Towards a Hybrid Dynamic Logic for Hybrid Dynamic Systems[1]

André Platzer[2]

Carnegie Mellon University, Pittsburgh, PA, USA
University of Oldenburg, Department of Computing Science, Germany
platzer@informatik.uni-oldenburg.de

Abstract

We introduce a hybrid variant of a dynamic logic with continuous state transitions along differential equations, and we present a sequent calculus for this extended hybrid dynamic logic. With the addition of satisfaction operators, this hybrid logic provides improved system introspection by referring to properties of states during system evolution. In addition to this, our calculus introduces state-based reasoning as a paradigm for delaying expansion of transitions using nominals as symbolic state labels. With these extensions, our hybrid dynamic logic advances the capabilities for compositional reasoning about (semialgebraic) hybrid dynamic systems. Moreover, the constructive reasoning support for goal-oriented analytic verification of hybrid dynamic systems carries over from the base calculus to our extended calculus.

Keywords: hybrid logic, dynamic logic, sequent calculus, compositional verification, real-time hybrid dynamic systems

1 Introduction

Failures in automotive industry, railway technology and avionics have a disastrous impact. A central feature prevalent in those safety-critical embedded systems is a tight amalgamation of mechanical, electrical and digital technology—leading to an interacting continuous and discrete system dynamics in, what is called, hybrid systems. Albeit, the complexity of the overall system dynamics in large-scale applications still exceeds the capabilities of today's verification tools, which depend on simple dynamics. Hence, a key idea is to decompose reasoning into: (*a*) a closer investigation of the actual complex dynamics of a single system component; and

[1] Despite the obtruding linguistic appeal of using hybrid dynamic logic for verifying hybrid dynamic systems, notice the incisive *gap in terminology*. While dynamic logics are multi-modal logics with modalities for reasoning about structured actions, the theory of dynamic systems provides mathematical concepts for states varying over time (along differential equations). Moreover, with hybrid logics being modal logics with nominals for labelling states, hybrid systems are dynamic systems with interacting discrete and continuous behaviour.

[2] This work was supported by a fellowship of the German Academic Exchange Service (DAAD) 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, see www.avacs.org).

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

(b) an integration of those local correctness results on a simplified abstract level into the overall system verification.

A companion paper [19] introduces a dynamic logic enriched by continuous state transitions along differential equations. This differential dynamic logic, $\mathsf{d}\mathcal{L}$, is suitable for verifying hybrid dynamic systems. In the present paper, we improve its capabilities for compositional reasoning (b) by combining the ideas of hybrid logic with $\mathsf{d}\mathcal{L}$ to form the hybrid dynamic logic $\mathsf{d}\mathcal{L}_h$. We refer the reader to [19] for details on (a) in the presence of continuous dynamics.

We use hybrid logic to achieve two important improvements with $\mathsf{d}\mathcal{L}_h$: state-based reasoning and introspection. Using nominals as abstract state labels, our hybrid calculus is able to leave state succession purely symbolic (*state-based reasoning*) and only expand symbolic state transitions when necessary for progress (expand-on-demand). This ability is increasingly important for computationally expensive state transitions caused by continuous dynamics. The second advantage results from improved system introspection by providing satisfaction operators to refer back to states during the system evolution.

On account of this, a prime contribution of this paper is the hybrid dynamic logic $\mathsf{d}\mathcal{L}_h$. It forms a foundation for compositional verification of hybrid dynamic systems. Another important contribution is a hybrid sequent calculus for the *first-order* dynamic logic $\mathsf{d}\mathcal{L}_h$, which allows the paradigm of state-based reasoning in an expand-on-demand style. A third contribution are satisfaction operators to refer back to states during system execution for improved system introspection.

Hybrid Dynamic Systems

Safety-critical systems in the aforementioned industries typically have a joint discrete and continuous state space. There, system behaviour depends on both, the state in a discrete controller and continuous physical measurands. Hybrid systems represent a mathematical model for those dynamic systems of interacting discrete and continuous behaviour. Such behaviour is governed by a combination of continuous evolution characterised by differential equations and of instantaneous projections for discrete jumps in the state space.

Compositional Verification

The key challenge for compositional verification is to provide both a specification language and a verification calculus that are prepared for integrability concerns of local correctness statements. Components often (re)occur in different contexts of

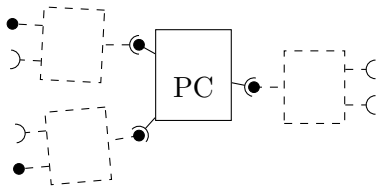

Fig. 1. Context of integrability for compositional verification

a system architecture. For verification, components thus need a systematic decoupling from their environment in order to permit a flexible assembly into their highly

generic context (Fig. 1 visualises a component (PC), in solid lines, with its interface to the environment, plotted with dashed lines). Hence, this flexibility of component integration also has to be respected in correctness statements with a rely-guarantee style of reasoning [10]. The basic requirement for the practical use of compositional verification in this connexion is a free *integrability* (this notion is adopted from general principles of component systems [2]) of correctness statements into the overall verification. This means that local correctness statements have to mimic the generic interaction structure of a component with its environment, in order to coalesce into a global correctness statement at every possible instantiation of the component. In particular, this requires the specification language to adhere to this interaction character of compositional correctness statements and the verification tools to deal faithfully with those aspects of integration.

We argue that the ability of dynamic logic to relate statements about multiple components is beneficial for compositional reasoning. All the more, by hybridising $\mathsf{d}\mathcal{L}$ to form $\mathsf{d}\mathcal{L}_h$, we improve the means for addressing compositionality by state-based reasoning (using nominals) and introspection (using satisfaction operators).

