J Autom Reasoning (2008) 41:143–189
DOI 10.1007/s10817-008-9103-8

Differential Dynamic Logic for Hybrid Systems

André Platzer

Received: 23 August 2007 / Accepted: 27 June 2008 / Published online: 20 August 2008

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 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.

Keywords Dynamic logic · Differential equations · Sequent calculus · Axiomatisation · Automated theorem proving · Verification of hybrid systems

1 Introduction

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 the 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.

A. Platzer (✉)
Department of Computing Science, University of Oldenburg,
26111 Oldenburg, Germany
e-mail: platzer@informatik.uni-oldenburg.de

Hybrid Systems As a common mathematical model for complex physical systems, *hybrid systems* are dynamical systems [45] where the system state evolves over time according to interacting laws of discrete and continuous dynamics [3, 9, 11, 21, 36, 55]. Continuous dynamics is specified by differential equations. It 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 interact, 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 [11], *hybrid automata* [3, 36] are the most widely used notation. They specify discrete and continuous dynamics in a graph, see Fig. 1 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 node *brake* of Fig. 1, the differential equations $z' = v, v' = a$ only apply within the invariant region $v \geq 0$ (the train does not move backwards by 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 Fig. 1 can take an edge to leave node *accel* when train z passed point s, set the acceleration to braking by $a := -b$, and enter mode *brake*.

Model Checking As a standard verification technique, model checking [15, 24] has been used successfully for verifying temporal logic properties [2, 24, 25, 50] of finite-state abstractions of automata-based transition structures by exhaustive state space exploration [3, 36, 37, 44]. The continuous state spaces of hybrid automata, however, do not admit equivalent finite-state abstractions [36]. Because of this, model checkers for hybrid automata use various approximations [3–5, 13, 14, 28, 36, 37, 44, 57] and

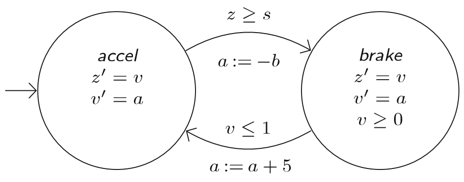

Fig. 1 Hybrid automaton for an (overly) simplified train control system

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 Fig. 1). It is, however, quite difficult to determine corresponding symbolic parameter constraints from concrete values of a counter-example trace produced by a model checker, especially if they rely on nonstructural state splitting [5, 13, 14, 29]. 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. Thus, standard model checking approaches [3, 13, 29, 36] cannot be used, as they require at most linear discrete dynamics.

Deductive Verification Deductive approaches [7, 8, 20, 21, 34, 35, 38, 59] 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 [21] 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 [2, 24, 25, 50], state transitions can be referred to using modal operators. In deductive approaches, temporal logics have been used to prove validity of formulas in calculi [21, 59]. 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.

Dynamic logic (DL) [34, 35, 51] is a successful approach for verifying infinite-state discrete systems deductively [7, 8, 34, 35, 38]. 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 following 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 DL 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, $\exists p([\alpha]\phi \land [\beta]\psi)$ 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 DL 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 DL so that operational models α of hybrid systems can be used in modal formulas like $[\alpha]\phi$.

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, e.g., $[\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2$ 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 so that $[\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2$ is equivalent to $[\alpha]\phi$, because of the dangling edges between the subgraphs α_i. For instance, the automaton in Fig. 1 cannot simply be verified by proving $[accel]\phi \wedge [brake]\phi$, 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 that allows 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 DL 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 paper is a thorough analysis of the logic $\mathsf{d}\mathcal{L}$ and its calculus.

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 [26, 27, 33] 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 [16, 54] 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 our previous work [47], we used separate side deductions for reducing the unquantified kernel $[\alpha]x \geq 0$ to some arithmetic formula ψ before returning to $\forall t\psi$ in the main proof. This is easy to understand and can be performed without much change in interactive theorem provers. It is, however, not necessarily well-suited for automation.

In this paper, we present an improved 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 [16, 54]. 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 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 d$\mathcal{L}$ calculus.

Related Work Model checking approaches work by state space exploration and require [36] various abstractions or approximations [3, 4, 14, 28, 36, 37, 44, 57] for hybrid automata, including numerical approximations [5, 13].

Beyond standard approaches [3, 29, 36] for linear systems with constant dynamics, Lafferriere et al. [41] 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 [13] presented polyhedral approximations of hybrid automata with polyhedral discrete dynamics, invariants, and initial state sets.

Fränzle [28] 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. [5] used piecewise linear numerical approximations in an approximate reachability algorithm for continuous systems with known Lipschitz bounds.

Mysore et al. [44] showed decidability of bounded-time and bounded switching reachability prefixes of semi-algebraic hybrid automata.

Because hybrid systems do not admit equivalent finite-state abstractions [36] and due to general limits of numerical approximation [49], model checkers are still more successful in falsification than in verification. To obtain a sound verification approach and for improved handling of free parameters [21], we follow a symbolic logic-based approach and support 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.

Zhou et al. [59] 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 [20, 21] 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.

Inspired by He [39], Zhou et al. [12] presented a hybrid variant of CSP as a language for describing hybrid systems. They gave a semantics in extended duration calculus [59] but no verification technique.

Rönkkö et al. [52] 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.

The strength of our logic primarily is that it is an expressive *first-order* DL: 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 [20, 21, 59] 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 [11, 13, 14, 18, 19, 36, 44, 49]. 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 [4, 45, 49].

Manna et al. [40, 42] and Ábrahám et al. [1] used theorem provers for checking invariants of hybrid automata in STeP [42] or PVS [1], 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 DL 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 [46].

Contributions Our main conceptual contribution is the differential dynamic logic $\mathsf{d}\mathcal{L}$ 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 $\mathsf{d}\mathcal{L}$ 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 $\mathsf{d}\mathcal{L}$ 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.

This paper extends our previous results [47] in various aspects. We generalise the logic to support systems of differential equations instead of one-dimensional change. We introduce a generalised free variable calculus that is significantly more suitable for automated theorem proving than previous calculi, and we prove our augmented calculus to be relatively complete. Finally, we extend our train case study to the general case with free acceleration and derive parameter constraints that are required for safety.

Structure of this Paper After introducing syntax and semantics of the differential dynamic logic $\mathsf{d}\mathcal{L}$ in Section 2, we introduce a free variable sequent calculus for $\mathsf{d}\mathcal{L}$ in Section 4 and prove soundness and relative completeness in Section 5. In Section 6, we use our calculus to prove an inductive safety property of a train control system presented in Section 3. We draw conclusions and discuss future work in Section 7.

2 Syntax and Semantics 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 DL 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 Fig. 1. 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., $q := \mathit{brake}$, 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 [8]. 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 Fig. 1. 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 [35] as control structure. For example, $q := \mathit{accel} \cup z'' = -b$ 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.

With these operations, hybrid systems can be represented naturally as hybrid programs. For example, Fig. 2 depicts a hybrid program rendition of the hybrid automaton in Fig. 1. 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 [3, 36], 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 Fig. 2 expresses that the transitions of a hybrid automaton can repeat indefinitely.

Fig. 2 Hybrid program rendition of hybrid automaton from Fig. 1

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

2.1 Syntax of Differential Dynamic Logic

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 $0, 1, +, -, \cdot, /, =, \leq, <, \geq, >$. System state variables are represented as real-valued constant symbols of Σ. Unlike fixed symbols like 1, state variables are *flexible* [8], 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. *Rigid* symbols like 1, instead, 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 $X_i \in V$. The set of formulas of *first-order logic* is defined as usual, giving first-order real arithmetic [54] augmented with Skolem terms. We will show the relationship to standard first-order real arithmetic without Skolem terms in Lemma 2 of Section 4.2.2.

2.1.1 Hybrid Programs

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

Definition 1 (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, further, 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 d$\mathcal{L}$ semantics allows arbitrary differential equations. To retain feasible arithmetic, some of our calculus rules assume that, like in [3, 28, 36, 44], 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., [4, 41, 45, 49, 58].

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 1, we only allow first-order formulas as tests. Instead, we could allow *rich tests*, i.e., arbitrary 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 d$\mathcal{L}$. To simplify the presentation, however, we refrain from allowing arbitrary d$\mathcal{L}$ formulas as tests, because that requires simultaneous inductive handling of hybrid programs and d$\mathcal{L}$ formulas in syntax, semantics, and completeness proofs, because 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 [35]. 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 [36] can be represented as hybrid programs using a straightforward generalisation of standard program encodings of automata. The fragment of hybrid programs without differential equations corresponds to conventional discrete programs generalised over the reals or to discrete-time dynamical systems [9]. The fragment without discrete jumps corresponds to switched continuous systems [9, 11], whereas the fragment of differential equations gives purely continuous dynamical systems [53]. Only the composition of mixed discrete jumps and continuous evolutions gives rise to truly hybrid behaviour.

2.1.2 Formulas of Differential Dynamic Logic

The formulas of d$\mathcal{L}$ are defined as in first-order dynamic logic [35]. 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 d$\mathcal{L}$ formula and α a hybrid program, then $[\alpha]\phi$, $\langle\alpha\rangle\phi$ are formulas (dynamic part).

