Differential Logic for Reasoning about Hybrid Systems*

André Platzer

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

Abstract. We propose a first-order dynamic logic for reasoning about hybrid systems. As a uniform model for discrete and continuous evolutions in hybrid systems, we introduce hybrid programs with differential actions. Our logic can be used to specify and verify correctness statements about hybrid programs, which are suitable for symbolic processing by calculus rules. Using first-order variables, our logic supports systems with symbolic parameters. With dynamic modalities, it is prepared to handle multiple system components.

Keywords: dynamic logic, hybrid systems, parametric verification

1 Introduction

A key idea for scalable verification of hybrid systems [1] is to decompose [2] reasoning into: (a) a closer investigation of the actual complex dynamics of a single system component; and (b) an integration of local correctness results into global system verification. Furthermore, both (a) and (b) need to handle parameters, which naturally arise from the degrees of freedom of how a single component can be instantiated in a system environment.

As first-order logic has widely proven its flexible power in handling symbolic parameters with logical variables, we extend it for reasoning about hybrid systems. Moreover, in order to be able to relate statements about a component and statements about its environment for compositional reasoning (b), we propose a dynamic logic in which such relations are naturally expressible [3]. Since hybrid systems are subject to continuous evolution along differential equations and discrete state change, we propose a first-order dynamic logic, $\mathsf{d}\mathcal{L}$, that provides both as fundamental system behaviour. Further, $\mathsf{d}\mathcal{L}$ can even be used for *parameter extraction*, i.e., automatic derivation of constraints for safety parameters.

Related work primarily uses propositional modal logic [4]. Unlike our first-order dynamic logic, propositional modal logic is restricted to handling abstract actions and does not support reasoning about concrete behaviour of hybrid systems like, for instance, continuous evolution along a differential equation $\ddot{z} = a$.

* This research 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).

A. Bemporad, A. Bichi, and G. Buttazzo (Eds.): HSCC 2007, LNCS 4416, pp. 746–749, 2007.

2 Differential Logic of Hybrid Programs

Dynamic Logic with Hybrid Programs. Dynamic logics (DL) [5] combine descriptions of system behaviour and correctness statements about the system state within a single specification language. By permitting arbitrary system operations α as actions of a labelled multi-modal logic, DL provides formulas of the form $[\alpha]\phi$ and $\langle\alpha\rangle\phi$. The formula $[\alpha]\phi$ expresses that all (terminating) runs of system α lead to states in which condition ϕ holds, whereas $\langle\alpha\rangle\phi$ expresses that there is at least one (terminating) run of α after which ϕ holds.

In this paper, we propose to extend DL to use hybrid systems for α. In particular, we propose a logic $\mathsf{d}\mathcal{L}$ that extends discrete DL [5] by *differential actions* such that α can display continuous evolution. Due to the symbolic nature of logic, it is beneficial to use simple system actions of an isolated effect in α. As a model for hybrid systems, we introduce hybrid programs, which are much more amenable to step-wise symbolic processing by calculus rules than graph structures of automata. Since hybrid automata [1] can be embedded, there is no loss of expressivity. Our differential logic $\mathsf{d}\mathcal{L}$ is a first-order dynamic logic with three basic characteristics to meet the requirements of hybrid systems:

Discrete jumps. Projections in state space are represented as instantaneous *assignments* of values to state variables. With this, mode switches like $\mathsf{mode} := 4$ or $\mathsf{signal} := 1$ can be expressed with discrete jumps, as well as resets $z := 0$ or discrete adjustments of control variables like $z := z - 2$.

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

Regular combinations. Discrete and continuous evolutions can be combined to form *hybrid programs* using regular expression operators $(\cup, ^*, ;)$ as structured behaviour of hybrid systems. For example, $\mathsf{mode} := 4 \cup \ddot{z} = -5$ describes a train controller that can choose by a nondeterministic choice $(\cup)$ to either switch its state to an alert mode (4) or initiate braking along the differential equation $\ddot{z} = -5$. In conjunction with other regular combinations, control constraints can be expressed using conditions like $z \geq 9?$ as guards for the system state.

Transition Semantics. There is a variety of slightly different semantics of hybrid system models. Since the interplay of discrete change with continuous evolution raises peculiar subtleties, we carefully motivate the advantages of our choice of semantics for $\mathsf{d}\mathcal{L}$ and hybrid programs. Consider the possible hybrid evolution with one system variable x over time t in Fig. 1. The semantics has to restrict the behaviour of the hybrid system during the continuous evolution phase, e.g., on the interval $[1, 2]$ to respect the differential equation $\dot{x} = f(x)$. Yet, the discrete jump at time 2 will necessarily lead to a discontinuity in the overall system trajectory.

A global system trajectory function g (where $g(t)$ records the value of x at time t) can only assume a *single* value at time 2, say the value $g(2) = 0.6$. Hence, the continuous evolution—as visible in g—will only be continuous on the open interval $(1, 2)$. Still, the evolution along $\dot{x} = f(x)$ has to be constrained at time 2 to possess a *left*-continuous continuation towards a projected value of 1, although this value will never be assumed by g. This complicates the well-posed definition of semantics on the basis of an overall system trajectory. Note that leaving out this condition of left-continuity would lead to a total transition relation with *all* states being reachable, which, of course, would not reflect the proper system behaviour.