Dynamic Logic

The principle of dynamic logic (DL) is to facilitate the formulation of statements about system behaviour by integrating system descriptions and formulas within a single specification language (see e.g. [13] for a general exposition of DL). By permitting arbitrary system descriptions α as actions of a labelled multi-modal logic, dynamic logic provides formulas of the form $[\alpha]\phi$ and $\langle\alpha\rangle\phi$, where $[\alpha]\phi$ expresses that all (terminating) runs of system α lead to states in which ϕ holds, whereas $\langle\alpha\rangle\phi$ expresses that there is at least one terminating run of α after which ϕ holds. A Hoare-style specification $\{\phi\}\alpha\{\psi\}$, for instance, can be expressed as $\phi \rightarrow [\alpha]\psi$. As in $\mathsf{d}\mathcal{L}$, descriptions of hybrid system actions play the role of α in $\mathsf{d}\mathcal{L}_h$.

Hybrid Logic

Hybrid logic internalises the modal satisfaction relation and state labels [4,6]. It introduces *nominals* as unique labels for states such that a nominal i designates a single state by being true at exactly one state. Further, the *satisfaction operator* $@_i\phi$ allows to refer to whether ϕ is true in the unique state labelled by nominal i.

Structure of this Paper

After introducing syntax and semantics of the hybrid dynamic logic $\mathsf{d}\mathcal{L}_h$ in Section 2, we introduce a sound sequent calculus for $\mathsf{d}\mathcal{L}_h$ in Section 3 and illustrate the advances in compositional reasoning using hybrid techniques. Section 4 contains related work. Conclusions and future work are in Section 5.

2 Syntax and Semantics of $\mathsf{d}\mathcal{L}_h$

2.1 Overview: The Basic Concepts of $\mathsf{d}\mathcal{L}_h$

The hybrid dynamic logic $\mathsf{d}\mathcal{L}_h$ has four basic characteristics to meet the requirements of hybrid dynamic systems.

Discrete jumps. Projections in discrete state space are represented as instantaneous assignments of values to state variables. With this, mode switches like $\mathsf{mode} := 4$ can be expressed with discrete jump system actions, as well as resets $z := 0$ or adjustments $x := x - 2$ of control variables.

Continuous evolution. Continuous variation in system dynamics is represented with differential equations as evolution constraints. For instance, evolution of a system with constant braking can be expressed with a system action for the differential equation $\ddot{z}{=}{-}5$ with second time-derivative $\ddot{z}$ of z.

Regular combinations. Elementary system behaviour of discrete and continuous change can be combined to structured behaviour of hybrid systems using regular combinations. For example, $\mathsf{mode} := 4 \cup \ddot{z}{=}{-}5$ describes a train controller that can either choose to switch its state to an alert mode (4) or initiate braking by the differential equation $\ddot{z} = -5$, by a nondeterministic choice. In conjunction with the other regular combinations, conditionals can express more constrained or even deterministic control choices.

Nominals. The combination $\mathsf{d}\mathcal{L}_h$ of hybrid logic with dynamic logic introduces nominals as unique markers for states during system runs and a satisfaction operator $@_i\phi$ to refer back to the truth-value of ϕ at the unique state labelled by nominal i. For instance, $@_i\mathsf{mode} = 4$ expresses that an alert (4) has been issued during the system evolution, namely at the state belonging to the nominal i.

2.2 Syntax of $\mathsf{d}\mathcal{L}_h$

Terms and Formulas

The formulas of $\mathsf{d}\mathcal{L}_h$ are built over a set V of real-valued variables (typically called x, y, z), a set N of nominals (with typical symbols i, j, k, r, s) and a fixed signature Σ of function and predicate symbols. The presence of sufficiently many variables and nominals is assumed by the calculus. The signature Σ is assumed to contain exclusively the usual function and predicate symbols for real arithmetic, such as $0, 1, +, \cdot, =, \leq, <, \geq, >$.

The set $\mathrm{Trm}(V)$ of *terms* with variables in V is defined as in classical first-order logic. The set $\mathrm{Fml}(V)$ of *formulas* of $\mathsf{d}\mathcal{L}_h$ is defined as common in hybrid logic, yet with modalities of first-order dynamic logic. That is, they are built using the connectives $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$ and the quantifiers $\forall, \exists$ for *variables* (first-order part). In addition, if ϕ is a formula and α a system action, then $[\alpha]\phi$ and $\langle\alpha\rangle\phi$ are formulas as in $\mathsf{d}\mathcal{L}$ (dynamic part). Moreover, if ϕ is a formula and i a nominal, then i and $@_i\phi$ are formulas of $\mathsf{d}\mathcal{L}_h$ (hybrid part). Note that, diametrically to CDL [17], $\mathsf{d}\mathcal{L}_h$ only allows variables but not nominals to be quantified. Formally, the sets $\mathrm{Fml}(V)$ of formulas, and $\mathrm{Act}(V)$ of system actions are simultaneously inductively defined in Definitions 2.1 and 2.2, respectively.

Definition 2.1 [Formulas] The set $\mathrm{Fml}(V)$ of *formulas* is the smallest set with:

- If $i \in N$ is a nominal symbol, then $i \in \mathrm{Fml}(V)$.
- If $p \in \Sigma$ is a predicate symbol and $t_i \in \mathrm{Trm}(\Sigma)$, then $p(t_1, \ldots, t_n) \in \mathrm{Fml}(V)$.
- If $\phi, \psi \in \mathrm{Fml}(V)$, then $\neg\phi, (\phi \wedge \psi), (\phi \vee \psi), (\phi \rightarrow \psi) \in \mathrm{Fml}(V)$.

- If $\phi \in \mathrm{Fml}(V)$ and $x \in V$, then $\forall x\, \phi, \exists x\, \phi \in \mathrm{Fml}(V)$.
- If $\phi \in \mathrm{Fml}(V)$ and $\alpha \in \mathrm{Act}(V)$, then $[\alpha]\phi, \langle\alpha\rangle\phi \in \mathrm{Fml}(V)$.
- If $i \in N$ is a nominal and $\phi \in \mathrm{Fml}(V)$, then $@_i\phi \in \mathrm{Fml}(V)$.

System Actions

As in $\mathsf{d}\mathcal{L}$ [19], elementary discrete jumps and continuous evolutions are combined with regular control structures to form more complex system actions of $\mathsf{d}\mathcal{L}_h$.

