Form Methods Syst Des (2008) 32: 57–83
DOI 10.1007/s10703-007-0044-3

Abstractions for hybrid systems

Ashish Tiwari

Published online: 5 December 2007

Abstract We present a procedure for constructing sound finite-state discrete abstractions of hybrid systems. This procedure uses ideas from predicate abstraction to abstract the discrete dynamics and qualitative reasoning to abstract the continuous dynamics of the hybrid system. It relies on the ability to decide satisfiability of quantifier-free formulas in some theory rich enough to encode the hybrid system. We characterize the sets of predicates that can be used to create high quality abstractions and we present new approaches to discover such useful sets of predicates. Under certain assumptions, the abstraction procedure can be applied compositionally to abstract a hybrid system described as a composition of two hybrid automata. We show that the constructed abstractions are always sound, but are relatively complete only under certain assumptions.

Keywords Hybrid systems · Predicate abstraction · Qualitative simulation

1 Introduction

Hybrid systems describe a wide class of systems that exhibit both discrete and continuous behaviors. The most natural examples of hybrid systems are obtained when a digital system is embedded in an analog environment. Several such systems operate in safety-critical domains, for example, inside automobiles, aircrafts, and chemical plants. Developing effective analysis techniques for hybrid systems will expedite the design process of embedded software while maintaining safety guarantees.

Hybrid automata [2, 35] provide a formalism for modeling hybrid systems by combining the discrete transition system formalism with continuous dynamical systems. The development of tools for analysis of hybrid automata is not an easy task. It has been shown that checking reachability for very simple class of hybrid systems is undecidable [24]. Several decidable classes have been identified, see [5] for a survey, but very often these classes are too weak to represent hybrid system models that arise in practical applications.

A. Tiwari (✉)
SRI International, 333 Ravenswood Ave., Menlo Park, CA, USA
e-mail: tiwari@csl.sri.com

Abstraction is a technique to reduce the complexity of a system design, while preserving some of its relevant behavior, so that the simplified system is more accessible to analysis tools and is still sufficient to establish certain safety properties. A powerful abstraction technique, called predicate abstraction [16], has been successfully used for analyzing discrete transition systems. In Sect. 3 of this paper, we present an abstraction methodology for hybrid systems. The abstraction mapping is defined in terms of a finite set of predicates over the continuous variables. The discrete transitions of the hybrid systems are abstracted using the standard predicate abstraction approach. The continuous behavior is, however, abstracted using qualitative reasoning [30, 44, 45]. In Sect. 2 the formal definition of a discrete abstraction of a hybrid system is given using a simulation relation between the hybrid system and a certain discretization of the hybrid system. The abstraction algorithm is proved correct in Sect. 3.

It is well known that the quality of the abstract system depends crucially on the choice of the abstraction predicates. The same is true in our case. We present a characterization for what constitutes a "good" set of predicates. Thereafter, we describe a collection of methods in Sect. 4 for discovering and generating such predicates by closely analyzing the *continuous* dynamics of the hybrid system. When the dynamics is linear ($\dot{X} = AX$), the eigenstructure of the matrix A plays a crucial role in defining these predicates (Sect. 4.2). When the dynamics is nonlinear, things are not so structured and we have to search for these predicates (Sect. 4.3).

The process of construction of the abstract system requires logical reasoning in some appropriate theory of the reals. The theory should be rich enough to express the hybrid system being abstracted. In our description of the procedure, we will leave the choice of theory open. But a canonical example is the first-order theory of real closed fields (over the signature $\langle +, -, *, >, = \rangle$), which is known to be decidable [47] and to admit some practical algorithms [11, 26, 29, 32]. The theorem proving issues and challenges are discussed in Sect. 6.

If a hybrid system is described as a composition of two hybrid automata, then, under certain conditions, we can abstract it by independently abstracting the two hybrid automata using our abstraction algorithm and composing the abstract transition systems. We precisely define the conditions and prove the correctness of the compositional abstraction algorithm in Sect. 7. The qualitative approach for abstracting continuous dynamics may appear weak, but if the set of predicates is saturated under the Lie derivative computation (as described in Sect. 4.1), then we can prove that the abstraction produced by our method is relatively complete (Sect. 8).

2 Hybrid systems

A *discrete state transition system DS* is a tuple $(Y, Init, t)$, where Y is a finite set of variables interpreted over some (countable or uncountable) domains, $\mathbf{Y}$ denotes the set of all valuations of Y over the respective domains, $Init \subseteq \mathbf{Y}$ is a set of initial states, and $t \subseteq \mathbf{Y} \times \mathbf{Y}$ is a set of transitions. The set $\mathbf{Y}$ is the state-space of *DS*. A *run* of *DS* is any mapping $\sigma : \mathbb{N} \mapsto \mathbf{Y}$ satisfying

(a) Initial condition: $\sigma(0) \in Init$, and
(b) Discrete evolution: for all $i \in \mathbb{N}$, $(\sigma(i), \sigma(i+1)) \in t$.

The set of all runs of *DS* is denoted by $[[DS]]$.

Since our interest is in formal verification of hybrid system models, we formally define *autonomous*, or input-free, hybrid automata.

Definition 1 An autonomous hybrid automaton HS is a tuple $(Q, X, Init, Inv, t, f)$, where Q is a finite set of variables interpreted over finite domains, $\mathbf{Q}$ denotes the finite set of all valuations of the variables Q over the respective finite domains, X is a finite set of variables interpreted over the reals $\mathbb{R}$, $\mathbf{X} = \mathbb{R}^X$ is the set of all valuations of the variables X, $Init \subseteq \mathbf{Q} \times \mathbf{X}$ is a set of initial states, $Inv : \mathbf{Q} \mapsto 2^{\mathbf{X}}$ assigns to each discrete state $q \in \mathbf{Q}$ an invariant set, $t \subset \mathbf{Q} \times \mathbf{X} \times \mathbf{Q} \times \mathbf{X}$ is a set of (guarded) discrete transitions, $f : \mathbf{Q} \mapsto (\mathbf{X} \mapsto T\mathbf{X})$ is a mapping from the discrete states to vector fields that specify the continuous flow in that discrete state.

The set $\mathbf{Q} \times \mathbf{X}$ is the state-space of the hybrid automaton. The variables in Q are said to be *discrete* whereas X are called *continuous*. We refer to $(\mathbf{q}, \mathbf{x}) \in \mathbf{Q} \times \mathbf{X}$ as the *state* of the hybrid automaton HS. We assume that f satisfies the standard assumptions for existence and uniqueness of solutions to ordinary differential equations. For example, f could be specified using polynomials over X.

Example 1 Consider a thermostat that controls the heating of a room. Assume that the thermostat turns the heater on when the temperature is between 68 and 70 and it turns the heater off when the temperature is between 80 and 82. Suppose the continuous dynamics in the on and off modes is specified by the equations $\dot{x} = -x + 100$ and $\dot{x} = -x$ respectively. If we assume that the heater is initially off and the room temperature is between 70 and 80, the hybrid automaton is given by $HS = (Q, X, Init, Inv, t, f)$, where $Q = \{q_1\}$ is the set of discrete variables, $\mathbf{Q} = \{on, off\}$ is the set of discrete states (thus, $q_1 \in \{on, off\}$), $X = \{x_1\}$ is the set of continuous variables, $\mathbf{X} = \mathbb{R}$ is the set of continuous states, $Init = \{(off, x) : 70 < x < 80\}$ is the initial condition, $Inv = \{(on, x) : x < 82\} \cup \{(off, x) : x > 68\}$ is the invariant set, $t = \{(on, x, off, x) : x \geq 80\} \cup \{(off, x, on, x) : x \leq 70\}$ is the set of discrete transitions, and $f(on) = -x + 100$ and $f(off) = -x$ specifies the continuous flows.

Let $\mathbf{S}$ be a (finite) set and $\alpha : \mathbf{Q} \times \mathbf{X} \mapsto \mathbf{S}$ be a function that maps the uncountable state-space of HS onto $\mathbf{S}$. The set $\mathbf{S}$ can be seen as the set of observed states of the hybrid system HS. In the context of a given mapping α, the semantics of the hybrid automaton HS can be specified by associating a discrete transition system HS_α with it.

Definition 2 Given a hybrid automaton $HS = (Q, X, Init, Inv, t, f)$ and a mapping $\alpha : \mathbf{Q} \times \mathbf{X} \mapsto \mathbf{S}$, the *discrete transition system corresponding to HS* is the system $HS_\alpha = (Q \cup X, Init, t_\alpha)$ with the same state-space and initial states, and the following transitions:

(a) Discrete transitions: $((\mathbf{q}, \mathbf{x}), (\mathbf{q}', \mathbf{x}')) \in t_\alpha$ if $(\mathbf{q}, \mathbf{x}, \mathbf{q}', \mathbf{x}') \in t$, and
(b) Continuous transitions: $((\mathbf{q}, \mathbf{x}), (\mathbf{q}, \mathbf{x}')) \in t_\alpha$ if there exists a $\delta > 0$ and a continuous function $F : [0, \delta] \mapsto \mathbf{X}$ such that for all $\tau \in (0, \delta)$, $\dot{F}(\tau) = f(\mathbf{q})(F(\tau))$, $F(\tau) \in Inv(\mathbf{q})$, *and* $\alpha((\mathbf{q}, F(\tau)))$ is a constant function on either the domain $[0, \delta)$ or the domain $(0, \delta]$, that is, either $\alpha((\mathbf{q}, F(\tau))) = \alpha((\mathbf{q}, F(0)))$ for all $\tau \in [0, \delta)$, or $\alpha((\mathbf{q}, F(\tau))) = \alpha((\mathbf{q}, F(\delta)))$ for all $\tau \in (0, \delta]$.

The set $[[HS]]$ of *runs* of the hybrid automaton HS with respect to the mapping α is now defined simply as $[[HS_\alpha]]$. Intuitively speaking, the discrete transition system HS_α captures all the different "observed" behaviors of the hybrid system HS.

Note that we have assumed that there are no inputs. When taking a discrete transition, both the discrete variables and the continuous variables can change. In other words, the discrete transitions could contain update functions for continuous variables.

The semantics of a hybrid system have been defined alternatively as a collection of runs, where each run is a mapping from a dense time interval to the state-space, or as an infinite-state discrete transition system where only the discrete transitions are observable [1, 20, 24]. The semantics we use here captures the behavior of the hybrid system during the discrete as well as the continuous evolutions. This is the reason for the extra side condition in Definition 2(b).

A hybrid automaton consisting of only one mode (that is, $Q = \emptyset$) and no discrete transitions (that is, $t = \emptyset$) is called a continuous dynamical system. It can be represented by the tuple $(X, \mathit{Init}, \mathit{Inv}, f)$, or simply by (X, Init, f) if the invariant is the whole space $\mathbf{X}$.

Example 2 (Hybrid Automata) The Delta and Notch proteins are involved in the process of cell differentiation through lateral inhibition. A simple model of a cell with these two proteins can be described by a hybrid system containing two state variables $X = \{x_d, x_n\}$ that store values of the concentrations of these two proteins in the cell [14]. The transcription of these two proteins could independently be either "on" or "off", so the cell can be in four modes. Thus, there are two discrete variables, $Q = \{q_d, q_n\}$ that take values over the domain $\{\mathit{off}, \mathit{on}\}$. The rules of the lateral inhibition mechanism assert that Notch inhibits the production of Delta in the same cell, whereas Delta promotes Notch production in the *adjacent* cell. Let x_u be a parameter representing Delta concentration in the environment. Hence, we have $HS = (Q, X, \mathit{Init}, \mathit{Inv}, t, f)$, where the continuous flow f is specified by

$$
\begin{aligned}
f(\{q_d = \mathit{off}, q_n = \mathit{off}\}) &= [-\lambda_d x_{di}, -\lambda_n x_{ni}],\\
f(\{q_d = \mathit{on}, q_n = \mathit{off}\}) &= [\Delta_d - \lambda_d x_{di}, -\lambda_n x_{ni}],\\
f(\{q_d = \mathit{off}, q_n = \mathit{on}\}) &= [-\lambda_d x_{di}, \Delta_n - \lambda_n x_{ni}],\\
f(\{q_d = \mathit{on}, q_n = \mathit{on}\}) &= [\Delta_d - \lambda_d x_{di}, \Delta_n - \lambda_n x_{ni}]
\end{aligned}
$$

and the discrete transitions t—twelve in all, one from each of the four modes to each other mode—are obtained using the rules that $q_n = \mathit{off}$ whenever $x_u < h_n$, $q_n = \mathit{on}$ whenever $x_u \geq h_n$, $q_d = \mathit{off}$ whenever $x_n > h_d$, and $q_d = \mathit{on}$ whenever $x_n \leq h_d$. Here $x_u, \lambda_d, \lambda_n, \Delta_d, \Delta_n, h_n, h_d$ are parameters that take values in $\mathbb{R}$.

We are now ready to define what we mean by a discrete (finite-state) abstraction of a hybrid system.

Definition 3 Let $HS = (Q, X, \mathit{Init}, \mathit{Inv}, t, f)$ be a hybrid automata and $DS = (Q', \mathit{Init}', t')$ be a discrete transitions system. We say DS is an *abstraction* for HS if there exists a mapping $\alpha : \mathbf{Q} \times \mathbf{X} \mapsto \mathbf{Q}'$ such that

(a) If $(\mathbf{q}, \mathbf{x}) \in \mathit{Init}$, then $\alpha(\mathbf{q}, \mathbf{x}) \in \mathit{Init}'$, and
(b) If $((\mathbf{q}, \mathbf{x}), (\mathbf{q}', \mathbf{x}')) \in t_\alpha$ is a transition in the discrete transition system HS_α corresponding to HS with respect to α, then there exists a transition $(\alpha(\mathbf{q}, \mathbf{x}), \alpha(\mathbf{q}', \mathbf{x}')) \in t'$ in DS.

In other words, the abstraction DS of a hybrid automaton HS is a discrete transition system that *simulates* the discrete system HS_α associated with HS, where α defines the corresponding *simulation relation* [33]. Consequently, if a $ACTL^*$ formula is true in the model DS, then it is also true in HS_α [18].

