DISCRETE ABSTRACTIONS OF HYBRID SYSTEMS

RAJEEV ALUR, THOMAS A. HENZINGER, GERARDO LAFFERRIERE, AND GEORGE J. PAPPAS

Abstract. A hybrid system is a dynamical system with both discrete and continuous state changes. For analysis purposes, it is often useful to abstract a system in a way that preserves the properties being analyzed while hiding the details that are of no interest. We show that interesting classes of hybrid systems can be abstracted to purely discrete systems while preserving all properties that are definable in temporal logic. The classes that permit discrete abstractions fall into two categories. Either the continuous dynamics must be restricted, as is the case for timed and rectangular hybrid systems, or the discrete dynamics must be restricted, as is the case for o-minimal hybrid systems. In this paper, we survey and unify results from both areas.

R. Alur, Department of Computer and Information Science, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104. Tel. (215) 573-7483, alur@cis.upenn.edu

T.A. Henzinger, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720. Tel. (510) 643-2430, tah@eecs.berkeley.edu

G. Lafferriere, Department of Mathematical Sciences, Portland State University, Portland, OR 97207. Tel. (503) 725-3662, gerardo@mth.pdx.edu

G.J. Pappas, Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104. Tel. (215) 898-9780, pappasg@ee.upenn.edu

A version of this paper appeared in the *Proceedings of the IEEE* **88**, 2000, pp. 971–984.

1. Introduction

Hybrid systems combine both digital and analog components, in a way that is useful for the analysis and design of distributed, embedded control systems. Hybrid systems have been used as mathematical models for many important applications, such as automated highway systems [40, 50, 79], air traffic management systems [49, 51, 74], embedded automotive controllers [12, 59], manufacturing systems [64], chemical processes [28], robotics [6, 71], real-time communication networks, and real-time circuits [53]. Their wide applicability has inspired a great deal of research from both control theory and theoretical computer science [1, 2, 7, 9, 10, 29, 31, 52, 75].

Many of the above motivating applications are *safety critical*, and require guarantees of safe operation. Consequently, much research focuses on formal *analysis* and *design* of hybrid systems. Formal analysis

of hybrid systems is concerned with verifying whether a hybrid system satisfies a desired specification, like avoiding an unsafe region of the state space. The process of formal design consists of synthesizing controllers for hybrid systems in order to meet a given specification. Both directions have received large attention in the hybrid systems community, and the reader is referred to [3, 11, 23, 25, 33, 42, 55, 73] for expositions to much of the research in the field.

In this paper, we are interested in the formal analysis of hybrid systems. The formal analysis of large scale, hybrid systems is typically a very difficult process due to the complexity and scale of the system. This makes the use of *computational* or *algorithmic* approaches to the verification of hybrid systems very desirable, whenever possible. We are therefore interested in developing computational procedures which, given a hybrid system and a desired property, will verify in a *finite* number of steps whether the system satisfies the specification or not. Given a class of hybrid systems $\mathcal{H}$, and a class of desired properties $\mathcal{P}$, a class of verification problems is called *decidable*, if there exists a computational procedure which, given *any* system $H \in \mathcal{H}$, and *any* property $P \in \mathcal{P}$, will decide in a finite number of steps whether H satisfies P. Decidability is not an issue in the verification of purely discrete systems modeled by finite state machines, since in the worst case verification can be performed by exhaustively searching the whole state space. However, in the case of hybrid systems, decidability is a central issue in algorithmic analysis, because of the uncountability of the state space. The main focus of this paper is on identifying decidable verification problems for hybrid systems.

A natural way to show that a class of analysis problems is decidable, is the process of *abstraction*. Given a hybrid system and some desired property, one extracts a finite, discrete system while preserving all properties of interest. This is achieved by constructing suitable, *finite* and *computable* partitions of the state space of the hybrid system. By obtaining discrete abstractions which are finite, and preserve properties of interest, analysis can be equivalently performed on the finite system, which requires only a finite number of steps. Checking the desired property on the abstracted system should be equivalent to checking the property on the original system. Only if no equivalent abstraction can be found, one may be content with a *sufficient* abstraction, where checking the desired property on the abstracted system is sufficient for checking the property on the original system [20].

In this paper, we focus on *equivalent discrete abstractions of hybrid systems* along with the classes of properties they preserve. We show that there are many interesting classes of hybrid systems which can be abstracted by finite systems for analysis purposes. Properties about the behavior of a system over time are naturally expressible in temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) [26]. Preserving LTL properties leads to special partitions of the state space given by *language equivalence relations*, whereas CTL properties are abstracted by *bisimulations*. A detailed exposition to the use of various logics in hybrid systems can be found

in [23]. Similar concepts and constructions, but from a hierarchical control perspective, can be found in [16, 61, 62, 63].

There are immediate obstacles due to undecidability. For example, in [37] it was shown that checking *reachability* (whether a certain region of the state space can be reached) is undecidable for a very simple class of hybrid systems, where the continuous dynamics involves only variables that proceed at two constant slopes. These results immediately imply that more general classes of hybrid systems cannot have finite bisimulation or language equivalence quotients. Therefore, our search for discrete abstractions of hybrid systems is limited by this result. Given this limit, we show that hybrid systems that can be abstracted fall into two classes. In the first class, the continuous behavior of the hybrid system must be restricted, as in the case of timed automata [5], multirate automata [4, 58], and rectangular automata [37, 68]. In the second class, the discrete behavior of the hybrid system must be restricted, as in the case of order-minimal hybrid systems [44, 45, 46].

In this paper, we present in a unified way all these results which collectively define a very tight boundary between decidable and undecidable questions about hybrid systems. We do not focus on complexity issues or the implementation of these algorithms by verification tools like KRONOS [24], COSPAN [8], UPAAL [48], and HYTECH [35]. It should be noted that, in practice, the algorithms implemented by the above tools work directly on the original system, and do not construct an equivalent finite abstraction first. However, the decidability results presented in this paper for finite abstractions provide correctness and termination arguments for the algorithms implemented by the tools [37, 38, 39]. Therefore, the approach described in this paper should be understood as theoretical background underlying the implementations.

More specifically, in Section 2, we introduce the reader to the notion of transition systems which should be thought of as graphs with a possibly infinite number of nodes (representing states) and edges (representing transitions). Desired properties of transition systems will be expressed as formulas in various temporal logics. We will review the important notions of language equivalences and bisimulations of transition systems, along with temporal logic properties they preserve, namely, Linear Temporal Logic and Computation Tree Logic. In Section 3, after a general definition of hybrid systems, we describe the transition systems generated by our hybrid system model. This allows us to apply the framework of Section 2 to the various classes of hybrid systems we consider in this paper. We then immediately present some undecidability results, which provide a clear boundary for applying the framework of Section 2. As a result, our search for decidable classes of hybrid systems is limited by this boundary. This forces us to consider hybrid systems with either simple continuous dynamics (Section 4), or simple discrete dynamics (Section 5). The latter are based on various first-order logical theories. A brief introduction to first order logic is given in Appendix A.

2. Transition Systems

Transition systems are graph models, possibly with an infinite number of states or transitions.

Definition 2.1 (Transition Systems). *A transition system $T = (Q, \Pi, \rightarrow, \models, Q_0)$ consists of:*

- *A (possibly infinite) set Q of states.*
- *A finite alphabet Π of propositions.*
- *A transition relation $\rightarrow \subseteq Q \times Q$.*
- *A satisfaction relation $\models \subseteq Q \times \Pi$.*
- *A set $Q_0 \subseteq Q$ of initial states.*

A state q_1 is *predecessor* of a state q_2, and q_2 is a *successor* of q_1, written $q_1 \rightarrow q_2$, if the transition relation $\rightarrow$ contains the pair (q_1, q_2). A state q *satisfies* a proposition π, written $q \models \pi$, if the satisfaction relation $\models$ contains the pair (q, π). The transition system T is *finite* if the cardinality of Q is finite, and it is infinite otherwise. We assume that every transition system is *deadlock free*, that is, for every state $q \in Q$, there exists a state $q' \in Q$ such that $q \rightarrow q'$.