Definition 2.2 [System actions] The set $\mathrm{Act}(V)$ of (hybrid dynamic) *system actions* is the smallest set such that:

- If $x \in V$ and $\theta \in \mathrm{Trm}(V)$, then $(x := \theta) \in \mathrm{Act}(V)$.
- If $x \in V$ and $\theta \in \mathrm{Trm}(V)$, then $(\dot{x} = \theta) \in \mathrm{Act}(V)$.
- If $\phi \in \mathrm{Fml}(V)$ is a quantifier-free first-order formula, then $(\phi?) \in \mathrm{Act}(V)$.
- If $\alpha, \gamma \in \mathrm{Act}(V)$ then $(\alpha; \gamma) \in \mathrm{Act}(V)$.
- If $\alpha, \gamma \in \mathrm{Act}(V)$ then $(\alpha \cup \gamma) \in \mathrm{Act}(V)$.
- If $\alpha \in \mathrm{Act}(V)$ then $(\alpha^*) \in \mathrm{Act}(V)$.

The effect of $x := \theta$ is an instantaneous discrete jump in the state space, while that of $\dot{x} = \theta$ is an ongoing continuous evolution that is governed by this differential equation (as in [19], our calculus imposes restrictions on the algebraic nature of θ). Extensions to systems of differential equations or higher-order derivatives are straightforward [19].

The semantics of $\phi?$ is that of a no-op if ϕ is true in the current state, and that of a dead end operator, which blocks all further evolution, otherwise. The non-deterministic choice $\alpha \cup \gamma$, sequential composition $\alpha; \gamma$ and non-deterministic repetition α^* of system actions are as customary. They can be combined with $\phi?$ to form more complicated control structures (see [13]). Independent parallel composition $\cap$ can be defined using nominals (cf. [17]).

Example 2.3 [ETCS case study] As our running example, consider part of the European Train Control System (ETCS) case study [11]. In ETCS, the movement of trains is controlled by decentralised Radio Block Centres (RBC), which grant or deny movement authorities to trains by wireless communication. The actual acceleration and braking behaviour is determined by the train and subject to movement authority limit, weather condition, slope of track etc. (in this paper, we focus on compositional hybrid dynamics and refer to [16] for pure real-time and communication aspects). For simplicity, assume that—depending on those conditions—the train motion control determines a safety envelope e around the train, within which it considers driving safe. When movement authority has been granted up to the track position m and the train is currently located at position z then $\mathsf{d}\mathcal{L}_h$ can analyse, for example, the following safety statement about the acceleration system (accel):

$$\psi \;\rightarrow\; \big[\big((m - z < e?; a := -b) \cup (m - z \geq 2e?; a := 0.1)\big); \ddot{z} = a\big]\; z < m$$

It expresses that—under a sanity condition ψ for the safety envelope—a train that decelerates using service brakes of constant force b if the safety envelope is underrun

$(m - z < e)$, but slowly accelerates if there is sufficient distance $(m - z \geq 2e)$, will always remain within its movement authority m. In reality, the evolution $\ddot{z} = a$ is interrupted for reassessment of driving conditions after a certain processing period, which we do not intend to investigate further here to retain a concise presentation.

2.3 Semantics

Hybrid dynamic systems evolve along a piecewise continuous trajectory in n-dimensional space as time passes. Discontinuities are caused by discrete jumps in the state space while continuous evolution segments are governed by differential equations.

The interpretations of $\mathsf{d}\mathcal{L}_h$ consist of worlds (states) that are assignments of variables with real values and depend on assignments of states to nominals. A potential behaviour of a hybrid system corresponds to a sequence of states that contain the observable values of system variables during its evolution. More precisely, the semantics of a single (compound or atomic) system action is captured by the state transitions that are possible using this action. Since function and predicate symbols are interpreted as common for real arithmetic, we suppress the first-order structures and set of states from the notation.

Definition 2.4 [Interpretation] An *interpretation* is a non-empty set of states with domain $\mathbb{R}$ that is assumed closed under system actions (see Def. 2.7). A state is a map $v{:}V{\to}\mathbb{R}$; the set of all states is denoted by $\mathrm{Sta}(V)$. In addition to this, an *assignment of nominals* is a mapping η from N to the set of states.

The passing of time is implicit and assumed to be defined according to the differential action $\dot{\tau} = 1$ upon which the symbol τ will represent the time passed within the model. Thus, $v(\tau)$ always denotes the absolute point in time to which the system state v belongs. Further, we use $v[x \mapsto d]$ to denote the *semantic modification* of a state v that is identical to v except for the interpretation of the symbol x, which is $d \in \mathbb{R}$. For terms and formulas, the valuation $val_\eta(v, \cdot)$ with respect to an assignment η and state v combines the usual semantics for first-order hybrid logic [5] and first-order modal or dynamic logic [12,13]. The underlying modal transition relation, $\rho_\eta(\alpha)$, is induced by the semantics of hybrid system α (Def. 2.7).

Definition 2.5 [Valuation of terms] The *valuation* $val_\eta(v, \cdot)$ *of terms* with respect to assignment η and state v is defined by:

(i) $val_\eta(v, x) = v(x)$ if x is a variable

(ii) $val_\eta(v, f(t_1, \ldots, t_n)) = f^\ell\big(val_\eta(v, t_1), \ldots, val_\eta(v, t_n)\big)$, where f^ℓ is the operation associated to the function symbol f

Definition 2.6 [Valuation of formulas] The *valuation* $val_\eta(v, \cdot)$ *of formulas* with respect to assignment η and state v is defined by:

(i) $val_\eta(v, p(t_1, \ldots, t_n)) = p^\ell\big(val_\eta(v, t_1), \ldots, val_\eta(v, t_n)\big)$, where p^ℓ is the relation associated to the predicate symbol p

(ii) $val_\eta(v, \phi \wedge \psi)$ is defined as usual, the same holds for $\neg, \vee, \to$