We consider the problem of constructing discrete transition system abstractions for continuous dynamical systems and hybrid systems in the sense of Definition 3.

2.1 Representation

We represent a set of states of a hybrid automaton by a formula in a suitable logical theory Th. For specifying hybrid systems, the theory of reals is pertinent. A *signature* is a set of function and constant symbols Σ_F, and predicate symbols Σ_P. For example, $\{+, -, \cdot, \hat{}, \exp, \log, \sin\}$ are function symbols and $\{=, >, \geq\}$ are examples of predicate symbols. The theory of real-closed fields, $\mathbb{R}$, for example, works over the signature $\{\mathbb{Q}, +, -, \cdot, =, >, \geq\}$, where $\mathbb{Q}$ is the set of rational constants. The set $\mathcal{T}(X)$ of terms over the variables X (and some signature) is defined in the usual way. For example, in the theory $\mathbb{R}$, the set of terms over a set X of variables corresponds to the set of polynomials $\mathbb{Q}[X]$. The set $ATM(X)$ of atomic formulas is obtained by applying a predicate symbol to terms from $\mathcal{T}(X)$. If the signature contains the minus symbol and $=, >, <, \geq, \leq$ are the only predicate symbols, then atomic formulas over reals can always be written as $p \sim 0$, where $\sim \in \{=, >, \geq\}$ and $p \in \mathcal{T}(X)$. The set $WFF(X)$ of well formed formulas (over X) is defined as the smallest set containing $ATM(X)$ and closed under the boolean operations (conjunction $\wedge$, disjunction $\vee$, and negation $\neg$) and quantification (existential $\exists$ and universal $\forall$). We denote formulas in $WFF(X)$ by Greek symbols ϕ, ψ, possibly with subscripts and use p to denote terms in the set $\mathcal{T}(X)$.

Let Th be some theory interpreted over the reals. We use the notation $Th \models \phi(X)$ to denote the fact that the (first-order) formula ϕ is true, in the theory of reals, for *all* valuations for X, that is, $Th \models \forall X : \phi(X)$. Recall that we use $\mathbf{x} \in \mathbb{R}^X$ to denote a valuation, or a point. Thus, the notation $Th \models \phi(\mathbf{x})$ denotes that ϕ evaluates to true under the given valuation $\mathbf{x}$. We say a term p occurs in a formula ϕ if there is an atomic formula $p \sim 0$ in ϕ.

Let X be a finite set of real valued variables. A formula $\phi(X)$ in $WFF(X)$ represents a set $\{\mathbf{x} \in \mathbf{X} : Th \models \phi(\mathbf{x})\}$ of states, which is denoted by $[[\phi]]$. For example, the formula $x_1 > 0 \wedge x_2 > 0$ represents the first quadrant in the 2-dimensional x_1-x_2 plane. If Q is a finite set of discrete variables, then we will assume that the set $\mathbf{Q}$ of discrete modes is finite, and hence we can use any explicit representation for subsets of $\mathbf{Q}$.

Consider a hybrid automaton $HS = (Q, X, Init, Inv, t, f)$. The sets Q and X can be specified by explicitly enumerating them. The sets $Init$ and Inv are represented using formulas from $WFF(X)$, one for each discrete mode. We assume that $Inv^* : \mathbf{Q} \mapsto WFF(X)$ represents the invariant. The set t of discrete transitions are specified by giving the source mode $\mathbf{q}$, the target mode $\mathbf{q}'$, the guard condition $\phi(X)$ (which should evaluate to true for the transition to be enabled), and a set of assignments $X := F(X)$, where $F(X)$ is a vector of terms over X. This is written as

$$\mathbf{q}, \phi(X) \longrightarrow \mathbf{q}', \quad X := F(X).$$

The set f is specified by enumerating all differential equations $\dot{x} = p$, one for each variable $x \in X$ and each mode in $\mathbf{Q}$.

2.1.1 Inside-out representation

Switched hybrid systems are special kinds of hybrid systems which do not involve any updates to the continuous variables during a discrete transition. In practice, the explicit representation of a switched hybrid automaton, as described above, may not be sufficiently succinct. Alternatively, the continuous dynamics (the f component) can be specified just once using additional parameters. The discrete transitions (the t component) are then specified using conditional assignments to these parameters, thus modifying the dynamics in the different discrete modes. This style of specification avoids any explicit enumeration of modes.

Any switched system can be represented in this way by introducing sufficiently many parameters. Our hybrid abstraction algorithm can use the succinct representation to optimize the process of abstracting the given switched hybrid system.

Example 3 Consider the hybrid system *HS* from Example 2. The hybrid automaton *HS* can be specified succinctly using an inside-out representation by giving the continuous dynamics as:

$$\dot{x_d} = D_d - \lambda_d x_d,$$
$$\dot{x_n} = D_n - \lambda_n x_n$$

and specifying the discrete transitions by the following conditional assignments to the newly introduced parameters D_d and D_n:

$$D_d = \text{ if } (x_n > h_d) \text{ then } 0 \text{ else } \Delta_d,$$
$$D_n = \text{ if } (x_u < h_n) \text{ then } 0 \text{ else } \Delta_n.$$

These four equations completely specify the dynamics of *HS*.

3 Abstracting hybrid systems

Predicate abstraction refers to the idea of using predicates on the original state variables as abstract variables in the new discrete system. These abstract variables take values on a boolean domain, thus resulting in finite state systems. In this paper, we fix a set $P \subset \mathcal{T}(X)$ of terms and abstract over the $3 * |P|$ predicates $\{p > 0, p = 0, p < 0 : p \in P\}$. But instead of using boolean variables in the abstract system, we will use variables interpreted over the three valued domain $\{neg, pos, zero\}$. Note that in the special case when $Th = \mathbb{R}$, the terms in P would be *polynomials* in $\mathbb{Q}[X]$.

We abstract a hybrid automaton $HS = (Q, X, InitX, Inv, t, f)$ over a given finite set $P \subset \mathcal{T}(X)$ of terms. The result will be a discrete state transition system $DS = (Q^A, Init^A, t^A)$ where $Q^A = Q \cup Q_P$ is the set of discrete variables, $Q_P = \{q_p : p \in P\}$, $Init^A \subseteq \mathbf{Q^A}$ is the set of initial states, and $t^A \subseteq \mathbf{Q^A} \times \mathbf{Q^A}$ is the set of transitions. The new discrete variables Q_P are interpreted over the domain $\{pos, neg, zero\}$. Thus, the set of states in the abstract system $\mathbf{Q^A}$ is $\mathbf{Q} \times \mathbf{Q_P}$ where $\mathbf{Q_P} = \{pos, neg, zero\}^{Q_P}$.

Let $\mathbf{q_P} \in \mathbf{Q_P}$ be the abstract state $((q_p = pos)_{p\in P_1}, (q_p = zero)_{p\in P_2}, (q_p = neg)_{p\in P_3})$, where $P_1 \cup P_2 \cup P_3$ is a partition of the set P. The concretization function $\gamma : \mathbf{Q^A} \mapsto 2^{\mathbf{Q}\times\mathbf{X}}$ is a mapping from the abstract states to subsets of the concrete states and is defined as follows:

$$\gamma((\mathbf{q}, \mathbf{q_P})) = \left\{(\mathbf{q}, \mathbf{x}) \in \mathbf{Q} \times \mathbf{X} | \bigwedge_{p\in P_1} p(\mathbf{x}) > 0 \wedge \bigwedge_{p\in P_2} p(\mathbf{x}) = 0 \wedge \bigwedge_{p\in P_3} p(\mathbf{x}) < 0\right\}. \quad (1)$$

Conversely, if $(\mathbf{q}, \mathbf{x}) \in \mathbf{Q} \times \mathbf{X}$ is a concrete state of the system *HS*, then the abstraction function, $\alpha : \mathbf{Q} \times \mathbf{X} \mapsto \mathbf{Q^A}$, is defined by,

$$\alpha((\mathbf{q}, \mathbf{x})) = (\mathbf{q}, ((q_p = pos)_{p\in P_1}, (q_p = zero)_{p\in P_2}, (q_p = neg)_{p\in P_3})), \quad (2)$$

where $P_1 \cup P_2 \cup P_3$ is a partition of the set P such that $p \in P_1$ iff $p(\mathbf{x}) > 0$, $p \in P_2$ iff $p(\mathbf{x}) = 0$, and $p \in P_3$ iff $p(\mathbf{x}) < 0$. Note that we do not abstract the discrete state space and only abstract the continuous state space to a finite set.

We will also use γ and α to denote the restriction of the corresponding functions to the second components so that $\gamma : \mathbf{Q_P} \mapsto 2^{\mathbf{X}}$ and $\alpha : \mathbf{X} \mapsto \mathbf{Q_P}$. The function α can be naturally extended to map *subsets* of the concrete states, $2^{\mathbf{X}}$, to *subsets* of the abstract states, $2^{\mathbf{Q_P}}$, that is, $\alpha : 2^{\mathbf{X}} \mapsto 2^{\mathbf{Q_P}}$. If $S \subseteq 2^{\mathbf{X}}$, we define $\alpha(S)$ as follows:

$$\alpha(S) = \bigcup_{\mathbf{x} \in S} \{\alpha(\mathbf{x})\}. \tag{3}$$

The context will disambiguate these usages of the abstraction and concretization functions.

3.1 The abstract initial states

Since we use formulas in $WFF(X)$ (over some theory Th) to represent elements of $2^{\mathbf{X}}$, we will be interested in the variants, $\alpha^* : WFF(X) \mapsto 2^{\mathbf{Q_P}}$ and $\gamma^* : \mathbf{Q_P} \mapsto WFF(X)$, of the abstraction and concretization functions, $\alpha : 2^{\mathbf{X}} \mapsto 2^{\mathbf{Q_P}}$ and $\gamma : \mathbf{Q_P} \mapsto 2^{\mathbf{X}}$. If $\mathbf{q_P}$ is abstract state $((q_p = pos)_{p \in P_1}, (q_p = zero)_{p \in P_2}, (q_p = neg)_{p \in P_3})$, then $\gamma^*(\mathbf{q_P})$ is defined as

$$\gamma^*(\mathbf{q_P}) = \bigwedge_{p \in P_1} p(\mathbf{x}) > 0 \ \wedge \ \bigwedge_{p \in P_2} p(\mathbf{x}) = 0 \ \wedge \ \bigwedge_{p \in P_3} p(\mathbf{x}) < 0. \tag{4}$$

We define α^* using Th-satisfiability.

$$\alpha^*(\phi(X)) = \{\mathbf{q_P} \in \mathbf{Q_P} : Th \models \exists X : \gamma^*(\mathbf{q_P}) \wedge \phi(X)\}. \tag{5}$$

If the initial states $Init$ of the hybrid system HS is $\bigcup_i \{(\mathbf{q_i}, \mathbf{x}) : \mathbf{x} \in [[\phi^i]]\}$, where each formula $\phi^i \in WFF(X)$ represents the initial states in a given discrete mode $\mathbf{q_i} \in \mathbf{Q}$, then the initial states $Init^A$ are obtained as $\bigcup_i \{(\mathbf{q_i}, \mathbf{q_P}) : \mathbf{q_P} \in \alpha^*(\phi^i)\}$.

Lemma 1 *If $\phi(X)$ is a formula representing a set in $2^{\mathbf{X}}$, then $\alpha([[\phi]]) \subseteq \alpha^*(\phi)$. In particular, if $HS = (Q, X, Init, Inv, t, f)$ is a hybrid system with initial states $Init = \bigcup_i \{(\mathbf{q_i}, \mathbf{x}) : \mathbf{x} \in [[\phi^i]]\}$ and DS, $Init^A$, and α are as defined as above, then, $\alpha(Init) \subseteq Init^A$.*

Proof Let $\mathbf{x} \in [[\phi]]$. We show that $\alpha(\mathbf{x}) \in \alpha^*(\phi)$. It follows from the definition of the γ^* and α functions that the formula $\gamma^*(\alpha(\mathbf{x}))$ evaluates to true on the point $\mathbf{x}$. By assumption, the same is true for the formula ϕ. Hence, $\alpha(\mathbf{x}) \in \alpha^*(\phi)$. Applying this result to each mode of the hybrid system, the second part of the claim is immediately proved. □

3.2 The abstract transition relation

The transitions t^A of the abstract system $DS = (Q^A, Init^A, t^A)$ are obtained by abstracting the discrete transitions and the continuous flow of the concrete hybrid system HS separately.

3.2.1 Abstractions of the discrete transitions

Let $(\mathbf{q}, \phi_1(X), \mathbf{q}', \phi_2(X))$ represent the set of discrete transitions $(\mathbf{q}, \mathbf{x}, \mathbf{q}', \mathbf{x}') \in t$ of the hybrid automaton HS, where $\mathbf{x} \in [[\phi_1]]$ and $\mathbf{x}' \in [[\phi_2]]$. Then, the corresponding abstract transitions t^A contain $((\mathbf{q}, \mathbf{q_P}), (\mathbf{q}', \mathbf{q}'_\mathbf{P}))$, where $\mathbf{q_P} \in \alpha^*(\phi_1(X))$ and $\mathbf{q}'_\mathbf{P} \in \alpha^*(\phi_2(X))$.