Definition 2 (Formulas) The set $\mathrm{Fml}(\Sigma, V)$ of *formulas*, 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 Fig. 2, the following d$\mathcal{L}$ formula states that the train is able to leave region $z < m$ when it starts in the same region:

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

Note that, according to Definition 2, hybrid programs are fully internalised in d$\mathcal{L}$ 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 [8] 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 $p/q = c$.

2.2 Semantics of Differential Dynamic Logic

We define the semantics of 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 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.

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 of 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.

We will use $\nu\left[x \mapsto d\right]$ 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, $\eta\left[x \mapsto d\right]$ agrees with the assignment η except on $x \in V$, which is assigned $d \in \mathbb{R}$.

For terms and formulas, the *valuation* $val_{I,\eta}(\nu, \cdot)$ is defined as usual for first-order modal logic [27, 35] with a distinction of rigid and flexible functions [8]. 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 5.

Definition 3 (Valuation of terms) The *valuation of terms* with respect to 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. $val_{I,\eta}(\nu, f(\theta_1, \ldots, \theta_n)) = I(f)(val_{I,\eta}(\nu, \theta_1), \ldots, val_{I,\eta}(\nu, \theta_n))$ when $f \in \Sigma$ is a rigid function symbol of arity $n \geq 0$.

Definition 4 (Valuation of formulas) The *valuation, $val_{I,\eta}(\nu, \cdot)$, of formulas* with respect to interpretation I, assignment, η and state ν is defined as

1. $val_{I,\eta}(\nu, p(\theta_1, \ldots, \theta_n)) = I(p)(val_{I,\eta}(\nu, \theta_1), \ldots, val_{I,\eta}(\nu, \theta_n))$
2. $val_{I,\eta}(\nu, \phi \wedge \psi) = \mathit{true}$ iff $val_{I,\eta}(\nu, \phi) = \mathit{true}$ and $val_{I,\eta}(\nu, \psi) = \mathit{true}$. Accordingly for $\neg, \vee, \to$
3. $val_{I,\eta}(\nu, \forall x \phi) = \mathit{true}$ iff $val_{I,\eta[x\mapsto d]}(\nu, \phi) = \mathit{true}$ for all $d \in \mathbb{R}$
4. $val_{I,\eta}(\nu, \exists x \phi) = \mathit{true}$ iff $val_{I,\eta[x\mapsto d]}(\nu, \phi) = \mathit{true}$ for some $d \in \mathbb{R}$
5. $val_{I,\eta}(\nu, [\alpha]\phi) = \mathit{true}$ iff $val_{I,\eta}(\omega, \phi) = \mathit{true}$ for all states ω with $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$
6. $val_{I,\eta}(\nu, \langle\alpha\rangle\phi) = \mathit{true}$ iff $val_{I,\eta}(\omega, \phi) = \mathit{true}$ for some state ω with $(\nu, \omega) \in \rho_{I,\eta}(\alpha)$

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.

Definition 5 (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. $(\nu,\omega) \in \rho_{I,\eta}(x_1 := \theta_1, \dots, x_n := \theta_n)$ iff $\nu[x_1 \longmapsto val_{I,\eta}(\nu,\theta_1)] \dots [x_n \longmapsto val_{I,\eta}(\nu,\theta_n)]$ equals state ω. Particularly, the value of the other variables $z \notin \{x_1, \dots, x_n\}$ remains constant, i.e., $val_{I,\eta}(\nu, z) = val_{I,\eta}(\omega, z)$
2. $(\nu,\omega) \in \rho_{I,\eta}(x'_1 = \theta_1, \dots, x'_n = \theta_n \& \chi)$ iff there is a *flow f* of some duration $r \geq 0$ from ν to ω along $x'_1 = \theta_1, \dots, x'_n = \theta_n \& \chi$, i.e., a function $f : [0,r] \to \mathrm{Sta}(\Sigma)$ with $f(0) = \nu$, $f(r) = \omega$ respecting 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 $z \notin \{x_1, \dots, x_n\}$ remains constant, i.e., $val_{I,\eta}(f(\zeta), z) = val_{I,\eta}(\nu, z)$ for all $\zeta \in [0, r]$. Further, the invariant is respected, i.e., $val_{I,\eta}(f(\zeta), \chi) = true$ for each $\zeta \in [0, r]$.
3. $\rho_{I,\eta}(?\chi) = \{(\nu,\nu) : val_{I,\eta}(\nu,\chi) = true\}$
4. $\rho_{I,\eta}(\alpha \cup \beta) = \rho_{I,\eta}(\alpha) \cup \rho_{I,\eta}(\beta)$
5. $\rho_{I,\eta}(\alpha;\beta) = \{(\nu,\omega) : (\nu,z) \in \rho_{I,\eta}(\alpha), (z,\omega) \in \rho_{I,\eta}(\beta)$ for a state $z\}$
6. $(\nu,\omega) \in \rho_{I,\eta}(\alpha^*)$ iff there are an $n \in \mathbb{N}$ and states $\nu = \nu_0, \dots, \nu_n = \omega$ 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 [8] for a compatible semantics and calculus handling concurrent modifications of the same x_i.

For differential equations like $x' = \theta$, Definition 5 characterises transitions along a continuous evolution respecting the differential equation, see Fig. 3a. 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 Fig. 3b.

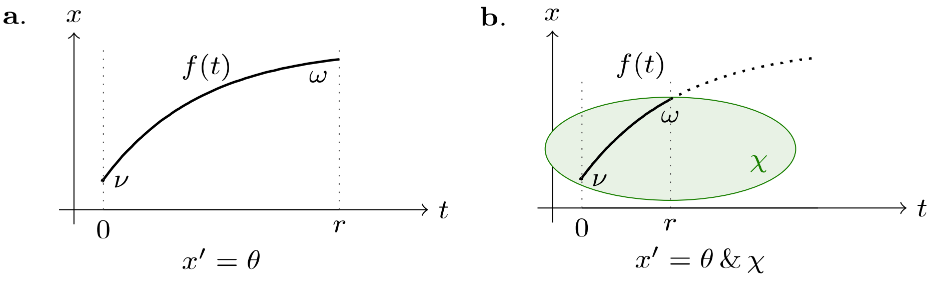

Fig. 3 Continuous flow along differential equation $x' = \theta$ over time t

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

Lemma 1 (Uniqueness) *Differential equations of* $\mathsf{d}\mathcal{L}$ *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* $f : [0, r] \longrightarrow \mathrm{Sta}(\Sigma)$ *satisfying the conditions of Case* 2 *of Definition* 5.

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 [58, Theorem 10.VI], which is also known as the Cauchy–Lipschitz theorem, 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, $\psi \rightarrow [\alpha]\phi$ is a natural rendition of the safety property that ϕ holds at all states reachable by α from initial states that satisfy ψ. Otherwise, $\mathsf{d}\mathcal{L}$ 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 [48].

3 Safety in the European Train Control System

As a case study to illustrate how $\mathsf{d}\mathcal{L}$ can be used for specifying and verifying hybrid systems, we examine a scenario of cooperating traffic agents in the *European Train Control System* (ETCS) [19]. 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 Fig. 4.

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

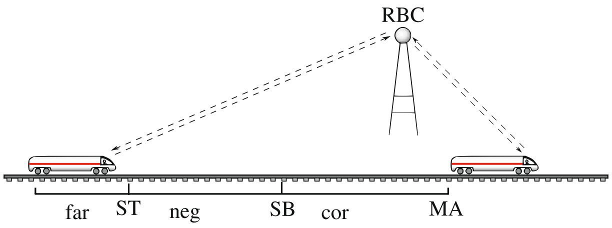

Fig. 4 ETCS train coordination protocol using dynamic movement authorities

for an MA-extension (negotiation phase *neg* of Fig. 4). 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 [18]. 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.

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 [18]. Depending on the current driving situation, the train controller determines a point SB (for start braking) up to 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 Fig. 4), 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., $m - s = \mathrm{SB}$). In this situation, $\mathsf{d}\mathcal{L}$ can analyse the following crucial safety property of ETCS:

$$\psi \rightarrow [(ctrl;\ drive)^*]z \le m \tag{1}$$

$$\text{where } ctrl \equiv (?m - z \le s; a := -b) \cup (?m - z \ge s; a := A)$$

$$drive \equiv \tau := 0; (z' = v, v' = a, \tau' = 1 \& v \ge 0 \land \tau \le \varepsilon).$$

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 [18], 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 $\rho_{I,\eta}((ctrl; drive)^*)$ is depicted in Fig. 5a. Figure 5b shows a possible run of the train where speed regulation successively decreases velocity v because MA has not been extended in time. 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.

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 (1) as in model checking approaches [19], 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 [18], we do not model weather conditions, slope of track, or train mass.

Because of its nonlinear behaviour and nontrivial reset relations, system (1) is beyond the modelling capabilities of linear hybrid automata [3, 29, 36] and beyond o-minimal automata [41]. Previous approaches need linear flows [3, 36], do not support the coupled dynamics caused by nontrivial resets [41], require polyhedral

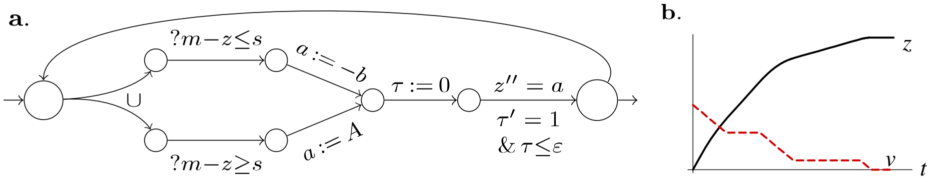

Fig. 5 ETCS transition structure and speed regulation of train speed control

initial sets and discrete dynamics [13], only handle robust systems with bounded regions [28], although parametric systems are not robust uniformly for all parameter choices, or they handle only bounded-time safety for systems with bounded switching [44]. Finally, in addition to general numerical limits [49], numerical approaches [5, 13] quickly become intractable due to the exponential impact of the number of variables.