(iii) $val_\eta(v, \forall x\, \phi) = \mathit{true}$ iff $val_\eta(v[x \mapsto d], \phi) = \mathit{true}$ for all $d \in \mathbb{R}$

(iv) $val_\eta(v, \exists x\, \phi) = \mathit{true}$ iff $val_\eta(v[x \mapsto d], \phi) = \mathit{true}$ for some $d \in \mathbb{R}$

(v) $val_\eta(v,[\alpha]\phi) = true$ iff $val_\eta(w,\phi) = true$ for all w such that $(v,w) \in \rho_\eta(\alpha)$

(vi) $val_\eta(v,\langle\alpha\rangle\phi) = true$ iff $val_\eta(w,\phi) = true$ for some w such that $(v,w) \in \rho_\eta(\alpha)$

(vii) $val_\eta(v,i) = true$ iff $\eta(i) = v$

(viii) $val_\eta(v,@_i\phi) = true$ iff $val_\eta(\eta(i),\phi) = true$

Unlike in [5], terms are *non-rigid*, i.e, their value can change from state to state. The semantics, $\rho_\eta(\alpha)$, of system actions is a hybrid variant of that in $\mathsf{d}\mathcal{L}$ [19]:

Definition 2.7 [Semantics of system actions] The *valuation, $\rho_\eta(\alpha)$, of a system action α*, is a *transition relation* on states. It specifies which state w is reachable from a state v by operations of the hybrid system α and is defined as:

(i) $(v,w) \in \rho_\eta(x := \theta)$ iff $w = v[x \mapsto val_\eta(v,\theta)]$

(ii) $(v,w) \in \rho_\eta(\dot{x} = \theta)$ iff there is a function $f : [v(\tau), w(\tau)] \to \mathrm{Sta}(V)$ such that $\gamma_x(\zeta) = val_\eta(f(\zeta), x)$ is continuous on $[v(\tau), w(\tau)]$ and differentiable[3] of value $\gamma_\theta(\zeta)$ at each time $\zeta \in (v(\tau), w(\tau))$, while γ_y is constant for each $y \neq x$ and $f(v(\tau)) = v, f(w(\tau)) = w$

(iii) $\rho_\eta(\phi?) = \{(v,v) \ : \ val_\eta(v,\phi) = true\}$

(iv) $\rho_\eta(\alpha;\gamma) = \rho_\eta(\alpha) \circ \rho_\eta(\gamma) = \{(v,w) \ : \ (v,z) \in \rho_\eta(\alpha), (z,w) \in \rho_\eta(\gamma) \text{ for a } z\}$

(v) $\rho_\eta(\alpha \cup \gamma) = \rho_\eta(\alpha) \cup \rho_\eta(\gamma)$

(vi) $(v,w) \in \rho_\eta(\alpha^*)$ iff there are $n \in \mathbb{N}$ and $v{=}v_0, \ldots, v_n{=}w$ with $(v_i, v_{i+1}) \in \rho_\eta(\alpha)$ for $0 \leq i < n$

Due to the implicit definition of time via the differential equation $\dot{\tau} = 1$, time does not pass during discrete jumps $x := \theta$ but only during continuous evolutions.

2.4 Compositionality by Introspection

Nominals add the capability to express that a state recurs during the run of a system. For instance, the following formula of $\mathsf{d}\mathcal{L}_h$ expresses that the system inevitably ends in a state (marked by i) that has already been visited after the first assignment: $[z := -y; i?; x := x - y; x := z + x]i$. We investigate how this can be exploited for compositional reasoning in the next example.

Example 2.8 [ETCS compositionality by introspection] Using nominals, $\mathsf{d}\mathcal{L}_h$ is able to express advanced system introspection properties in the ETCS scenario of Example 2.3. Such system introspection is useful for isolating reasoning about components of tight interplay. For instance, the following formula expresses that—from the perspective of one component—the only cause of underrunning the safety envelope could be due to a change in track slope (or weather conditions) at state i during the system evolution and due to a failure to react by properly having adjusted the braking distance estimates at state j (the use of nominals for introspection is not limited to sequential composition, though):

$$[\mathsf{poll\text{-}sensor}; i?; \mathsf{accel}; j?; \ddot{z} = a](m - z < e \to @_i(slope \vee weather) \wedge @_j \neg adjust)$$

[3] Solutions are defined accordingly for differential equation systems, which is where the full capabilities of f with domain $\mathrm{Sta}(V)$ rather than $\mathbb{R}$ come into play. Then γ_y remains constant for each variable y with no equation in the system ($\dot{x} = \theta$).

Once this conjecture has been established, a separate analysis could be used to show that a misjudgement leading to this sequence of events (as observed at i followed by j) is prevented by another system component that is responsible for assessing track conditions. Yet, the latter would hold for reasons completely irrespective of the functionality of the acceleration system itself. Similarly, reasoning about the full train control depends on messages received from the RBC during system evolution. Nominals help to isolate reasoning about train control and reasoning about RBCs, even though safety depends on their cooperation.

Here, nominals help to structure verification of components, thereby unleashing a great potential for performance improvements during verification. In general, it is easier to prove several conjectures about smaller subsystems than a single correctness statement about the full system [8]. Using nominals for introspection, this effect can even be exploited to decompose reasoning for components when correctness depends on a very tight interplay.

3 A Sequent Calculus for $\mathsf{d}\mathcal{L}_h$

3.1 Overview

In this section, we introduce a sequent calculus for $\mathsf{d}\mathcal{L}_h$. With the basic idea being to perform a symbolic evaluation, system actions are successively transformed into logical formulas describing their effects. Yet, rule applications for first-order reasoning and system reasoning are not separated but intertwined.