However, the discrete transitions of a hybrid automaton are often specified by a guarded command

$$\mathbf{q}, \phi(X) \longrightarrow \mathbf{q}', \quad X := F(X), \tag{6}$$

where $\mathbf{q}$ is the source mode, $\mathbf{q}'$ is the target mode, $\phi(X)$ is the guard condition, and $X := F(X)$ is a set of assignments. In this case, for each function $p_i(X) \in P$, we compute the subsets $Q_{i1}, Q_{i2}, Q_{i3} \subseteq 2^{\mathbf{Q_P}}$ of abstract states such that

$$Th \models \phi(X) \wedge Inv^*(\mathbf{q}) \wedge \bigvee_{\mathbf{q_P} \in Q_{i1}} \gamma^*(\mathbf{q_P}) \quad \Rightarrow \quad p_i[X/F(X)] \geq 0,$$

$$Th \models \phi(X) \wedge Inv^*(\mathbf{q}) \wedge \bigvee_{\mathbf{q_P} \in Q_{i2}} \gamma^*(\mathbf{q_P}) \quad \Rightarrow \quad p_i[X/F(X)] \neq 0,$$

$$Th \models \phi(X) \wedge Inv^*(\mathbf{q}) \wedge \bigvee_{\mathbf{q_P} \in Q_{i3}} \gamma^*(\mathbf{q_P}) \quad \Rightarrow \quad p_i[X/F(X)] \leq 0.$$

The notation $p[X/F(X)]$ represents the term obtained by replacing the variables X in p by terms $F(X)$. For every $\mathbf{q_P} \in \alpha^*(\phi)$, we add the following transition in the abstract system:

$$\mathbf{q}, \mathbf{q_P} \longrightarrow \mathbf{q}', \quad (q_{p_i} :\in ITE1(\mathbf{q_P}, Q_{i1}, Q_{i2}, Q_{i3}))_{p_i \in P}, \tag{7}$$

where the notation $a :\in A$ means that a is nondeterministically *assigned to* some element in A and the function $ITE1(\mathbf{q_P}, Q_1, Q_2, Q_3)$ is defined as

$$\begin{aligned} ITE1(\mathbf{q_P}, Q_1, Q_2, Q_3) = \ & \text{if } (\mathbf{q_P} \in Q_1 \cap Q_2) \text{ then } \{pos\}, \\ & \text{elsif } (\mathbf{q_P} \in Q_3 \cap Q_2) \text{ then } \{neg\}, \\ & \text{elsif } (\mathbf{q_P} \in Q_1 \cap Q_3) \text{ then } \{zero\}, \\ & \text{elsif } (\mathbf{q_P} \in Q_1) \text{ then } \{pos, zero\}, \\ & \text{elsif } (\mathbf{q_P} \in Q_2) \text{ then } \{pos, neg\}, \\ & \text{elsif } (\mathbf{q_P} \in Q_3) \text{ then } \{neg, zero\}, \\ & \text{else } \{pos, neg, zero\}. \end{aligned}$$

All the different abstract transitions (7) arising from the same guarded command (6) can be represented by a single guarded command. Each discrete transition in the hybrid automaton *HS* is abstracted in this way. This completes the abstraction of discrete transitions with updates. Note that if $p_i(X)$ does not contain any of the variables that are updated by the assignments, then the value of q_{p_i} can be left unchanged (and we need not compute the sets Q_{i1}, Q_{i2}, and Q_{i3} in this case).

3.2.2 Abstracting the continuous flow

The continuous evolution of the hybrid system is abstracted using qualitative reasoning. Let $X = \{x_i : i = 1, 2, \ldots, n\}$ be the n real valued variables. Let $p \in P$ be a term over the variables X. The notation $\vec{d}p$ denotes the $1 \times n$ row vector, or the 1-form, consisting of partial derivatives of p with respect to the n variables, that is, $\vec{d}p = [\partial p/\partial x_1, \partial p/\partial x_2, \ldots, \partial p/\partial x_n]$. If $\mathbf{q}$ is a mode of the hybrid automaton *HS* and $f(\mathbf{q})$ is the vector field in that mode, then the Lie derivative, $L_{f(\mathbf{q})}(p)$, of p with respect to the vector field $f(\mathbf{q})$ is defined as

$$L_{f(\mathbf{q})}(p) = \vec{d}pf(\mathbf{q}) = \frac{\partial p}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial p}{\partial x_2}\frac{dx_2}{dt} + \cdots + \frac{\partial p}{\partial x_n}\frac{dx_n}{dt}. \tag{8}$$

Thus, the Lie derivative $L_{f(\mathbf{q})}(p)$ is just the derivative dp/dt of p with respect to time in mode $\mathbf{q}$.

Let $\mathbf{q}$ be a mode of the hybrid automaton HS and $f(\mathbf{q})$ be the vector field in that mode. We add an abstract transition $((\mathbf{q}, \mathbf{q_P}), (\mathbf{q}, \mathbf{q'_P})) \in t^A$ whenever all of the following conditions hold (for all $p \in P$):

(a) If $q_p = neg$ in the state $\mathbf{q_P}$, then
  (a1) If $Th \models \gamma^*(\mathbf{q_P}) \wedge Inv^*(\mathbf{q}) \Rightarrow L_{f(\mathbf{q})}(p) \leq 0$, then $q_p = neg$ in $\mathbf{q'_P}$;
  (a2) If not, then either $q_p = neg$ or $q_p = zero$ holds in $\mathbf{q'_P}$.
(b) If $q_p = zero$ in the state $\mathbf{q_P}$, then
  (b1) If $Th \models \gamma^*(\mathbf{q_P}) \wedge Inv^*(\mathbf{q}) \Rightarrow L_{f(\mathbf{q})}(p) < 0$, then $q_p = neg$ in $\mathbf{q'_P}$;
  (b2) If $Th \models \gamma^*(\mathbf{q_P}) \wedge Inv^*(\mathbf{q}) \Rightarrow L_{f(\mathbf{q})}(p) = 0$, then $q_p = zero$ in $\mathbf{q'_P}$;
  (b3) If $Th \models \gamma^*(\mathbf{q_P}) \wedge Inv^*(\mathbf{q}) \Rightarrow L_{f(\mathbf{q})}(p) > 0$, then $q_p = pos$ in $\mathbf{q'_P}$;
  (b4) If not, then either $q_p = pos$, $q_p = zero$, or $q_p = neg$ holds in $\mathbf{q'_P}$.
(c) If $q_p = pos$ in the state $\mathbf{q_P}$, then
  (c1) If $Th \models \gamma^*(\mathbf{q_P}) \wedge Inv^*(\mathbf{q}) \Rightarrow L_{f(\mathbf{q})}(p) \geq 0$, then $q_p = pos$ in $\mathbf{q'_P}$;
  (c2) If not, then either $q_p = pos$ or $q_p = zero$ holds in $\mathbf{q'_P}$.

This procedure is implemented in the following way. For each mode $\mathbf{q} \in \mathbf{Q}$ and for each term $p_i \in P$, we first compute $L_{f(\mathbf{q})}(p_i)$ symbolically. Thereafter, we compute the subsets $Q_{i1}, Q_{i2}, Q_{i3} \subseteq 2^{\mathbf{Q_P}}$ of abstract states such that

$$Th \models Inv^*(\mathbf{q}) \wedge \bigvee_{\mathbf{q_P} \in Q_{i1}} \gamma^*(\mathbf{q_P}) \quad \Rightarrow \quad L_{f(\mathbf{q})}(p_i) \geq 0,$$

$$Th \models Inv^*(\mathbf{q}) \wedge \bigvee_{\mathbf{q_P} \in Q_{i2}} \gamma^*(\mathbf{q_P}) \quad \Rightarrow \quad L_{f(\mathbf{q})}(p_i) \neq 0,$$

$$Th \models Inv^*(\mathbf{q}) \wedge \bigvee_{\mathbf{q_P} \in Q_{i3}} \gamma^*(\mathbf{q_P}) \quad \Rightarrow \quad L_{f(\mathbf{q})}(p_i) \leq 0.$$

Subsequently, for each $\mathbf{q_P} \in \alpha(Inv(\mathbf{q}))$, we generate the following discrete abstract transition

$$\mathbf{q}, \mathbf{q_P} \longrightarrow \mathbf{q}, \quad (q_{p_i} :\in ITE2(\mathbf{q_P}, q_{p_i}, Q_{i1}, Q_{i2}, Q_{i3}))_{p_i \in P},$$

where the mode $\mathbf{q}$ is left unchanged, but the abstract variables q_{p_i} are assigned a value from the set returned by the function $ITE2(\mathbf{q_P}, q_{p_i}, Q_{i1}, Q_{i2}, Q_{i3})$, which is defined as follows:

$$\begin{aligned} ITE2(\mathbf{q_P}, q_p, Q_1, Q_2, Q_3) = \ & \text{if } q_p = pos, \\ & \text{if } \mathbf{q_P} \in Q_1 \text{ then } \{pos\} \text{ else } \{pos, zero\}, \\ & \text{elsif } q_p = neg, \\ & \text{if } \mathbf{q_P} \in Q_3 \text{ then } \{neg\} \text{ else } \{neg, zero\}, \\ & \text{else}, \\ & \text{if } (\mathbf{q_P} \in Q_1 \cap Q_2) \text{ then } \{pos\}, \\ & \text{elsif } (\mathbf{q_P} \in Q_3 \cap Q_2) \text{ then } \{neg\}, \\ & \text{elsif } (\mathbf{q_P} \in Q_1 \cap Q_3) \text{ then } \{zero\}, \\ & \text{else } \{pos, neg, zero\}. \end{aligned}$$

We repeat the process for each mode and compute the abstract transitions arising from the continuous dynamics of each mode separately. This completes the phase of adding transitions to the abstract system.

Note that we can handle cases where the set $\mathbf{Q}$ of discrete states in *HS* is infinite as long as the number of distinct "modes" (which can each be specified as a formula over Q) are finite.

Theorem 1 *Let $HS = (Q, X, Init, Inv, t, f)$ be a hybrid automaton and $P \subset \mathcal{T}(X)$ be a finite set of terms over the set X of real variables. If $DS = (Q^A = Q \cup Q_P, Init^A, t^A)$ is the discrete transition system constructed by the above method, then DS is an abstraction for HS.*

Proof If $(\mathbf{q}, \mathbf{x}) \in \mathbf{Q} \times \mathbf{X}$ is a concrete state of the hybrid system *HS*, then the abstraction mapping α is defined by (2). Lemma 1 establishes condition (a) of Definition 3. In order to establish condition (b), let $((\mathbf{q}, \mathbf{x}), (\mathbf{q}', \mathbf{x}')) \in t_\alpha$ be a transition in the discrete system HS_α corresponding to *HS* with respect to the mapping α. There are two cases based on whether this is a discrete (Definition 2(a)) or continuous (Definition 2(b)) transition.

Continuous transition In this case $\mathbf{q}' = \mathbf{q}$. Let F be as in Definition 2(b). Without loss of generality, assume that $\alpha(\mathbf{q}, F(\tau))$ is a constant function on $[0, \delta)$. We claim that $(\alpha(\mathbf{q}, \mathbf{x}), \alpha(\mathbf{q}, \mathbf{x}')) \in t^A$. Let

$$\alpha(\mathbf{x}) = ((q_p = pos)_{p \in P_1}, (q_p = zero)_{p \in P_2}, (q_p = neg)_{p \in P_3}).$$

There are two subcases

(1) $\alpha(\mathbf{x}) = \alpha(\mathbf{x}')$: Consider $p \in P_2$. We note that $p(F(\tau)) = 0$ for all $\tau \in [0, \delta]$. It follows that $L_{f(\mathbf{q})}(p) = 0$ at all points $F(\tau)$, in particular $L_{f(\mathbf{q})}(p) = 0$ at $F(0) = \mathbf{x}$. At point $\mathbf{x}$, the formula $\gamma^*(\alpha(\mathbf{x}))$ evaluates to true, the formula $Inv^*(\mathbf{q})$ is true, and the formula $L_{f(\mathbf{q})}(p) = 0$ is true. Therefore, it is *not possible* to prove the theorems in cases (b1) or (b3). Hence, we would apply either case (b2) or (b4) and in both these cases we have the choice of maintaining $q_p = zero$. Finally, for $p \in P_1 \cup P_3$, all cases in (a) and (c) allow for the possibility of keeping the sign of q_p unchanged.

(2) $\alpha(\mathbf{x}) \neq \alpha(\mathbf{x}')$: If for some $p \in P_1$, $p(F(\delta)) = 0$, then for some $\tau \in (0, \delta)$, $L_{f(\mathbf{q})}(p) < 0$ at the point $F(\tau) \in \mathbb{R}^n$. Let $\mathbf{x}''$ be this intermediate point $F(\tau)$. Now, at this point, $\gamma^*(\alpha(\mathbf{x}))$ evaluates to true (since, by assumption, $\alpha(\mathbf{x}) = \alpha(\mathbf{x}'')$) and $Inv^*(\mathbf{q})$ also evaluates to true (since again, by assumption, the invariant is true at all intermediate points), but $L_{f(\mathbf{q})}(p) \geq 0$ evaluates to false. Consequently, case (c1) cannot be applicable and we have to use case (c2), which shows that q_p can be chosen to be *zero*. Using similar arguments for all other cases for $p \in P_2$ and $p \in P_3$, we conclude that there will be a transition $(\alpha(\mathbf{q}, \mathbf{x}), \alpha(\mathbf{q}, \mathbf{x}')) \in t^A$.