4 Free Variable Calculus for Differential Dynamic Logic

In this section, we introduce a sequent calculus for verifying hybrid systems by proving corresponding d$\mathcal{L}$ 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.

4.1 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 d$\mathcal{L}$ 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 $1 \le i \le n$) is defined as usual; it is denoted by $\phi_{x_1}^{\theta_1} \ldots {}_{x_n}^{\theta_n}$. We assume α-conversion for renaming as needed. In the d$\mathcal{L}$ calculus, only admissible substitutions are applicable, which is crucial for soundness.

Definition 6 (Admissible substitution) An application of a substitution σ is *admissible* if no replaced term t occurs in the scope of a quantifier or modality binding a (logical or state) variable of t or of the replacement $\sigma(t)$. 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 $x' = \theta$).

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 [26, 27, 33], 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 [26, 27, 33]. The basis of d$\mathcal{L}$, instead, is first-order logic interpreted over the reals or in the theory of real-closed fields [54]. 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 [16, 54], 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 4.2).

Definition 7 (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 d$\mathcal{L}$ calculus, Fig. 6 provides *rule schemata* to which the following definition associates the calculus rules that are applicable in 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).

Definition 8 (Rules) The *rule schemata* in Fig. 6 induce *calculus rules* by:

1. If

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

is an instance of a P, G, or F1–F5 rule schema in Fig. 6, then

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

can be applied as a proof rule of the 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 Fig. 6, then

$$\frac{\Gamma \vdash \langle \mathcal{J} \rangle\, \phi_1, \Delta}{\Gamma \vdash \langle \mathcal{J} \rangle\, \phi_0, \Delta} \quad \text{and} \quad \frac{\Gamma, \langle \mathcal{J} \rangle\, \phi_1 \vdash \Delta}{\Gamma, \langle \mathcal{J} \rangle\, \phi_0 \vdash \Delta}$$

can both be applied as proof rules of the 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).

$$\text{(P1)}\ \frac{\phi \vdash}{\vdash \neg\phi} \quad \text{(P3)}\ \frac{\vdash \phi, \psi}{\vdash \phi \lor \psi} \quad \text{(P5)}\ \frac{\vdash \phi \quad \vdash \psi}{\vdash \phi \land \psi} \quad \text{(P7)}\ \frac{\phi \vdash \psi}{\vdash \phi \to \psi} \quad \text{(P9)}\ \frac{}{\phi \vdash \phi}$$

$$\text{(P2)}\ \frac{\vdash \phi}{\neg\phi \vdash} \quad \text{(P4)}\ \frac{\phi \vdash \quad \psi \vdash}{\phi \lor \psi \vdash} \quad \text{(P6)}\ \frac{\phi, \psi \vdash}{\phi \land \psi \vdash} \quad \text{(P8)}\ \frac{\vdash \phi \quad \psi \vdash}{\phi \to \psi \vdash} \quad \text{(P10)}\ \frac{\vdash \phi \quad \phi \vdash}{\vdash}$$

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

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

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

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

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

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

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

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

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

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.

Fig. 6 Rule schemata of the free variable calculus for differential dynamic logic

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 $\Phi_1 \vdash \Psi_1, .., \Phi_n \vdash \Psi_n$ 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.

P-Rules For propositional logic, standard rules P1–P9 with cut P10 are listed in Fig. 6. 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 $\phi \rightarrow \psi$ 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.

F-Rules The quantifier rules F1 and F2 correspond to the liberalised δ^+-rule of Hähnle and Schmitt [33]. F4 and F5 resemble the usual γ-rule but, unlike in [26, 27, 30, 33], 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 4.2.

D-Rules The rules for dynamic modalities transform a hybrid program into simpler logical formulas. Rules D1–D8 are as in discrete DL [8, 35]. Sequential compositions are proven using nested modalities (D1–D2), and nondeterministic 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 5).

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 8) 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* subsequent discrete jumps into a single discrete jump set (see previous results [8] for a compatible calculus detailing jump set merging, which works without the need to add vacuous identity jumps $a := a$):

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

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 $\langle x_1 := \vartheta_1{}^{\theta_1}_{x_1} \cdots {}^{\theta_n}_{x_n}, \ldots, x_n := \vartheta_n{}^{\theta_1}_{x_1} \cdots {}^{\theta_n}_{x_n}\rangle\phi$.

Given first-order definable flows for their differential equations, D11–D12 handle continuous evolutions (see [4, 41, 49] 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 $\tilde{t} \le t$. This constraint simplifies to *true* if χ is *true*. Similar simplifications can be made for convex invariant conditions (Section 6).

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 6) of the respective hybrid program α. This universal closure is required for soundness in the presence of contexts Γ, Δ (Definition 8) 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 the antecedent of their conclusion and their premiss holds.

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 [35] 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 $v \le 0$ 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:

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

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

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:

Definition 9 (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 8) 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 $\Phi \vdash_{d\mathcal{L}} \psi$, 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 $\Phi_0 \vdash \psi$.

4.2 Deduction Modulo with Invertible Quantifiers and Real Quantifier Elimination

The F-rules lift quantifier elimination to d$\mathcal{L}$ by following a generalised deduction modulo approach. They integrate decision procedures, e.g., for real quantifier elimination as a background prover [6] into the deductive proof system. Yet, unlike in the approaches of Dowek et al. [23] and Tinelli [56], the information given to the background prover is not restricted to ground formulas [56] or atomic formulas [23]. Further, real quantifier elimination is quite different from uninterpreted logic [26, 30, 33] 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 7) and has a decidable ground theory, for instance, the first-order theory of real arithmetic (i.e., the theory of real-closed fields [16, 54]). 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 d$\mathcal{L}$, 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.

4.2.1 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 d$\mathcal{L}$ interact to combine arithmetic with dynamic reasoning in hybrid systems, we analyse the braking behaviour in train control. The proof in Fig. 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 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 Fig. 7, the conjunction of its two goals can be handled by QE and simplification, yielding the resulting sub-goal:

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

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 $\exists a(-b \leq a \leq A \land [z'' = a]\, z \leq m)$.

4.2.2 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.

For the moment, suppose the rules did not contain QE. The requirement for admissible substitutions (Definition 6) ensures that the proof attempt of an invalid

$$\begin{array}{l|c}
 & v \geq 0, z < m \vdash v^2 > 2b(m - z) \\
\hline
\text{P7,P6} & \vdash v \geq 0 \land z < m \to v^2 > 2b(m - z) \\
\hline
 & v \geq 0, z < m \vdash -\frac{b}{2}T^2 + vT + z > m \\
\cline{2-2}
\text{F6} \quad \text{D9} & v \geq 0, z < m \vdash T \geq 0 \qquad v \geq 0, z < m \vdash \langle z := -\frac{b}{2}T^2 + vT + z\rangle z > m \\
\hline
\text{P5} & v \geq 0, z < m \vdash T \geq 0 \land \langle z := -\frac{b}{2}T^2 + vT + z\rangle z > m \\
\hline
\text{F4} & v \geq 0, z < m \vdash \exists t{\geq}0\, \langle z := -\frac{b}{2}t^2 + vt + z\rangle z > m \\
\hline
\text{D11} & v \geq 0, z < m \vdash \langle z' = v, v' = -b\rangle z > m \\
\hline
\text{P7,P6} & \vdash v \geq 0 \land z < m \to \langle z' = v, v' = -b\rangle z > m
\end{array}$$

Fig. 7 Deduction modulo for analysis of MA-violation in braking mode

formula in Fig. 8a cannot close in the 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 [16, 54]. Yet, when eliminating X in Fig. 8a, 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:

Lemma 2 (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 QE(ψ) be the quantifier-free formula belonging to ψ according to Definition 7. Then QE$(\psi)_{z_1}^{\theta_1} \ldots {}_{z_n}^{\theta_n}$ satisfies the requirements of Definition 7 for ϕ, because $\vDash \psi_{z_1}^{\theta_1} \ldots {}_{z_n}^{\theta_n} \leftrightarrow \text{QE}(\psi)_{z_1}^{\theta_1} \ldots {}_{z_n}^{\theta_n}$: For F defined as $\psi \leftrightarrow \text{QE}(\psi)$, 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, 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 Fig. 8a, $\exists X(2X + 1 < s(X))$ is no instance of first-order real arithmetic, because, by α-renaming, $(\exists X(2X + 1 < z))_z^{s(X)}$

$$\begin{array}{rl}
 & \text{F3 is not applicable} \\ \cline{2-2}
 & \vdash \text{QE}(\exists X\,(2X + 1 < s(X))) \\ \cline{2-2}
\text{F6} & \vdash 2X + 1 < s(X) \\ \cline{2-2}
\text{D9} & \vdash \langle x := 2X + 1\rangle (x < s(X)) \\ \cline{2-2}
\text{F1} & \vdash \forall y\, \langle x := 2X + 1\rangle (x < y) \\ \cline{2-2}
\text{F4} & \vdash \exists x\, \forall y\, \langle x := 2x + 1\rangle (x < y)
\end{array}$$

a.

$$\begin{array}{rl}
 & \vdash \overbrace{\text{QE}\,(\exists X\ \text{QE}(\forall s\,(2X + 1 < s)))}^{\text{false}} \\ \cline{2-2}
\text{F6} & \vdash \text{QE}(\forall s\,(2X + 1 < s)) \\ \cline{2-2}
\text{F3} & \vdash 2X + 1 < s(X) \\ \cline{2-2}
\text{D9} & \vdash \langle x := 2X + 1\rangle (x < s(X)) \\ \cline{2-2}
\text{F1} & \vdash \forall y\, \langle x := 2X + 1\rangle (x < y) \\ \cline{2-2}
\text{F4} & \vdash \exists x\, \forall y\, \langle x := 2x + 1\rangle (x < y)
\end{array}$$

b.