For propositional logic, standard rules R1–R9 are listed in Figure 2. For dealing with nominals, the satisfaction operator and symbolic state transitions with nominals, $\mathsf{d}\mathcal{L}_h$ uses standard rules R14–R21 of hybrid logic [4,5]. Most of the other rules for dynamic modalities are hybrid variants of $\mathsf{d}\mathcal{L}$ rules [19]. They transform system actions into "simpler" logical formulas, thereby relating the meaning of actions and formulas. The idea of using hybrid modality rules R14–R17, however, is to facilitate state-based reasoning by introducing nominals as symbolic state labels. Expanding those purely symbolic transitions with the other dynamic rules can be delayed by this technique until inevitable for proof progress, which is particularly useful for computationally expensive cases like R22. Unlike first-order logic, quantifiers are dealt with using quantifier elimination [21] over the reals for continuous evolutions, which is presented in a companion paper [19].

3.2 Rules of the Calculus

A *sequent* is of the form $\Gamma \vdash \Delta$, where Γ and Δ are finite (multi)sets of formulas. Its informal semantics is the same as that of the formula $\bigwedge_{\phi\in\Gamma}\phi \rightarrow \bigvee_{\psi\in\Delta}\psi$.

Definition 3.1 [Provability, Derivability] A formula ψ is *provable* from a set Φ of formulas, denoted by $\Phi \vdash_{\mathsf{d}\mathcal{L}_h} \psi$ iff there is a finite set $\Phi_0 \subseteq \Phi$ for which the sequent $\Phi_0 \vdash \psi$ is derivable. In turn, a sequent of the form $\Gamma, \Phi \vdash \Psi, \Delta$ (for some finite multisets Γ, Δ of context formulas) is *derivable* iff there is an instance

$$\frac{\Phi_1 \vdash \Psi_1 \qquad \ldots \qquad \Phi_n \vdash \Psi_n}{\Phi \vdash \Psi}$$

$$\text{(R1)}\ \frac{\vdash @_s A}{@_s \neg A \vdash} \qquad \text{(R4)}\ \frac{@_s A, @_s B \vdash}{@_s (A \wedge B) \vdash} \qquad \text{(R7)}\ \frac{@_s A \vdash \quad @_s B \vdash}{@_s (A \vee B) \vdash}$$

$$\text{(R2)}\ \frac{@_s A \vdash}{\vdash @_s \neg A} \qquad \text{(R5)}\ \frac{\vdash @_s A \quad \vdash @_s B}{\vdash @_s (A \wedge B)} \qquad \text{(R8)}\ \frac{\vdash @_s A, @_s B}{\vdash @_s (A \vee B)}$$

$$\text{(R3)}\ \frac{@_s A \vdash @_s B}{\vdash @_s (A \rightarrow B)} \qquad \text{(R6)}\ \frac{\vdash @_s A \quad @_s B \vdash}{@_s (A \rightarrow B) \vdash} \qquad \text{(R9)}\ \frac{}{@_s A \vdash @_s A}$$

$$\text{(R10)}\ \frac{\phi, \phi \vdash}{\phi \vdash} \qquad \text{(R21)}\ \frac{@_r \phi}{@_s @_r \phi}$$

$$\text{(R11)}\ \frac{\vdash \phi, \phi}{\vdash \phi} \qquad \text{(R22)}\ \frac{@_s \exists c\, (\hat{x}_c(0) = x \wedge \exists t{\geq}0\, \langle x := \hat{x}_c(t) \rangle \phi)}{@_s \langle \dot{x} = \theta \rangle \phi}$$

$$\text{(R12)}\ \frac{@_s \langle x := \theta \rangle r, @_s F_x^\theta \vdash}{@_s \langle x := \theta \rangle r, @_r F \vdash} \qquad \text{(R23)}\ \frac{@_s \forall c\, (\hat{x}_c(0) = x \rightarrow \forall t{\geq}0\, [x := \hat{x}_c(t)] \phi)}{@_s [\dot{x} = \theta] \phi}$$

$$\text{(R13)}\ \frac{@_s \langle x := \theta \rangle r \vdash @_s F_x^\theta}{@_s \langle x := \theta \rangle r \vdash @_r F} \qquad \text{(R24)}\ \frac{@_s \langle \alpha \rangle \langle \gamma \rangle \phi}{@_s \langle \alpha ; \gamma \rangle \phi}$$

$$\text{(R14)}\ \frac{@_s \langle \alpha \rangle a, @_a \phi \vdash}{@_s \langle \alpha \rangle \phi \vdash} \qquad \text{(R25)}\ \frac{@_s [\alpha][\gamma] \phi}{@_s [\alpha ; \gamma] \phi}$$

$$\text{(R15)}\ \frac{@_r \phi \vdash}{@_s [\alpha] \phi, @_s \langle \alpha \rangle r \vdash} \qquad \text{(R26)}\ \frac{@_s (\phi \wedge \psi)}{@_s \langle \phi ? \rangle \psi}$$

$$\text{(R16)}\ \frac{\vdash @_r \phi}{@_s \langle \alpha \rangle r \vdash @_s \langle \alpha \rangle \phi} \qquad \text{(R27)}\ \frac{@_s (\phi \rightarrow \psi)}{@_s [\phi ?] \psi}$$

$$\text{(R17)}\ \frac{@_s \langle \alpha \rangle a \vdash @_a \phi}{\vdash @_s [\alpha] \phi} \qquad \text{(R28)}\ \frac{@_s (\langle \alpha \rangle \phi \vee \langle \gamma \rangle \phi)}{@_s (\langle \alpha \cup \gamma \rangle \phi)}$$

$$\text{(R18)}\ \frac{@_s s \vdash}{\vdash} \qquad \text{(R29)}\ \frac{@_s ([\alpha] \phi \wedge [\gamma] \phi)}{@_s [\alpha \cup \gamma] \phi}$$

$$\text{(R19)}\ \frac{@_r \phi \vdash}{@_s r, @_s \phi \vdash} \qquad \text{(R30)}\ \frac{@_s (\phi \vee \langle \alpha ; \alpha^* \rangle \phi)}{@_s \langle \alpha^* \rangle \phi}$$

$$\text{(R20)}\ \frac{@_s \langle \alpha \rangle u \vdash}{@_s \langle \alpha \rangle r, @_r u \vdash} \qquad \text{(R31)}\ \frac{@_s (\phi \wedge [\alpha ; \alpha^*] \phi)}{@_s [\alpha^*] \phi}$$

In the above rule schemata, r, s, u are nominals. In R12–R13, F is a first-order formula and the substitution does not introduce new bindings. In R22–R23 the differential equation has expressible solution $\hat{x}_c$, where c, t are fresh variables. In R14 and R17, a is a new nominal; further, $\phi \notin N$ in R14.