A *region* is a subset $P \subseteq Q$ of the states. The sets of predecessor and successor states of P are

$$Pre(P) = \{q \in Q \mid \exists p \in P.\ q \rightarrow p\} \tag{2.1}$$

$$Post(P) = \{q \in Q \mid \exists p \in P.\ p \rightarrow q\} \tag{2.2}$$

The set of states that are accessible from P in two transitions is $Post(Post(P))$, and is denoted $Post^2(P)$. In general, $Post^i(P)$ consists of the states that are accessible from P in i transitions. $Pre^i(P)$ is defined similarly. Then

$$Pre^*(P) = \bigcup_{i \in \mathbb{N}} Pre^i(P) \tag{2.3}$$

$$Post^*(P) = \bigcup_{i \in \mathbb{N}} Post^i(P) \tag{2.4}$$

are the set of states that are *backward* and *forward reachable* from P, that is, accessible in any number of transitions. In particular, $Post^*(Q_0)$ is the set of *reachable states* of the transition system T, and is denoted by $Reach(T)$.

A problem that is of great interest for transition systems is the reachability problem. Given a proposition $\pi \in \Pi$, we write $[\![\pi]\!] = \{q \in Q \mid q \models \pi\}$ for the set of states that satisfy π.

Problem 2.2 (Reachability Problem). *Given a transition system $T = (Q, \Pi, \rightarrow, \models, Q_0)$ and a proposition $\pi \in \Pi$, is $Reach(T) \cap [\![\pi]\!] \neq \emptyset$?*

If the proposition π encodes an undesirable or unsafe region of the state space, then solving reachability corresponds to checking if the system is safe. In this paper, we are interested in computational approaches to the solution of the reachability problem. The following algorithm computes the reachable space until either a state satisfying π is reached, or no more reachable states can be added.

Algorithm 1 (Forward Reachability Algorithm)

initially $R := Q_0$;
while *true* **do**
  if $R \cap [\![\pi]\!] \neq \emptyset$ **then return** "unsafe" **end if;**
  if $Post(R) \subseteq R$ **then return** "safe" **end if**;
  $R := R \cup Post(R)$
end while

A backward reachability algorithm which starts with $[\![\pi]\!]$ and checks whether $Pre^*([\![\pi]\!]) \cap Q_0 \neq \emptyset$ can be similarly constructed. Such iterative algorithmic approaches to checking system properties are guaranteed to terminate if the state space of the transition system is finite, since in the worst case they can only visit a finite number of states. If the state space is infinite, then there is, in general, no guarantee that the forward reachability algorithm will terminate within a finite number of iterations of the loop. It could continue adding states forever without ever reaching the target region $[\![\pi]\!]$ or a fixed point R such that $Post(R) \subseteq R$. In this paper, our goal is to find classes of infinite transition systems whose analysis can be performed on *equivalent* but *finite* transition systems. This is accomplished by constructing suitable finite quotients or *discrete abstractions* of the original system in the sense that they preserve the properties of interest while omitting detail.

In addition to reachability, the desired system specification may require more detailed system properties. For example, one may wish to encode the requirement that a system failure is eventually followed by a return to the normal mode of operation. More abstractly, if the transition system visits a region P_1, encoding a failure, then eventually it will reach a region P_2, encoding normal operation. Such properties can be encoded as formulas in temporal logic [65]. Formulas of temporal logic are thus used to formally specify properties of systems, such as reachability, invariance, or response properties. In the sequel, after defining the notion of quotient transition systems, two kinds of equivalence relations, *language equivalences* and *bisimulations*, are considered along with two popular temporal logics, Linear Temporal Logic (LTL) and Computation Tree Logic (CTL), whose properties they preserve.

An equivalence relation $\sim \subseteq Q \times Q$ on the state space is *proposition preserving* if for all states $p, q \in Q$ and all propositions $\pi \in \Pi$, if $p \sim q$ and $p \models \pi$, then $q \models \pi$; that is, the region $[\![\pi]\!]$ is a union of equivalence classes. Given a proposition-preserving equivalence relation $\sim$, the definition of *quotient*

transition system $T/_\sim$ is natural. Let $Q/_\sim$ denote the quotient space, that is, the set of equivalence classes. For a region P, we denote by $P/_\sim$ the collection of all equivalence classes which intersect P. The transition relation $\rightarrow_\sim$ on the quotient space is defined as follows: for $P_1, P_2 \in Q/_\sim$, we have $P_1 \rightarrow_\sim P_2$ iff there exist two states $q_1 \in P_1$ and $q_2 \in P_2$ such that $q_1 \rightarrow q_2$. The satisfaction relation $\models_\sim$ on the quotient space is defined as follows: for $P \in Q/_\sim$, we have $P \models_\sim \pi$ iff there exists a state $q \in P$ such that $q \models \pi$. The quotient transition system is then $T/_\sim = (Q/_\sim, \Pi, \rightarrow_\sim, \models_\sim, Q_0/_\sim)$.

2.1. **Language equivalences preserve linear temporal properties.** Let $q \in Q$ be a state of the transition system $T = (Q, \Pi, \rightarrow, \models, Q_0)$. Given a state $q \in Q$, let $\Pi_q = \{\pi \in \Pi \mid q \models \pi\}$ be the set of propositions that are satisfied by q. A *trajectory* generated from q is an infinite sequence $q_0 q_1 q_2 \ldots$ such that $q_0 = q$ and for all $i \in \mathbb{N}$, we have $q_i \rightarrow q_{i+1}$. This trajectory defines the *word* $\Pi_{q_0} \Pi_{q_1} \Pi_{q_2} \ldots$ The set of words that are defined by trajectories generated from q is denoted by $L(q)$, and called the *language* of the state q. The set $\bigcup_{q \in Q_0} L(q)$ of words that are defined by trajectories generated from initial states is denoted by $L(T)$, and called the *language* of the transition system T.

Definition 2.3 (Language Equivalences)**.** *Let T be a transition system with state space Q. An equivalence relation $\sim_L$ on Q is a language equivalence of T if for all states $p, q \in Q$, if $p \sim_L q$, then $L(p) = L(q)$.*

Note that every language equivalence is proposition preserving. Every language equivalence $\sim_L$ partitions the state space and gives rise to the quotient transition system $T/_{\sim_L}$, which is called a *language equivalence quotient* of T. The formulas of Linear Temporal Logic (LTL) are interpreted over words, and hence the properties expressed in LTL are preserved by language equivalence quotients.

