timing effects of iterative background closures, nondeterminstic term orderings based on memory references etc. Finally, computational backends allow for variations in overall performance.

Still, qualitative performance differences can be read off from the measurements. For instance, it makes quite a remarkably dramatic practical difference whether a strategy is able to prove a case study within only 100s or cannot even come up with an answer after 5 hours.

Appendix B.

Hybrid Automata

Contents

Synopsis

To formally relate the notions of hybrid programs and hybrid automata, we show that hybrid automata can be embedded canonically into hybrid programs. Further, reachability in hybrid automata directly corresponds to satisfying models of associated $\mathsf{d}\mathcal{L}$ formulas, safety corresponds to validity of the $\mathsf{d}\mathcal{L}$ formulas.

B.1. Hybrid Automata

Even though hybrid automata are the most standard notation for hybrid systems, there are several slightly different notions of hybrid automata or automata-based models for hybrid systems [Tav87, ACHH92, NOSY92, ACH+95, Bra95b, Hen96, AHH96, BBM98, LPY99, DN00, PAM+05, DHO06]. We follow the notion of hybrid automata from Henzinger [Hen96] most closely, with corresponding care on actual definability of the relations as in other approaches [Frä99, LPY99, PAM+05, PC07] and with invariants that are required to hold at all times following [ACH+95, DHO06, PC07].

Hybrid automata are graph models with two kinds of transitions: discrete jumps in the state space caused by mode switches (edges), and continuous evolution along flows within a mode (vertex).

B.1 Definition (Hybrid automata). A *hybrid automaton A* consists of

- a continuous state space $\mathbb{R}^n$;
- a finite directed graph (*control graph*) with vertices Q (as *modes*) and edges E (*control switches*);
- flow conditions $\mathit{flow}_q \subseteq \mathbb{R}^n \times \mathbb{R}^n$ that determine the relationship of the continuous state $x \in \mathbb{R}^n$ and its time-derivative $x' \in \mathbb{R}^n$ during continuous evolution in mode $q \in Q$;
- invariant conditions $\mathit{inv}_q \subseteq \mathbb{R}^n$ that have to be true while in mode $q \in Q$;
- jump relations $\mathit{jump}_e \subseteq \mathbb{R}^n \times \mathbb{R}^n$ that determine the new value of the continuous state $x \in \mathbb{R}^n$ depending on its old value when following edge $e \in E$;

where jump_e and inv_q are definable in first-order real arithmetic [Tar51] and additional restrictions apply for flow_q depending on the class of hybrid automata.

Typically, the jump relation jump_e is given as a conjunction of transition guards $\mathit{guard}_e \subseteq \mathbb{R}^n$, which determine from which states an edge can be taken, and variable resets $\mathit{reset}_e \subseteq \mathbb{R}^n \times \mathbb{R}^n$, which adjust the state value to its new value. In most cases, the reset relation is specified by a list of assignments $x_1 := \theta_1, \ldots x_n := \theta_n$, which correspond to a discrete jump set of $\mathsf{d}\mathcal{L}$ (Definition 2.3). Further, flow conditions are usually just specified by a set of differential equations $x_1' = \theta_1, \ldots, x_n' = \theta_n$. See Figure 1.3 on page 4 for an example.

Although often neglected, definability of the constituent relations of hybrid automata, e.g., in first-order real arithmetic, is a crucial prerequisite for dealing with any state reachability question. In a hybrid automaton that uses the Mandelbrot set as invariant, it would already be undecidable whether a state $x \in \mathbb{R}^n$ satisfies the invariant of the current mode, even in the strong computational model of real Turing machines by Blum et al. [BCSS98].

B.2 Definition (Transition semantics of hybrid automata). The *transition system* of a hybrid automaton A is a transition relation $\curvearrowright$ defined as follows

- the state space is defined as $S := \{(q, x) \in Q \times \mathbb{R}^n \ : \ x \in \mathit{inv}_q\}$;
- the transition relation is defined by $\curvearrowright := \bigcup_{e \in E} \overset{e}{\curvearrowright} \cup \bigcup_{q \in Q} \overset{q}{\curvearrowright}$ where
  1. $(q, x) \overset{e}{\curvearrowright} (\tilde{q}, \tilde{x})$ iff $(x, \tilde{x}) \in \mathit{jump}_e$ and $e \in E$ is an edge from $q \in Q$ to $\tilde{q} \in Q$ (*discrete transition*).
  2. $(q, x) \overset{q}{\curvearrowright} (q, \tilde{x})$ iff there is a function $f : [0, r] \to \mathbb{R}^n$ with time-derivative $f' : (0, r) \to \mathbb{R}^n$ with $f(0) = x, f(r) = \tilde{x}$ and $(f(\zeta), f'(\zeta)) \in \mathit{flow}_q$ at each $\zeta \in (0, r)$. Further, $f(\zeta) \in \mathit{inv}_q$ for each $\zeta \in [0, r]$ (*continuous transition*).

State $\sigma \in S$ is *reachable* from state $\sigma_0 \in S$, denoted by $\sigma_0 \curvearrowright^* \sigma$, iff, for some $n \in \mathbb{N}$, there is a sequence of states $\sigma_1, \sigma_2, \ldots, \sigma_n = \sigma \in S$ such that $\sigma_{i-1} \curvearrowright \sigma_i$ for $1 \leq i \leq n$.

Most often, the semantics of hybrid automata is further restricted to nonzeno traces [Hen96, DHO06, DN00].

B.2. Embedding Hybrid Automata into Hybrid Programs

A hybrid automaton like in Figure 1.3 on page 4 can be represented faithfully as the hybrid program in Figure 2.1 on page 21. More generally, we show that it is always possible to represent hybrid automata as hybrid programs, thus showing that hybrid automata can be embedded faithfully into $\mathsf{d}\mathcal{L}$.

B.3 Proposition (Hybrid automata embedding). *There is an effective mapping ι from hybrid automata to hybrid programs / DA-programs such that the following diagram commutes:*

$$\begin{array}{ccc} HA & \xhookrightarrow{\ \iota\ } & \mathrm{HP}(\Sigma, V) \\ \big\downarrow{\scriptstyle \curvearrowright^*} & \circlearrowright & \big\downarrow{\scriptstyle \rho} \\ S^2 & \rightleftarrows & \mathrm{Sta}(\Sigma)^2 \end{array}$$

Proof. Let $\Sigma = \{q, x_1, \ldots, x_n, {x_1}^+, \ldots, {x_n}^+\}$, using vectorial notation x for the vector $(x_1, \ldots, x_n)$ and x^+ for $({x_1}^+, \ldots, {x_n}^+)$. We define ι as the function that maps hybrid automaton A to the following HP:

$$\begin{array}{l} \big(?q = q_i;\ \mathit{flow}_{q_i}(x, x')\ \&\ \mathit{inv}_{q_1} \\ \quad \cup\ ?q = q_i;\ (x^+ := *;\ ?\mathit{jump}_e(x, x^+);\ x := x^+);\ ?\mathit{inv}_{q_j};\ q := q_j \\ \quad \cup\ \ldots \\ \big)^* \end{array}$$

The respective lines in this HP are subject to a choice for each mode q_i or each edge e from some state q_i to some state q_j. Let α^* denote this program $\iota(A)$.