Fig. 8 Deduction modulo with invertible quantifiers. **a** Wrong rearrangement attempt. **b** Correct reintroduction order

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 Fig. 8b. 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.

4.2.3 Quantifier Elimination and Modalities

Quantifier elimination over the first-order theory of reals cannot handle modal formulas. Hence, the 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:

$$\mathrm{QE}(\exists X(X<0\land((\langle y:=2y+1\rangle\, y>0)\to X>y)))\equiv(\langle y:=2y+1\rangle\, y>0)\to y<0.$$

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 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 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.

Lemma 3 (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 $val_{I,\eta}(\nu,\cdot)$ and $\rho_{I,\eta}(\cdot)$ in Definitions 3–5. □

4.2.4 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 9). Universal quantifiers can be handled separately for conjunctions by $\forall x(\phi\land\psi)\equiv\forall x\phi\land\forall x\psi$. 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 [26]. By Lemma 3 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 [30]. 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.

5 Soundness and 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.

5.1 Soundness

We prove that a successful deduction in the $\mathsf{d}\mathcal{L}$ calculus always produces correct verification results about hybrid systems: The $\mathsf{d}\mathcal{L}$ 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 [33] to sequent calculus.

A formula ϕ is *satisfiable* [33] (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).

Theorem 1 (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 8, 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.

D9 Rule D9 is locally sound. Assume the premiss holds in I, η, ν, i.e., $I, \eta, \nu \models \phi_{x_1}^{\theta_1} \ldots_{x_n}^{\theta_n}$. 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 6). 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 that 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 $I, \eta^r, \nu \models \bar{\chi} \wedge \langle \mathcal{S}_t \rangle \phi$. Let $\mathcal{D}$ abbreviate $x_1' = \theta_1, \ldots, x_n' = \theta_n \& \chi$. We have to show that $I, \eta, \nu \models \langle \mathcal{D} \rangle \phi$. Equivalently, by Lemma 3, 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 5 for $\mathcal{D}$. In fact, f obeys the continuity and differentiability properties of Definition 5 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 $val_{I,\eta^r}(f(\zeta), \chi) = true$ 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 1.

F1 The proof is a sequent calculus adaptation of that in [33]. 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 $I, \eta, \nu \models \exists x \neg\phi(x)$. Then we construct an 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 3, 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 8 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 quantifier elimination 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 that $I, \eta, \nu \models \Psi(s(X_1, \ldots, X_n))$ by choosing $val_{I,\eta}(\nu, s(X_1, \ldots, X_n))$ for X in the premiss. By admissibility of substitutions, variables $X_1, \dots, X_n$ are free at all occurrences of $s(X_1, \dots, X_n)$, hence their value is the same in all occurrences.

F4 F4 is locally sound by a simplified version of the proof in [33]. 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 3, 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 [35] using universal closures for local soundness. G1–G2 are simple refinements of Lemma 3 using that $\forall^\alpha$ 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 $\forall^\alpha$, the premiss implies $I, \eta, \nu' \models \phi \rightarrow \psi$, hence $I, \eta, \nu' \models \psi$ and $I, \eta, \nu \models \langle\alpha\rangle \psi$. The proof of G1 is accordingly.

G3 For any I, η, ν with $I, \eta, \nu \models \forall^\alpha(\phi \rightarrow [\alpha]\phi)$, we conclude that $I, \eta, \nu' \models \phi \rightarrow [\alpha]\phi$ for all ν' with $(\nu, \nu') \in \rho_{I,\eta}(\alpha)$. As these share the same η, we can further conclude $I, \eta, \nu \models \phi \rightarrow [\alpha^*]\phi$ 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 $I, \eta[v \mapsto d], \nu' \models v > 0 \land \varphi(v) \rightarrow \langle\alpha\rangle\varphi(v-1)$ 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 $I, \eta[v \mapsto d], \nu \models v > 0 \land \varphi(v) \rightarrow \langle\alpha\rangle\varphi(v-1)$. 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 $d \leq 0$. $\square$

5.2 Completeness

Theorem 1 shows that all provable closed d$\mathcal{L}$ formulas are valid. The converse question is whether the d$\mathcal{L}$ calculus is *complete*, i.e., all valid d$\mathcal{L}$ formulas are provable. Combining completeness for first-order logic [33] and decidability of real-arithmetic [16], 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, d$\mathcal{L}$ is not axiomatisable and, unlike its basis of first-order *real* arithmetic, $\mathsf{d}\mathcal{L}$ 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 [36], we show that even the purely discrete and purely continuous parts of $\mathsf{d}\mathcal{L}$ are not effectively axiomatisable. Hence, valid $\mathsf{d}\mathcal{L}$ formulas are not always provable.

Theorem 2 (Incompleteness) *Both the discrete fragment and the continuous fragment of* $\mathsf{d}\mathcal{L}$ *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 $\mathsf{d}\mathcal{L}$ interpretations in both fragments. Then these fragments extend first-order *integer* arithmetic such that the incompleteness theorem of Gödel [31] applies. Natural numbers are definable in the discrete fragment without continuous evolutions using repetitive additions:

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

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

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

These differential equations characterise sin and cos as unique solutions for s and c, 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 $k \in \mathbb{N}$. The nonzero initial values for s and c prevent the trivial solution identical to 0. □

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 [17, 34, 35]. In calculi for discrete programs, completeness is proven relative to the handling of data [17, 34, 35]. For hybrid systems, this is inadequate: By Theorem 2, 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 [54].

According to Theorem 2, 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 [17, 34, 35] 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 $[x_1'=\theta_1, \dots, x_n'=\theta_n]F$ with a first-order formula F. Dually, $\langle x_1'=\theta_1, \dots, x_n'=\theta_n\rangle F$ is expressible as $\neg[x_1'=\theta_1, \dots, x_n'=\theta_n]\neg F$.

Theorem 3 (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 Appendix A, adapts the techniques of Cook [17] and Harel [34, 35] 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 $\mathsf{d}\mathcal{L}$. For this, we show that $\mathsf{d}\mathcal{L}$ formulas can be expressed equivalently in FOD, and that valid $\mathsf{d}\mathcal{L}$ formulas can be derived from corresponding FOD axioms in the $\mathsf{d}\mathcal{L}$ 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 3 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 Theorems 1 and 3, the $\mathsf{d}\mathcal{L}$ calculus axiomatises the transition behaviour of hybrid systems completely relative to the handling of differential equations!

5.3 Relatively Semidecidable Fragments

To strengthen the completeness result from Theorem 3, we consider fragments of $\mathsf{d}\mathcal{L}$ 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 $\mathsf{d}\mathcal{L}$ 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 $\mathsf{d}\mathcal{L}$ calculus. If an imperfect oracle chooses inadequate (in)variants, applying the $\mathsf{d}\mathcal{L}$ calculus rules results in goals that are not valid, which is again decidable by quantifier elimination in the $\mathsf{d}\mathcal{L}$ calculus.

Theorem 4 (Relatively semidecidable fragment) *Relative to an oracle generating first-order invariants and variants, the* $\mathsf{d}\mathcal{L}$ *calculus gives a backtracking-free semideci-*

sion procedure for (closed) d$\mathcal{L}$ *formulas with differential equations having first-order definable flows.*

Proof (*Outline*) The (constructive) proof, which, in full, is contained in Appendix B, shows that there are always applicable d$\mathcal{L}$ rules that transform the formulas equivalently and that formulas in this d$\mathcal{L}$ 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 semi-decision procedure for the fragment of Theorem 4. As a corollary to Theorem 2 and Theorem 4, there are valid d$\mathcal{L}$ formulas that need proper d$\mathcal{L}$ (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 4 can be obtained by unrolling, e.g., by D5.