Discrete transition. If $((\mathbf{q}, \mathbf{x}), (\mathbf{q}', \mathbf{x}')) \in t_\alpha$ is a discrete transition (from Definition 2(b)), then by definition $(\mathbf{q}, \mathbf{x}, \mathbf{q}', \mathbf{x}') \in t$. Assume that this transition is captured in *HS* by the guarded assignment, $\mathbf{q}, \phi(X) \longrightarrow \mathbf{q}', X := F(X)$. Therefore, the formula ϕ evaluates to true at the point $\mathbf{x}$. It follows from Lemma 1 that $\alpha(\mathbf{x})$ is in $\alpha^*(\phi)$. Finally, we note that for $p \in P$, the value of p after the updates will be identical to the value of $p[X/F(X)]$ before the transition. It now follows from the way the discrete transition is abstracted that there is a transition $(\mathbf{q}, \alpha(\mathbf{x}), \mathbf{q}', \alpha(\mathbf{x}')) \in t^A$. This completes the proof. □

Example 4 We abstract the thermostat model from Example 1 using the terms $P = \{x, x - 68, x - 70, x - 80, x - 82, x - 100\}$, labeled $p_1, \ldots, p_6$. The abstract discrete transition system is defined over seven variables, $\{q_1, q_{p_1}, \ldots, q_{p_6}\}$. The dynamics are given

as:

$$
\begin{aligned}
&q_1 = on, \quad q_{p_4} = pos \quad \longrightarrow \quad q_1 := off,\\
&q_1 = off, \quad q_{p_3} = neg \quad \longrightarrow \quad q_1 := on,\\
&q_1 = on, \quad q_{p_5} = neg \quad \longrightarrow \quad (q_{p_i} :\in VN(q_{p_1}, \ldots, q_{p_6}, i, 6))_{i=1,\ldots,6},\\
&q_1 = off, \quad q_{p_2} = pos \quad \longrightarrow \quad (q_{p_i} :\in VN(q_{p_1}, \ldots, q_{p_6}, i, 1))_{i=1,\ldots,6},
\end{aligned}
$$

where the function $VN(a_1, a_2, a_3, a_4, a_5, a_6, i, j) = ITE2((a_1, \ldots, a_6), a_i, \{\mathbf{q_P} \in \mathbf{Q_P} : q_{p_j} \neq pos\}, \{\mathbf{q_P} \in \mathbf{Q_P} : q_{p_j} \neq zero\}, \{\mathbf{q_P} \in \mathbf{Q_P} : q_{p_j} \neq neg\})$. Note that the first two transitions above are abstractions of the discrete transitions, while the latter two are abstractions of the continuous dynamics. We have simplified the formulas here. For example, $x > 80$ should be abstracted to $q_{p_4} = pos \ \wedge \ q_{p_3} = pos \ \cdots$, but we have just included $q_{p_4} = pos$ above. We also note that certain abstract states are infeasible, for example, $q_{p_4} = pos \ \wedge \ q_{p_3} = zero$. In Sect. 5, we will add information to the abstract system to guarantee that the abstract trajectories remain inside the feasible region and the invariant set.

4 Choosing the abstraction mapping

The quality of the abstraction computed by our method depends crucially on the terms P over which the abstraction is computed. In this section, we discuss the approaches to identify (compute) P. It is clear that the terms that occur in the statement of the property we wish to verify are ideal candidates to include in P. For example, if we wish to verify that $x_1 > x_2$ always, then we should add the term $x_1 - x_2$ to P. Similarly, the terms that occur in the guards of discrete transitions are added to P. The set P thus constructed is the seed set. We will add more elements to P using the techniques described below.

4.1 Higher-order lie derivatives

Consider a set $P_0 = \{p_1, p_2, \ldots, p_k\}$ of k terms over variables X. Let $\psi \in WFF(X)$ be a formula containing free variables from X. Define the *extended monoid* of the set P_0 (relative to ψ) as the minimal set $EMon_\psi(P_0)$ such that (i) $P_0 \subset EMon_\psi(P_0)$, (ii) $r \in EMon_\psi(P_0)$ if r is zero, positive or negative definite relative to ψ (that is, $Th \models \psi \Rightarrow r = 0$, or $Th \models \psi \Rightarrow r > 0$, or $Th \models \psi \Rightarrow r < 0$), (iii) $p_1 p_2 \in EMon_\psi(P_0)$ whenever $p_1, p_2 \in EMon_\psi(P_0)$. In other words, the set $EMon_\psi(P_0)$ is the monoid over P_0 *and* all relatively positive and negative definite functions. An important property of the set $EMon_\psi(P_0)$ is that if the signs (positive, negative, or zero) of all the terms in P_0 are known, then the sign of any term in $EMon_\psi(P_0)$ can be uniquely determined (assuming ψ holds).

We add new functions to the set P by saturating the existing terms in P under the Lie derivative computation (with respect to vector field in a particular mode). Let $\mathbf{q}$ be a mode of the hybrid system HS and $f(\mathbf{q})$ be the corresponding vector field. The rule for adding new terms to P is the following: *If $p \in P$ is a term, then we add the term $L_{f(\mathbf{q})}(p)$ to P unless $L_{f(\mathbf{q})}(p) \in EMon_{Inv^*(\mathbf{q})}(P)$*. Thus, the process of constructing new terms to add to P involves computing the Lie derivative $L_{f(\mathbf{q})}(p)$ and testing if $L_{f(\mathbf{q})}(p) \in EMon_{Inv^*(\mathbf{q})}(P)$. Since the latter test could be expensive, we sometimes replace it by the following weaker tests

$$
\begin{aligned}
&Th \models Inv^*(\mathbf{q}) \quad \Rightarrow \quad L_{f(\mathbf{q})}(p) \sim 0 \quad \sim \in \{>, =, <\},\\
&Th \models Inv^*(\mathbf{q}) \quad \Rightarrow \quad L_{f(\mathbf{q})}(p) = cp',
\end{aligned}
$$

for some constant $c \in \mathbb{R}$ and $p' \in P$. If either of these proof obligations can be proved, then clearly $L_{f(\mathbf{q})}(p) \in EMon_{Inv^*(\mathbf{q})}(P)$ and hence $L_{f(\mathbf{q})}(p)$ is not added to P.

The functions added to P by the saturation process are useful when qualitatively abstracting the continuous dynamics of *HS*, as outlined in the previous section.

Note that for general vector fields $f(\mathbf{q})$ the saturation process might not terminate. But there are special cases where this process is guaranteed to terminate. For example, suppose a mode $\mathbf{q}$ has linear dynamics given by a nilpotent matrix A, that is, $f(\mathbf{q})(X) = AX$ such that $A^k = 0$. Let $p \in P$ be a polynomial with degree d. Then the $(d * k)$-th Lie derivative of p with respect to $f(\mathbf{q})$ will be identically zero. As a second example, consider a mode $\mathbf{q}$ that has linear dynamics given by a matrix A, which satisfies the equation $A^n = rA^m$ for some constant $r \in \mathbb{R}$ and $n, m \in \mathbb{N}$. If $p \in P$ is a *linear* polynomial, then the saturation process can be shown to terminate. In particular, if $p = \vec{a}^T X$ is a linear polynomial, then the n-th Lie derivative of p will r times the m-th Lie derivative. Since the n-th derivative of p is a constant multiple of the m-th derivative of p, it does not get added to the set P of polynomials in the saturation process.

We remark here that the termination of the saturation process is determined by *both* the initial set P of seed terms *and* the vector fields f (in all the different modes). However, the termination of the saturation phase is not necessary for creating an abstraction. We can stop at any point in the saturation process and compute the abstraction using the set P computed up to that point. A larger set P yields a finer abstraction as it results in a larger state space in the resulting abstract system.

Example 5 Consider the hybrid automaton *HS* from Examples 2 and 3. The guards of the discrete transitions give two terms $P = \{x_n - h_d, x_u - h_n\}$. Let us assume that all symbolic parameters, such as h_d, h_n, λ_d, are constrained to be positive constants by the invariant. Note that the parameters are unchanging, that is, for example, $\dot{x_u} = 0$ in all modes. Consequently, the Lie derivative of $x_u - h_n$ is identically zero in all four modes. Hence, nothing is added to the set P. Next, we compute the Lie derivative of $x_n - h_d$ in the four modes. We get two distinct values, $-\lambda_n x_n$ and $\Delta_n - \lambda_n x_n$. These get added to P. Next, we compute the Lie derivative of $-\lambda_n x_n$ with respect to the four distinct vector fields. We get two different terms again, but these are just $-\lambda_n$ times the old terms. Hence, nothing new is added to P. Similarly, when we compute the Lie derivative of $\Delta_n - \lambda_n x_n$, we get the same answers and do nothing. After saturation, finally $P = \{x_n - h_d, x_u - h_n, -\lambda_n x_n, \Delta_n - \lambda_n x_n\}$.

4.1.1 Heuristic for identifying important terms

We say that a set P_0 of (polynomial) functions is *closed under Lie derivative computation with respect to* $f(\mathbf{q})$ *and relative to* $Inv^*(\mathbf{q})$ if for every function $p_i \in P_0$, the Lie derivative of p_i w.r.t. $f(\mathbf{q})$ is in $EMon_{Inv^*(\mathbf{q})}(P_0)$. Clearly, if P_0 is a set which satisfies the above property, then inclusion of P_0 into P incurs no further additions to P due to saturation. Hence, an important heuristic for identifying new terms for inclusion into P is the following:

> *A set P_0 of terms is good if, for each mode $\mathbf{q}$ of the hybrid system, the set P_0 is closed under Lie derivative computation with respect to $f(\mathbf{q})$ and relative to $Inv^*(\mathbf{q})$.*

In Sects. 4.2 and 4.3, we will be guided by this basic heuristic rule to compute important sets P_0 and include them in P. Note that even if a set P_0 may not satisfy the above condition for every mode, it could still be useful if P_0 satisfies the condition for *some* modes.

4.2 Linear dynamics

Useful linear functions can be computed for inclusion in P if there are modes with linear dynamics. Let $\mathbf{q}$ be a mode of the hybrid system HS with vector field given as $f(\mathbf{q})(X) = AX$, where $A \in \mathbb{Q}^{n\times n}$. If λ is a real eigenvalue of A and $\vec{c} = [c_1, \ldots, c_n]^T$ is an eigenvector of A^T corresponding to λ, then we add the linear function $p = \vec{c}^T X$ to the set P.

The reason why this particular p is a useful term to use for abstraction becomes clear when we compute the Lie derivative of p with respect to $f(\mathbf{q})$,

$$L_{f(\mathbf{q})}(p) = \vec{c}^T \dot{X} = \vec{c}^T AX = (A^T\vec{c})^T X = (\lambda\vec{c})^T X = \lambda p.$$

This shows that the set $\{p\}$ is closed under Lie derivative computation in mode $\mathbf{q}$. In this specific case, $L_{f(\mathbf{q})}(p)$ vanishes on the surface $p = 0$ and hence flows will not cross this surface. If the system starts on one side of this surface, it will continue to remain in that half space. This information is preserved in the abstraction if p is included in the set P. Hence, $p = 0$ can potentially be a *barrier certificates* [39]. We also remark here that the linear forms p can be used to approximate reachability sets explicitly [49, 53, 54].

Each distinct real eigenvalue of A can be used to generate a suitable linear function p for inclusion in the set P of abstraction terms. If the eigenvalue is rational, then the computation and representation of the corresponding eigenvector is straightforward. If not, then we use polynomials to represent the coefficients of p.

Example 6 Consider a part of the leader control developed in [15] and also discussed in [42] for collision avoidance in automated cruise control in automobiles. The control is applied during safety-critical situations when the inter-vehicle distance is small, or the relative velocity between vehicles is large. Let gap, v_f, v, and a respectively represent the gap between the two cars, the velocity of the leading car, and the velocity and acceleration of the rear car. We are given,

$$\dot{v} = a, \qquad \dot{a} = -3a - 3(v - v_f) + (gap - (v + 10)), \qquad \dot{gap} = v_f - v.$$

Formally, this describes a linear dynamical system with $X = \{v, v_f, a, gap\}$. Assuming the variable v_f is a parameter (unchanging symbolic constant), the dynamics can be written as $\dot{X} = AX + B$, where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -4 & 3 & -3 & 1 \\ -1 & 1 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 \\ 0 \\ -10 \\ 0 \end{bmatrix}.$$

By a change of variables, $rgap \leftarrow gap - 10$, we get $\dot{X} = AX$, where $X = [v, v_f, a, rgap]^T$. We leave the set *Init* of initial states and the invariant sets unspecified. For a given set of possible initial states, the problem is to verify that the rear car would never collide with the car in front, that is, always $gap > 0$, or $rgap > -10$.

Now, the characteristic polynomial for A, $\lambda(\lambda^3 + 3\lambda^2 + 4\lambda + 1)$, has exactly two real zeros. The nonzero real eigenvalue, denoted by λ, lies between $-1/3$ and $-1/4$. Now, if $\vec{c} = [c_1, c_2, c_3, c_4]^T$ is an eigenvector of A^T corresponding to λ, then $A^T\vec{c} = \lambda\vec{c}$. We assume, without loss of generality, that $c_4 = 1$, and hence we get an eigenvector $[c_1, c_2, c_3, 1]^T$ where c_1, c_2 and c_3 satisfy the equations $c_3 = \lambda$, $c_1 = \lambda^2 + 3\lambda$, and $c_2 = -c_1 - 1$. Therefore, the linear term corresponding to this eigenvector is $p = c_1 v - (c_1 + 1)v_f + c_3 a + rgap$. We add

this term to the set P. In fact, for certain initial states, the surface $p = 0$ acts as a barrier between the initial states and the unsafe region and suffices to prove the safety problem (under certain assumptions on the invariant set). We refer the interested reader to [49] for further analysis of this example.

4.2.1 Complex eigenvalues

Let $y^2 + ay + b$ be a factor of the characteristic polynomial of A, where $a, b \in \mathbb{R}$ and $a \neq 0, 4b > a^2$. Let W denote the null space of the transformation $(A^T)^2 + aA^T + bI$, that is,

$$W = \{\vec{c} \in \mathbb{R}^n : ((A^T)^2 + aA^T + bI)\vec{c} = 0\}.$$

Since $A^T \in \mathbb{Q}^{n\times n}$, $a \in \mathbb{R}$, and $b \in \mathbb{R}$, it follows that W is nonempty.

Let $\vec{c} \in \mathbb{R}^n$ be a nonzero vector in W. Consider the linear function $p = \vec{c}^T X$ over the state variables corresponding to this vector. Let $\dot{p}$ denote $L_{f(\mathbf{q})}(p)$ and $\ddot{p}$ denote $L_{f(\mathbf{q})}(\dot{p})$. We have the following relation between $\ddot{p}$, $\dot{p}$, and p.

$$\begin{aligned}\ddot{p} + a\dot{p} + bp &= \vec{c}^T\ddot{X} + a\vec{c}^T\dot{X} + b\vec{c}^T X = \vec{c}^T A^2 X + a\vec{c}^T AX + b\vec{c}^T X \\ &= \vec{c}^T(A^2 + aA + bI)X = (((A^T)^2 + aA^T + bI)\vec{c})^T X = 0.\end{aligned}$$

We add p and $\dot{p}$ to the set P. Note that we can infer the sign of $\ddot{p}$ from that of p and $\dot{p}$ in many cases.