Definition 2.4 (Linear Temporal Logic [66, 54])**.** *The formulas of Linear Temporal Logic (LTL) are defined inductively as follows:*

- **Propositions** *Every proposition π is a formula.*
- **Formulas** *If ϕ_1 and ϕ_2 are formulas, then the following are also formulas:*

$$\phi_1 \vee \phi_2 \qquad \neg\phi_1 \qquad \bigcirc \phi_1 \qquad \phi_1 \mathcal{U} \phi_2$$

The formulas of LTL are interpreted over infinite sequences of sets of propositions. Consider a word $w = \Pi_0 \Pi_1 \Pi_2 \ldots$, where each Π_i is a set of propositions. The satisfaction of a proposition π at position $i \in \mathbb{N}$ of word w is denoted by $(w, i) \models_L \pi$ (which should not be confused with the satisfaction relation $\models$ which tells us whether a state satisfies a proposition), and holds iff $\pi \in \Pi_i$. We can then recursively define the semantics for any LTL formula as follows:

- $(w, i) \models_L \phi_1 \vee \phi_2$ if either $(w, i) \models_L \phi_1$ or $(w, i) \models_L \phi_2$
- $(w, i) \models_L \neg\phi_1$ if $(w, i) \not\models_L \phi_1$

- $(w, i) \models_L \bigcirc\phi_1$ if $(w, i+1) \models \phi_1$
- $(w, i) \models_L \phi_1 \mathcal{U} \phi_2$ if there is a $j \geq i$ such that $(w, j) \models_L \phi_2$ and for all $i \leq k < j$, we have $(w, k) \models_L \phi_1$

A word w *satisfies* an LTL formula ϕ if $(w, 0) \models_L \phi$. From $\neg$ and $\vee$, which stand for negation and disjunction, respectively, we can also define conjunction $\wedge$, implication $\Rightarrow$, and equivalence $\Leftrightarrow$. The *temporal operators* $\bigcirc$ and $\mathcal{U}$ are called the *next* and *until* operators. The $\bigcirc\phi_1$ formula holds for a word $\Pi_0\Pi_1\Pi_2\ldots$ iff the subformula ϕ_1 is true for the suffix $\Pi_1\Pi_2\ldots$ The formula $\phi_1 \mathcal{U} \phi_2$ intuitively expresses the property that ϕ_1 is true until ϕ_2 becomes true. Using the next and until operators, we can also define the following temporal operators in LTL:

- Eventually: $\Diamond\phi = \mathit{true}\,\mathcal{U}\phi$
- Always: $\Box\phi = \neg\Diamond\neg\phi$

Therefore, $\Diamond\phi$ indicates that ϕ becomes eventually true, whereas $\Box\phi$ indicates that ϕ is true at all positions of a word. The LTL formula $\Box\Diamond\phi$ is true for words that satisfy ϕ infinitely often, whereas a word satisfies $\Diamond\Box\phi$ if ϕ becomes eventually true and then stays true forever.

A transition system T *satisfies* an LTL formula ϕ if some word in the language $L(T)$ satisfies ϕ. For example, if π is a proposition encoding an unsafe region, then violation of safety can be expressed as $\Diamond\pi$. Violation of the more elaborate requirement that visiting region $[\![\pi_1]\!]$ will eventually be followed by visiting region $[\![\pi_2]\!]$, is expressed by the formula $\Diamond(\pi_1 \wedge \Box\neg\pi_2)$.

Problem 2.5 (LTL Model Checking Problem)**.** *Given a transition system T and an LTL formula ϕ, determine if T satisfies ϕ.*

Since reachability can be expressed by an LTL formula of the form $\Diamond\pi$, it is immediate that Problem 2.2 is contained in Problem 2.5. Given the definition of language equivalence, the following theorem should come as no surprise.

Theorem 2.6 (Language equivalences preserves LTL properties)**.** *Let T be a transition system and let $\sim_L$ be a language equivalence of T. Then T satisfies the LTL formula ϕ if and only if the language equivalence quotient $T/_{\sim_L}$ satisfies ϕ.*

Therefore, given a transition system T and an LTL formula ϕ, we can equivalently perform the model checking problem on $T/_{\sim_L}$. In general, language equivalence quotients are not finite. If, however, we are *given* a finite language equivalence quotient of a transition system T, then using the above theorem, LTL model checking can be decided for T.

2.2. **Bisimulations preserve branching temporal properties.** We now define a different way of partitioning the state space along with a class of properties it preserves.

Definition 2.7 (Bisimulations [57]). *Let $T = (Q, \Pi, \rightarrow, \models, Q_0)$ be a transition system. A proposition-preserving equivalence relation $\sim_B$ on Q is a bisimulation of T if for all states $p, q \in Q$, if $p \sim_B q$, then for all states $p' \in Q$, if $p \rightarrow p'$, then there exists a state $q' \in Q$ such that $q \rightarrow q'$ and $p' \sim_B q'$.*

If $\sim_B$ is a bisimulation, then the quotient transition system $T/_{\sim_B}$ is called a *bisimulation quotient* of T. The crucial property of bisimulations is that for every equivalence class $P \in Q/_{\sim_B}$, the predecessor region $Pre(P)$ is a union of equivalence classes. Therefore, if $P_1, P_2 \in Q/_{\sim_B}$, then $Pre(P_1) \cap P_2$ is either the empty set or all of P_2. It is not difficult to check that every bisimulation is a language equivalence, but a language equivalence is not necessarily a bisimulation.

Computation Tree Logic (CTL) is a temporal logic, which contrary to LTL, contains existential quantifiers that range over trajectories.

Definition 2.8 (Computation Tree Logic [19, 69]). *The formulas of Computation Tree Logic (CTL) are defined inductively as follows:*

- **Propositions** *Every proposition π is a formula.*
- **Formulas** *If ϕ_1 and ϕ_2 are formulas, then the following are also formulas:*

$$\phi_1 \vee \phi_2 \qquad \neg\phi_1 \qquad \exists \bigcirc \phi_1 \qquad \exists \Box \phi_1 \qquad \phi_1 \exists \mathcal{U} \phi_2$$

The difference between the semantics of LTL and CTL is that LTL formulas are interpreted over words, whereas CTL formulas are interpreted over the tree of trajectories generated from a given state of a transition system. More precisely, the state q_0 of the transition system T satisfies the proposition π if $q_0 \models \pi$, as usual, and the semantics of any CTL formula is then recursively defined as follows:

- $q_0 \models \phi_1 \vee \phi_2$ if either $q_0 \models \phi_1$ or $q_0 \models \phi_2$
- $q_0 \models \neg\phi_1$ if $q_0 \not\models \phi_1$
- $q_0 \models \exists \bigcirc \phi_1$ if there exists a state $q_1 \in Q$ such that $q_0 \rightarrow q_1$ and $q_1 \models \phi_1$
- $q_0 \models \exists \Box \phi_1$ if there exists a trajectory $q_0 q_1 q_2 \ldots$ generated from q_0 such that for all $i \geq 0$, we have $q_i \models \phi_1$
- $q_0 \models \phi_1 \exists \mathcal{U} \phi_2$ if there exists a trajectory $q_0 q_1 q_2 \ldots$ generated from q_0 such that $q_i \models \phi_2$ for some $i \geq 0$, and for all $0 \leq j < i$, we have $q_j \models \phi_1$

As in LTL, we can define $\wedge$, $\Rightarrow$, and $\Leftrightarrow$ from $\neg$ and $\vee$. The temporal operators $\exists\bigcirc$, $\exists\Box$, and $\exists\mathcal{U}$ are called *possibly-next*, *possibly-always*, and *possibly-until*, as they refer to the existence of a trajectory from a given state. The *possibly-eventually* operator $\exists\Diamond\phi$ is defined as *true* $\exists\mathcal{U}\phi$. Additional temporal operators, which refer to all trajectories from a given state, can be defined as follows:

- Inevitably-next: $\forall \bigcirc \phi = \neg\exists \bigcirc \neg\phi$

- Inevitably-always: $\forall\Box\phi = \neg\exists\Diamond\neg\phi$
- Inevitably-eventually: $\forall\Diamond\phi = \neg\exists\Box\neg\phi$

A transition system T *satisfies* a CTL formula ϕ if some initial state of T satisfies ϕ. For example, reachability can be captured in CTL by the formula $\exists\Diamond\pi$. The CTL formula $\exists\Diamond\forall\Box\pi$ encodes the requirement that there is some reachable state from which all trajectories stay within the region $[\![\pi]\!]$.