6 Verifying Safety in the European Train Control System

Finding Inductive Candidates We want to prove safety statement (1) of the ETCS from Section 3. 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 Fig. 7, an unwinding of the loop in (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:

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

Inductive Verification Using G3 to prove (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 *ctrl*; *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 has been given, since we want to identify the prerequisite ψ for safe driving by proof analysis. In the proof of the induction step $\phi \to [ctrl; drive]\phi$, 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 can be proven in d$\mathcal{L}$ 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):

$$\text{G3}\;\dfrac{\text{F1}\;\dfrac{\text{P7}\;\dfrac{\text{D2}\;\dfrac{\text{D4,P5}\;\dfrac{\dfrac{\dots}{\phi \vdash \langle a := -b\rangle[drive]\phi} \qquad \text{D8,P7}\;\dfrac{\dfrac{\dots}{\phi, m - z \geq s \vdash \langle a := A\rangle[drive]\phi}}{\phi \vdash [?m-z \geq s; a := A][drive]\phi}}{\phi \vdash [ctrl][drive]\phi}}{\phi \vdash [ctrl; drive]\phi}}{\vdash \phi \to [ctrl; drive]\phi}}{\vdash \forall^\alpha (\phi \to [ctrl; drive]\phi)}}{\phi \vdash [(ctrl; drive)^*]\phi}$$

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 [58, §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 $\forall 0{\le}\tilde{t}{\le}t\langle \mathcal{S}_{\tilde{t}}\rangle\chi$ of D12 can be simplified to $\langle \mathcal{S}_{\tilde{t}}\rangle\chi$ 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 as $*$):

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

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 $\langle \mathcal{S}_t\rangle$.

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

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:

$$s \geq \frac{v^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v\right). \tag{3}$$

This constraint can be discovered automatically in the above proof by the indicated application of F3 using quantifier elimination with some simplifications. Constraint (3) is required to make sure invariant (2) still holds after accelerating. In fact, augmenting the case study with (3) makes the argument inductive, and the whole proof of the safety statement (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) is a controllability constraint, see Section 4.2.1.

Quite unlike in the acceleration-free case [47], constraint (3) 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 (3), which admits complex behaviour like in Fig. 5b. 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 d$\mathcal{L}$ calculus, similar constraints can be derived [48] 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 Fig. 4.

For the resulting ETCS system, liveness can be proven in the d$\mathcal{L}$ calculus by showing that the train can pass every point p by an appropriate choice of m by the RBC:

$$v = v_0 > 0 \land \varepsilon > 0 \land b > 0 \land A \geq 0 \to \forall p \exists m \langle (ctrl; drive)^* \rangle z \geq p \tag{4}$$

The proof of property (4) uses the variant $\varphi(n) \equiv z + n\varepsilon v_0 \geq p \land v \geq v_0$ 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.

In this example, we can see the effect of the 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 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 [46] for the d$\mathcal{L}$ calculus, the safety statement (1) with invariant (2) can be verified automatically in a theorem prover that invokes Mathematica for D11–D12, F3, and F6.

7 Conclusions and Future Work

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, d$\mathcal{L}$, we have presented a concise generalised free variable proof calculus over the reals.

Our sequent calculus for d$\mathcal{L}$ is a generalisation of classical calculi for discrete dynamic logic [7, 8, 34, 35] 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 [8] to support reasoning about dynamically reconfiguring system structures, which we want to extend to hybrid systems in future work. Further, we will develop Lyapunov-like techniques to handle differential equations in $\mathsf{d}\mathcal{L}$ without solving them. 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.

Acknowledgements I am grateful to the anonymous referees for their most valuable and helpful comments. I also thank Ernst-Rüdiger Olderog and Andreas Schäfer for their proofreading remarks. This research was partially supported by a fellowship of the German Academic Exchange Service (DAAD) during a research visit at the Carnegie Mellon University (CMU) and by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS).

Appendix A: Relative Completeness Proof

In this section, we present a fully constructive proof of Theorem 3, which generalises the techniques of Harel [34, 35] and Cook [17] to the hybrid case. It shows that for every valid $\mathsf{d}\mathcal{L}$ formula, there is a finite set of valid FOD-formulas from which it can be derived in the $\mathsf{d}\mathcal{L}$ calculus. See the proof outline in Section 5.2 for a road map of the proof.

Natural numbers are definable in FOD by Theorem 2. In this section, we abbreviate quantifiers over natural numbers, e.g., $\forall x(nat(x) \to \phi)$ by $\forall x : \mathbb{N}\, \phi$ and $\exists x$ by $\exists x : \mathbb{N}\, \phi$. Likewise, we abbreviate quantifiers over integers, e.g., $\forall x(nat(x) \lor nat(-x)) \to \phi$ by $\forall x : \mathbb{Z}\, \phi$.

A.1 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 [43], 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.

Lemma 4 ($\mathbb{R}$-Gödel encoding) *The formula* $\mathrm{at}(Z, n, j, z)$*, 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 a position j with $1 \le j \le n$), is definable in FOD. For a formula $\phi(z)$ we abbreviate* $\exists z(\mathrm{at}(Z, n, j, z) \land \phi(z))$ *by* $\phi(Z_j^{(n)})$.

Proof The basic idea of the $\mathbb{R}$-Gödel encoding is to interleave the bits of real numbers as depicted in Fig. 9a (for a pairing of $n = 2$ numbers a and b). For defin-

$$\begin{tikzcd}
\displaystyle\sum_{i=0}^{\infty}\frac{a_i}{2^i} = a_0.a_1a_2\ldots \arrow[dr, leftrightarrow] & \\
& \displaystyle\sum_{i=0}^{\infty}\left(\frac{a_i}{2^{2i-1}}+\frac{b_i}{2^{2i}}\right) = a_0b_0.a_1b_1a_2b_2\ldots \\
\displaystyle\sum_{i=0}^{\infty}\frac{b_i}{2^i} = b_0.b_1b_2\ldots \arrow[ur, leftrightarrow] &
\end{tikzcd}$$

a.

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

b.

Fig. 9 Characterising Gödel encoding of $\mathbb{R}$-sequences in one real number. **a** Fractional encoding principle by bit interleaving. **b** Definition of $\mathbb{R}$-Gödel encoding in FOD

ing $\mathrm{at}(Z,n,j,z)$, we use several auxiliary functions to improve readability, see Fig. 9b. 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 $\mathrm{bit}(a,i)$ gives the i-th bit of $a\in\mathbb{R}$ when represented with basis 2. For $i>0$, $\mathrm{bit}(a,i)$ yields fractional bits, and, for $i\leq 0$, it yields bits of the integer part. For instance, $\mathrm{bit}(a,1)$ yields the first fractional bit, $\mathrm{bit}(a,0)$ 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, $\mathrm{intpart}(a)$ and $\mathrm{bit}(a,i)$ 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 $i<0$. □

A.2 Expressibility and Rendition of Hybrid Program Semantics

In order to show that $\mathsf{d}\mathcal{L}$ 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 [10, 32], 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 $\mathbb{R}^k$).

Lemma 5 (Program rendition) *For every hybrid program α with variables $\vec{x} = x_1, \ldots, x_k$ there is a FOD-formula $\mathcal{S}_\alpha(\vec{x}, \vec{v})$ with variables among the $2k$ distinct variables $\vec{x} = x_1, \ldots, x_k$ and $\vec{v} = v_1, \ldots, v_k$ 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,

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

Proof By Lemma 3, 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 Fig. 10. To simplify the notation, we assume that all variables $x_1, \ldots, x_k$ 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.

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 1) 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

$$\mathcal{S}_{x_1:=\theta_1,..,x_k:=\theta_k}(\vec{x}, \vec{v}) \equiv \bigwedge_{i=1}^{k} (v_i = \theta_i)$$

$$\mathcal{S}_{x'_1=\theta_1,..,x'_k=\theta_k}(\vec{x}, \vec{v}) \equiv \langle x'_1 = \theta_1, .., x'_k = \theta_k\rangle\, \vec{v} = \vec{x}$$

$$\mathcal{S}_{x'_1=\theta_1,..,x'_k=\theta_k \,\&\, \chi}(\vec{x}, \vec{v}) \equiv \exists t\, \big(t = 0 \land \langle x'_1 = \theta_1, .., x'_k = \theta_k, t' = 1\rangle \big(\vec{v} = \vec{x} \land [x'_1 = -\theta_1, .., x'_k = -\theta_k, t' = -1](t \geq 0 \to \chi)\big)\big)$$

$$\mathcal{S}_{?\chi}(\vec{x}, \vec{v}) \equiv \vec{v} = \vec{x} \land \chi$$

$$\mathcal{S}_{\beta\cup\gamma}(\vec{x}, \vec{v}) \equiv \mathcal{S}_\beta(\vec{x}, \vec{v}) \lor \mathcal{S}_\gamma(\vec{x}, \vec{v})$$

$$\mathcal{S}_{\beta;\gamma}(\vec{x}, \vec{v}) \equiv \exists \vec{z} (\mathcal{S}_\beta(\vec{x}, \vec{z}) \land \mathcal{S}_\gamma(\vec{z}, \vec{v}))$$

$$\mathcal{S}_{\beta^*}(\vec{x}, \vec{v}) \equiv \exists Z\, \exists n : \mathbb{N}\, \big(Z_1^{(n)} = \vec{x} \land Z_n^{(n)} = \vec{v} \land \forall i : \mathbb{N}\, (1 \leq i < n \to \mathcal{S}_\beta(Z_i^{(n)}, Z_{i+1}^{(n)}))\big)$$

Fig. 10 Explicit rendition of hybrid program transition semantics in FOD

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 Fig. 11.