Fig. 2. Rule schemata of the hybrid $\mathsf{d}\mathcal{L}_h$ calculus

of a rule schema of the $\mathsf{d}\mathcal{L}_h$ calculus in Figure 2 such that $\Gamma, \Phi_i \vdash \Psi_i, \Delta$ is derivable for each $1 \leq i \leq n$. Schemata R21–R31 can be applied on either side of the sequent.

As usual in sequent calculus, note that—although the direction of entailment is from premisses (above rule bar) to conclusion (below)—the order of reasoning is *converse* in practice. Rules are applied analytically, starting with the proof obligation at the bottom. As usual in hybrid logic [4], the calculus is further built for satisfaction formulas $@_s\phi$ and retains this structure during the proof. The choice of multisets and the contraction rules R10–R11 is only necessary for a concise formulation of R15,R16,R18–R20 (see [4] for details).

For handling discrete change, the $\mathsf{d}\mathcal{L}_h$ rules R12–R13 use substitutions once the remaining formulas are first-order (improvements for resolving assignments earlier are presented in [3]; observe how hybrid logic gives an elegant alternative to update prefixes). The same rules are assumed for $[x := \theta]$ rather than $\langle x := \theta\rangle$, here, as discrete change is deterministic. The result of applying to ϕ the substitution that replaces x by θ is defined as usual; it is denoted by ϕ^θ_x.

In R22–R23, solving differential equations amounts to a symbolic initial value problem. Yet, differential equations are restricted to having flows as solutions that are expressible in $\mathsf{d}\mathcal{L}_h$ (i.e., semialgebraic):

Definition 3.2 [Expressible solutions] The differential equation $\dot{x} = \theta$ has *expressible solutions* in $\mathsf{d}\mathcal{L}_h$, if its general solution $\hat{x}_c$ is parametric in $c = (c_1, \ldots, c_n) \in \mathbb{R}^n$ and $\hat{x}_c \in \mathrm{Trm}(V)$.

Quantifiers arising from R22–R23 are handled using the technique in [19].

3.3 Compositional Verification with State-based Reasoning

To illustrate state-based reasoning and focus on the aspects involving hybrid logic, we consider a conjecture about an acceleration system accel (simplified in comparison to Example 2.3). It is embedded in another component tctl, assumed to assess weather conditions and communication with the RBC:

$$@_s\big([\mathsf{tctl}]\neg\langle \ddot{z} = \text{-}b\rangle z \geq m \;\rightarrow\; \neg\langle \mathsf{tctl}; \mathsf{accel}\rangle \mathit{crash}\big)$$

Fig. 3 shows a proof using state-based reasoning (subsequent weakening is indicated by gray type; the application of R12 is slightly generalised to replace a by $-b$ in differential equations, see [19]). Observe how expanding the modality of $@_s[\mathsf{tctl}]$ is avoided by R14 and replaced by an abstract symbolic successor t. Using the state-based reasoning capabilities of $\mathsf{d}\mathcal{L}_h$, this abstract transition to t (thus the whole system action tctl) never needs to be expanded during the proof. The same state-based reasoning technique can be used to skip computation of the continuous evolution in $@_t\langle \ddot{z} = \text{-}b\rangle$ and close (indicated by $*$) the left branch by simple reasoning that crash states (cr) imply $z \geq m$. Closing the right branch is omitted (indicated by ellipsis). It would require reasoning about the actual sanity condition of the safety envelope and is beyond the scope of this paper. Techniques for proving the antecedent, which actually involve investigating the continuous dynamics, will be presented in [19].

$$
\dfrac{\dfrac{}{R9\ \ @_t\langle a := \text{-}b\rangle r, @_t\langle \ddot{z} = \text{-}b\rangle cr \vdash @_t\langle \ddot{z} = \text{-}b\rangle z \geq m}{R12\ \ @_t\langle a := \text{-}b\rangle r, @_r\langle \ddot{z} = a\rangle cr \vdash @_t\langle \ddot{z} = \text{-}b\rangle z \geq m}\ * \qquad \dfrac{}{@_t\langle c_2?; a := 0.1\rangle r, \ldots \vdash \ldots}\ \ldots}{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{R2\ \ @_s[\mathsf{tctl}]\neg\langle \ddot{z} = \text{-}b\rangle z \geq m \vdash @_s\neg\langle \mathsf{tctl}; \mathsf{accel}\rangle cr}}{R24\ \ @_s[\mathsf{tctl}]\neg\langle \ddot{z} = \text{-}b\rangle z \geq m, @_s\langle \mathsf{tctl}; \mathsf{accel}\rangle cr \vdash}}{R14\ \ @_s[\mathsf{tctl}]\neg\langle \ddot{z} = \text{-}b\rangle z \geq m, @_s\langle \mathsf{tctl}\rangle\langle \mathsf{accel}\rangle cr \vdash}}{R15\ \ @_s[\mathsf{tctl}]\neg\langle \ddot{z} = \text{-}b\rangle z \geq m, @_s\langle \mathsf{tctl}\rangle t, @_t\langle \mathsf{accel}\rangle cr \vdash}}{R1\ \ @_t\neg\langle \ddot{z} = \text{-}b\rangle z \geq m, @_s\langle \mathsf{tctl}\rangle t, @_t\langle \mathsf{accel}\rangle cr \vdash}}{R24\ \ @_t\langle \mathsf{accel}\rangle cr \vdash @_t\langle \ddot{z} = \text{-}b\rangle z \geq m}}{R14\ \ @_t\langle a := \text{-}b \cup (c_2?; a := 0.1)\rangle\langle \ddot{z} = a\rangle cr \vdash @_t\langle \ddot{z} = \text{-}b\rangle z \geq m}}{R28\ \ @_t\langle a := \text{-}b \cup (c_2?; a := 0.1)\rangle r, @_r\langle \ddot{z} = a\rangle cr \vdash @_t\langle \ddot{z} = \text{-}b\rangle z \geq m}}{R7\ \ @_t(\langle a := \text{-}b\rangle r \vee \langle c_2?; a := 0.1\rangle r), @_r\langle \ddot{z} = a\rangle cr \vdash @_t\langle \ddot{z} = \text{-}b\rangle z \geq m}
$$