States of the hybrid automaton A and states of its HP $\iota(A)$ immediately correspond to each other using the bijection $\Phi : S \to \mathrm{Sta}(\Sigma)$ that maps $(\tilde{q}, \tilde{x}) \in S$ to the state ν that is defined as $\nu(q) = \tilde{q}$ and $\nu(x_i) = \tilde{x}_i$ for $1 \leq i \leq n$. Observe that Φ is a bijection up to forgetful projections of internal variables x^+. In the following, we use the state identification Φ implicitly as necessary to simplify the notation.

We have to show that the diagram commutes, that is, $\curvearrowright^* = \rho \circ \iota$ (up to identification of states by Φ, i.e., $\Phi \circ \curvearrowright^* = \rho \circ \iota$).

"$\subseteq$" Let $\sigma_0 \curvearrowright^* \sigma$, that is, let $n \in \mathbb{N}$ and $\sigma_1, \sigma_2, \ldots, \sigma_n = \sigma \in S$ such that $\sigma_{i-1} \curvearrowright \sigma_i$ for all $1 \leq i \leq n$. The proof is by induction on n.

IA $n = 0$ then $(\sigma_0, \sigma) \in \rho(\alpha^*)$ using zero repetitions.

IS By induction hypothesis, we can assume that $(\sigma_0, \sigma_{n-1}) \in \rho(\alpha^*)$. We have to show $(\sigma_{n-1}, \sigma_n) \in \rho(\alpha)$, thereby implying $(\sigma_0, \sigma_n) \in \rho(\alpha^*)$.

Consider the case where the last transition $\sigma_{n-1} \curvearrowright \sigma_n$ is a continuous transition in mode q_i of some duration $r \geq 0$. Then, up to identification by Φ, there is a state flow $\varphi : [0, r] \to \mathrm{Sta}(\Sigma)$ with $\varphi(0) = \sigma_{n-1}, \varphi(r) = \sigma_n$ and $\varphi \models \mathit{flow}_{q_i} \wedge \mathit{inv}_{q_i}$. Thus, α can copy the transition as $(\sigma_{n-1}, \sigma_n) \in \rho(\alpha)$ using the choice $?q = q_i; \mathit{flow}_{q_i}(x, x') \,\&\, \mathit{inv}_{q_i}$. The test succeeds, because $\Phi(\sigma_{n-1})(q) = q_i$.

Consider the case where the last transition $\sigma_{n-1} \curvearrowright \sigma_n$ is a discrete transition from mode q_i to q_j along edge e Then $(\sigma_{n-1}, \sigma) \in \mathit{jump}_e$. Thus, by choosing the values of σ_n for x^+, we have that $(\sigma_{n-1}, \sigma_n) \in \rho(\alpha)$ by the choice $?q = q_i; (x^+ := *; ?\mathit{jump}_e(x, x^+); x := x^+); ?\mathit{inv}_{q_j}; q := q_j$.

"$\supseteq$" Let $(\sigma_0, \sigma_n) \in \rho(\alpha^*)$ following n repetitions of α, i.e., let $\sigma_1, \ldots, \sigma_{n-1} \in \mathrm{Sta}(\Sigma)$ such that $(\sigma_{i-1}, \sigma_i) \in \rho(\alpha)$ for all $1 \leq i \leq n$. The proof is by induction on n.

IA For $n = 0$, there is nothing to show.

IS By induction hypothesis, we can assume that $\sigma_{i-1} \curvearrowright \sigma_i$ for all $1 \leq i < n$. We have to show that $\sigma_{n-1} \curvearrowright \sigma_n$, thereby showing that $\sigma_0 \curvearrowright^* \sigma_n$. If $q_i := \sigma_{n-1}(q) = \sigma_n(q)$, then it is easy to see from the structure of α that $(\sigma_{n-1}, \sigma_n) \in \rho(?q = q_i; \mathit{flow}_{q_i}(x, x') \,\&\, \mathit{inv}_{q_i})$. Thus, $\sigma_{n-1} \overset{q_i}{\curvearrowright} \sigma_n$ by a continuous transition.

If, however, $q_i := \sigma_{n-1}(q)$, $q_j = \sigma_n(q)$, then it is easy to see that

$$(\sigma_{n-1}, \sigma_n) \in \rho(?q = q_i; (x^+ := *; ?\mathit{jump}_e(x, x^+); x := x^+); ?\mathit{inv}_{q_j}; q := q_j)$$

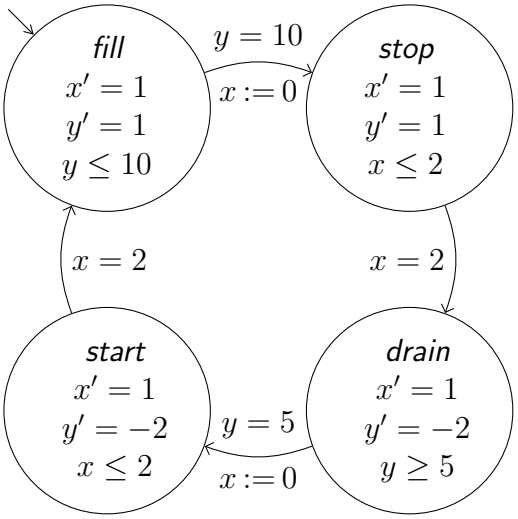
according to a line of α that originates from some edge e from q_i to q_j. Thus, $(\sigma_{n-1}, \sigma_n) \in \mathit{jump}_e$ and $\sigma_n \models \mathit{inv}_{q_j}$, hence, $\sigma_{n-1} \overset{e}{\curvearrowright} \sigma_n$ by a discrete transition. □

B.4 Corollary. *There is an effective mapping from safety properties of hybrid automata to* $\mathsf{d}\mathcal{L}$*/ DA-program formulas such that the hybrid automaton A, starting in initial mode q_0, safely remains in the region $F \in \mathrm{Fml}_{\mathrm{FOL}}(\Sigma, V)$ if and only if the corresponding* $\mathsf{d}\mathcal{L}$ *formula is valid.*

Proof. Let α^* the the HP $\iota(A)$ belonging to A according to Proposition B.3. Then, when starting A in initial mode q_0, it safely remains in the region F iff the following formula is valid

$$q = q_0 \wedge \mathit{inv}_{q_0} \to [\alpha^*]F$$ □

B.1 Example (Water tank). Consider the classical simple water tank example, which regulates water level y between 1 and 12 by filling or emptying the water tank. The control in the hybrid automaton of Figure B.1a further uses a clock variable x to model delayed reactions of pumps or valves.