To show reversibility, let $(\nu, \omega) \in \rho_{I,\eta}(x_1' = \theta_1, \dots, x_k' = \theta_k)$, that is, let $f : [0, r] \to \mathrm{Sta}(\Sigma)$ be a solution of $x_1' = \theta_1, \dots, x_k' = \theta_k$ starting in 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 $x_1' = -\theta_1, \dots, x_k' = -\theta_k$, which can be seen for $1 \le i \le k$ as follows:

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

Unlike all other cases, case $\mathcal{S}_{x_1'=\theta_1,\dots,x_k'=\theta_k \& \chi}(\vec{x}, \vec{v})$ in Fig. 10 uses nested FOD modalities. Yet nested modalities can be avoided in $\mathcal{S}_\alpha(\vec{x}, \vec{v})$ using an equivalent FOD formula without them, see Fig. 11:

$$\exists t \exists r \big(t = 0 \wedge \langle x_1' = \theta_1, \dots, x_k' = \theta_k, t' = 1\rangle (\vec{v} = \vec{x} \wedge r = t) \wedge$$
$$\forall \vec{x} \forall t (\vec{x} = \vec{v} \wedge t = r \to [x_1' = -\theta_1, \dots, x_k' = -\theta_k, t' = -1](t \ge 0 \to \chi))\big)\,.$$

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 4 in Fig. 10 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 5 to characterise modalities, we prove that every d$\mathcal{L}$ formula can be expressed equivalently in FOD by structural induction.

Lemma 6 (Expressibility) d$\mathcal{L}$ *is* expressible *in FOD: for all* 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.,* $\models \phi \leftrightarrow \phi^\#$. *The converse holds trivially.*

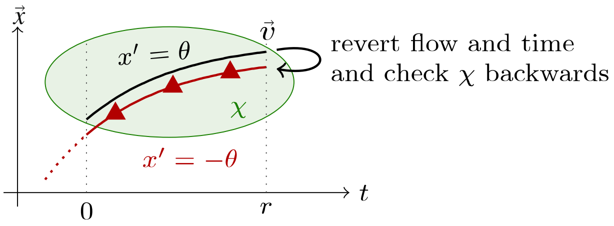

Fig. 11 Invariant region checks along backwards flow over time t

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 $\models \varphi \leftrightarrow \varphi^{\#}$ and $\models \psi \leftrightarrow \psi^{\#}$, from which we can conclude by congruence that $\models (\varphi \vee \psi) \leftrightarrow (\varphi^{\#} \vee \psi^{\#})$ giving $\models \phi \leftrightarrow \phi^{\#}$ 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 5:

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

3. The case where ϕ is $[\alpha]\psi$ is again a consequence of Lemma 5:

$$\models \langle\alpha\rangle\,\psi \leftrightarrow \forall\vec{v}(\mathcal{S}_\alpha(\vec{x},\vec{v}) \rightarrow \psi^{\#}{}_{\vec{x}}^{\vec{v}}) \ .$$

□

The above proofs directly carry over to rich test d$\mathcal{L}$, i.e., the logic where 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 Fig. 10. Accordingly, nested modalities can be avoided in FOD by using the following formula for $\mathcal{S}_{x_1'=\theta_1,..,x_k'=\theta_k\&\chi}(\vec{x},\vec{v})$:

$$\exists t\exists r\big(t=0 \wedge \big\langle x_1'=\theta_1,..,x_k'=\theta_k, t'=1\big\rangle(\vec{v}=\vec{x} \wedge r=t) \wedge$$
$$\forall\vec{z}(\exists\vec{x}\exists t(\vec{x}=\vec{v} \wedge t=r \wedge \langle x_1'=-\theta_1,..,x_k'=-\theta_k, t'=-1\rangle(t \geq 0 \wedge \vec{z}=\vec{x})) \rightarrow \chi^{\#}{}_{\vec{x}}^{\vec{z}})\big) \ .$$

A.3 Relative Completeness of First-order Assertions

As special cases of Theorem 3, 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 d$\mathcal{L}$ formula ϕ is derivable (Definition 9) from a set of FOD-tautologies, which is equivalent to saying that ϕ is derivable in the 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: $\exists x\phi \equiv \big\langle x'=1\big\rangle\phi \vee \big\langle x'=-1\big\rangle\phi$. 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 8, and we use it implicitly together with P10 in the following. Derivability of sequents and corresponding formulas is equivalent by the following lemma.

Lemma 7 (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 $\phi \vdash_{\mathcal{D}} \psi$:

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

The converse direction is by an application of P7. □

Lemma 8 (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 1, it is easy to see for discrete jump sets $\mathcal{I}$ and $\mathcal{J}$ that the d$\mathcal{L}$ 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 8. Applicability conditions of rules do not depend on jump prefixes, as Definition 8 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

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

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

$$\dfrac{\Gamma, \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_1 \vdash \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Psi_1, \Delta \quad \dots \quad \Gamma, \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_n \vdash \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Psi_n, \Delta}{\Gamma, \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Phi_0 \vdash \langle \mathcal{I} \rangle \langle \mathcal{J} \rangle \Psi_0, \Delta} \tag{5}$$

Next, we show that these nested jump prefixes can be reduced to a single jump prefix as Definition 8 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 4.1. We replace each rule application (with nested prefixes) of the form (5) by the following derivation with only a single prefix (assuming $n = 1$ for notational convenience):

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

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 (5), which is applicable on $\Phi_0 \vdash \Psi_0$ for *any* context by Definition 8, 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 (5). 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 (5):

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

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 6). Rule F1 is applicable for Skolem constant s as no free logical variables occur in the proof. □

Proposition 1 (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

$$\vDash F \to [\alpha]G \textit{ implies } \vdash_{\mathcal{D}} F \to [\alpha]G \quad (\textit{and } F \vdash_{\mathcal{D}} [\alpha]G \textit{ by Lemma } 7).$$

Proof We generalise the relative completeness proof by Cook [17] to 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 $x_1 := \theta_1 \dots, x_n := \theta_n$, $?\chi$, $\beta \cup \gamma$, 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 6, 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 8, 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 7 or P7 yield $\vdash_{\mathcal{D}} F \to [\beta;\gamma]G$).

3. $\models F \to [x_1' = \theta_1 \dots, x_n{}' = \theta_n]G$ is a FOD-formula and hence derivable as a $\mathcal{D}$ axiom. Continuous evolution $x_1' = \theta_1 \dots, x_n' = \theta_n \& \chi$ with invariant regions is definable in FOD by Lemma 5, which we consider as an abbreviation in this proof.
4. $\models F \to [\beta^*]G$ 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 6:

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

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 7. By Lemma 8 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 8, 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 $F \vdash_{\mathcal{D}} [\beta^*]G$. □

Proposition 2 (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

$$\models F \to \langle\alpha\rangle G \text{ implies } \vdash_{\mathcal{D}} F \to \langle\alpha\rangle G \text{ (and } F \vdash_{\mathcal{D}} \langle\alpha\rangle G \text{ by Lemma 7)} .$$

Proof We generalise the arithmetic completeness proof by Harel [34] to the hybrid case. Most cases of the proof are simple adaptations of the corresponding cases in Proposition 1. What remains to be shown is the case of repetitions. Assume that $\models F \to \langle\beta^*\rangle G$. 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 5–6 as $(\langle\beta^*\rangle G)^{\#} \equiv \exists\vec{v}(\mathcal{S}_{\beta^*}(\vec{x}, \vec{v}) \land G_{\vec{x}}^{\vec{v}})$, *except* that the quantifier on the repetition count n is removed such that n becomes a free variable (plus index shifting to count repetitions):

$$\varphi(n-1) \equiv \exists\vec{v}\,\exists Z\big(Z_1^{(n)} = \vec{x} \land Z_n^{(n)} = \vec{v} \land \forall i : \mathbb{N}\ (1 \le i < n \to \mathcal{S}_{\beta}(Z_i^{(n)}, Z_{i+1}^{(n)})) \land G_{\vec{x}}^{\vec{v}}\big) .$$

By Lemma 4, $\varphi(n)$ can only hold true if n is a natural number.

According to the loop semantics, $\models n > 0 \land \varphi(n) \to \langle\beta\rangle\varphi(n-1)$ 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 8, we extend this derivation to $\vdash_{\mathcal{D}} \forall^{\beta}\forall n{>}0(\varphi(n) \to \langle\beta\rangle\varphi(n-1))$. Thus $\exists v\,\varphi(v) \vdash_{\mathcal{D}} \langle\beta^*\rangle\exists v{\le}0\varphi(v)$ 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:

- $\models F \to \exists v\,\varphi(v)$, because $\models F \to \langle\beta^*\rangle G$, and
- $\models (\exists v{\le}0\varphi(v)) \to G$, because $v{\le}0$ and the fact, that, by Lemma 4, $\varphi(v)$ 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 $\vdash_{\mathcal{D}} \forall^{\beta}(\exists v{\le}0\,\varphi(v) \to G)$ by Lemma 8 and extend the derivation to $\langle\beta^*\rangle\exists v{\le}0\,\varphi(v) \vdash_{\mathcal{D}} \langle\beta^*\rangle G$ by G2. From $\vdash_{\mathcal{D}} F \to \exists v\,\varphi(v)$ we conclude $F \vdash_{\mathcal{D}} \exists v\,\varphi(v)$ by Lemma 7. Now, the above derivations can be combined propositionally by a cut with $\langle\beta^*\rangle\exists v{\le}0\,\varphi(v)$ and with $\exists v\,\varphi(v)$ to yield $F \vdash_{\mathcal{D}} \langle\beta^*\rangle G$. □

A.4 Relative Completeness of the Differential Logic Calculus

Having succeeded with the proofs of the above statements we can finish the proof of the Theorem 3, which is the central result of this work.