4.2.2 Computability issues

If eigenvalues are rational, then computation of the left eigenvector would just involve simple arithmetic manipulation over rationals. However, if we have to deal with real numbers and the left eigenvectors are composed of real numbers, then we require the ability to represent and reason with algebraic numbers. This requires theorem proving capability for a theory of reals defined over $\{+, -, *, =\}$.

4.3 Nonlinear dynamics

We present some heuristic approaches to find useful terms for inclusion in the set P when the dynamics in a given mode are nonlinear.

4.3.1 Linear barriers

We first discover important linear functions for inclusion into P. Let $\dot{X} = f(\mathbf{q})$ specify the dynamics in mode $\mathbf{q}$. Separate the nonlinear component from the linear component and rewrite the above equation as

$$\dot{X} = AX + BY,$$

where Y is the vector of non-linear functions of the state variables X, see also Example 7. Here A is an $n \times n$ matrix, B is an $n \times m$ matrix, X is a $n \times 1$ vector, and Y is a $m \times 1$ vector. Let $\vec{c}$ be a real eigenvector of A^T which is also in the *kernel* of B^T (that is, the linear subspace of zeros of B^T), that is,

$$A^T\vec{c} = \lambda\vec{c}, \qquad B^T\vec{c} = \vec{0},$$

where the components of $\vec{c}$ are reals. The transpose $\vec{c}^T$ of the vector $\vec{c}$ is a 1-form. Consider the linear function $p = \vec{c}^T X$.

$$L_{f(\mathbf{q})}(p) = \vec{c}^T \dot{X} = \vec{c}^T (AX + BY) = (A^T \vec{c})^T X + (B^T \vec{c})^T Y$$
$$= (\lambda \vec{c})^T X + \vec{0} = \lambda p.$$

This shows that the set $\{p\}$ is closed under Lie derivative computation and hence, it is useful to include p in the set P. We note here that the matrix A and B are unique up to permutations of their rows. It is easy to see that any of these choices for B will result in the same outcome.

Example 7 If x_1 and x_2 represent concentrations of two proteins that can bind together to form a dimer, then the law of mass action gives the following differential equations governing the dynamics of x_1 and x_2:

$$\dot{x_1} = \Delta_1 - \lambda x_1 - k x_1 x_2,$$
$$\dot{x_2} = \Delta_2 - \lambda x_2 - k x_1 x_2.$$

If we introduce a new variable x_3 to homogenize the above (intuitively x_3 is always 1), we get $A = [-\lambda, 0, \Delta_1; 0, -\lambda, \Delta_2; 0, 0, 0]$ is a (3×3)-matrix, $X = [x_1; x_2; x_3]$ is a column vector, $B = [-k; -k; 0]$ is a (3×1)-matrix, and $Y = [x_1 x_2]$ is a vector with only one element. The above equations can be written as $\dot{X} = AX + BY$. If we use the method outlined above and replace x_3 by 1, we get the linear function $p = -\lambda x_1 + \lambda x_2 + \Delta_1 - \Delta_2$. We immediately observe that $L_f(p) = -\lambda p$.

4.3.2 Nonlinear invariants

Assume that the dynamics in the mode $\mathbf{q}$ are given by $\dot{X} = f(\mathbf{q})$, where $f(\mathbf{q}) \in \mathbb{Q}[X]^n$ is a *polynomial* vector field. A syzygy of the vector field $f(\mathbf{q})$ is a 1-form $\vec{h}^T$ such that $\vec{h}^T f(\mathbf{q}) = 0$. A 1-form $\vec{h}^T$ is *exact* if there exists a smooth function (polynomial, in our case) p such that $\vec{h}^T = \vec{d}p$. Suppose there is a syzygy $\vec{q}^T$ of the vector field $f(\mathbf{q})$ which is also exact. In other words, there is a polynomial p such that

$$\partial p/\partial x_1 = q_1, \qquad \partial p/\partial x_2 = q_2, \qquad \ldots, \qquad \partial p/\partial x_n = q_n \quad \text{and}$$
$$q_1 * (dx_1/dt) + q_2 * (dx_2/dt) + \cdots + q_n * (dx_n/dt) = 0.$$

Under these assumptions, it is easy to note that the Lie derivative of p with respect to the vector field $f(\mathbf{q})$ vanishes, that is, $dp/dt = \vec{d}p\dot{X} = \vec{d}pf(\mathbf{q}) = 0$. The set $\{p\}$ is closed under Lie derivative computation. In fact, in this case the value of the expression $p(x_1, x_2, \ldots, x_n)$ remains invariant through the time evolution of the nonlinear system and it is fruitful to include p in the set P.

Example 8 Consider the nonlinear dynamical system:

$$\dot{x_1} = x_1 x_2, \qquad \dot{x_2} = -x_1.$$

It is the case that $1x_1x_2 + x_2(-x_1) = 0$ and hence $(1, x_2)$ is a syzygy of the polynomials $x_1x_2, -x_1$. A solution for p that satisfies both $\partial p/\partial x_1 = 1$ and $\partial p/\partial x_2 = x_2$, is $V = x_1 + x_2^2/2$. It is easily observed that $\dot{p} = 0$ and hence p is an invariant of the above dynamical system.

4.3.3 Nonlinear barriers

Assume again that the dynamics in the mode $\mathbf{q}$ are given by $\dot{X} = f(\mathbf{q})$, where $f(\mathbf{q}) \in \mathbb{Q}[X]^n$ is a polynomial vector field. Suppose that there exists a polynomial r such that $r = \vec{q}^T f(\mathbf{q})$, the 1-form $\vec{q}^T$ is exact (that is, $\mathbf{d}p = \vec{q}^T$ for some $p \in \mathbb{Q}[X]$), and r is nonnegative or nonpositive definite relative to $Inv^*(\mathbf{q}) \wedge p \sim 0$, where $\sim \in \{>, <\}$. In other words, we have

$$r = \vec{q}^T f(\mathbf{q}), \qquad \vec{q}^T = \mathbf{d}p,$$
$$Th \models Inv^*(\mathbf{q}) \wedge p \sim 0 \quad \Rightarrow \quad r \sim' 0, \quad \sim' \in \{\geq, \leq\}.$$

In this case, we can add p to the set P, since $L_{f(\mathbf{q})}(p) = r$ and we can infer the sign of r given the sign of p. Note that r is in the ideal generated by the polynomials in $f(\mathbf{q})$.

4.3.4 Computability issues

The computation of nonlinear invariants and barriers, as described above, requires the ability to (a) compute a syzygy basis for a finite set of polynomials, (b) check for exactness of a given 1-form, and integrate an exact 1-form, (c) enumerate elements in the ideal generated by a finite set polynomials, and (d) test a polynomial for being relatively nonnegative or nonpositive definite. Questions pertaining to ideal membership can be solved using Gröbner basis computation and syzygy basis can also be computed using standard algorithms from computational algebraic geometry. Exactness of a 1-form $\vec{h}^T$ can be tested by checking if $\partial h_i/\partial x_j = \partial h_j/\partial x_i$, for all i, j [52]. Exact polynomial 1-forms can be integrated symbolically. The test for nonnegative or nonpositive definiteness requires theorem proving capability.

Unlike the case of linear dynamics, the discovery of invariants and barriers for nonlinear dynamics involves a search. We have to enumerate syzygies or ideal members, and test if they satisfy the other constraints. It is left for future work to determine if the sets of nonlinear invariants and barriers are (algorithmically) *computable*.

Example 9 Consider the nonlinear system

$$\dot{x_1} = x_1 - x_2 + x_1 x_2, \qquad \dot{x_2} = -x_2 - x_2^2.$$

The nonnegative definite polynomial x_2^2 is in the ideal generated by $x_1 - x_2 + x_1 x_2$ and $-x_2 - x_2^2$ and we correspondingly have $x_2^2 = -x_2(x_1 - x_2 + x_1 x_2) - x_1(-x_2 - x_2^2)$. The 1-form $[-x_2, -x_1]$ is exact since $\partial(-x_2)/\partial x_2 = \partial(-x_1)/\partial x_1 = -1$. Symbolically integrating this 1-form, we get the polynomial $-x_1 x_2$. In all, we conclude that $d/dt(-x_1 x_2) = x_2^2$. Since $x_2^2 \geq 0$ is always true, we can infer that the value of $-x_1 x_2$ is nondecreasing.

4.4 Barrier certificates

Consider a continuous dynamical system with dynamics given by $\dot{X} = f(X)$ and initial states *Init*. Suppose that we are also given an unsafe region $\mathbf{X}_u$. A function p such that (a) $L_{f(\mathbf{q})}(p) \leq 0$ whenever $p = 0$, (b) $p(\mathbf{x}) > 0$ whenever $\mathbf{x} \in \mathbf{X}_u$, and (c) $p(\mathbf{x}) \leq 0$ whenever $\mathbf{x} \in Init$, is called a *barrier certificate* [39, 40]. An existence of a barrier certificate demonstrates that the unsafe region is not reachable from the initial states. Several of the functions proposed by us to be included in the set P satisfy conditions similar to condition (a). For example, the linear functions defined by the left eigenvectors for linear dynamics satisfy

condition (a). Similarly, some of the functions proposed above for nonlinear dynamics also satisfy condition (a). These functions can become barrier certificates *if* the initial and the unsafe regions additionally satisfy conditions (b) and (c). Techniques based on convex optimization have been proposed for the computation of barrier certificates [40]. Any functions computed in this way can also be added to the set P.

Barrier certificates are good choices for inclusion into the set P since they satisfy our general characterization of the set of good predicates. Alternatively, our general characterization can be seen as a generalization of the notion of barrier certificates to sets of functions, which if used together, can yield useful reachability information about the hybrid system.

Example 10 The Volterra predator-prey model [52] is given by

$$\dot{x_1} = -x_1 + x_1x_2, \qquad \dot{x_2} = x_2 - x_1x_2,$$

where x_1 indicates the number of predators and x_2 indicates the number of prey. It is an easy exercise to note that the set of four polynomials $P_0 = \{x_1, x_2, (x_2 - 1), (x_1 - 1)\}$ is closed under Lie derivative computation. The qualitative abstraction of this model over the four polynomials in P_0 shows the possible oscillatory behavior of the system. Another choice of a closed set is $\{x_1 + x_2, x_2 - x_1, x_1 + x_2 - 2x_1x_2, 1 - 2x_1x_2\}$ and this can be used to refine the above abstraction.

5 Refining the abstraction and other optimizations

The process of abstracting the continuous evolution of the hybrid system *HS* is done using qualitative rules in our approach as outlined above. As a consequence of this, the abstract transitions are obtained as updates to the abstract variables, each one independent of the other, as in Cartesian abstraction [7]. This means that the concretization of a new abstract state reached by taking an abstract transition could be infeasible. Furthermore, certain abstract states can also be deleted because they are explicitly disallowed by the given invariant set *Inv* of the concrete system.

More specifically, any transition to the abstract state $(\mathbf{q}, \mathbf{q_P})$ can be deleted if

$$Th \not\models \exists X : \gamma^*(\mathbf{q_P}) \wedge Inv^*(\mathbf{q})(X).$$

Note that this process also removes infeasible abstract states, that is, states $(\mathbf{q}, \mathbf{q_P})$ such that $Th \not\models \exists X : \gamma^*(\mathbf{q_P})$. If the infeasible abstract states were not removed, then we could have cases where we (spuriously) reach a feasible abstract state via an infeasible abstract state. Thus, removing the infeasible abstract states improves the quality of abstraction by eliminating such spurious trajectories.

This refinement step is implemented by first computing the set $Feas = \{\mathbf{q_P} \in \mathbf{Q_P} : Th \models \exists X : \gamma^*(\mathbf{q_P})\}$ during the process of constructing the abstraction. We modify each abstract transition by adding an extra condition to check if the destination state is feasible and satisfies the invariant of the destination mode. For example, if the original abstract transition was

$$\mathbf{q}, \psi_1(Q_P) \longrightarrow \mathbf{q}', \quad Q_P := F_Q(Q_P)$$

then we replace it by the new abstract transition

$$\mathbf{q}, \psi_1(Q_P) \longrightarrow \mathbf{q}', \quad Q_P := F_Q(Q_P), \quad Q_P \in Feas \cap \alpha(Inv(\mathbf{q}')),$$

where Q_P refer to the *new* values of these variables in the newly added condition. The transition is allowed only if the additional condition holds.

5.1 Reducing the size of the abstract state space

The size of the abstract state space, $O(|\mathbf{Q}| \times 3^{|P|})$, grows exponentially with the number of terms in the set P. This size can be reduced if not all modes are abstracted using the same set P of terms. We can choose a different subset of terms for different modes of the hybrid system.

Let $P_{\mathbf{q}}$ be a set of terms indexed by the modes $\mathbf{q} \in \mathbf{Q}$. When abstracting a concrete predicate $\phi(X)$ in mode $\mathbf{q}$, we will only use the terms in $P_{\mathbf{q}}$, so that (5) is replaced by,

$$\alpha^*_{P_{\mathbf{q}}}(\phi(X)) = \{\mathbf{q}' \in \mathbf{Q}_{\mathbf{P_q}} : Th \models \exists X : \gamma^*(\mathbf{q}') \wedge \phi(X)\}. \tag{9}$$

The initial states can be abstracted using this modified equation now.

A discrete transition $\mathbf{q}, \phi(X) \longrightarrow \mathbf{q}', X := F(X)$ is now abstracted by the following transitions, defined for each $\mathbf{q_P} \in \alpha^*_{P_{\mathbf{q}}}(\phi(X))$,

$$\mathbf{q}, \mathbf{q_P} \longrightarrow \mathbf{q}', (q_p := ITE1(\mathbf{q_P}, Q_{p1}, Q_{p2}, Q_{p3}))_{p \in P_{\mathbf{q}'}},$$

where Q_{p1}, Q_{p2}, Q_{p3} are now subsets of $2^{\mathbf{Q_{P_q}}}$. The only change in the abstraction of the continuous evolution is that in mode $\mathbf{q} \in \mathbf{Q}$ we only make changes to the abstract variables q_p, where $p \in P_{\mathbf{q}}$. The size of the state space of the resulting abstract system is now $O(\Pi_{\mathbf{q} \in \mathbf{Q}} |P_{\mathbf{q}}|)$.