B.1a: Hybrid automaton

$$
\begin{aligned}
& q = \mathit{fill} \rightarrow [(\\
& \quad (?q = \mathit{fill};\ x' = 1, y' = 1 \,\&\, y \leq 10) \\
& \cup (?q = \mathit{fill} \wedge y = 10;\ x := 0;\ q := \mathit{stop}) \\
& \cup (?q = \mathit{stop};\ x' = 1, y' = 1 \,\&\, x \leq 2) \\
& \cup (?q = \mathit{stop} \wedge x = 2;\ q := \mathit{drain}) \\
& \cup (?q = \mathit{drain};\ x' = 1, y' = -2 \,\&\, y \geq 5) \\
& \cup (?q = \mathit{drain} \wedge y = 5; x := 0; q := \mathit{start}) \\
& \cup (?q = \mathit{start};\ x' = 1, y' = -2 \,\&\, x \leq 2) \\
& \cup (?q = \mathit{start} \wedge x = 2;\ q := \mathit{fill}) \\
&)^*] \, (1 \leq y \wedge y \leq 12)
\end{aligned}
$$

B.1b: Hybrid program

Figure B.1.: Water tank

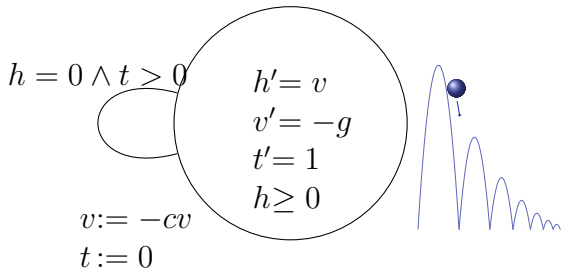
Figure B.1b shows a corresponding representation of the hybrid automaton in Figure B.1a as a hybrid program. Each line of the hybrid program corresponds to a discrete or continuous transition of the water tank hybrid automaton. The constants *fill*, *stop*, *drain*, *start* are pairwise different. The water tank is provable with the following state-dependent invariant:

$$1 \leq y \leq 12 \wedge (q = \mathit{start} \rightarrow y \geq 5 - 2x) \wedge (q = \mathit{stop} \rightarrow y \leq 10 + x)$$

The transformation in Proposition B.3 is a canonical embedding but hybrid programs allow for more flexible programming structures with which more natural characterisations of the system behaviour can be obtained.

B.2 Example (Parametric bouncing ball). Consider the well-known bouncing ball example [EJSL99]. A ball falls from height h and bounces back from the ground (which corresponds to $h = 0$) after an elastic deformation. The current speed of the ball is denoted by v, and t is a clock measuring the falling time. We assume an arbitrary positive gravity force g and that the ball loses energy according to a damping factor $0 \leq c < 1$. Figure B.2 depicts the hybrid automaton, an illustration of the system dynamics, and a representation of the system as a hybrid program.

The ball loses energy at every bounce, thus the ball never bounces higher than the initial height. This can be expressed by the safety property $0 \leq h \leq H$, where H



$$\texttt{Ball} \equiv \big(\; h' = v, v' = -g, t' = 1 \,\&\, h \geq 0; \; \text{if } (h = 0 \;\wedge\; t > 0) \text{ then } v := -cv;\; t := 0 \; \text{fi} \big)^*$$

Figure B.2.: Parametric bouncing ball

denotes the initial energy level, i.e., the initial height if $v = 0$. For instance, we can prove the following property:

$$(v^2 \leq 2g(H - h) \wedge h \geq 0 \wedge g > 0 \wedge H \geq 0) \rightarrow [\texttt{Ball}](0 \leq h \leq H)$$

This specification follows the pattern of Hoare-triples. It expresses that the bouncing ball, when started in an initial state satisfying the precondition always respects the postcondition $0 \leq h \leq H$.

Appendix C.

Background Material

Contents

C.1. Differential Equations

In this section, we summarise some classical results for differential equations, and refer to [Wal98] for details. As usual, $C^k(D, \mathbb{R}^n)$ denotes the space of k-times continuously differentiable functions from D to $\mathbb{R}^n$.

C.1.1. Ordinary Differential Equations

An ordinary differential equation in explicit form is an equation $y'(x) = f(x, y)$ involving a derivative $y'(x)$ of y with respect to x. A solution is a function that obeys said relation of its derivative:

C.1 Definition (Ordinary differential equation). Let $f : D \to \mathbb{R}^n$ be a function on a domain $D \subseteq \mathbb{R} \times \mathbb{R}^n$. The function $Y : I \to \mathbb{R}^n$ is a *solution* on the interval $I \subseteq \mathbb{R}$ of the *initial-value problem*

$$\begin{bmatrix} y'(x) = & f(x, y) \\ y(x_0) = & y_0 \end{bmatrix} \tag{C.1}$$

with *ordinary differential equation* (ODE) $y' = f(x, y)$, if, for all $x \in I$

1. $(x, Y(x)) \in D$

2. $Y'(x)$ exists and $Y'(x) = f(x, Y(x))$.

3. $Y(x_0) = y_0$

If $f : D \to \mathbb{R}^n$ is continuous, then $Y : I \to \mathbb{R}^n$ is continuously differentiable. The definition is accordingly for higher-order differential equations, i.e., differential equations involving higher-order derivatives $y^{(n)}(x)$ for $n > 1$.

Several differential equations have no explicit closed-form solution with elementary functions, for instance, $x''(t) = e^{t^2}$, see [Zei03]. Likewise, differential equations like $y'(t) = \frac{2}{t^3}y$ can have non-analytic smooth solutions like $y(t) = e^{-\frac{1}{t^2}}$.

C.1.2. Existence and Uniqueness Theorems

There are several classical theorems that guarantee existence and/or uniqueness of solutions for differential equations (not necessarily closed-form solutions with elementary functions, though). The classical existence theorem is due to Peano [Pea90].

C.2 Theorem (Existence theorem of Peano [Wal98, Theorem 10.IX]). *Let $f : D \to \mathbb{R}^n$ be a continuous function on an open, connected domain $D \subseteq \mathbb{R} \times \mathbb{R}^n$. Then, the initial-value problem* (C.1) *with $(x_0, y_0) \in D$ has a solution. Further, every solution of* (C.1) *can be continued arbitrarily close to the border of D.*

A function $f : D \to \mathbb{R}^n$ with $D \subseteq \mathbb{R} \times \mathbb{R}^n$ is *Lipschitz-continuous* with respect to y iff there is an $L \in \mathbb{R}$ such that for all $(x, y), (x, \bar{y}) \in D$:

$$\|f(x, y) - f(x, \bar{y})\| \leq L\|y - \bar{y}\| \ .$$

If, for instance, $\frac{\partial f(x,y)}{\partial y}$ exists and is bounded on D then f is Lipschitz-continuous with $\max_{(x,y)\in D} \|\frac{\partial f(x,y)}{\partial y}\|$ by mean-value theorem. Similarly, f is *locally Lipschitz-continuous* iff for each $(x, y) \in D$, there is a neighbourhood in which f is Lipschitz-continuous.

Most importantly, the classical Picard-Lindelöf's theorem [Lin94], which is also known as the Cauchy-Lipschitz theorem, guarantees existence and uniqueness of solutions. As restrictions of solutions are always solutions, we understand uniqueness up to restrictions.

C.3 Theorem (Uniqueness theorem of Picard-Lindelöf [Wal98, Theorem 10.VI]). *In addition to the premisses of Theorem C.2, let f be locally Lipschitz-continuous with respect to y (for instance, $f \in C^1(D, \mathbb{R}^n)$ is sufficient). Then, there is a unique solution of* (C.1).

C.4 Proposition (Global uniqueness theorem of Picard-Lindelöf [Wal98, Proposition 10.VII]). *Let $f : [0, a] \times \mathbb{R}^n \to \mathbb{R}^n$ be a continuous function that is Lipschitz-continuous with respect to y. Then, there is a unique solution of* (C.1) *on* $[0, a]$.