Proof (of Theorem 3) The proof follows a basic structure analogous to that of Harel's proof for the discrete case [34, Theorem 3.1]. We have to show that every valid d$\mathcal{L}$ formula ϕ can be proven from FOD axioms within the 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 $\phi_1 \to [\alpha]\phi_2$ or $\phi_1 \to \langle\alpha\rangle\phi_2$ logically. Next, we express subformulas ϕ_i equivalently in FOD by Lemma 6, and use Proposition 1 and 2 to resolve these first-order safety or liveness assertions. Finally, we prove that the original 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 by $\exists x\phi \equiv \langle x' = 1\rangle\phi \vee \langle x' = -1\rangle\phi$ 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):

$$\phi_1 \vee [\alpha]\phi_2$$
$$\phi_1 \vee \langle\alpha\rangle\phi_2$$

   As a unified notation for those cases we use $\phi_1 \vee \{\!|\alpha|\!\}\phi_2$. Then, $|\phi_2| < |\phi|$, since ϕ_2 has less modalities. Likewise, $|\phi_1| < |\phi|$ because $\{\!|\alpha|\!\}\phi_2$ contributes one modality to $|\phi|$ that is not part of ϕ_1.
   According to Lemma 6 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 that $\vDash \phi_1^{\#} \vee \{\!|\alpha|\!\}\phi_2^{\#}$, which directly implies $\vDash \neg\phi_1^{\#} \to \{\!|\alpha|\!\}\phi_2^{\#}$. Then by Proposition 1 or 2, respectively, we can derive

$$\neg\phi_1^{\#} \vdash_{\mathcal{D}} \{\!|\alpha|\!\}\phi_2^{\#}. \tag{6}$$

Further $\vDash \phi_1 \leftrightarrow \phi_1^\#$ implies $\vDash \neg\phi_1 \to \neg\phi_1^\#$, which is derivable by IH, because $|\phi_1| < |\phi|$. By Lemma 7, we obtain $\neg\phi_1 \vdash_{\mathcal{D}} \neg\phi_1^\#$, which we combine with (6) by a cut with $\neg\phi_1^\#$ to

$$\neg\phi_1 \vdash_{\mathcal{D}} \langle\!\![\alpha]\!\!\rangle\phi_2^\# \ . \tag{7}$$

Likewise $\vDash \phi_2 \leftrightarrow \phi_2^\#$ implies $\vDash \phi_2^\# \to \phi_2$, which is derivable by IH, as $|\phi_2| < |\phi|$. We can extend the derivation of $\vdash_{\mathcal{D}} \phi_2^\# \to \phi_2$ to one of $\vdash_{\mathcal{D}} \forall^\alpha(\phi_2^\# \to \phi_2)$ by Lemma 8 and conclude $\langle\!\![\alpha]\!\!\rangle\phi_2^\# \vdash_{\mathcal{D}} \langle\!\![\alpha]\!\!\rangle\phi_2$ by G1–G2. Finally we combine the latter propositionally with (7) by a cut with $\langle\!\![\alpha]\!\!\rangle\phi_2^\#$ to derive $\neg\phi_1 \vdash_{\mathcal{D}} \langle\!\![\alpha]\!\!\rangle\phi_2$, from which $\vdash_{\mathcal{D}} \phi_1 \vee \langle\!\![\alpha]\!\!\rangle\phi_2$ can be obtained, again using P10, to complete the proof. □

Appendix B: Relative Semidecidability Proof

As an auxiliary result for proving Theorem 4, we show that, in d$\mathcal{L}$ proofs, Skolem symbols occur in a uniform way, i.e., a Skolem symbol s always occurs with the same list of arguments.

Lemma 9 (Uniform Skolem symbols) Let ϕ be a d$\mathcal{L}$ 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 d$\mathcal{L}$ 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 $s(X_1, \dots, X_n)$ introduced by rules F1–F2 are of the required form as the X_i are precisely the free logical variables. In addition, the symbol $s(X_1, \dots, X_n)$ 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), 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 d$\mathcal{L}$ calculus preserve the property as they never replace arguments of Skolem function symbols (which are free variables by induction hypothesis). □

Proof (of Theorem 4) 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 $\mathsf{d}\mathcal{L}$ 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 $\mathsf{d}\mathcal{L}$ 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 7), 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 4.1. 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 [22]. It lifts to sequents in rule applications (Definition 8) 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 [22].

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 P6 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 $[x_1' = \theta_1, \dots, x_n' = \theta_n \& \chi]\psi$, 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 $\langle x_1' = \theta_1, \dots, x_n' = \theta_n \& \chi\rangle\psi$ 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 $\langle x_1 := \theta_1, \dots, x_n := \theta_n\rangle\psi$, there are two cases. If D9 is applicable, it yields equivalent simpler sequents. Otherwise, we have $\psi \prec \langle x_1 := \theta_1, \dots, x_n := \theta_n\rangle\psi$. Thus, by IH, there is a finite sequence of rule applications on ψ yielding equivalent sequents with atomic formulas. Prefixing the resulting proof with $\langle x_1 := \theta_1, \dots, x_n := \theta_n\rangle$, yields a corresponding proof for $\Gamma \vdash \phi, \Delta$ by Lemma 8. The formulas of its open branches resulting from ϕ are of the form $\langle x_1 := \theta_1, \dots, x_n := \theta_n\rangle G$ for atomic formulas G, where, at the latest, D9 is applicable, as substitutions are admissible on atomic formulas. Case $[x_1 := \theta_1, \dots, x_n := \theta_n]\psi$ is similar, using D10 first.
6. If ϕ is of the form $\forall x \psi(x)$, we can apply F1 giving $\psi(s(X_1, \dots, X_n))$. Now, we have $\psi(s(X_1, \dots, X_n)) \prec \forall x \psi(x)$, hence, by IH, $\psi(s(X_1, \dots, X_n))$ can be transformed equivalently to a set of sequents of the form $\Phi_i(s(X_1, \dots, X_n)) \vdash \Psi_i(s(X_1, \dots, X_n))$ with atomic formulas (without loss of generality, we can assume $s(X_1, \dots, X_n)$ 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: $\Gamma \vdash \psi(s(X_1, \dots, X_n), \Delta$ is equivalent to $\bigwedge_i(\Phi_i(s(X_1, \dots, X_n)) \vdash \Psi_i(s(X_1, \dots, X_n)))$ 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 $\mathrm{QE}(\exists X(\bigwedge_i(\Phi_i \vdash \Psi_i)))$, 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 $s(X_1, \dots, X, \dots, X_n)$ occurs in a $\Phi_i \vdash \Psi_i$, then, by IH, the Skolem function s already occurred in $\psi(X)$. By Lemma 9, $s(X_1, \dots, X, \dots, X_n)$ 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 9. After applying F6 the additional free variable X disappears. □

References

1. Ábrahám-Mumm, E., Steffen, M., Hannemann, U.: Verification of hybrid systems: formalization and proof rules in PVS. In: ICECCS, pp. 48–57. IEEE Computer Society, Los Alamitos (2001). doi:10.1109/ICECCS.2001.930163
2. Alur, R., Courcoubetis, C., Dill, D.L.: Model-checking for real-time systems. In: LICS, pp. 414–425. IEEE Computer Society, Los Alamitos (1990)
3. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theor. Comput. Sci. **138**(1), 3–34 (1995). doi:10.1016/0304-3975(94)00202-T
4. Anai, H., Weispfenning, V.: Reach set computations using real quantifier elimination. In: Benedetto, M.D.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC, *LNCS*, vol. 2034, pp. 63–76. Springer, Berlin (2001). doi:10.1007/3-540-45351-2_9
5. Asarin, E., Dang, T., Girard, A.: Reachability analysis of nonlinear systems using conservative approximation. In: Maler, O., Pnueli, A. (eds.) Hybrid Systems: Computation and Control, 6th International Workshop, HSCC 2003 Prague, Czech Republic, April 3–5, 2003, Proceedings, *LNCS*, vol. 2623, pp. 20–35. Springer, Berlin (2003). doi:10.1007/3-540-36580-X_5
6. Beckert, B.: Equality and other theories. In: D'Agostino, M., Gabbay, D., Hähnle, R., Posegga, J. (eds.) Handbook of Tableau Methods. Kluwer, Deventer (1999)