Problem 2.9 (CTL Model Checking Problem)**.** *Given a transition system T and a CTL formula ϕ, determine if T satisfies ϕ.*

As in LTL model checking, Problem 2.2 is contained in Problem 2.9. However, Problem 2.5 is incomparable to Problem 2.9, as there are requirements which can be expressed in LTL but not in CTL (such as the requirement $\Box\Diamond\pi$), and there are requirements which can be expressed in CTL but not in LTL (such as the requirement $\exists\Diamond\forall\Box\pi$) [26]. The following theorem shows that bisimulations preserve CTL properties.

Theorem 2.10 (Bisimulation preserves CTL properties [15])**.** *Let T be a transition system and let $\sim_B$ be a bisimulation of T. Then T satisfies the CTL formula ϕ if and only if the bisimulation quotient $T/_{\sim_B}$ satisfies ϕ.*

Therefore, CTL model checking for T can be performed equivalently on $T/_{\sim_B}$. Bisimulations can be computed using the following algorithm. If the algorithm terminates within a finite number of iterations of the loop, then there is a finite bisimulation quotient, and the algorithm returns a finite partition of the state space which is the coarsest bisimulation (i.e., the bisimulation with the fewest equivalence classes).

Algorithm 2 (Bisimulation Algorithm [14, 41]**)**

initially $Q/_{\sim_B} := \{[\![\pi]\!] \mid \pi \in \Pi\}$;
while there exist $P, P' \in Q/_{\sim_B}$ such that $\emptyset \subsetneq P \cap Pre(P') \subsetneq P$ **do**
    $P_1 := P \cap Pre(P')$; $P_2 = P \setminus Pre(P')$;
    $Q/_{\sim_B} := (Q/_{\sim_B} \setminus \{P\}) \cup \{P_1, P_2\}$
end while;
return $Q/_{\sim_B}$

Therefore, in order to show that CTL model checking can be decided for a transition system T, it suffices to show that the bisimulation algorithm terminates on T, and that each step of the algorithm is *computable* or *effective*. This means that we must be able to represent (possibly infinite) state sets symbolically, perform boolean operations, check emptiness, and compute the predecessor operation Pre on the symbolic representation of state sets [33].

Even though LTL and CTL are incomparable, they are both sublogics of CTL*, a more expressive temporal logic, and of a fixed point logic called the *μ-calculus* [23, 26]. Bisimulations preserve not only CTL properties according to Theorem 2.10, but also all CTL* and μ-calculus properties [15].

3. Hybrid Systems

In this section, we apply the framework presented in Section 2 to transition systems generated by hybrid systems. We then immediately present various barriers for obtaining finite discrete abstractions for general hybrid systems, by showing classes of hybrid systems whose reachability problems are undecidable. We start with a definition of hybrid systems.

Definition 3.1 (Hybrid Systems [3]). *A hybrid system is a tuple $H = (V, n, X_0, F, Inv, R)$ with the following components:*

- *V is a finite set of* locations*, and $n \geq 0$ is a nonnegative integer called the* dimension *of H. The state space of H is $X = V \times \mathbb{R}^n$. Each state thus has the form (ℓ, x), where $\ell \in V$ is the* discrete *part of the state, and $x \in \mathbb{R}^n$ is the* continuous *part.*
- *$X_0 \subseteq X$ is the set of initial states.*
- *F: $X \to 2^{\mathbb{R}^n}$ assigns to each state $(\ell, x) \in X$ a* set *$F(\ell, x) \subseteq \mathbb{R}^n$ which constrains the time derivative of the continuous part of the state. Thus in discrete location ℓ, the continuous part of the state satisfies the* differential inclusion *$\dot{x} \in F(\ell, x)$.*
- *Inv: $V \to 2^{\mathbb{R}^n}$ assigns to each location $\ell \in V$ an* invariant *set $Inv(\ell) \subseteq \mathbb{R}^n$ which constrains the value of the continuous part of the state while the discrete part is ℓ.*
- *$R \subseteq X \times X$ is a relation capturing discontinuous state changes.*

We refer to the n individual coordinates of the continuous part $\mathbb{R}^n$ of the state space as real-valued *variables*, and we view the continuous part $x = (x_1, \ldots, x_n)$ of a state as an assignment of values to the variables.

Hybrid systems are typically represented as finite graphs with vertices V, and edges E defined by

$$E = \{(\ell, \ell') \in V \times V \mid ((\ell, x), (\ell', x')) \in R \text{ for some } x \in Inv(\ell) \text{ and } x' \in Inv(\ell')\}.$$

With each vertex $\ell \in V$ we associate an *initial* set defined as

$$Init(\ell) = \{x \in Inv(\ell) \mid (\ell, x) \in X_0\}.$$

With each edge $e = (\ell, \ell') \in E$ we associate a *guard* set defined as

$$Guard(e) = \{x \in Inv(\ell) \mid ((\ell, x), (\ell', x')) \in R \text{ for some } x' \in Inv(\ell')\}$$

and a set-valued *reset* map

$$Reset(e, x) = \{x' \in Inv(\ell') \mid ((\ell, x), (\ell', x')) \in R\}.$$

Trajectories of the hybrid system H originate at any initial state $(\ell, x) \in X_0$ and consist of concatenations of *continuous flows* and *discrete jumps*. Continuous flows keep the discrete part ℓ of the state constant, and the continuous part evolves over time according to the differential inclusions $\dot{x} \in F(\ell, x)$, as long as x remains inside the invariant set $Inv(\ell)$. If during the continuous flow, it happens that $x \in Guard(e)$ for some $e = (\ell, \ell') \in E$, then the edge e becomes *enabled*. The state of the hybrid system may then instantaneously jump from (ℓ, x) to any (ℓ', x') with $x' \in Reset(e, x)$. Then the process repeats, and the continuous part of the state evolves according to the differential inclusions $\dot{x} \in F(\ell', x)$. Even though Definition 3.1 places no well-posedness conditions on the class of hybrid systems we consider, the results presented in this paper will assume strong restrictions regarding the types of X_0, F, Inv, and R which are permitted.

Example 3.2. Figure 1 is a graphical illustration of a special kind of hybrid system, called a *timed automaton*, which is a finite state machine coupled with real-valued *clock* variables. This timed automaton consists of two locations ℓ_1 and ℓ_2, and two variables x and y which always evolve in $\mathbb{R}$ under the differential equations $\dot{x} = 1$ and $\dot{y} = 1$. Therefore x and y simply measure time. The initial state of the system is $(\ell_1, x = 0, y = 0)$ and the invariant sets associated with the locations ℓ_1 and ℓ_2 are $x < 5$ and $y < 10$, respectively. There are two edges, $e_1 = (\ell_1, \ell_2)$ and $e_2 = (\ell_2, \ell_1)$. The guard of e_1 is the set $x > 4$ and the reset map is $R(e_1, x, y) = \{(10, 3)\}$, whereas the guard and reset of e_2 are $y > 9$ and $R(e_2, x, y) = \{(x, 0)\}$, respectively. Notice that the identity map on the x variable on the e_2 edge is suppressed from Figure 1. A simple reachability specification may require that the timed automaton never enters the region $\{(\ell_2, x, y) \mid x > 7 \text{ and } y < 6\}$.