The following result is a minor componentwise generalisation of [Wal98, Proposition 6.VI] to vectorial differential equations.

C.5 Proposition (Continuation of solutions [Wal98, Proposition 6.VI]). *Let $f : D \to \mathbb{R}^n$ be a continuous function on the open, connected domain $D \subseteq \mathbb{R} \times \mathbb{R}^n$. If φ is a solution of $y' = f(x, y)$ on $[0, b)$ whose images lies within a compact set $A \subseteq D$, then φ can be continued to a solution on $[0, b]$.*

C.1.3. Linear Differential Equations with Constant Coefficients

For linear differential equation systems with constant coefficients there is a well-established constructive theory for obtaining closed-form solutions using classical techniques from linear algebra.

C.6 Proposition (Linear systems with constant coefficients [Wal98, §18.X]). *Let the matrix $A \in \mathbb{R}^{n \times n}$ be constant. Then, the initial-value problem*

$$\left[\begin{array}{ll} y'(t) = & Ay(t) + b(t) \\ y(\tau) = & \eta \end{array}\right]$$

has the solution

$$y(t) = e^{A(t-\tau)}\eta + \int_\tau^t e^{A(t-s)}b(s)ds$$

where exponentiation of matrices is defined by the usual power series

$$e^{At} = \sum_{n=0}^{\infty} \frac{1}{n!} A^n t^n \ .$$

Bibliography

[ABB+05] Wolfgang Ahrendt, Thomas Baar, Bernhard Beckert, Richard Bubel, Martin Giese, Reiner Hähnle, Wolfram Menzel, Wojciech Mostowski, Andreas Roth, Steffen Schlager, and Peter H. Schmitt. The KeY tool. *Software and System Modeling*, 4:32–54, 2005.

[ABD08] Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors. *Automated Reasoning, Third International Joint Conference, IJCAR 2008, Sydney, Australia, Proceedings*, volume 5195 of *LNCS*. Springer, 2008.

[ACD90] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for real-time systems. In *LICS*, pages 414–425. IEEE Computer Society, 1990.

[ACH+95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138(1):3–34, 1995.

[ACHH92] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Grossman et al. [GNRR93], pages 209–229.

[ACM84a] Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM J. Comput.*, 13(4):865–877, 1984.

[ACM84b] Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical algebraic decomposition II: An adjacency algorithm for the plane. *SIAM J. Comput.*, 13(4):878–889, 1984.

[ADG+01] Andrew Adams, Martin Dunstan, Hanne Gottliebsen, Tom Kelsey, Ursula Martin, and Sam Owre. Computer algebra meets automated theorem proving: Integrating Maple and PVS. In Richard J. Boulton

and Paul B. Jackson, editors, *TPHOLs*, volume 2152 of *LNCS*, pages 27–42. Springer, 2001.

[ADG03] Eugene Asarin, Thao Dang, and Antoine Girard. Reachability analysis of nonlinear systems using conservative approximation. In Maler and Pnueli [MP03], pages 20–35.

[AHH96] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE T. Software. Eng.*, 22(3):181–201, 1996.

[AHS96] Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors. *Hybrid Systems III: Verification and Control, Proceedings of the DIMACS/SYCON Workshop, October 22-25, 1995, Ruttgers University, New Brunswick, NJ, USA*, volume 1066 of *LNCS*. Springer, 1996.

[ÁMSH01] Erika Ábrahám-Mumm, Martin Steffen, and Ulrich Hannemann. Verification of hybrid systems: Formalization and proof rules in PVS. In Andler and Offutt [AO01], pages 48–57.

[AO01] Sten F. Andler and Jeff Offutt, editors. *7th International Conference on Engineering of Complex Computer Systems (ICECCS 2001), 11-13 June 2001, Skövde, Sweden*, Los Alamitos, 2001. IEEE Computer Society.

[AP04] Rajeev Alur and George J. Pappas, editors. *Hybrid Systems: Computation and Control, 7th International Workshop, HSCC 2004, Philadelphia, PA, USA, March 25-27, 2004, Proceedings*, volume 2993 of *LNCS*. Springer, 2004.

[AW01] Hirokazu Anai and Volker Weispfenning. Reach set computations using real quantifier elimination. In Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *LNCS*, pages 63–76. Springer, 2001.

[Bas99] Saugata Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *J. ACM*, 46(4):537–555, 1999.

[BBB07] Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors. *Hybrid Systems: Computation and Control, 10th International Conference, HSCC 2007, Pisa, Italy, Proceedings*, volume 4416 of *LNCS*. Springer, 2007.

[BBM98] Michael S. Branicky, Vivek S. Borkar, and Sanjoy K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE T. Automat. Contr.*, 43(1):31–45, 1998.

[BBW07] Grégory Batt, Calin Belta, and Ron Weiss. Model checking genetic regulatory networks with parameter uncertainty. In Bemporad et al. [BBB07], pages 61–75.

[BCR98] Jacek Bochnak, Michel Coste, and Marie-Francoise Roy. *Real Algebraic Geometry*, volume 36 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer, 1998.

[BCSS98] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation*. Springer, Secaucus, NJ, USA, 1998.

[BCZ98] Andrej Bauer, Edmund M. Clarke, and Xudong Zhao. Analytica - an experiment in combining theorem proving and symbolic computation. *J. Autom. Reasoning*, 21(3):295–325, 1998.

[BD07] Christopher W. Brown and James H. Davenport. The complexity of quantifier elimination and cylindrical algebraic decomposition. In Dongming Wang, editor, *ISSAC*, pages 54–60. ACM, 2007.

[Bec99] Bernhard Beckert. Equality and other theories. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer, 1999.

[BFU04] Investigation report. Technical Report AX001-1-2/02, German Federal Bureau of Aircraft Accidents Investigation, May 2004.

[BGH+04] Bernhard Beckert, Martin Giese, Elmar Habermalz, Reiner Hähnle, Andreas Roth, Philipp Rümmer, and Steffen Schlager. Taclets: A new paradigm for constructing interactive theorem provers. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales, Serie A: Matemáticas (RACSAM)*, 98(1), 2004.

[BGH+07] Bernhard Beckert, Martin Giese, Reiner Hähnle, Vladimir Klebanov, Philipp Rümmer, Steffen Schlager, and Peter H. Schmitt. The KeY System 1.0 (deduction component). In Frank Pfenning, editor, *Proceedings, International Conference on Automated Deduction, Bremen, Germany*, LNCS. Springer, 2007.

[BHS07] Bernhard Beckert, Reiner Hähnle, and Peter H. Schmitt, editors. *Verification of Object-Oriented Software: The KeY Approach*, volume 4334 of *LNCS*. Springer, 2007.

[BJK+97] Bruno Buchberger, Tudor Jebelean, Franz Kriftner, Mircea Marin, Elena Tomuta, and Daniela Vasaru. A survey of the Theorema project. In *ISSAC*, pages 384–391, 1997.

[BK06] Bernhard Beckert and Vladimir Klebanov. Must program verification systems and calculi be verified? In *3rd International Verification Workshop VERIFY'06, at FLoC, Seattle, USA*, pages 34–41, 2006.

[Bla00] Patrick Blackburn. Internalizing labelled deduction. *J. Log. Comput.*, 10(1):137–168, 2000.