Abbreviations: $c_2 \equiv (m{-}z{\geq}2e)$ and $\mathsf{accel} \equiv \big(a := \text{-}b \cup (c_2?; a := 0.1)\big); \ddot{z} = a$

Fig. 3. Proof with state-based reasoning for compositional verification.

3.4 Soundness and Incompleteness

Soundness is proven like (local) soundness of the base logic $\mathsf{d}\mathcal{L}$ in [19].

Theorem 3.3 (Soundness) *The $\mathsf{d}\mathcal{L}_h$ calculus (Def. 3.1) is* sound*, which means that derivable formulas are* valid *(true in all states of all interpretations with all assignments of nominals).*

The notion of repetition in $\mathsf{d}\mathcal{L}_h$ permits to define natural numbers in real arithmetic, yielding the following result [19].

Proposition 3.4 (Incompleteness) *The logic $\mathsf{d}\mathcal{L}_h$ is inherently* incomplete*, i.e., no sound calculus for $\mathsf{d}\mathcal{L}_h$ can ever be complete.*

Proposition 3.5 (Reducibility) *The hybrid logic $\mathsf{d}\mathcal{L}_h$ is reducible to $\mathsf{d}\mathcal{L}$.*

Proof. (Sketch) The proof relies on the key observation that states of $\mathsf{d}\mathcal{L}_h$ can be characterised completely by (quantified) variable assignments. Assume x is the vector of all variables occurring in a formula and $\boldsymbol{i}$ is a vector with fresh variables of the same dimension for each nominal i. With this, a reduction can be achieved using the following replacements for satisfaction operators or propositional occurrences of nominals, respectively:

$$
\begin{aligned}
i &\rightsquigarrow \boldsymbol{i} = x \\
@_i\psi &\rightsquigarrow \langle x := \boldsymbol{i}\rangle\psi
\end{aligned}
$$

The correspondence now results from assigning to the vector of variables $\boldsymbol{i}$ the values of the vector of variables x at the state labelled by the nominal i. □

While hybrid reasoning can be emulated in the base logic $\mathsf{d}\mathcal{L}$ by this result, we argue that a built-in treatment of nominals and satisfaction operators in the $\mathsf{d}\mathcal{L}_h$ calculus is important for practical prover performance. In particular, the simplicity of the reduction in the above proof is lost when the vector notation is expanded.

4 Related Work.

Chaochen *et al.* [7] presented a proof system for the duration calculus (DC) extended by durational assertions about mathematical expressions in real-valued time-dependent states and their derivatives. It has no constructive treatment of arithmetic, but only incorporates an oracle, which needs to guess the mathematical conjectures that are needed for proof progress.

Rönkkö *et al.* [20] presented action systems (inspired by the guarded command language) with differential relations over variables and time-derivatives. To each action they associate a weakest precondition in higher-order logic with built-in derivatives of quantified function symbols. Without providing a means for verification of this higher-order logic, the potential for practical applications is more or less limited to providing a notational variant of classical mathematics. They primarily focus on refinement and parallel compositions.

Davoren and Nerode [9] presented a Hilbert-style proof system for a modal μ-calculus $\mathbf{L}\mu$. Their actual calculus only has an "opaque" treatment of differential equations, importing them as anonymous atomic actions irrespective of the corresponding differential equation or flow. They give a model checking procedure for transition systems of hybrid automata that can be expressed in the quantifier-free theory of real-closed fields. Davoren and Nerode argue for a broader scope of applicability of deductive methods, though. They further elaborate on topological issues of hybrid systems expressed in $\mathbf{L}\mu$.

All those approaches have in common that they rely on a semantical treatment of the differential equations determining hybrid evolution instead of incorporating a constructive goal-oriented interface to mathematical problem solving as required for practical verification of hybrid dynamic systems. Further, they do not take into account aspects of compositional reasoning.

Passy and Tinchev [17] presented and analysed a propositional dynamic logic, CPDL, that is extended with nominals and a universal transition relation (this relation is definable in $\mathsf{d}\mathcal{L}_h$ as $\forall y \langle x := y\rangle \phi$, using vector notation x and y as in Proposition 3.5). They also add an intersection operator $\cap$ on programs. Their extension, CDL, of propositional dynamic logic retains opaque atomic programs and only permits quantification over nominals. Thus CDL is not a first-order dynamic logic and not suitable for verification.

In a series of three papers, Piazza *et al.* [18] proposed model checking for semialgebraic hybrid automata in systems biology using quantifier elimination. This work suggests that quantifier elimination is a reasonable choice for handling semialgebraic continuous dynamics.

In [15], Lafferriere *et al.* presented a subclass of linear hybrid systems where reachability is decidable. Decidability is obtained by combining a bisimulation refinement algorithm for backward reachability based on quantifier elimination in real arithmetic over $\{+,-,\cdot,<,0,1\}$ with o-minimality results. Yet, like the HyTech approach [14] to verification of hybrid automata, the philosophy is different in that we focus on investigating a single integrated logic by giving a proof system rather than model checking algorithms for reachability in a separate machine model. Also, tools like HyTech are much more successful in falsification rather than verification [1], which we intend to improve using a theorem prover approach.