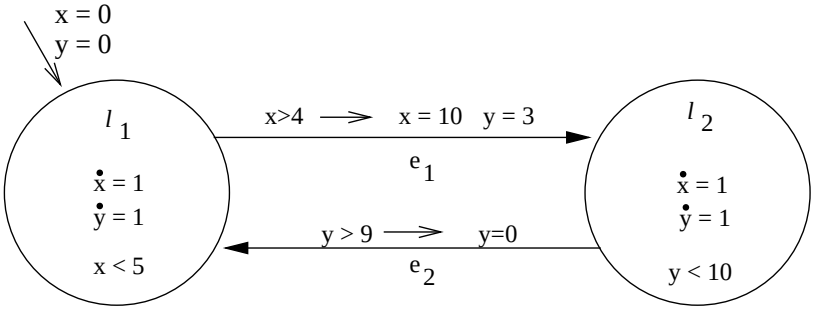


FIGURE 1. A timed automaton

3.1. **Rectangular, multirate, and timed automata.** Consider the space $\mathbb{R}^n$ with the variables $x_1, \ldots, x_n$. A *rectangular set* is defined by a conjunction of linear (in)equalities of the form $x_i \approx c$,

where $\approx$ is one of $<, \leq, =, \geq, >$, and $c \in \mathbb{Q}$. For a rectangular set B, let B_i be its projection onto the i-th coordinate. Thus a rectangular set $B \subseteq \mathbb{R}^n$ is of the form $B = B_1 \times \cdots \times B_n$, where each B_i is a bounded or unbounded interval.

Definition 3.3 (Rectangular Automata [37]). *A rectangular automaton is a hybrid system that satisfies the following constraints.*

- *For every location $\ell \in V$, the sets $Init(\ell)$ and $Inv(\ell)$ are rectangular sets.*
- *For every location $\ell \in V$, there is a rectangular set B^ℓ such that $F(\ell, x) = B^\ell$ for all $x \in \mathbb{R}^n$.*
- *For every edge $e \in E$, the set $Guard(e)$ is a rectangular set, and there is a rectangular set B^e and a subset $J^e \subseteq \{1, \ldots, n\}$ such that for all $x \in \mathbb{R}^n$,*

  $$Reset(e, x) = \{(x'_1, \ldots, x'_n) \in \mathbb{R}^n \mid \text{for all } 1 \leq i \leq n, \text{ if } i \in J^e \text{ then } x'_i \in B^e_i \text{ else } x'_i = x_i\}.$$

Therefore, in a rectangular automaton, the derivative of each variable stays between two fixed bounds, which may be different in different locations. This is because in each location ℓ, the differential inclusions are constant and coordinate-wise decoupled, that is, $\dot{x}_i \in B^\ell_i$ for all $1 \leq i \leq n$. With each discrete jump across an edge e, the value of a variable x_i is either left unchanged (if $i \notin J^e$), or reset nondeterministically to a new value within some fixed, constant interval B^e_i (if $i \in J^e$). An example of a rectangular automaton is shown in Figure 2.

A rectangular automaton is *initialized* if for every edge $e = (\ell, \ell') \in E$ and all $1 \leq i \leq n$, if $Reset(e, x)_i = x_i$, then $F(\ell', x)_i = F(\ell, x)_i$. In other words, if after a discrete jump the bounds on the derivative of a variable change, then its value must be nondeterministically reset ("reinitialized") within a fixed interval. The rectangular automaton of Figure 2 is initialized.

Definition 3.4 (Multirate Automata [3]). *A multirate automaton is a rectangular automaton that satisfies the following constraints:*

- *For each location $\ell \in V$, the set $Init(\ell)$ is either empty or a singleton set.*
- *For each edge $e \in E$, the set B^e is a singleton set.*
- *For each location $\ell \in V$, the set B^ℓ is a singleton set.*

Therefore, in a multirate automaton, each variable follows constant, rational slope, which may be different in different locations. Multirate automata may or may not be initialized.

Definition 3.5 (Timed Automata [5]). *A timed automaton is a multirate automaton such that $B^\ell = \{(1, 1, \ldots, 1)\}$ for each location $\ell \in V$.*

Therefore, in a timed automaton, in every location each variable follows the constant slope 1, that is, $\dot{x}_i = 1$ for all $1 \leq i \leq n$. Each x_i is thus referred to as a clock variable. Notice that timed automata are initialized by definition, because the differential inclusion never changes.

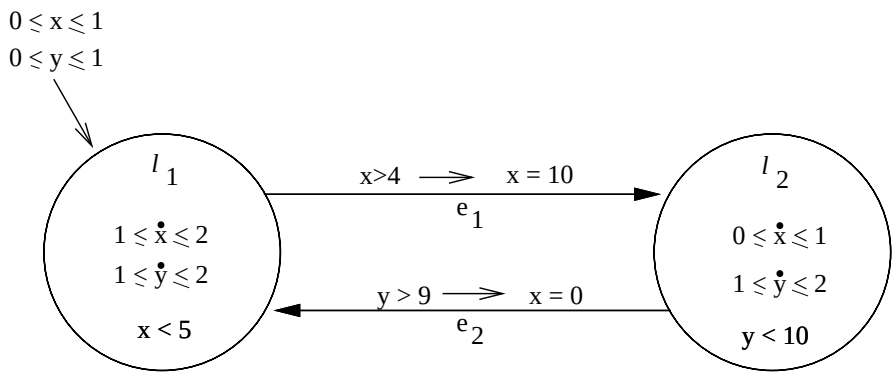


FIGURE 2. A rectangular automaton

3.2. **Transition systems of hybrid systems.** Let $H = (V, n, X_0, F, Inv, R)$ be a hybrid system, and let Σ be a finite set of subsets of $\mathbb{R}^n$. The hybrid system H generates a transition system $T_{H,\Sigma} = (Q, \Pi, \rightarrow, \models, Q_0)$ with respect to Σ. Set $Q = X = V \times \mathbb{R}^n$ and $Q_0 = X_0$. Set $\Pi = V \cup \Sigma$, that is, the propositions are the locations and the given sets in Σ. For $\pi \in V$, define $(\ell, x) \models \pi$ iff $\ell = \pi$, and for $\pi \in \Sigma$, define $(\ell, x) \models \pi$ iff $x \in \pi$. Finally, define $\rightarrow = (\cup_{e \in E} \xrightarrow{e}) \cup \xrightarrow{\tau}$ as follows.

Discrete transitions: $(\ell, x) \xrightarrow{e} (\ell', x')$ for $e = (\ell, \ell') \in E$ iff $x \in Guard(e)$ and $x' \in Reset(e, x)$.

Continuous transitions: $(\ell_1, x_1) \xrightarrow{\tau} (\ell_2, x_2)$ iff $\ell_1 = \ell_2$ and there exists a real $\delta \geq 0$ and a differentiable curve x: $[0, \delta] \rightarrow \mathbb{R}^n$ with $x(0) = x_1$, $x(\delta) = x_2$, for all $t \in [0, \delta]$ we have $x(t) \in Inv(\ell_1)$, and for all $t \in (0, \delta)$ we have $\dot{x}(t) \in F(\ell_1, x(t))$.

The continuous τ transitions are time-abstract transitions in the sense that the time it takes to reach one state from another is ignored.

Having defined the transition system of a hybrid system allows us to proceed with the conceptual framework presented in Section 2, and determine language equivalence and bisimulation quotients of hybrid systems. The next subsection presents some immediate barriers in obtaining such discrete abstractions which are finite.

3.3. **Undecidability barriers.** A variable x_i is a *two slope* variable if there exist $k_1, k_2 \in \mathbb{Q}$ such that for all locations $\ell \in V$, either $F(\ell, x)_i = \{k_1\}$ or $F(\ell, x)_i = \{k_2\}$. The rationals k_1 and k_2 are the *slopes* of x_i. The variable x_i is a *one slope* variable if $k_1 = k_2$. Note that a clock variable is a one slope variable with slope $k_1 = k_2 = 1$. The following theorem presents an immediate obstacle in obtaining finite discrete abstractions of hybrid systems.