[Bor99] Brian Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 11(1-4):613–623, 1999.

[Bou72] Nicolas Bourbaki. *Elements of mathematics. Commutative algebra.* Hermann, Paris, 1972. Translated from the French.

[Bou89] Nicolas Bourbaki. *Algebra I.* Springer, Berlin, 1989. Originally published as *Éléments de Mathématique, Algèbre.*

[BP06] Bernhard Beckert and André Platzer. Dynamic logic with non-rigid functions: A basis for object-oriented program verification. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, Proceedings*, volume 4130 of *LNCS*, pages 266–280. Springer, 2006.

[BPR96] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. ACM*, 43(6):1002–1045, 1996.

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry.* Springer, 2nd edition, 2006.

[Bra95a] Michael S. Branicky. General hybrid dynamical systems: Modeling, analysis, and control. In Alur et al. [AHS96], pages 186–200.

[Bra95b] Michael S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* PhD thesis, Dept. Elec. Eng. and Computer Sci., Massachusetts Inst. Technol., Cambridge, MA, 1995.

[Bra95c] Michael S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theor. Comput. Sci.*, 138(1):67–100, 1995.

[Bro03] Christopher W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *SIGSAM Bull.*, 37(4):97–108, 2003.

[BS01] Bernhard Beckert and Steffen Schlager. A sequent calculus for first-order dynamic logic with trace modalities. In Goré et al. [GLN01], pages 626–641.

[Buc65] Bruno Buchberger. *An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal*. PhD thesis, University of Innsbruck, 1965.

[Bue08] Martin Buehler. Summary of DGC 2005 results. *Journal of Field Robotics*, 23:465–466, 2008.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BW98] Thomas Becker and Volker Weispfenning. *Gröbner Bases*, volume 141 of *Springer Graduate Texts in Mathematics*. Springer, 1998.

[CFH+03] Edmund M. Clarke, Ansgar Fehnker, Zhi Han, Bruce H. Krogh, Joël Ouaknine, Olaf Stursberg, and Michael Theobald. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *Int. J. Found. Comput. Sci.*, 14(4):583–604, 2003.

[CGJ+03] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[CGP99] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

[CH91] George E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symb. Comput.*, 12(3):299–328, 1991.

[Chv83] Vasek Chvátal. *Linear Programming*. Freeman, 1983.

[CJ89] George E. Collins and Jeremy R. Johnson. Quantifier elimination and the sign variation method for real root isolation. In *ISSAC*, pages 264–271, 1989.

[CJ98] Bob F. Caviness and Jeremy R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition, Texts and Monographs in Symbolic Computation*. Springer Verlag, 1998.

[CJK90] George E. Collins, Jeremy R. Johnson, and Wolfgang Küchlin. Parallel real root isolation using the coefficient sign variation method. In Richard Zippel, editor, *CAP*, volume 584 of *LNCS*, pages 71–87. Springer, 1990.

[CJK02] George E. Collins, Jeremy R. Johnson, and Werner Krandick. Interval arithmetic in cylindrical algebraic decomposition. *J. Symb. Comput.*, 34(2):145–157, 2002.

[CJR95] Zhou Chaochen, Wang Ji, and Anders P. Ravn. A formal description of hybrid systems. In Alur et al. [AHS96], pages 511–530.

[CK03] Alongkrit Chutinan and Bruce H. Krogh. Computational techniques for hybrid system verification. *IEEE T. Automat. Contr.*, 48(1):64–75, 2003.

[CL05] Pieter Collins and John Lygeros. Computability of finite-time reachable sets for hybrid systems. In *CDC-ECC'05*, pages 4688– 4693. IEEE, Dec 2005.

[Cla79] Edmund M. Clarke. Program invariants as fixedpoints. *Computing*, 21(4):273–294, 1979.

[CLO92] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. Springer, New York, 1992.

[Coh69] Paul J. Cohen. Decision procedures for real and p-adic fields. *Communications in Pure and Applied Mathematics*, 22:131–151, 1969.

[Col75] George E. Collins. Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Barkhage, editor, *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

[Coo78] Stephen A. Cook. Soundness and completeness of an axiom system for program verification. *SIAM J. Comput.*, 7(1):70–90, 1978.

[Cra57] William Craig. Three uses of the herbrand-gentzen theorem in relating model theory and proof theory. *J. Symb. Log.*, 22(3):269–285, 1957.

[Dav97] Jennifer Mary Davoren. On hybrid systems and the modal μ-calculus. In Panos J. Antsaklis, Wolf Kohn, Michael D. Lemmon, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems*, volume 1567 of *LNCS*, pages 38–69. Springer, 1997.

[DCMM04] Jennifer Mary Davoren, Vaughan Coulthard, Nicolas Markey, and Thomas Moor. Non-deterministic temporal logics for general flow systems. In Alur and Pappas [AP04], pages 280–295.

[DE73] George B. Dantzig and B. Curtis Eaves. Fourier-Motzkin elimination and its dual. *J. Comb. Theory, Ser. A*, 14(3):288–297, 1973.

[DH88] James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *J. Symb. Comput.*, 5(1/2):29–35, 1988.

[DHK03] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. *J. Autom. Reasoning*, 31(1):33–72, 2003.

[DHO03] Werner Damm, Hardi Hungar, and Ernst-Rüdiger Olderog. On the verification of cooperating traffic agents. In Frank S. de Boer, Marcello M. Bonsangue, Susanne Graf, and Willem P. de Roever, editors, *FMCO*, volume 3188 of *LNCS*, pages 77–110. Springer, 2003.

[DHO06] Werner Damm, Hardi Hungar, and Ernst-Rüdiger Olderog. Verification of cooperating traffic agents. *International Journal of Control*, 79(5):395–421, May 2006.

[DM79] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, 1979.

[DM07] Alexandre Donzé and Oded Maler. Systematic simulation using sensitivity analysis. In Bemporad et al. [BBB07], pages 174–189.

[DMC05] Gilles Dowek, César Muñoz, and Víctor A. Carreño. Provably safe coordinated strategy for distributed conflict resolution. In *Proceedings of the AIAA Guidance Navigation, and Control Conference and Exhibit 2005, AIAA-2005-6047*, 2005.

[DMO+07] Werner Damm, Alfred Mikschl, Jens Oehlerking, Ernst-Rüdiger Olderog, Jun Pang, André Platzer, Marc Segelken, and Boris Wirtz. Automating verification of cooperation, control, and design in traffic applications. In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *Formal Methods and Hybrid Real-Time Systems, Essays in Honor of Dines Bjørner and Chaochen Zhou on the Occasion of Their 70th Birthdays*, volume 4700 of *LNCS*, pages 115–169. Springer, 2007.

[DN00] Jennifer Mary Davoren and Anil Nerode. Logics for hybrid systems. *IEEE*, 88(7):985–1010, July 2000.

[DPR05] Werner Damm, Guilherme Pinto, and Stefan Ratschan. Guaranteed termination in the verification of LTL properties of non-linear robust discrete time hybrid systems. In Peled and Tsay [PT05], pages 99–113.

[EC82] E. Allen Emerson and Edmund M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.*, 2(3):241–266, 1982.

[EH86] E. Allen Emerson and Joseph Y. Halpern. "Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.