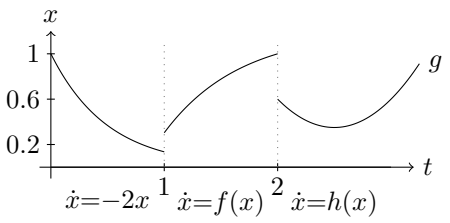

Fig. 1. Discontinuous hybrid trajectory.

In contrast to this, the $\mathsf{d}\mathcal{L}$ semantics inflates points in time with instantaneous discrete progression by associating an *individual* trajectory for each continuous evolution or instant jump phase, e.g. the phases $[0, 1], \{1\}, [1, 2], \{2\}, [2, 4]$. Hence, the trajectories remain continuous within each differential evolution phase and discontinuities are isolated purely in discrete jump transitions. Thereby, the $\mathsf{d}\mathcal{L}$ semantics directly traces the succession of values assumed during the hybrid evolution, even if they belong to states which occur without model time passing in between. In addition to the fact that those so-called *super-dense* time effects naturally occur at mode switches between differential evolutions, they are necessary for joint mode switches of several system variables at once, like in $x := 3; y := 5$. We argue that the resulting $\mathsf{d}\mathcal{L}$ semantics is much simpler to define than for approaches with a global overall system trajectory as, for example, in [2].

3 Parametric Verification of Train Control Systems

Symbolic parameters occurring in system dynamics raise a couple of challenges. Firstly, even very simple parametric flows and guards are *non-linear*: With parameter p, the flow constraint $2x + py \leq 5$ is an algebraic inequality but not linear. Thus, our logic needs to handle dynamics in full real arithmetic. Secondly, parameters often arise from system decomposition, e.g. in [2]. For this, safety statements about a parametric component typically have to take its interaction with the environment into account. In particular, local correctness statements need to reflect this interaction to obtain global correctness for every possible instantiation. Thus, the verification logic needs to support this interactive character with rely-guarantee reasoning; see, e.g. [2].

We argue that logic is the right level for handling the symbolic nature of parameters. All the more, the ability of dynamic logic to relate statements about multiple components is extremely valuable for compositional reasoning [3].

In the European Train Control System (ETCS) [6], the movement of trains is controlled by decentralised Radio Block Centres (RBC), which grant or deny

movement authorities (MA) to trains by wireless communication. In case of an emergency, trains always have to stop within the MA issued by the RBC. In ETCS, the actual acceleration and braking behaviour is determined by the train and subject to MA limits, weather conditions, slope of track etc. For simplicity, assume that—depending on those conditions—the train motion control determines a safety envelope s around the train, within which it considers driving safe. When an MA has been granted up to the track position m and the train is currently located at position z then $\mathsf{d}\mathcal{L}$ can analyse, for example, the following safety statement about the (simplified) acceleration system:

$$\psi \;\rightarrow\; [\big((m-z<s?;a:=-b)\cup(m-z\geq 2s?;a:=0.1)\big);\ddot{z}=a]\; z<m \;. \quad (1)$$

It expresses that, under a condition ψ about parameters, trains always remain within their MA m. Further, it specifies that the train decelerates using engine brakes of force b if the safety envelope is underrun ($m-z<s$). It slowly accelerates if there is sufficient distance ($m-z\geq 2s$). To give a more concise program, we have omitted the case where the train keeps its current speed if there is no need to brake nor sufficient distance (i.e., $s\leq m-z<2s$). The resulting transition structure for the hybrid program in (1) is depicted in Fig. 2. Formula (1) can be analysed successfully by our calculus for verifying $\mathsf{d}\mathcal{L}$ formulas, which is similar to the one in [3]. With such an analysis, parameter constraints on the free variables of (1) can be discovered.

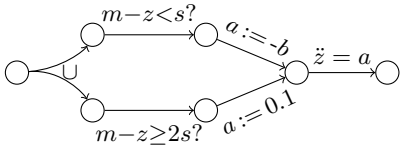

Fig. 2. Acceleration transitions.

The behaviour of the program in (1) can be analysed in $\mathsf{d}\mathcal{L}$, which is a first-order dynamic logic. In contrast, the hybrid program in (1) would collapse to a mere abstract shape $((\alpha_1;\alpha_2)\cup(\alpha_3;\alpha_4));\alpha_5$ in propositional modal logics [4]. There, the truth of (1), which depends on the actual effects of the α_i, cannot be analysed, since the state changes induced by the abstract actions α_i are unknown in propositional programs. For this reason, $\mathsf{d}\mathcal{L}$ is devised as a first-order logic.

References

1. Henzinger, T.A.: The theory of hybrid automata. In: LICS. (1996) 278–292
2. Damm, W., Hungar, H., Olderog, E.R.: Verification of cooperating travel agents. International Journal of Control **79**(5) (2006) 395–421
3. Platzer, A.: Towards a hybrid dynamic logic for hybrid dynamic systems. In Blackburn, P., Bolander, T., Braüner, T., de Paiva, V., Villadsen, J., eds.: Proc., LICS International Workshop on Hybrid Logic, Seattle, USA. ENTCS (2006)
4. Davoren, J.M., Nerode, A.: Logics for hybrid systems. Proceedings of the IEEE **88**(7) (2000) 985–1010
5. Harel, D., Kozen, D., Tiuryn, J.: Dynamic logic. MIT Press (2000)
6. Faber, J., Meyer, R.: Model checking data-dependent real-time properties of the European train control system. In: FMCAD, IEEE Computer Society Press (2006)