The process for obtaining terms to include in P, as outlined in Sect. 4, works on discrete modes individually. If the term p is generated when working in mode $\mathbf{q}$, then p is added to $P_{\mathbf{q}}$. For example, in the saturation process that computes higher-order Lie derivatives, if $p' = L_{f(\mathbf{q})}(p)$ for $p \in P_{\mathbf{q}}$, then p' is added to $P_{\mathbf{q}}$ and it is not added to any of the other sets $P_{\mathbf{q}'}, \mathbf{q}' \neq \mathbf{q}$.

6 Computability and theorem proving obligations

We have described the abstraction procedure assuming that we have a procedure to handle the theorem proving obligations that are generated. In fact, we have left the choice of the theory *Th* open until now. In this section, we will briefly discuss the issues related to automating the theorem proving support required for constructing abstractions of hybrid systems as described in this paper.

We recapitulate the requirements on the theorem proving capability required over the theory *Th*. We need the ability to decide satisfiability of quantifier-free formulas (that is, implicitly existentially quantified formulas) over *Th*. First, this is required to abstract initial states and guards of discrete transitions, as given by (5). Second, the proof obligations arising in the process of abstracting the updates on discrete transitions and abstracting the continuous dynamics are of the form $Th \models \psi \Rightarrow p \sim 0$, where $\sim \in \{\geq, \neq, \leq\}$. These are equivalent to testing $Th \models \exists X : (\psi \wedge p \sim' 0)$, where $\sim'$ is respectively in $\{<, =, >\}$. Finally, we note that we need to decide satisfiability of quantifier-free formulas to eliminate the infeasible states from the abstract system. The signature of *Th* is required to contain the symbols $\{-, >, =\}$ and any other symbol required either to specify the input hybrid system or to express the Lie derivative of terms in *Th* with respect to the system dynamics.

The time complexity of the basic abstraction procedure of Sect. 3, ignoring the phase of generating the set P, is $O(|\mathbf{Q}||P|3^{|P|}T_{Th}(N) + |\mathbf{Q}|^2 3^{|P|}T_{Th}(N))$, where $|S|$ denotes the cardinality of the set S, T_{Th} is the time-complexity of the satisfiability procedure for theory Th, and N is the size of the input hybrid system HS. The first term results from the phase of abstracting the continuous dynamics (Sect. 3.2.2) and the second term is contributed by the discrete transitions (Sect. 3.2.1). In our implementation, we attempt to overcome the two most expensive factors, $3^{|P|}$ and T_{Th}, in the complexity.

In the process of constructing an abstraction, we do not explicitly enumerate the $3^{|P|}$ states in $\mathbf{Q_P}$. In the case of abstracting the initial states, we do a depth-first enumeration of these exponentially many states and cache witnesses of disproofs to avoid repeated theorem proving effort. In the case of abstracting the dynamics, we use several heuristics, such as slicing, to select a small subset $P_1 \subseteq P$, and enumerate only over the 3^{P_1} states. These approximations do not compromise the soundness of the abstraction, though they can potentially affect its quality.

Special classes of hybrid systems, such as timed automata, can be specified using the signature $\{\mathbb{Q}, +, -, >, =\}$, which excludes the multiplication operator. The theory of reals over this signature, often called the theory of *linear* arithmetic over reals, satisfies all our constraints in this case and it can be used for abstracting such systems. Satisfiability of a conjunction of atomic formulas is efficiently decidable for this theory.

If the theory Th is the theory of reals defined over the signature $\{\mathbb{Q}, +, -, *, >, =\}$, then we can use polynomial expressions to specify the dynamics of the hybrid system. This theory is called the theory of the real closed fields and it can be used to abstract such polynomial systems. This theory is known to be decidable [11, 47]. In particular, the satisfiability problem is decidable. However, the decision procedure is computationally expensive.

One can also use richer signatures by including the trigonometric functions or the exponential function in the signature. The theory of reals over such richer signatures loses some of its nice decidability results.

The abstraction procedures uses theorem proving in a "failure-tolerant" mode, that is, the correctness of the procedure is preserved even when the theorem prover ceases to be complete as long as it is sound. By soundness, we mean that whenever the prover says a formula is unsatisfiable, then it really should be unsatisfiable. Completeness requires that if the prover says satisfiable, then the formula should indeed be satisfiable. The proof of Theorem 1 only requires that the theorem prover be sound. The incompleteness in the theorem prover will only result in a coarser abstract system. Thus, we can use efficient, sound, but incomplete, procedures to test satisfiability of quantifier-free formulas in the theory Th for constructing abstractions. This is especially useful if it is computationally expensive, or impossible, to obtain sound and complete decision procedures.

7 Compositional abstraction

Let $HS_1 = (Q_1 \cup Q_1^{in}, X_1 \cup X_1^{in}, Init_1, Inv_1, t_1, f_1)$ and $HS_2 = (Q_2 \cup Q_2^{in}, X_2 \cup X_2^{in}, Init_2, Inv_2, t_2, f_2)$ be a pair of hybrid automata, where, for $i = 1, 2$, Q_i^{in} is a finite set of input discrete variables, X_i^{in} is a finite set of input real variables, $Init_i \subseteq \mathbf{Q_i} \times \mathbf{X_i}$, $Inv_i : \mathbf{Q_i} \times \mathbf{Q_i^{in}} \mapsto 2^{(\mathbf{X_i} \times \mathbf{X_i^{in}})}$, $t_i : \mathbf{Q_i} \times \mathbf{Q_i^{in}} \times \mathbf{X_i} \times \mathbf{X_i^{in}} \times \mathbf{Q_i} \times \mathbf{X_i}$, and $f_i : \mathbf{Q_i} \times \mathbf{Q_i^{in}} \mapsto (\mathbf{X_i} \times \mathbf{X_i^{in}} \mapsto T\mathbf{X_i})$. Let $\sigma_1 : (Q_1^{in} \cup X_1^{in}) \mapsto (Q_2 \cup X_2)$ and $\sigma_2 : (Q_2^{in} \cup X_2^{in}) \mapsto (Q_1 \cup X_1)$ be two renaming functions that map the input variables to other state variables.[1] The result of composing HS_1 and HS_2

[1] For simplicity, we assume here that the composed hybrid automaton HS is autonomous, that is, it has no inputs and all inputs of HS_1 and HS_2 are closed by suitably renaming them to other state variables.

(with respect to σ_1 and σ_2) is the hybrid automaton $HS = HS_1 \times HS_2 = (Q, X, Init, Inv, t, f)$, where $Q = Q_1 \cup Q_2$ is the set of discrete variables so that the discrete modes of HS is given by $\mathbf{Q} = \mathbf{Q_1} \times \mathbf{Q_2}$ and $X = X_1 \cup X_2$ is the set of real variables and the continuous state-space of HS is $\mathbf{X} = \mathbf{X_1} \times \mathbf{X_2}$. If $(\mathbf{q_1}, \mathbf{q_1^{in}}, \mathbf{x_1}, \mathbf{x_1^{in}}, \mathbf{q_1'}, \mathbf{x_1'}) \in t_1$ in HS_1, then $(\mathbf{q_1}, \mathbf{q_2}, \mathbf{x_1}, \mathbf{x_2}, \mathbf{q_1'}, \mathbf{q_2}, \mathbf{x_1'}, \mathbf{x_2}) \in t$ in HS *if* $\mathbf{q_1^{in}}$ matches $\mathbf{q_2}$ on the variables made identical by σ_1 and $\mathbf{x_1^{in}}$ matches $\mathbf{x_2}$ on the variables made identical by σ_1. Similarly, a discrete transition in t_2 induces a discrete transition in HS. Note that the hybrid automaton HS can make a discrete transition exactly when one of its components can make a discrete transition. If $\dot{X}_1 = f_1(\mathbf{q_1})(X_1, X_1^{in})$ is the continuous flow equation in the discrete mode $\mathbf{q_1}$ of the automaton HS_1, and $\dot{X}_2 = f_2(\mathbf{q_2})(X_2, X_2^{in})$ is the continuous flow equation in the discrete mode $\mathbf{q_2}$ of the automaton HS_2, then in the discrete state $(\mathbf{q_1}, \mathbf{q_2})$ of HS, there is a continuous flow given by the equations $\dot{X}_1 = f_1(\mathbf{q_1})(X_1, \sigma_1(X_1^{in}))$ and $\dot{X}_2 = f_2(\mathbf{q_2})(X_2, \sigma_2(X_2^{in}))$. Note that the time evolutions of the component hybrid automata happen simultaneously.

Example 11 The Delta-Notch lateral inhibition model in Example 2 shows interesting behavior when there is more than one cell. A model of two such cells is obtained as a composition of two hybrid automata, one for each cell. For $i = 1, 2$, the hybrid automaton $HS_i = (Q_i \cup Q_i^{in}, X_i \cup X_i^{in}, Init_i, Inv_i, t_i, f_i)$ is obtained from the automaton HS described in Example 2 by (a) appending index i in the subscript of names of all variables, (b) setting $Q_i^{in} = \emptyset$, and (c) setting $X_1^{in} = \{x_{u1}\}$ and $X_2^{in} = \{x_{u2}\}$. Define the renaming functions σ_1 and σ_2 so that $\sigma_1(x_{u1}) = x_{d2}$ and $\sigma_2(x_{u2}) = x_{d1}$. The two-cell Delta-Notch lateral signaling model is now obtained by composing HS_1 and HS_2 with respect to σ_1 and σ_2.

7.1 Abstraction

Let $HS_1 = (Q_1 \cup Q_1^{in}, X_1 \cup X_1^{in}, Init_1, Inv_1, t_1, f_1)$ and $HS_2 = (Q_2 \cup Q_2^{in}, X_2 \cup X_2^{in}, Init_2, Inv_2, t_2, f_2)$ be a pair of hybrid automata. We wish to *compositionally* abstract the hybrid automaton $HS = HS_1 \times HS_2$ obtained by composing HS_1 and HS_2 under given renaming functions σ_1, σ_2. However, this is only possible under some strong assumptions on the level of interaction between HS_1 and HS_2.

Assume that for $i = 1, 2$, it is the case that the hybrid automaton $HS_i = (Q_i \cup Q_i^{in}, X_i \cup X_i^{in}, Init_i, Inv_i, t_i, f_i)$ satisfies the following two conditions:

(A1) The vector field $f_i(\mathbf{q})$ does not depend on the input variables X_i^{in}.
(A2) There is no term containing variables from both X_i and X_i^{in} that occurs in the guard of discrete transitions and invariants.

As a consequence of Condition (A2), we can assume that all atomic formulas in a guard or invariant in HS_i are of the form $p \sim 0$, where $p \in \mathcal{T}(X_i)$ or $p \in \mathcal{T}(X_i^{in})$. If either of these two conditions is violated by either HS_1 or HS_2, then the compositional abstraction algorithm fails. This algorithm is described as follows:

1. For $i = 1, 2$ let

$$P_i^{in} := \{p \in \mathcal{T}(X_i^{in}) : p \sim 0 \text{ occurs in } HS_i\},$$
$$P_i := \{p \in \mathcal{T}(X_i) : p \sim 0 \text{ occurs in } HS_i\}.$$

2. Let $P_1 := P_1 \cup P_2^{in}$ and $P_2 := P_2 \cup P_1^{in}$. Obtain the final sets P_i of terms, over X_i, using saturation and other methods, for use in abstracting HS_i.

3. Abstract HS_i using the terms in P_i to get a discrete transition system DS_i under the assumption that the input variables are unchanged during both continuous evolutions and discrete transitions. Note that $DS_i = (Q_i \cup Q_i^{in} \cup Q_{P_i} \cup Q_{P_i^{in}}, Init_i^A, t_i^A)$, where $Q_{P_i^{in}}$ and Q_i^{in} are *input* variables.
4. Construct $DS = (Q_1 \cup Q_2 \cup Q_{P_1} \cup Q_{P_2}, Init_1^A \times Init_2^A, t^A)$, where t^A contains (a) all transitions of t_1^A and t_2^A that correspond to abstractions of discrete transitions of either HS_1 or HS_2, and (b) the cross-product of all the transitions that are abstractions of the continuous dynamics. Return DS.

Note that Assumption (A1) guarantees that the continuous evolutions of HS_1 and HS_2 are completely independent of each other. Hence, to track the evolution of a pure term p over X_i along a flow, we only need to consider the continuous dynamics of HS_i. However, the discrete transitions of HS_1 (HS_2) depend on variables that are set by HS_2 (HS_1) via the terms p in guards and reset functions. Hence, HS_2 (HS_1) needs to "track" such terms and this is the reason for Step 2 above.

Theorem 2 *Let $HS_i = (Q_i \cup Q_i^{in}, X_i \cup X_i^{in}, Init_i, Inv_i, t_i, f_i)$ be two hybrid automata and $\rho_1 : Q_1^{in} \cup X_1^{in} \mapsto Q_2 \cup X_2$ and $\rho_2 : Q_2^{in} \cup X_2^{in} \mapsto Q_1 \cup X_1$ be two renamings and HS be the result of composing HS_1 and HS_2 with respect to the two renamings. Let P_1, P_2, and $DS = (Q^A = Q_1 \cup Q_2 \cup Q_{P_1} \cup Q_{P_2}, Init^A, t^A)$ be as computed by the procedure outlined above. Then, DS is an abstraction of HS.*

Proof If $((\mathbf{q_1}, \mathbf{q_2}), (\mathbf{x_1}, \mathbf{x_2})) \in \mathbf{Q_1} \times \mathbf{Q_2} \times \mathbf{X_1} \times \mathbf{X_2}$ is a concrete state of the hybrid system HS, then the abstraction mapping α is

$$\alpha(((\mathbf{q_1}, \mathbf{q_2}), (\mathbf{x_1}, \mathbf{x_2}))) = ((\mathbf{q_1}, \mathbf{q_2}), (\alpha_{P_1}(\mathbf{x_1}), \alpha_{P_2}(\mathbf{x_2}))),$$

where α_{P_i} are defined by (2). Since $Init_i^A$ is a correct abstraction of $Init_i$, it follows that $Init_1^A \times Init_2^A$ is a correct abstraction of $Init_1 \times Init_2$.