[Eis99] David Eisenbud. *Commutative algebra with a view toward algebraic geometry*, volume 150 of *Graduate Texts in Mathematics*. Springer, New York, 3 edition, 1999.

[EJSL99] Magnus Egerstedt, Karl Hernik Johansson, Shankar Sastry, and John Lygeros. On the regularization of Zeno hybrid automata. *Systems and Control Letters*, 38:141–150, 1999.

[Eme90] Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. MIT Press, 1990.

[ERT02] ERTMS User Group. ERTMS/ETCS System requirements specification. `http://www.aeif.org/ccm/`, 2002. Version 2.2.2.

[Fau99] Jean-Charles Faugére. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[Fau02] Jean-Charles Faugére. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *ISAAC*. ACM Press, 2002.

[Fit96] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, New York, 2nd edition, 1996.

[FM99] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*. Kluwer, Norwell, MA, USA, 1999.

[FM06] Johannes Faber and Roland Meyer. Model checking data-dependent real-time properties of the European Train Control System. In *FM-CAD*, pages 76–77. IEEE Computer Society Press, Nov 2006.

[Frä99] Martin Fränzle. Analysis of hybrid systems: An ounce of realism can save an infinity of states. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *CSL*, volume 1683 of *LNCS*, pages 126–140. Springer, 1999.

[Fre05] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In Morari and Thiele [MT05], pages 258–273.

[GCB07] Daniel Silva Graça, Manuel L. Campagnolo, and Jorge Buescu. Computability with polynomial differential equations. *Advances in Applied Mathematics*, 2007.

[Gea88] Charles W. Gear. Differential-algebraic equations index transformations. *SIAM J. Sci. Stat. Comput.*, 9(1):39–47, 1988.

[Gie01] Martin Giese. Incremental closure of free variable tableaux. In Goré et al. [GLN01], pages 545–560.

[GLN01] Rajeev Goré, Alexander Leitsch, and Tobias Nipkow, editors. *Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 18-23, 2001, Proceedings*, volume 2083 of *LNCS*. Springer, 2001.

[GM93] Michael J. C. Gordon and Tom F. Melham. *Introduction to HOL: A Theorem-Proving Environment for Higher-Order Logic*. Cambridge Univ. Press, 1993.

[GM08] Aarti Gupta and Sharad Malik, editors. *Computer Aided Verification, CAV 2008, Princeton, NJ, USA, Proceedings*, volume 5123 of *LNCS*. Springer, 2008.

[GMAR07] André L. Galdino, César Muñoz, and Mauricio Ayala-Rincón. Formal verification of an optimal air traffic conflict resolution and recovery algorithm. In Daniel Leivant and Ruy de Queiroz, editors, *WoLLIC*, volume 4576 of *LNCS*, pages 177–188. Springer, 2007.

[GNRR93] Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors. *Hybrid Systems*, volume 736 of *LNCS*. Springer, 1993.

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Mon.hefte Math. Phys.*, 38:173–198, 1931.

[Gri88] Dima Grigoriev. Complexity of deciding Tarski algebra. *J. Symb. Comput.*, 5(1/2):65–108, 1988.

[Gro07] Jean Gross. Schlussbericht über die Entgleisung von Güterzug 43647 der BLS AG auf der Weiche 34 (Einfahrt Lötschberg-Basisstrecke). Technical Report 07101601, Unfalluntersuchungsstelle Bahnen und Schiffe, Oct 2007.

[GT08] Sumit Gulwani and Ashish Tiwari. Constraint-based approach for analysis of hybrid systems. In Gupta and Malik [GM08], pages 190–203.

[GV88] Dima Grigoriev and Nicolai Vorobjov. Solving systems of polynomial inequalities in subexponential time. *J. Symb. Comput.*, 5(1/2):37–64, 1988.

[Har79] David Harel. *First-Order Dynamic Logic*. Springer, New York, 1979.

[Har07] John Harrison. Verifying nonlinear real formulas via sums of squares. In Klaus Schneider and Jens Brandt, editors, *TPHOLs*, volume 4732 of *LNCS*, pages 102–118. Springer, 2007.

[Hen96] Thomas A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, Los Alamitos, 1996. IEEE Computer Society.

[HESV91] Ann Hsu, Farokh Eskafi, Sonia Sachs, and Pravin Varaiya. Design of platoon maneuver protocols for IVHS. PATH Research Report UCB-ITS-PRR-91-6, Institute of Transportation Studies, University of California, Berkeley, 1991.

[HH94] Thomas A. Henzinger and Pei-Hsin Ho. HYTECH: The Cornell HYbrid TECHnology tools. In Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems*, volume 999 of *LNCS*, pages 265–293. Springer, 1994.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, Cambridge, 2000.

[HKT07] Inseok Hwang, Jegyom Kim, and Claire Tomlin. Protocol-based conflict resolution for air traffic control. *Air Traffic Control Quarterly*, 15(1), 2007.

[HLS+96] Dieter Hutter, Bruno Langenstein, Claus Sengler, Jörg H. Siekmann, Werner Stephan, and Andreas Wolpers. Deduction in the verification support environment (VSE). In Marie-Claude Gaudel and Jim Woodcock, editors, *FME*, volume 1051 of *LNCS*, pages 268–286. Springer, 1996.

[HNSY92] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. In *LICS*, pages 394–406. IEEE Computer Society, 1992.

[Hod93] Wilfrid Hodges. *Model Theory*. Cambridge University Press, 1993.

[HS94] Reiner Hähnle and Peter H. Schmitt. The liberalized δ-rule in free variable semantic tableaux. *J. Autom. Reasoning*, 13(2):211–221, 1994.

[Jif94] He Jifeng. From CSP to hybrid systems. In A. W. Roscoe, editor, *A classical mind: essays in honour of C. A. R. Hoare*, pages 171–189, Hertfordshire, UK, 1994. Prentice Hall.

[JSZL01] Karl Hernik Johansson, Shankar Sastry, Jun Zhang, and John Lygeros. Zeno hybrid systems. *International Journal of Robust & Nonlinear Control*, 11:435–451, 2001.

[KH96] Anatole Katok and Boris Hasselblatt. *Introduction to the Modern Theory of Dynamical Systems*. Cambridge University Press, New York, NY, 1996.

[KK71] Georg Kreisel and Jean-Louis Krivine. *Elements of mathematical logic: Model Theory*. North-Holland, 2 edition, 1971.

[KMP00] Yonit Kesten, Zohar Manna, and Amir Pnueli. Verification of clocked and hybrid systems. *Acta Inf.*, 36(11):837–912, 2000.

[Kol72] Ellis Robert Kolchin. *Differential Algebra and Algebraic Groups*. Academic Press, New York, 1972.

[KP07] Stephanie Kemper and André Platzer. SAT-based abstraction refinement for real-time systems. In Frank S. de Boer and Vladimir Mencl, editors, *Formal Aspects of Component Software, Third International Workshop, FACS 2006, Prague, Czech Republic, Proceedings*, volume 182 of *ENTCS*, pages 107–122, 2007.

[Lan78] Serge Lang. *Algebra*. Addison-Wesley Publishing Company, 1978.

[Lib03] Daniel Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhäuser, Boston, MA, 2003.