Theorem 3.6 (Undecidability of uninitialized multirate automata [37])**.** *Consider the class of multirate automata with $n - 1$ clock variables and one two slope variable with slopes $k_1 \neq k_2$. The reachability problem (Problem 2.2) is undecidable for this class.*

In other words, there is no computational procedure that takes as input any multirate automaton H from the given class, and a proposition π, and determines if any trajectory visits a state that satisfies π. The proof of the undecidability result proceeds by a reduction from the halting problem for two counter machines, and can be found in [37]. Theorem 3.6 shows that initialization is a necessary condition for decidability. An additional necessary condition is provided by the following theorem, which shows that any violation of rectangularity, namely the coupling variables, also leads to undecidability.

Theorem 3.7 (Undecidability of coupling variables in multirate automata [37])**.** *Suppose we generalize the definition of multirate automata so to permit either (1) the intersection of rectangular guard sets $Guard(e)$ with inequalities of the form $x_i \leq x_j$, or (2) the intersection of rectangular invariant sets $Inv(e)$ with inequalities of the form $x_i \leq x_j$, or (3) reset maps of the form $Reset(e, x)_i = x_j$, for $j \neq i$. Consider a class of multirate automata which are generalized in one of these three ways, and which have $n - 1$ clock variables and a one slope variable with slope $k \neq 1$. The reachability problem (Problem 2.2) is undecidable for this class.*

Since the reachability problem is a special case of LTL and CTL model checking, it is clear from Theorems 3.6 and 3.7 that Problems 2.5 and 2.9 are also undecidable for very restrictive classes of hybrid systems. Consequently, it must be impossible to construct finite language equivalence or bisimulation quotients for transition systems $T_{H,\Sigma}$, where H is a hybrid system of Theorem 3.6 or 3.7, and $\Sigma = \emptyset$.

The above negative results force us to consider hybrid systems with either simpler discrete dynamics or simpler continuous dynamics, in order for the framework of Section 2 to be successful. In the next two sections, we survey such results, which, in conjunction with Theorems 3.6 and 3.7, define a tight boundary between decidability and undecidability for model checking of hybrid systems.

4. Restricting the Flows

In this section, we obtain discrete abstraction of hybrid systems with restricted continuous dynamics. We first consider timed automata, which have finite bisimulation quotients of a very intuitive structure.

4.1. **Timed and multirate automata.** A timed automaton H is defined by a finite graph (V, E), a dimension n, and linear inequalities of the form $x_i \approx c$, where $c \in \mathbb{Q}$, which define initial, invariant, and guard sets, as well as reset maps. Even though the timed automata defined in Section 3.1 allow rational constants in their definition, in this section we consider timed automata with only integer constants. There is no loss of generality in this assumption, because a finite number of rationals

can always be rescaled to integers. Furthermore, we restrict the clock variables to range over the nonnegative reals. There is also no loss of generality in this assumption, because every clock variable of a timed automaton is bounded from below by initial sets and reset maps. Let C_i be the largest integer that x_i is compared to in the definition of H. For example, in Figure 1, the largest integer that x is compared to is 10 (in the reset map of e_1), which is also the largest integer to which y is compared (in the invariant set of ℓ_2).

Given a nonnegative real $x \in \mathbb{R}_{\geq 0}$, let $\lfloor x \rfloor$ stand for the floor function, let $\lceil x \rceil$ stand for the ceiling function, and let $\langle x \rangle$ stand for the fractional part of x; that is, $\langle x \rangle = x - \lfloor x \rfloor$. We define the following equivalence relations on $\mathbb{R}^n_{\geq 0}$ and on $X = V \times \mathbb{R}^n_{\geq 0}$, the state space of H.

Definition 4.1 (Region Equivalence [5]). *Two vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ in $\mathbb{R}^n_{\geq 0}$ are region equivalent, written $x \sim^R y$, if the following two conditions are satisfied:*

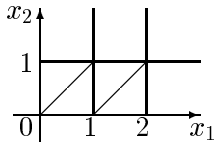
- *For all $1 \leq i \leq n$, we have either both $\lfloor x_i \rfloor = \lfloor y_i \rfloor$ and $\lceil x_i \rceil = \lceil y_i \rceil \leq C_i$, or both $\lceil x_i \rceil > C_i$ and $\lceil y_i \rceil > C_i$.*
- *For all $1 \leq i, j \leq n$, if $\lceil x_i \rceil \leq C_i$ and $\lceil x_j \rceil \leq C_j$, then $\langle x_i \rangle \leq \langle x_j \rangle$ iff $\langle y_i \rangle \leq \langle y_j \rangle$.*

Two states (ℓ_1, x_1) and (ℓ_1, x_2) in X are region equivalent, $(\ell_1, x_1) \sim^R_H (\ell_2, x_2)$, if both $\ell_1 = \ell_2$ and $x_1 \sim^R x_2$.

Therefore two states of H are region equivalent if they agree on the discrete parts, on the integral parts of all clock values, and on the ordering of the fractional parts of all clock values. The integral parts of the clock values determine whether or not a particular clock constraint is met, whereas the ordering of the fractional parts determines which clock will change its integral part first. For example, if two clocks x and y are between 0 and 1 in a state, then an edge whose guard set is defined by the clock constraint $x = 1$ can be followed by an edge which is guarded by the clock constraint $y = 1$, depending on whether or not the current clock values satisfy $x < y$. Furthermore, since each clock variable x_i is never compared with constants greater than C_i, then the actual value of x_i, once it exceeds C_i, is of no consequence in determining the validity of any clock constraints.

Example 4.2. The nature of the equivalence classes defined by $\sim^R$ can be best understood using a planar example. Consider $(x_1, x_2) \in \mathbb{R}^2_{\geq 0}$ with $C_1 = 2$ and $C_2 = 1$. The equivalence classes are shown in Figure 3. Note that there are only a finite number of classes, at most $n! \cdot 2^n \cdot \Pi_{i=1}^n (2C_i + 2)$, where n is the number of clock variables. Thus, the number of classes is exponential in the dimension and in the size of clock constraints (each constant C_i requires $\log C_i$ bits for representation in a clock constraint).

If we are given a finite set Σ of rectangular sets, then we define the region equivalence relation $\sim^R_{H,\Sigma}$ on the states of the timed automaton H just like $\sim^R_H$, except that the constants C_i are taken to



6 corner points: e.g., $\{(0,1)\}$
14 open line segments: e.g., $\{(x_1, x_2) \mid 0 < x_1 = x_2 < 1\}$
8 open regions: e.g., $\{(x_1, x_2) \mid 0 < x_1 < x_2 < 1\}$

FIGURE 3. Equivalence classes of planar region equivalence

be maximal also with respect to the constants that define the sets in Σ. The following is the main theorem about timed automata.

Theorem 4.3 (Bisimulations of timed automata [5]). *Let H be a timed automaton, and let Σ be a finite set of rectangular sets. Then the region equivalence relation $\sim^R_{H,\Sigma}$ is a bisimulation of the transition system $T_{H,\Sigma}$.*

Since the region equivalence relation $\sim^R_{H,\Sigma}$ has a finite number of equivalence classes, and the corresponding quotient transition system can be constructed effectively, we obtain the following corollary.

Corollary 4.4. *The LTL and CTL model checking problems (Problems 2.5 and 2.9) can be decided for timed automata, provided every proposition occurring in temporal formulas is either an automaton location or a rectangular set.*

The above result was the first successful extraction of a finite discrete abstraction from a hybrid system, and has inspired much research in this direction along with the development of verification tools.The result can be generalized as follows to multirate automata.