Suppose that $(((\mathbf{q_1}, \mathbf{q_2}), (\mathbf{x_1}, \mathbf{x_2})), ((\mathbf{q_1'}, \mathbf{q_2'}), (\mathbf{x_1'}, \mathbf{x_2'}))) \in t_\alpha$ is a transition in the discrete system HS_α corresponding to HS with respect to the abstraction mapping α. Let this transition be a result of continuous evolution (Definition 2(b)). Since the continuous dynamics of HS_1 (HS_2) are independent of the state variables of HS_2 (HS_1), we know that there is a transition $(\mathbf{q_i}, \alpha(\mathbf{x_i}), \mathbf{q_i}, \alpha(\mathbf{x_i'})) \in t_i^A$ and hence, by definition of DS, there is the required transition in DS.

Next suppose that $(((\mathbf{q_1}, \mathbf{q_2}), (\mathbf{x_1}, \mathbf{x_2})), ((\mathbf{q_1'}, \mathbf{q_2'}), (\mathbf{x_1'}, \mathbf{x_2'}))) \in t_\alpha$ is due to a discrete transition (Definition 2(a)) taken by, say, HS_1. Now, the guard and the updates of this discrete transition of HS_1 may depend on state variables of HS_2. But the state variables of HS_2 do not change in this transition, and hence $\mathbf{q_2'} = \mathbf{q_2}$ and $\mathbf{x_2'} = \mathbf{x_2}$. By correctness of the abstraction on HS_1, it follows that $((\mathbf{q_1}, \alpha(\mathbf{x_1})), (\mathbf{q_1'}, \alpha(\mathbf{x_1'}))) \in t_1^A$. By definition of DS, it follows that $((\mathbf{q_1}, \mathbf{q_2}, \alpha(\mathbf{x_1}), \alpha(\mathbf{x_2})), (\mathbf{q_1'}, \mathbf{q_2}, \alpha(\mathbf{x_1'}), \alpha(\mathbf{x_2}))) \in t^A$. This completes the proof. □

Example 12 Consider the hybrid automaton HS obtained by composing HS_1 and HS_2 from Example 11. In Step 1 of the procedure, we compute $P_1^{in} = \{x_{d2} - h_n\}$ and $P_1 = \{x_{n1} - h_d\}$. Similarly we get the values of P_2^{in} and P_2. Now, in Step 2, P_1 is set to $\{x_{n1} - h_d, x_{d1} - h_n\}$ and after saturation (also see Example 5), we get $P_i = \{x_{ni}, x_{ni} - h_d, x_{ni} - \Delta_n/\lambda_n, x_{di}, x_{di} - h_n, x_{di} - \Delta_d/\lambda_d\}$. Note that we have sanitized these saturated sets by multiplying them with appropriate positive or negative values. Label the elements of P_i as $p_{i1}, \ldots, p_{i6}$, in that

order. In Step 3, we compute the abstractions DS_1 and DS_2. DS_1 has six state variables, $q_{p_{1i}}, i = 1, \ldots, 6$ and one input variable $q_{p_{25}}$.

$$\begin{aligned}(q_{p_{1i}} :\in \text{ if } (p_{25} = neg) \text{ then } & VN(q_{p_{11}}, \ldots, q_{p_{16}}, 1i, 11),\\ & \text{else } VN(q_{p_{11}}, \ldots, q_{p_{16}}, 1i, 13))_{i=1,2,3},\\ (q_{p_{1i}} :\in \text{ if } (p_{12} = pos) \text{ then } & VN(q_{p_{11}}, \ldots, q_{p_{16}}, 1i, 14),\\ & \text{else } VN(q_{p_{11}}, \ldots, q_{p_{16}}, 1i, 16))_{i=4,5,6}.\end{aligned}$$

Similarly, DS_2 has six local variables and one input variable $q_{p_{15}}$ and it can be represented as above. Finally, the abstract system DS is obtained by putting DS_1 and DS_2 together as in Step (4).

8 Relative completeness

Our approach for abstracting hybrid systems uses a combination of predicate abstraction and qualitative reasoning techniques. The qualitative approach for abstracting continuous dynamics appears to be very weak. But there are indications that this approach is not far fetched, and it can give good abstractions even when applied to purely continuous dynamical systems. Tabuada [46] has recently shown that the sign abstraction based on qualitative reasoning gives a system which is bisimilar to the original system in certain cases. In the following theorem, we establish a relative completeness result for our qualitative approach.

Consider a simple system with one state variable, $\dot{x} = -x$, undergoing exponential decay. Let $x = 1$ initially. Clearly, the set of all reachable states of the system is exactly $x > 0 \wedge x \leq 1$. Thus, the terms x and $x - 1$ are sufficient to precisely describe the reach set of this example. If we abstract this system using $P = \{x, x - 1\}$ and compute the reachable set on the abstract system and concretize it, we get $x \geq 0 \wedge x - 1 \leq 0$ as the reach set, which is the *closure* of the set described by $0 < x \leq 1$. The following theorem formally states this observation for general continuous dynamical systems.

We make standard assumptions on the vector field so as to guarantee the existence of solutions to the differential equations. We also assume that we use a sound and complete theorem prover to discharge the proof obligations in constructing the abstraction for purposes of the following theorem.

Theorem 3 *Let $CS = (X, Init, f)$ be a continuous dynamical system. Let $\phi(X)$ be a quantifier-free formula in the theory Th that represents the set of all reachable states of CS, that is, $[[\phi]] = Reach(CS)$. Let $P_0 = \{p \in \mathcal{T}(X) : p \sim 0$ occurs in $\phi\}$ and assume that the saturation process of P_0 under higher-order Lie derivative computation terminates in the set P. Let $DS = (Q_P, Init^A, t^A)$ be the abstraction of CS with respect to the set P. If ψ is the reachable set of DS, then $[[\phi]] \subseteq \gamma(\psi) \subseteq [[\phi]]^c$, where $[[\phi]]^c$ denotes the closure of $[[\phi]]$.*

Proof By correctness of the abstraction procedure, it follows that $[[\phi]] \subseteq \gamma(\psi)$. We next show that $\gamma(\psi) \subseteq [[\phi]]^c$.

For $\mathbf{q} \in \mathbf{Q_P}$, we will call the set $\gamma(\mathbf{q})$, whenever it is nonempty, a *region*. We note that $[[\phi]]$ is just a finite union of such regions. Consequently, if any point in a region is reachable, then all points in that region are necessarily reachable. If $\mathbf{q} = ((q_p = pos)_{p \in P_1}, (q_p = zero)_{p \in P_2}, (q_p = neg)_{p \in P_3})$, then we specify the region $\gamma(\mathbf{q})$ by the triple (P_1, P_2, P_3). We observe that *the region* (P_1, P_2, P_3) *is on the boundary of the region* (P_1', P_2', P_3'), *if* $P_1 \subseteq P_1'$, $P_3 \subseteq P_3'$, *and* $P_2' \subset P_2$. This fact is used in the proof below.

Since $Init \subseteq [[\phi]]$, and all predicates in ϕ are included in the set of abstraction predicates, it follows that all the regions contained in $\gamma(Init^A)$ are also contained in $[[\phi]]$, and hence $\gamma(Init^A) \subseteq [[\phi]]$. Now, if $\gamma(\psi) \not\subseteq [[\phi]]^c$, then there are two abstract *feasible* states $\mathbf{q}$ and $\mathbf{q}'$ (and two corresponding regions (P_1, P_2, P_3) and (P_1', P_2', P_3')), such that $(\mathbf{q}, \mathbf{q}') \in t^A$, and the region $\gamma(\mathbf{q}')$ is not, while the region $\gamma(\mathbf{q})$ is, contained in the closure $[[\phi]]^c$ of the reachable set $[[\phi]]$.

First consider the case when $\gamma(\mathbf{q}) \subseteq [[\phi]]$, that is, all points in the region $\gamma(\mathbf{q})$ are reachable in CS. Let $\mathbf{x} \in \gamma(\mathbf{q})$ be any point in this region. Since we have assumed the existence of a solution of the differential equation on $\mathbb{R}^n$, we have that starting from point $\mathbf{x}$ the system reaches a new point $\mathbf{x}''$ in a suitably *small* time step such that for all $p \in P_1$, $p(\mathbf{x}'') > 0$; for all $p \in P_3$, $p(\mathbf{x}'') < 0$; and for all $p \in P_2$, $p(\mathbf{x}'')$ is either positive, negative, or zero, depending on the sign of $L_f(p)$ at point $\mathbf{x}$. In other words, if $\mathbf{x}''$ belongs to the region (P_1'', P_2'', P_3''), then $P_1'' = P_1 \cup P_2^+$, $P_2'' = P_2 - P_2^+ - P_2^-$, and $P_3'' = P_3 \cup P_2^-$, where $P_2^+ = \{p \in P_2 : L_f(p)(\mathbf{x}) > 0\}$ and $P_2^- = \{p \in P_2 : L_f(p)(\mathbf{x}) < 0\}$. Since $\mathbf{x}''$ is reachable, the full region (P_1'', P_2'', P_3'') is contained in $[[\phi]]$ and is reachable. Using the qualitative abstraction rules (a)–(c) of Sect. 3.2.2 and the fact that for all $p \in P$, the sign of $L_f(p)$ can be uniquely inferred in a given region (thanks to the closure of P under Lie derivative computation), we conclude that $P_2'' \subseteq P_2'$, $P_1' \subseteq P_1''$, and $P_3' \subseteq P_3''$. If $P_2'' = P_2'$, then $P_1'' = P_1'$ and $P_3'' = P_3'$, and the new reachable region (P_1'', P_2'', P_3'') is identically equal to the region $\gamma(\mathbf{q}')$ thus showing that $\gamma(\mathbf{q}') \subseteq [[\phi]]$, a contradiction. If $P_2'' \subset P_2'$, then using the above observation we infer that the region $\gamma(\mathbf{q}')$ is on the boundary of the region (P_1'', P_2'', P_3''). This shows that $\gamma(\mathbf{q}') \subseteq [[\phi]]^c$, leading to a contradiction again.

Next consider the case when $\gamma(\mathbf{q})$ is part of $[[\phi]]^c - [[\phi]]$, that is, the set $\gamma(\mathbf{q})$ is on the boundary of the set of reachable states of CS. As in the previous case, we start with a point $\mathbf{x} \in \gamma(\mathbf{q})$, and construct a new point $\mathbf{x}''$ reachable from $\mathbf{x}$ in a small time step such that $\gamma(\mathbf{q}') \subseteq \gamma(\alpha(\mathbf{x}''))^c$. But now, since $\mathbf{x}$ is not a reachable state, $\mathbf{x}''$ need not be reachable. However, by assumption, there are reachable points in the neighborhood of $\mathbf{x}$. We pick a point $\mathbf{x_0}$ sufficiently close to $\mathbf{x}$ so that $\mathbf{x_0}$ reaches a point $\mathbf{x_0''}$ that is sufficiently close to $\mathbf{x}''$. Since $\mathbf{x} \in \gamma(\alpha(\mathbf{x_0}))^c$, it follows that $\mathbf{x}'' \in \gamma(\alpha(\mathbf{x_0''}))^c$. Combining this fact with the fact that $\gamma(\mathbf{q}') \subseteq \gamma(\alpha(\mathbf{x}''))^c$, we get $\gamma(\mathbf{q}') \subseteq \gamma(\alpha(\mathbf{x_0''}))^c$. But the point $\mathbf{x}''$ and all points in $\gamma(\alpha(\mathbf{x}''))$ are reachable, and this shows that $\gamma(\mathbf{q}') \subseteq [[\phi]]^c$, which contradicts our assumption. This completes the proof. □

The above theorem cannot be easily generalized to hybrid systems. Even if all the assumptions made in Theorem 3 were true within each mode, the abstract system could reach boundaries that are unreachable in the concrete system. If there are discrete transitions from these boundary points, then the abstract system would take the corresponding abstract transitions and reach parts of state space unreachable in the concrete. For example, in the exponential decay example discussed above, suppose that there is a discrete transition on the condition that $x = 0$, which takes the system into a new mode $\mathbf{q}'$. The new mode $\mathbf{q}'$ is unreachable in the concrete system, but it would be reachable in the abstract system.

9 Related work

Qualitative reasoning has been used by researchers in Artificial Intelligence for modeling and analyzing physical systems in the face of incomplete knowledge of the system dynamics [44]. The idea is to interpret a continuous variable, say x, over an abstract domain of the form $\{(-\infty, c_0), c_0, (c_0, c_1), c1, (c_1, c_2), c2, \ldots, c_n, (c_n, \infty)\}$, where $c_0, \ldots, c_n \in \mathbb{R}$ are

constants. Model construction involves keeping track of the sign of the derivative of x. In [44], the authors give a method for proving temporal properties about systems specified (incompletely) using *qualitative* differential equations. In [30] and [45], the authors assume a (more) completely specified input model (using differential equations, for example) and construct an abstraction either incrementally [30] or directly [45]. We extend the ideas of qualitative reasoning for analyzing hybrid models by using arbitrary functions over the state variables, and not just state *variables*, for defining the qualitative state space. The methods to seek the right functions by analyzing the differential equations systematically is also novel in our approach. An important realization is that a set of functions closed under Lie derivative computation is important for achieving effective abstractions of the original hybrid system. We also use powerful theorem proving support in creating the abstraction. As a result, the abstractions we obtain have more information and are more useful from an analysis point of view.

There has been a lot of work on constructing abstractions for hybrid systems. These works can be categorized based on the semantics of the hybrid system considered, the class of formulas preserved, the class of hybrid systems considered, the class of abstract systems generated, and whether the abstractions are conservative or accurate. Accurate abstractions, or bisimulations, lead to decidability results [5]. In [36], the interest is in abstracting certain restricted classes of linear hybrid systems into another simpler class of hybrid systems called *timed* automata. The paper [23, 25] translates nonlinear hybrid systems into linear hybrid automata, whereas timed automata approximations of the original nonlinear system are considered in [21]. These approaches can be viewed as abstracting a hybrid system by another hybrid system.