[Lin94] M. Ernst Lindelöf. Sur l'application de la méthode des approximations successives aux équations différentielles ordinaires du premier ordre. *Comptes rendus hebdomadaires des s'eances de l'Acad'emie des sciences*, 114:454–457, 1894.

[LLL00] Carolos Livadas, John Lygeros, and Nancy A. Lynch. High-level modeling and analysis of TCAS. *Proc. IEEE - Special Issue on Hybrid Systems: Theory & Applications*, 88(7):926–947, 2000.

[Łoj64] Stanisław Łojasiewicz. Triangulations of semi-analytic sets. *Annali della Scuola Normale Superiore di Pisa*, 18:449–474, 1964.

[LPS00] Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, March 2000.

[LPY99] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. A new class of decidable hybrid systems. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *HSCC*, volume 1569 of *LNCS*, pages 137–151. Springer, 1999.

[LPY01] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *J. Symb. Comput.*, 32(3):231–253, 2001.

[LT05] Ruggero Lanotte and Simone Tini. Taylor approximation for hybrid systems. In Morari and Thiele [MT05], pages 402–416.

[LW93] Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *Comput. J.*, 36(5):450–462, 1993.

[Man91] Elizabeth L. Mansfield. *Differential Gröbner Bases*. PhD thesis, University of Sydney, 1991.

[MF01] Mieke Massink and Nicoletta De Francesco. Modelling free flight with collision avoidance. In Andler and Offutt [AO01], pages 270–280.

[MH05] Sean McLaughlin and John Harrison. A proof-producing decision procedure for real arithmetic. In Robert Nieuwenhuis, editor, *CADE*, volume 3632 of *LNCS*, pages 295–314. Springer, 2005.

[Mil72] Robin Milner. Logic for computable functions: description of a machine implementation. Technical report, Stanford University, Stanford, CA, USA, 1972.

[Mor87] Michał Morayne. On differentiability of Peano type functions. *Colloquium Mathematicum*, LIII:129–132, 1987.

[Mor05] Teo Mora. *Solving Polynomial Equation Systems II: Macaulay's Paradigm and Gröbner Technology*. Cambridge University Press, 2005.

[MP03] Oded Maler and Amir Pnueli, editors. *Hybrid Systems: Computation and Control, 6th International Workshop, HSCC 2003 Prague, Czech Republic, April 3-5, 2003, Proceedings*, volume 2623 of *LNCS*. Springer, 2003.

[MPM05] Venkatesh Mysore, Carla Piazza, and Bud Mishra. Algorithmic algebraic model checking II: Decidability of semi-algebraic model checking and its applications to systems biology. In Peled and Tsay [PT05], pages 217–233.

[MS98] Zohar Manna and Henny Sipma. Deductive verification of hybrid systems using STeP. In Thomas A. Henzinger and Shankar Sastry, editors, *HSCC*, volume 1386 of *LNCS*, pages 305–318. Springer, 1998.

[MT05] Manfred Morari and Lothar Thiele, editors. *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC 2005, Zurich, Switzerland, March 9-11, 2005, Proceedings*, volume 3414 of *LNCS*. Springer, 2005.

[MTW73] Charles W. Misner, Kip S. Thorne, and John Archibald Wheeler. *Gravitation*. W.H. Freeman, New York, 1973.

[Nip08] Tobias Nipkow. Linear quantifier elimination. In Armando et al. [ABD08].

[NOSY92] Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. An approach to the description and analysis of hybrid systems. In Grossman et al. [GNRR93], pages 149–178.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[PAM+05] Carla Piazza, Marco Antoniotti, Venkatesh Mysore, Alberto Policriti, Franz Winkler, and Bud Mishra. Algorithmic algebraic model checking I: Challenges from systems biology. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *LNCS*, pages 5–19. Springer, 2005.

[Par03] Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Math. Program.*, 96(2):293–320, May 2003.

[PC07] André Platzer and Edmund M. Clarke. The image computation problem in hybrid systems model checking. In Bemporad et al. [BBB07], pages 473–486.

[PC08a] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In Gupta and Malik [GM08], pages 176–189.

[PC08b] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. Technical Report CMU-CS-08-103, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2008.

[PDP01] Robert S. Parker, Francis J. Doyle, and Nicholas A. Peppas. The intravenous route to blood glucose control. *IEEE Engineering in Medicine and Biology*, 20(1):65–73, Jan 2001.

[Pea90] Giuseppe Peano. Demonstration de l'intégrabilité des équations différentielles ordinaires. *Mathematische Annalen*, 37:182–228, 1890.

[PER79] Marian Boykan Pour-El and Ian Richards. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic*, 17:61–90, 1979.

[Per91] Lawrence Perko. *Differential equations and dynamical systems*. Springer, New York, NY, USA, 1991.

[PGHD04] Jan Peleska, Daniel Große, Anne Elisabeth Haxthausen, and Rolf Drechsler. Automated verification for train control systems. In *Proceedings of Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMAT 2004)), Braunschweig*, 2004.

[PJ04] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In Alur and Pappas [AP04], pages 477–492.

[PJP07] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE T. Automat. Contr.*, 52(8):1415–1429, 2007.

[PK02] Amir Pnueli and Yonit Kesten. A deductive proof system for CTL*. In Lubos Brim, Petr Jancar, Mojmír Kretínský, and Antonín Kucera, editors, *CONCUR*, volume 2421 of *LNCS*, pages 24–40. Springer, 2002.

[Pla04a] André Platzer. An object-oriented dynamic logic with updates. Master's thesis, University of Karlsruhe, Department of Computer Science. Institute for Logic, Complexity and Deduction Systems, 2004.

[Pla04b] André Platzer. Using a program verification calculus for constructing specifications from implementations. Minor thesis, University of Karlsruhe, Department of Computer Science. Institute for Logic, Complexity and Deduction Systems, 2004.

[Pla07a] André Platzer. Combining deduction and algebraic constraints for hybrid system analysis. In Bernhard Beckert, editor, *4th International Verification Workshop VERIFY'07, at CADE-21, Bremen, Germany, July 15-16, 2007*, volume 259 of *CEUR Workshop Proceedings*, pages 164–178. CEUR-WS.org, 2007.

[Pla07b] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In Nicola Olivetti, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, 16th International Conference, TABLEAUX 2007, Aix en Provence, France, July 3-6, 2007, Proceedings*, volume 4548 of *LNCS*, pages 216–232. Springer, 2007.

[Pla07c] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. Technical Report 15, Reports of SFB/TR 14 AVACS, May 2007. ISSN: 1860-9821, http://www.avacs.org.

[Pla07d] André Platzer. Differential logic for reasoning about hybrid systems. In Bemporad et al. [BBB07], pages 746–749.

[Pla07e] André Platzer. A temporal dynamic logic for verifying hybrid system invariants. In Sergei N. Artëmov and Anil Nerode, editors, *Logical Foundations of Computer Science, 5th International Symposium, LFCS'07, New York, USA, June 4-7, 2007, Proceedings*, volume 4514 of *LNCS*, pages 457–471. Springer, 2007.

[Pla07f] André Platzer. A temporal dynamic logic for verifying hybrid system invariants. Technical Report 12, Reports of SFB/TR 14 AVACS, Feb 2007. ISSN: 1860-9821, http://www.avacs.org.