Theorem 4.5 (Bisimulations of initialized multirate automata [3]). *Let H be an initialized multirate automaton, and let Σ be a finite set of rectangular sets. Then the transition system $T_{H,\Sigma}$ has a finite bisimulation quotient, which can be constructed effectively.*

The proof of Theorem 4.5 is based on rescaling the slope of each variable to 1, by appropriately adjusting all initial, invariant, and guard sets, as well as reset maps. From the region equivalence of the resulting timed automaton we obtain a bisimulation of the initialized multirate automaton.

Corollary 4.6. *The LTL and CTL model checking problems (Problems 2.5 and 2.9) can be decided for initialized multirate automata, provided every proposition occurring in temporal formulas is either an automaton location or a rectangular set.*

Notice that restricting ourselves to *initialized* multirate automata in Theorem 4.5 does not violate the conditions of Theorem 3.6, by which multirate automata which are not initialized cannot, in

general, have a finite bisimulation quotient. Similarly, restricting ourselves to propositions which are rectangular sets in Corollary 4.6 does not violate the spirit of Theorems 3.7.

4.2. **Rectangular automata.** Up to this point, the restricted classes of hybrid systems that we have presented admit finite bisimulation quotients. In this section, we show that more general hybrid automata do not admit finite bisimulation quotients, but may admit finite language-equivalence quotients, which are coarser quotients.

Theorem 4.7 (Language equivalences of initialized rectangular automata [37, 38])**.** *Let H be an initialized rectangular automaton, and let Σ be a finite set of rectangular sets. Then the transition system $T_{H,\Sigma}$ has a finite language-equivalence quotient, which can be constructed effectively.*

The main idea of the proof is to convert an initialized rectangular automaton to an initialized multirate automaton, by replacing each variable x_i, which satisfies a differential inclusion of the form $\dot{x}_i \in [a_i, b_i]$, by two variables named x_i^l and x_i^u, which satisfy $\dot{x}_i^l = a_i$ and $\dot{x}_i^u = b_i$, respectively. The variables x_i^l and x_i^u keep track of the lower and upper bounds of x_i. The initial, invariant, and guard sets, as well as the reset maps must be adjusted accordingly. For example, if the guard set is defined by $x_i \leq 3$, then it is replaced by $x_i^l \leq 3$, and if $x_i^u > 3$ then x_i^u is reset to 3. This conversion from the rectangular to a multirate automaton is language preserving. Hence, from the finite bisimulation of the initialized multirate automaton (Theorem 4.5) we can construct a finite language equivalence of the original initialized rectangular automaton.

Corollary 4.8. *The LTL model checking problem (Problem 2.5) can be decided for initialized rectangular automata, provided every proposition occurring in temporal formulas is either an automaton location or a rectangular set.*

The conversion from initialized rectangular automata to initialized multirate automata may not preserve branching properties, such as those expressible in CTL. In general, initialized rectangular automata do not admit finite bisimulation quotients.

Theorem 4.9 (Lack of finite bisimulation quotients for initialized rectangular automata [32])**.** *There exist an initialized rectangular automaton H and a finite set Σ of rectangular sets such that every bisimulation of the transition system $T_{H,\Sigma}$ has infinitely many equivalence classes.*

In order to simplify the proof of the above theorem, we consider a slight extension of Definition 3.3 and allow more than one edge between a pair of locations.

Example 4.10. Consider the simple rectangular automaton H shown in Figure 4. The automaton has only one location, ℓ, is trivially initialized, and has two variables, x and y, which are allowed to

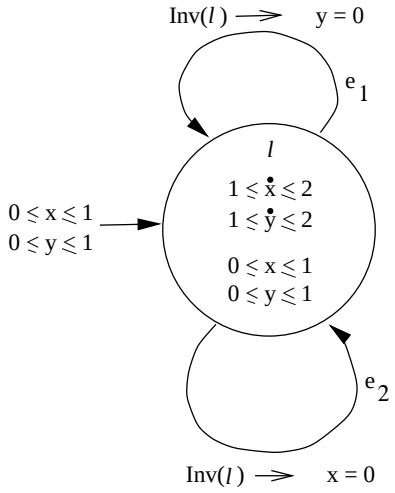


FIGURE 4. An initialized rectangular automaton without finite bisimulation quotient

live on the unit square; that is, $Inv(\ell) = \{(x, y) \in \mathbb{R}^2 \mid 0 \leq x \leq 1 \text{ and } 0 \leq y \leq 1\}$. Furthermore, $Init(\ell) = Inv(\ell)$. Both x and y satisfy the differential inclusion $\dot{x} \in [1, 2]$ and $\dot{y} \in [1, 2]$. There are two edges from ℓ to itself, e_1 and e_2, with $Guard(e_1) = Guard(e_2) = Inv(\ell)$. Furthermore, $Reset(e_1, (x, y)) = \{(x, 0)\}$ and $Reset(e_2, (x, y)) = \{(0, y)\}$; that is, e_1 and e_2 reset y or x to 0, respectively. Let Σ consist of the two rectangular sets defined by $x = 1$ and $y = 1$. Then the bisimulation algorithm (Algorithm 2.2) does not terminate on the transition system $T_{H,\Sigma}$.

The classes of hybrid systems presented in this section are expressive enough to model many systems arising in real-time communication networks, real-time circuits, as well as real-time software. Timed automata allow us to model accurate clocks, and rectangular automata allow us to model clocks with bounded drift. However, the continuous dynamics (flows) that can captured directly by rectangular automata is rather limited for control applications, and generally involves approximations [36, 67]. In order to capture more complicated continuous dynamics directly without violating the undecidability results of Section 3.3, one needs to restrict the discrete dynamics (jumps) of a hybrid system.

5. Restricting the Jumps

Our goal in this section is to apply the framework of Section 2 to hybrid systems with more complicated continuous behavior. However, the following example shows that, even in the absence of discrete dynamics, the bisimulation algorithm does not terminate.

Example 5.1. Consider the trivial hybrid system with only one discrete location ℓ, no discrete jumps, and let F be the linear vector field on $\mathbb{R}^2$

$$\begin{aligned} \dot{x}_1 &= 0.2 \cdot x_1 + x_2 \\ \dot{x}_2 &= -x_1 + 0.2 \cdot x_2 \end{aligned}$$

Assume the partition of $\mathbb{R}^2$ consists of the following three sets (see Figure 5):

$$\begin{aligned} P_1 &= \{(x,0) : 0 \leq x \leq 4\} \\ P_2 &= \{(x,0) : -4 \leq x < 0\} \\ P_3 &= \mathbb{R}^2 \setminus (P_1 \cup P_2) \end{aligned}$$

The trajectories of F are spirals moving away from the origin. The first iteration of the algorithm

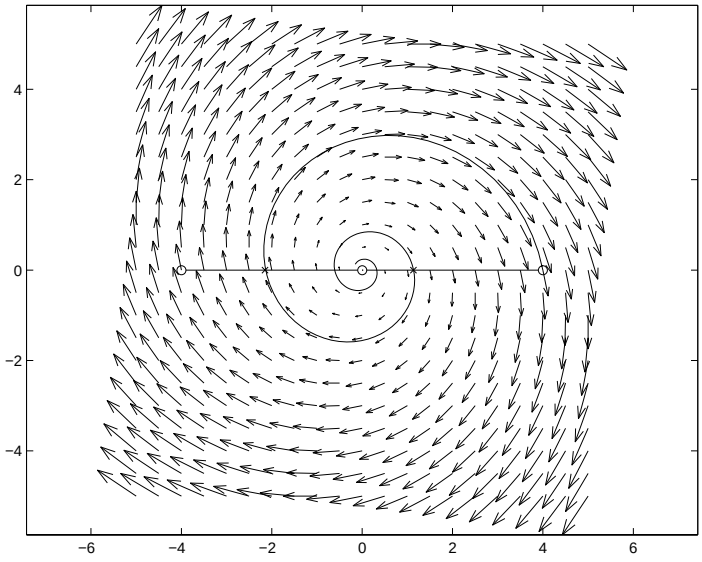


FIGURE 5. Bisimulation algorithm does not terminate

partitions P_2 into $P_4 = P_2 \cap Pre(P_1) = \{(x,0) : z_1 \leq x < 0\}$ and $P_2 \setminus Pre(P_1)$, where $z_1 < 0$ is the x_1-coordinate of the first intersection point of the spiral through $(4,0)$ with P_2. The second iteration subdivides P_1 into $P_5 = P_1 \cap Pre(P_4) = \{(x,0) : 0 \leq x \leq z_2\}$ and $P_1 \setminus Pre(P_4)$ where $z_2 > 0$ is the x_1-coordinate of the next point of intersection of the spiral with P_1. This process continues indefinitely since the spiral intersects P_1 in infinitely many points, and therefore the algorithm does not terminate. In fact, the bisimilarity quotient is not finite.