Approaches based on predicate abstraction have also been proposed for timed systems [34] and more recently for hybrid systems [4]. Finite approximations of the system are also constructed in [9]. In both approaches, the continuous dynamics are abstracted using differential equation solvers. Similar approaches, but applied directly to overapproximating reach sets (rather than constructing an abstraction), have been investigated in [23, 27]. There is also some recent work on refining abstractions based on the spurious counter-examples produced [3, 10, 34].

One can naturally associate an infinite state transition system, with uncountably many states, with a continuous dynamical system or a hybrid system. However, different abstractions attempt to preserve different behaviors of this infinite transition system. In [24], certain discrete transitions of the hybrid system are marked *observables*, and the abstraction attempts to preserve the observable behavior. In other cases [19], the discrete states in a run of the system are observed and the abstraction attempts to preserve this sequence of discrete states. In [8], the constructed abstraction captures all the discrete transition made by the system. Therefore, one semantic step in [8] corresponds to a continuous flow followed by exactly one discrete step. In our work, the overall behavior of the hybrid system is abstracted with respect to a finite set of polynomials and the original discrete states. In particular, the behavior inside a continuous evolution is captured too. Our approach guarantees that the abstractions are sound. A price we pay for this is that our method for computing the abstract transitions is more approximate (and consequently much simpler computationally) than some of these other methods. The work [43] is in the discrete-time framework and assumes that the dynamics are specified by a solved function, rather than differential equations, that also satisfies certain conditions on the existence of inverse.

A related thread of work consists of the use of abstract interpretation techniques, like widening, to accelerate reachability (or fixed point) computation [12, 22]. In these works, abstract systems are not generated as we do here, but the widening operation can be interpreted as "working" on the abstraction.

Our work is also closely related to the work on over-approximating reach sets for special kinds of systems. Exact reachability sets can be computed for linear vector fields where the matrix A is diagonalizable with all rational or all purely imaginary eigenvalues [28]. The linear functions constructed from left eigenvectors for use in the abstraction process capture partial reachability information. Thus, our techniques can handle linear systems with mixed (reals and imaginary) eigenvalues, unlike results by Lafferriere et. al. [5, 28], but we only get approximate reachable sets. For more recent work on approximating reachable sets using information from the eigenstructure of the A matrix, see [53].

Some of the good functions p we generate for purposes of abstracting the original system are just "energy" or Lyapunov functions. Such functions have been used to get analytical descriptions of trajectories and provide arguments for stability or periodicity. However, the problem of generating these functions and issues about computability have not been addressed. Moreover, such functions have not been used for defining abstraction mapping and getting over approximations of the reach sets. These features distinguish this work from the well established theory of nonlinear systems.

The sum of squares based method provides an alternative approach to discharging the theorem proving obligations generated by our approach [37, 38, 41]. In our implementation, we use a sound, and incomplete, decision procedure for satisfiability of conjunction of atomic formulas in the theory of real closed fields [48].

10 Conclusion

We have presented a procedure for constructing sound abstractions for hybrid systems. The procedure selects a finite set P of terms over some theory and the abstraction mapping interprets these terms over the three values sign domain, $\{pos, zero, neg\}$. The discrete transitions are abstracted using the standard predicate abstraction approach, while the continuous dynamics are abstracted using qualitative reasoning. Both rely on the ability to decide satisfiability of quantifier-free formulas over the theory of reals, defined over a suitable choice of signature.

The abstraction procedure can be applied compositionally, under certain assumptions, to abstract a hybrid system composed of two hybrid automata that interact through a well-defined set of input and output state variables. We also show that our abstraction procedure is sound and relatively complete.

The quality of the generated abstraction depends crucially on the choice P of terms used. We have identified an important property, viz. closure under Lie derivative computation, which appears to be critical for providing good abstraction mappings. We have outlined some approaches for computing good abstraction predicates.

Acknowledgements The author would like to thank the reviewers for helpful comments and pointing out an error in a proof in an earlier version.

This research was supported in part by the National Science Foundation under grants CCR-0311348 and CCR-0326540, NASA Langley Research Center contract NAS1-00108 to Rannoch Corporation, and the DARPA BioSpice contract DE-AC03-765F00098 to Lawrence Berkeley Laboratory. Some of the results described in this paper also appear in [49–51].

References

1. Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho P-H, Nicollin X, Olivero A, Sifakis J, Yovine S (1995) The algorithmic analysis of hybrid systems. Theor Comput Sci 138(3):3–34

2. Alur R, Courcoubetis C, Henzinger TA, Ho P-H Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Grossman et al [17], pp 209–229
3. Alur R, Dang T, Ivancic F Counter-example guided predicate abstraction of hybrid systems. In: Garavel and Hatcliff [13], pp 208–223
4. Alur R, Dang T, Ivancic F Progress on reachability analysis of hybrid systems using predicate abstraction. In: Maler and Pnueli [31]
5. Alur R, Henzinger T, Lafferriere G, Pappas GJ (2000) Discrete abstractions of hybrid systems. Proc IEEE 88(2):971–984
6. Alur R, Pappas GJ (eds) (2004) Hybrid systems: computation and control, 7th international workshop, HSCC 2004, Philadelphia, PA, March 25–27, 2004, Proceedings. Lecture notes in computer science, vol 2993. Springer, Berlin
7. Ball T, Podelski A, Rajamani SK (2001) Boolean and Cartesian abstraction for model checking C programs. In: Proc of the 7th intl conf on tools and algorithms for the construction and analysis of systems, TACAS 2001. Lecture notes in computer science. Springer, Berlin, pp 268–283
8. Chutinan A, Krogh BH (1999) Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In: Vaandrager FW, van Schuppen JH (eds) HSCC. Lecture notes in computer science, vol 1569. Springer, Berlin, pp 76–90
9. Chutinan A, Krogh BH (2001) Verification of infinite-state dynamic systems using approximate quotient transition systems. IEEE Trans Autom Control 46(9):1401–1410
10. Clarke EM, Fehnker A, Han Z, Krogh BH, Stursberg O, Theobald M Verification of hybrid systems based on counterexample-guided abstraction refinement. In: Garavel and Hatcliff [13], pp 192–207
11. Collins GE (1975) Quantifier elimination for the elementary theory of real closed fields by cylindrical algebraic decomposition. In: Proc 2nd GI conf automata theory and formal languages. Lecture notes in computer science, vol 33. Springer, Berlin, pp 134–183
12. Dang T, Maler O (1998) Reachability analysis via face lifting. In: Henzinger TA, Sastry S (eds) HSCC. Lecture notes in computer science, vol 1386. Springer, Berlin, pp 96–109
13. Garavel H, Hatcliff J (eds) (2003) In: 9th intl conf on tools and algorithms for the construction and analysis of systems, TACAS 2003. Lecture notes in computer science, vol 2619. Springer, Berlin
14. Ghosh R, Tomlin CJ (2001) Lateral inhibition through delta-notch signaling: a piecewise affine hybrid model. In: Hybrid systems: computation and control, HSCC 2001. LNCS, vol 2034. Springer, Berlin, pp 232–246
15. Godbole D, Lygeros J (1994) Longitudinal control of the lead car of a platoon. IEEE Trans Veh Technol 43(4):1125–1135
16. Graf S, Saïdi H (1997) Construction of abstract state graphs with PVS. In: Grumberg O (ed) Proc 9th conference on computer-aided verification (CAV'97). Lecture notes in computer science, vol 1254. Springer, Berlin, pp 72–83
17. Grossman RL, Nerode A, Ravn AP, Rischel H (eds) (1993) In: Hybrid systems. Lecture notes in computer science, vol 736. Springer, Berlin
18. Grumberg O, Long DE (1994) Model checking and modular verification. ACM Trans Program Lang Syst 16(3):843–871
19. Henzinger MR, Henzinger TA, Kopke PW (1995) Computing simulations on finite and infinite graphs. In: Proc 36th annual IEEE symp on foundations of computer science FOCS, pp 453–462
20. Henzinger TA (1995) Hybrid automata with finite bisimulations. In: Proc 22nd intl colloquium on automata, languages, and programming, ICALP 1995. Lecture notes in computer science, vol 944. Springer, Berlin, pp 324–335
21. Henzinger TA, Ho P-H (1995) Algorithmic analysis of nonlinear hybrid systems. In: Wolper P (ed) Computer aided verification, Proc of the 7th intl conf, CAV '95. Lecture notes in computer science, vol 939. Springer, Berlin, pp 225–238
22. Henzinger TA, Ho P-H (1995) A note on abstract interpretation strategies for hybrid automata. In: Antsaklis P, Kohn W, Nerode A (eds) Hybrid systems II. Lecture notes in computer science, vol 999. Springer, Berlin, pp 252–264
23. Henzinger TA, Ho P-H, Wong-Toi H (1998) Algorithmic analysis of nonlinear hybrid systems. IEEE Trans Autom Control 43:540–554
24. Henzinger TA, Kopke PW, Puri A, Varaiya P (1998) What's decidable about hybrid automata? J Comput Syst Sci 57:94–124
25. Henzinger TA, Wong-Toi H (1996) Linear phase-portrait approximations for nonlinear systems. In: Alur R, Henzinger T, Sontag ED (eds) Hybrid systems III. Lecture notes in computer science, vol 1066. Springer, Berlin, pp 377–388
26. Hong H (1990) An improvement of the projection operator in cylindrical algebraic decomposition. In: Proc ISAAC 90, pp 261–264

27. Krogh BH, Stursberg O On efficient representation and computation of reachable sets for hybrid systems. In Maler and Pnueli [31]
28. Lafferriere G, Pappas GJ, Yovine S (2001) Symbolic reachability computations for families of linear vector fields. J Symb Comput 32(3):231–253
29. Lazard D (1990) An improved projection for cylindrical algebraic decomposition. Technical report, Informatique, Universite Paris IV, F-75252 Paris Cedex 05, France
30. Loeser T, Iwasaki Y, Fikes R (1998) Safety verification proofs for physical systems. In: Proc of the 12th intl workshop on qualitative reasoning. AAAI Press, Menlo Park, pp 88–95. Also published as a Knowledge Systems Lab, Stanford University, technical report KSL-98-14
31. Maler O, Pnueli A (eds) (2003) In: Hybrid systems: computation and control, 6th international workshop, HSCC 2003 Prague, Czech Republic, April 3–5, 2003. Proceedings. Lecture notes in computer science, vol 2623. Springer, Berlin
32. McCallum S (1988) An improved projection operator for cylindrical algebraic decomposition of three dimensional space. J Symb Comput 5:141–161
33. Milner R (1971) An algebraic definition of simulation between programs. In: Proc. 2nd IJCAI, pp 481–489
34. Möller MO, Rueß H, Sorea M (2002) Predicate abstraction for dense real-time systems. Electron Notes Theor Comput Sci 65(6). http://www.elsevier.com/locate/entcs/volume65.html
35. Nicollin X, Olivero A, Sifakis J, Yovine S An approach to the description and analysis of hybrid systems. In Grossman et al [17], pp 149–178
36. Olivero A, Sifakis J, Yovine S (1994) Using abstractions for the verification of linear hybrid systems. In: Proc of the 6th computer-aided verification, CAV. Lecture notes in computer science, vol 818. Springer, Berlin, pp 81–94
37. Parrilo PA (2000) Structured semidefinite programs and semialgebraic geometric methods in robustness and optimization. PhD thesis, California Institute of Technology, Pasadena
38. Parrilo PA, Sturmfels B (2003) Minimizing polynomial functions. In: Algorithmic and quantitative real algebraic geometry. DIMACS series in discrete mathematics and theoretical computer science, vol 60, pp 83–99. http://www/arxiv.org/abs/math.OC/0103170
39. Prajna S (2003) Barrier certificates for nonlinear model validation. In: Proc IEEE conference on decision and control
40. Prajna S, Jadbabaie A Safety verification of hybrid systems using barrier certificates. In Alur and Pappas [6], pp 477–492
41. Prajna S, Papachristodoulou A, Parrilo PA (2002) SOSTOOLS: sum of square optimization toolbox for MATLAB, http://www.cds.caltech.edu/sostools
42. Puri A, Varaiya P (1995) Driving safely in smart cars. In: Proc of the 1995 American control conference
43. Raisch J, O'Young S (1997) A totally ordered set of discrete abstractions for a given hybrid or continuous system. In: Hybrid systems IV. Lecture notes in computer science, vol 1273. Springer, Berlin, pp 342–360
44. Shults B, Kuipers BJ (1997) Proving properties of continuous systems: qualitative simulation and temporal logic. AI J 92:91–129
45. Sokolsky O, Hong HS Qualitative modeling of hybrid systems. In: Proc of the Montreal workshop, 2001. Available from http://www.cis.upenn.edu/~rtg/rtg_papers.htm
46. Tabuada P (2004) Flatness and finite bisimulations in continuous time. In: Proc 16th intl symp on mathematical theory of networks and systems
47. Tarski A (1948) A decision method for elementary algebra and geometry, 2nd edn. University of California Press, Berkeley
48. Tiwari A (2003) Abstraction based theorem proving: an example from the theory of reals. In: Proc CADE-19 workshop on pragmatics of decision procedures in automated deduction, PDPAR 2003. INRIA, Nancy, pp 40–52
49. Tiwari A Approximate reachability for linear systems. In Maler and Pnueli [31], pp 514–525
50. Tiwari A, Khanna G (2002) Series of abstractions for hybrid automata. In: Tomlin C, Greenstreet MR (eds) HSCC. Lecture notes in computer science, vol 2289. Springer, Berlin, pp 465–478
51. Tiwari A, Khanna G Nonlinear systems: approximating reach sets. In: Alur and Pappas [6], pp 600–614
52. Vidyasagar M (1993) Nonlinear systems analysis. Prentice Hall, New York
53. Yazarel H, Pappas GJ (2004) Geometric programming relaxations for linear system reachability. In: Proc 2004 American control conference
54. Yazarel H, Prajna S, Pappas GJ (2004) S.O.S. for safety. In: Proc 43rd IEEE conference on decision and control, vol 1, pp 461–466