7. Beckert, B., Hähnle, R., Schmitt, P.H. (eds.): Verification of Object-Oriented Software: The KeY Approach, *LNCS*, vol. 4334. Springer, Berlin (2007)
8. Beckert, B., Platzer, A.: Dynamic logic with non-rigid functions: a basis for object-oriented program verification. In: Furbach, U., Shankar, N. (eds.) IJCAR, *LNCS*, vol. 4130, pp. 266–280. Springer, Berlin (2006)
9. Branicky, M.S.: Studies in hybrid systems: modeling, analysis, and control. Ph.D. thesis, Dept. Elec. Eng. and Computer Sci., Massachusetts Inst. Technol., Cambridge, MA (1995)
10. Branicky, M.S.: Universal computation and other capabilities of hybrid and continuous dynamical systems. Theor. Comput. Sci. **138**(1), 67–100 (1995). doi:10.1016/0304-3975(94)00147-B
11. Branicky, M.S., Borkar, V.S., Mitter, S.K.: A unified framework for hybrid control: model and optimal control theory. IEEE Trans. Automat. Contr. **43**(1), 31–45 (1998). doi:10.1109/9.654885
12. Chaochen, Z., Ji, W., Ravn, A.P.: A formal description of hybrid systems. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) Hybrid Systems, *LNCS*, vol. 1066, pp. 511–530. Springer, Berlin (1995)
13. Chutinan, A., Krogh, B.H.: Computational techniques for hybrid system verification. IEEE Trans. Automat. Contr. **48**(1), 64–75 (2003). doi:10.1109/TAC.2002.806655
14. Clarke, E.M., Fehnker, A., Han, Z., Krogh, B.H., Ouaknine, J., Stursberg, O., Theobald, M.: Abstraction and counterexample-guided refinement in model checking of hybrid systems. Int. J. Found. Comput. Sci. **14**(4), 583–604 (2003)
15. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT, Cambridge (1999)
16. Collins, G.E., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. J. Symb. Comput. **12**(3), 299–328 (1991)
17. Cook, S.A.: Soundness and completeness of an axiom system for program verification. SIAM J. Comput. **7**(1), 70–90 (1978). doi:10.1137/0207005
18. Damm, W., Hungar, H., Olderog, E.R.: Verification of cooperating travel agents. Int. J. Control **79**(5), 395–421 (2006)
19. Damm, W., Mikschl, A., Oehlerking, J., Olderog, E.R., Pang, J., Platzer, A., Segelken, M., Wirtz, B.: Automating verification of cooperation, control, and design in traffic applications. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) Formal Methods and Hybrid Real-Time Systems, *LNCS*, vol. 4700, pp. 115–169. Springer, Berlin (2007)
20. Davoren, J.M.: On hybrid systems and the modal μ-calculus. In: Antsaklis, P.J., Kohn, W., Lemmon, M.D., Nerode, A., Sastry, S. (eds.) Hybrid Systems, *LNCS*, vol. 1567, pp. 38–69. Springer, Berlin (1997). doi:10.1007/3-540-49163-5_3
21. Davoren, J.M., Nerode, A.: Logics for hybrid systems. Proc. IEEE **88**(7), 985–1010 (2000). doi:10.1109/5.871305
22. Dershowitz, N., Manna, Z.: Proving termination with multiset orderings. Commun. ACM **22**(8), 465–476 (1979). doi:10.1145/359138.359142
23. Dowek, G., Hardin, T., Kirchner, C.: Theorem proving modulo. J. Autom. Reason. **31**(1), 33–72 (2003)
24. Emerson, E.A., Clarke, E.M.: Using branching time temporal logic to synthesize synchronization skeletons. Sci. Comput. Program. **2**(3), 241–266 (1982)
25. Emerson, E.A., Halpern, J.Y.: "Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic. J. Assoc. Comput. Mach. **33**(1), 151–178 (1986)
26. Fitting, M.: First-Order Logic and Automated Theorem Proving, 2nd edn. Springer, New York (1996)
27. Fitting, M., Mendelsohn, R.L.: First-Order Modal Logic. Kluwer, Norwell (1999)
28. Fränzle, M.: Analysis of hybrid systems: an ounce of realism can save an infinity of states. In: Flum, J., Rodríguez-Artalejo, M. (eds.) CSL, *LNCS*, vol. 1683, pp. 126–140. Springer, Berlin (1999)
29. Frehse, G.: PHAVer: algorithmic verification of hybrid systems past HyTech. In: Morari, M., Thiele, L. (eds.) HSCC, *LNCS*, vol. 3414, pp. 258–273. Springer, Berlin (2005). doi:10.1007/b106766
30. Giese, M.: Incremental closure of free variable tableaux. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) IJCAR, *LNCS*, vol. 2083, pp. 545–560. Springer, Berlin (2001). doi:10.1007/3-540-45744-5_46
31. Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Mon.hefte Math. Phys. **38**, 173–198 (1931). doi:10.1007/BF01700692
32. Graça, D.S., Campagnolo, M.L., Buescu, J.: Computability with polynomial differential equations. Adv. Appl. Math. **40**, 330–349 (2007)
33. Hähnle, R., Schmitt, P.H.: The liberalized δ-rule in free variable semantic tableaux. J. Autom. Reason. **13**(2), 211–221 (1994). doi:10.1007/BF00881956
34. Harel, D.: First-Order Dynamic Logic. Springer, New York (1979)
35. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT, Cambridge (2000)

36. Henzinger, T.A.: The theory of hybrid automata. In: LICS, pp. 278–292. IEEE Computer Society, Los Alamitos (1996)
37. Henzinger, T.A., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. In: LICS, pp. 394–406. IEEE Computer Society, Los Alamitos (1992)
38. Hutter, D., Langenstein, B., Sengler, C., Siekmann, J.H., Stephan, W., Wolpers, A.: Deduction in the verification support environment (VSE). In: Gaudel, M.C., Woodcock, J. (eds.) FME, *LNCS*, vol. 1051, pp. 268–286. Springer, Berlin (1996)
39. Jifeng, H.: From CSP to hybrid systems. In: Roscoe, A.W. (ed.) A Classical Mind: Essays in Honour of C. A. R. Hoare, pp. 171–189. Prentice Hall, Hertfordshire (1994)
40. Kesten, Y., Manna, Z., Pnueli, A.: Verification of clocked and hybrid systems. Acta Inf. **36**(11), 837–912 (2000). doi:10.1007/s002360050177
41. Lafferriere, G., Pappas, G.J., Yovine, S.: A new class of decidable hybrid systems. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC, *LNCS*, vol. 1569, pp. 137–151. Springer, Berlin (1999)
42. Manna, Z., Sipma, H.: Deductive verification of hybrid systems using STeP. In: Henzinger, T.A., Sastry, S. (eds.) HSCC, *LNCS*, vol. 1386, pp. 305–318. Springer, Berlin (1998). doi:10.1007/3-540-64358-3_47
43. Morayne, M.: On differentiability of Peano type functions. Colloq. Math. **LIII**, 129–132 (1987)
44. Mysore, V., Piazza, C., Mishra, B.: Algorithmic algebraic model checking II: Decidability of semi-algebraic model checking and its applications to systems biology. In: Peled, D., Tsay, Y.K. (eds.) ATVA, *LNCS*, vol. 3707, pp. 217–233. Springer, Berlin (2005)
45. Perko, L.: Differential equations and dynamical systems. Springer, New York (1991)
46. Platzer, A.: Combining deduction and algebraic constraints for hybrid system analysis. In: Beckert, B. (ed.) VERIFY'07 at CADE, Bremen, Germany, *CEUR Workshop Proceedings*, vol. 259, pp. 164–178. CEUR-WS.org (2007)
47. Platzer, A.: Differential dynamic logic for verifying parametric hybrid systems. In: Olivetti, N. (ed.) TABLEAUX, *LNCS*, vol. 4548, pp. 216–232. Springer, Berlin (2007)
48. Platzer, A.: A temporal dynamic logic for verifying hybrid system invariants. In: Artëmov, S.N., Nerode, A. (eds.) LFCS, *LNCS*, vol. 4514, pp. 457–471. Springer, Berlin (2007)
49. Platzer, A., Clarke, E.M.: The image computation problem in hybrid systems model checking. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC, *LNCS*, vol. 4416, pp. 473–486. Springer, Berlin (2007)
50. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57. IEEE, Piscataway (1977)
51. Pratt, V.R.: Semantical considerations on Floyd-Hoare logic. In: FOCS, pp. 109–121. IEEE, Piscataway (1976)
52. Rönkkö, M., Ravn, A.P., Sere, K.: Hybrid action systems. Theor. Comput. Sci. **290**(1), 937–973 (2003)
53. Sibirsky, K.S.: Introduction to Topological Dynamics. Noordhoff, Leyden (1975)
54. Tarski, A.: A Decision Method for Elementary Algebra and Geometry, 2nd edn. University of California Press, Berkeley (1951)
55. Tavernini, L.: Differential automata and their discrete simulators. Nonlinear Anal. **11**(6), 665–683 (1987). doi:10.1016/0362-546X(87)90034-4
56. Tinelli, C.: Cooperation of background reasoners in theory reasoning by residue sharing. J. Autom. Reason. **30**(1), 1–31 (2003)
57. Tiwari, A.: Approximate reachability for linear systems. In: Maler, O., Pnueli, A. (eds.) Hybrid Systems: Computation and Control, 6th International Workshop, HSCC 2003 Prague, Czech Republic, April 3–5, 2003, Proceedings, *LNCS*, vol 2623, pp. 514–525. Springer, Berlin (2003). doi:10.1007/3-540-36580-X_37
58. Walter, W.: Ordinary Differential Equations. Springer, Berlin (1998)
59. Zhou, C., Ravn, A.P., Hansen, M.R.: An extended duration calculus for hybrid real-time systems. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems, LNCS, vol. 736, pp. 36–59. Springer, Berlin (1992)