[Pla07g] André Platzer. Towards a hybrid dynamic logic for hybrid dynamic systems. In Patrick Blackburn, Thomas Bolander, Torben Braüner, Valeria de Paiva, and Jørgen Villadsen, editors, *LICS International Workshop on Hybrid Logic, HyLo'06, Seattle, USA, Proceedings*, volume 174 of *ENTCS*, pages 63–77, Jun 2007.

[Pla08a] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *Journal of Logic and Computation*, 2008. To appear.

[Pla08b] André Platzer. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning*, 41(2):143–189, 2008.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[PQ08a] André Platzer and Jan-David Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In Armando et al. [ABD08], pages 171–178.

[PQ08b] André Platzer and Jan-David Quesel. Logical verification and systematic parametric analysis in train control. In Magnus Egerstedt and Bud Mishra, editors, *Hybrid Systems: Computation and Control, 10th International Conference, HSCC 2008, St. Louis, USA, Proceedings*, volume 4981 of *LNCS*, pages 646–649. Springer, 2008.

[Pra76] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *FOCS*, pages 109–121. IEEE, 1976.

[Pra79] Vaughan R. Pratt. Process logic. In *POPL*, pages 93–100, 1979.

[PT05] Doron Peled and Yih-Kuen Tsay, editors. *Automated Technology for Verification and Analysis, Third International Symposium, ATVA 2005, Taipei, Taiwan, October 4-7, 2005, Proceedings*, volume 3707 of *LNCS*. Springer, 2005.

[Que07] Jan-David Quesel. A theorem prover for differential dynamic logic: Deductive verification of hybrid systems. Master's thesis, University of Oldenburg, Department of Computing Science. Correct System Design Group, Apr 2007.

[RCT05] Enric Rodríguez-Carbonell and Ashish Tiwari. Generating polynomial invariants for hybrid systems. In Morari and Thiele [MT05], pages 590–605.

[Ren92a] James Renegar. On the computational complexity and geometry of the first-order theory of the reals, part I: Introduction. preliminaries. the geometry of semi-algebraic sets. the decision problem for the existential theory of the reals. *J. Symb. Comput.*, 13(3):255–300, 1992.

[Ren92b] James Renegar. On the computational complexity and geometry of the first-order theory of the reals, part II: The general decision problem. preliminaries for quantifier elimination. *J. Symb. Comput.*, 13(3):301–328, 1992.

[Ren92c] James Renegar. On the computational complexity and geometry of the first-order theory of the reals, part III: Quantifier elimination. *J. Symb. Comput.*, 13(3):329–352, 1992.

[Rey01] Mark Reynolds. An axiomatization of full computation tree logic. *J. Symb. Log.*, 66(3):1011–1057, 2001.

[Rey05] Mark Reynolds. An axiomatization of PCTL*. *Inf. Comput.*, 201(1):72–119, 2005.

[Ris88] Jean-Jacques Risler. Some aspects of complexity in real algebraic geometry. *J. Symb. Comput.*, 5(1/2):109–119, 1988.

[Rob56] Abraham Robinson. *Complete Theories*. North-Holland, 1956.

[RRS03] Mauno Rönkkö, Anders P. Ravn, and Kaisa Sere. Hybrid action systems. *Theor. Comput. Sci.*, 290(1):937–973, 2003.

[Rüm07] Philipp Rümmer. A sequent calculus for integer arithmetic with counterexample generation. In Bernhard Beckert, editor, *VERIFY'07 at CADE, Bremen, Germany*, volume 259 of *CEUR Workshop Proceedings*, pages 179–194. CEUR-WS.org, 2007.

[Sei54] Abraham Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, 60:365–374, 1954.

[Sib75] Konstantin Sergeevich Sibirsky. *Introduction to Topological Dynamics*. Noordhoff, Leyden, 1975.

[SRH02] Pierre-Yves Schobbens, Jean-François Raskin, and Thomas A. Henzinger. Axioms for real-time logics. *Theor. Comput. Sci.*, 274(1-2):151–182, 2002.

[SSM04] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Constructing invariants for hybrid systems. In Alur and Pappas [AP04], pages 539–554.

[Ste73] Gilbert Stengle. A Nullstellensatz and a Positivstellensatz in semialgebraic geometry. *Math. Ann.*, 207(2):87–97, 1973.

[Sti92] Colin Stirling. Modal and temporal logics. In *Handbook of logic in computer science (vol. 2): background: computational structures*, pages 477–563. Oxford University Press, Inc., New York, NY, USA, 1992.

[Str06] Adam W. Strzebonski. Cylindrical algebraic decomposition using validated numerics. *J. Symb. Comput.*, 41(9):1021–1038, 2006.

[Stu35] Charles-François Sturm. Mémoire sue la résolution des équations numériques. *Mémoires des Savants Etrangers*, 6:271–318, 1835.

[Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 2nd edition, 1951.

[Tav87] Lucio Tavernini. Differential automata and their discrete simulators. *Non-Linear Anal.*, 11(6):665–683, 1987.

[Tin03] Cesare Tinelli. Cooperation of background reasoners in theory reasoning by residue sharing. *J. Autom. Reasoning*, 30(1):1–31, 2003.

[Tiw03] Ashish Tiwari. Approximate reachability for linear systems. In Maler and Pnueli [MP03], pages 514–525.

[Tiw05] Ashish Tiwari. An algebraic approach for the unsatisfiability of nonlinear constraints. In C.-H. Luke Ong, editor, *CSL*, volume 3634 of *LNCS*, pages 248–262. Springer, 2005.

[TPS98] Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management: a study in multi-agent hybrid systems. *IEEE T. Automat. Contr.*, 43(4):509–521, 1998.

[vdD88] Lou van den Dries. Alfred Tarski's elimination theory for real closed fields. *J. Symb. Log.*, 53(1):7–19, 1988.

[vdD98] Lou van den Dries. *Tame Topology and O-minimal Structures*. Cambridge University Press, 1998.

[Wal98] Wolfgang Walter. *Ordinary Differential Equations*. Springer, 1998.

[Wei88] Volker Weispfenning. The complexity of linear problems in fields. *J. Symb. Comput.*, 5(1/2):3–27, 1988.

[Wei05] Klaus Weihrauch. *Computable Analysis*. Springer, 2005.

[Wol05] Wolfram Research, Inc., Champaign, IL. *Mathematica*, version 5.2 edition, 2005. `http://www.wolfram.com/`.

[Zei03] Eberhard Zeidler, editor. *Teubner-Taschenbuch der Mathematik I.* Teubner, 2003.

[ZRH92] Chaochen Zhou, Anders P. Ravn, and Michael R. Hansen. An extended duration calculus for hybrid real-time systems. In Grossman et al. [GNRR93], pages 36–59.

Index

Curriculum Vitae

1998	Abitur
1998 – 1999	social service
1998 – 1999	study mathematics at Fernuni Hagen
1999 – 2004	study computer science and mathematics at University of Karlsruhe (TH)
September 2004	Diplom (M.S. degree) in Computer Science Thesis: "An Object-Oriented Dynamic Logic with Updates"
since 2004	Research Assistant in the working group of Prof. Ernst-Rüdiger Olderog at the Carl-von-Ossietzky University, Oldenburg.
since 2005	PhD student in the Graduate School TrustSoft, Oldenburg.
December 19, 2008	defence of the dissertation