5 Conclusions and Future Work

We have introduced a hybrid dynamic logic, $\mathsf{d}\mathcal{L}_h$, as a hybrid variant of the differential dynamic logic $\mathsf{d}\mathcal{L}$ [19], and we presented a sound sequent calculus for $\mathsf{d}\mathcal{L}_h$. The design of the logic $\mathsf{d}\mathcal{L}_h$ is guided by the ambition to improve the capabilities of compositional reasoning about (semialgebraic) hybrid dynamic systems.

The hybrid logic $\mathsf{d}\mathcal{L}_h$ extends $\mathsf{d}\mathcal{L}$, a dynamic logic with continuous state transitions along differential equations [19]. In comparison to $\mathsf{d}\mathcal{L}$, the hybrid variant $\mathsf{d}\mathcal{L}_h$ provides built-in support for system introspection using hybrid satisfaction operators and state-based reasoning with nominals. Both techniques improve its potential for compositional verification.

The $\mathsf{d}\mathcal{L}_h$ calculus combines the ideas underlying the sequent calculus for $\mathsf{d}\mathcal{L}$ [19] and calculi for hybrid logic [4]. Our calculus provides state-based reasoning, i.e, it exploits nominals to reason about purely abstract symbolic transitions and only expands transitions on demand when necessary for progress. Such a delay is crucial to avoid computationally expensive continuous evolutions that turn out to be of no relevance for the particular verification subgoal during the proof.

In addition to the future work for the base logic $\mathsf{d}\mathcal{L}$ like operator completion with parallelism, aspects concerning the particular hybrid nature of $\mathsf{d}\mathcal{L}_h$ include investigating the correspondence between deterministic evolutions or assignments (cf. update prefix in [3]) with nominals in the calculus, yielding a theory of structured nominals. Further, unique initial-value problems for continuous evolutions give rise to deterministic state transitions. In $\mathsf{d}\mathcal{L}_h$, this would result in first-order nominals $s(t)$ that export the time t of evolution as a parameter. Another line of research is on guidance of state-based reasoning rules in automatic theorem provers.

In short, we have demonstrated that combining $\mathsf{d}\mathcal{L}$ with hybrid logic yields a hybrid dynamic logic $\mathsf{d}\mathcal{L}_h$ that is capable of system introspection, and whose calculus is suitable for state-based reasoning about hybrid dynamic systems.

Acknowledgement

Earlier versions of this work have benefited from discussions with Ernst-Rüdiger Olderog, Martin Fränzle, Edmund M. Clarke and Bruce H. Krogh. The author also thanks Peter H. Schmitt, Bernhard Beckert and Reiner Hähnle for feedback on a precursory logic, hylo, presented at the KeY symposium 2005.

References

[1] Rajeev Alur. Private communication, Lipari Summer School on Formal Methods, 2005.

[2] Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.

[3] Bernhard Beckert and André Platzer. Dynamic logic with non-rigid functions: A basis for object-oriented program verification. In U. Furbach and N. Shankar, editors, *Proc., International Joint Conference on Automated Reasoning (IJCAR)*, volume 4130 of *LNCS*, pages 266–280. Springer, 2006.

[4] Patrick Blackburn. Internalizing labelled deduction. *Journal of Logic and Computation*, 10(1):137–168, 2000.

[5] Patrick Blackburn and Maarten Marx. Tableaux for quantified hybrid logic. In Uwe Egly and Christian G. Fermüller, editors, *TABLEAUX*, volume 2381 of *LNCS*, pages 38–52. Springer, 2002.

[6] Patrick Blackburn and Jerry Seligman. What are hybrid languages? In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 1, pages 41–62. CSLI Publications, Stanford, 1998.

[7] Zhou Chaochen, Anders P. Ravn, and Michael R. Hansen. An extended duration calculus for hybrid real-time systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, volume 736 of *LNCS*, pages 36–59. Springer, 1992.

[8] Werner Damm, Hardi Hungar, and Ernst-Rüdiger Olderog. Verification of cooperating travel agents. *Int. J. Control*, 79(5):395–421, May 2006.

[9] Jennifer M. Davoren and Anil Nerode. Logics for hybrid systems. *Proceedings of the IEEE*, 88(7):985–1010, July 2000.

[10] Willem Paul de Roever, Frank S. de Boer, Ulrich Hannemann, Jozef Hooman, Yassine Lakhnech, Mannes Poel, and Job Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*. Cambridge University Press, 2001.

[11] Johannes Faber and Roland Meyer. Model checking data-dependent real-time properties of the European train control system. In *Proc., Formal Methods in Computer Aided Design (FMCAD)*. IEEE Computer Society Press, 2006.

[12] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[13] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, 2000.

[14] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: A model checker for hybrid systems. In Orna Grumberg, editor, *CAV*, volume 1254 of *LNCS*, pages 460–463. Springer, 1997.

[15] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. A new class of decidable hybrid systems. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *HSCC*, volume 1569 of *LNCS*, pages 137–151. Springer, 1999.

[16] Roland Meyer, Johannes Faber, and Andrey Rybalchenko. Model checking duration calculus: A practical approach. In Kamel Barkaoui, Ana Cavalcanti, and Antonio Cerone, editors, *Proc., International Colloquium on Theoretical Aspects of Computing, (ICTAC)*, volume 4281 of *LNCS*, pages 332–346. Springer, 2006.

[17] Solomon Passy and Tinko Tinchev. An essay in combinatory dynamic logic. *Inf. Comput.*, 93(2):263–332, 1991.

[18] Carla Piazza, Marco Antoniotti, Venkatesh Mysore, Alberto Policriti, Franz Winkler, and Bud Mishra. Algorithmic algebraic model checking I: Challenges from systems biology. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *LNCS*, pages 5–19. Springer, 2005.

[19] André Platzer. Differential logic for reasoning about hybrid systems. Submitted, 2006.

[20] Mauno Rönkkö, Anders P. Ravn, and Kaisa Sere. Hybrid action systems. *Theor. Comput. Sci.*, 290(1):937–973, 2003.

[21] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 2nd edition, 1951.