From the above example it is clear that the critical problem one must investigate is how the trajectories of $F(\ell, \cdot)$ interact with the sets inside a single location ℓ. This requires that the trajectories

of the vector field $F(\ell, \cdot)$ have *nice* intersection properties with such sets. Since the goal is to obtain finite partitions, it will become important that we restrict the study to classes of sets with *global finiteness* properties, for example, sets with finitely many connected components. Even though these desirable properties are geometric in nature, they are captured by the notion of order-minimality (o-minimality) from model theory.

5.1. **O-Minimal Structures.** In this section we provide a brief introduction to o-minimal structures [77], and then use it to construct finite bisimulations of certain classes of hybrid systems. A brief introduction to first-order logic can be found in Appendix A. More introductory material on first-order logic can be found in [27, 76], and the use of various logics for hybrid systems is detailed in [23].

Definition 5.2 (O-Minimal Structure)**.** *A (model-theoretic) structure over the reals is called o-minimal (order minimal) if every definable subset (with parameters) of* $\mathbb{R}$ *is a finite union of points and open intervals (possibly unbounded).*

For structures which extend $(\mathbb{R}, <, +, -, 0, 1)$, this is equivalent to checking the above property for sets definable without parameters [56]. For example, consider the subset of the reals defined by $\{x \in \mathbb{R} \mid p(x) \geq 0\}$, where $p(x)$ is some polynomial. Then, since every polynomial has a finite number of roots, the set where it is not negative is a finite union of points and intervals. This finiteness property must hold for any definable set in the structure, $\{x \in \mathbb{R} \mid \phi(x)\}$, even if the formula $\phi(x)$ contains quantifiers.

The class of o-minimal structures over the reals is quite rich. In [72] it was shown that the structure $(\mathbb{R}, <, +, -, \cdot, 0, 1)$, admits elimination of quantifiers, by proposing an algorithm which given any formula in $(\mathbb{R}, <, +, -, \cdot, 0, 1)$ converts it to an equivalent formula without quantifiers. This, together with an analysis of the sets definable by quantifier-free formulas shows that the structure is o-minimal. Tarski was also interested in extending this result to $(\mathbb{R}, <, +, -, \cdot, e^x, 0, 1)$, where there is an additional symbol in the language for the exponential function. While this structure does not admit elimination of quantifiers, it was shown in [80] that this structure is o-minimal. Another important extension is obtained as follows. Assume f is a real-analytic function in a neighborhood of the cube $[-1, 1]^n \subset \mathbb{R}^n$. Let $\hat{f} \colon \mathbb{R}^n \to \mathbb{R}$ be the function defined by

$$\hat{f}(x) = \begin{cases} f(x) & \text{if } x \in [-1, 1]^n \\ 0 & \text{otherwise} \end{cases}$$

We call such functions *restricted analytic functions*. These functions are useful to describe the behavior of some periodic trajectories. For example, the functions sin and cos restricted to a period are sufficient to define closed orbits of some linear systems. In [78], the structure $(\mathbb{R}, <, +, -, \cdot, e^x, \{\hat{f}\}, 0, 1)$,

which is an extension of $(\mathbb{R}, <, +, -, \cdot, \{\hat{f}\}, 0, 1)$, was shown to be o-minimal. The following table summarizes o-minimal structures over the reals along with some *examples* of sets and vector field trajectories that are definable in these theories.

Table 1 : O-Minimal Structures

Structure	Sample Definable Sets	Sample Definable Trajectories
$(\mathbb{R}, <, +, -, 0, 1)$	Polyhedral sets	Linear trajectories
$(\mathbb{R}, <, +, -, \cdot, 0, 1)$	Semialgebraic sets	Polynomial trajectories
$(\mathbb{R}, <, +, -, \cdot, \{\hat{f}\}, 0, 1)$	Subanalytic sets	Polynomial trajectories
$(\mathbb{R}, <, +, -, \cdot, e^x, 0, 1)$	Semialgebraic sets	Exponential trajectories
$(\mathbb{R}, <, +, -, \cdot, e^x, \{\hat{f}\}, 0, 1)$	Subanalytic sets	Exponential trajectories

Based on the notion of o-minimality, the following class of hybrid systems is defined.

Definition 5.3 (O-Minimal Hybrid Systems)**.** *A hybrid system H is called o-minimal if*

- *for each $\ell \in V$, $F(\ell, \cdot)$ is a differential equation whose flow is complete (defined for all time)*
- *for each $e \in E$, the reset map $Reset(e, x)$ is a piecewise constant (with finite number of pieces) but set valued map.*
- *for each $\ell \in V$ and all edges $e \in E$, the sets $Inv(\ell)$, $Init(\ell)$, and $Guard(e)$, and the flow of $F(\ell, \cdot)$ are definable in the same o-minimal structure over the reals.*

Note that o-minimal hybrid systems place a restriction on the discrete jumps, namely that every time a discrete jump is taken, all states must be reinitialized, possibly nondeterministically. Notice, however, that we do allow piecewise constant set valued maps, which can be used to overapproximate, arbitrarily closely, useful reset maps like the identity map. A more detailed analysis of set valued maps can be found in [22]. This restriction on the discrete dynamics along with the powerful structure of o-minimal structures, allows us to prove the following theorem without violating the results of Section 3.3. Even though the following theorem is proved in [44] for constant, set valued reset maps, the proof can be easily adapted to handle piecewise constant, set valued resets.

Theorem 5.4 (Bisimulations of O-Minimal Hybrid Systems [44])**.** *Let H be an o-minimal hybrid system, and let Σ be a finite collection of sets definable in the same o-minimal structure. Then the transition system $T_{H,\Sigma}$ has a finite bisimulation quotient.*

Theorem 5.4 is appealing since it can capture hybrid systems with more complicated continuous dynamics. To illustrate the continuous behavior that can be captured, we apply Theorem 5.4 for each o-minimal structure of Table 1, and we provide examples of definable, o-minimal hybrid systems.

$(\mathbb{R}, <, +, -, 0, 1)$. The definable sets in this structure capture polyhedral sets whereas the definable flows capture linear flows. In particular, it captures timed and multirate automata in the special case where all reset maps are constant. Timed and multirate automata, in general, allow more complicated reset maps, like the identity map, in their discrete jumps.

$(\mathbb{R}, <, +, -, \cdot, 0, 1)$. In [72], it was shown that $(\mathbb{R}, <, +, -, \cdot, 0, 1)$ is decidable. In fact, the decision procedure consisted of two parts: first an algorithm for eliminating quantifiers, and second an algorithm for deciding quantifier free formulas. Because of these results, the definable sets with parameters in this structure are the *semialgebraic sets*, which are defined as boolean combinations of sets of the form $\{x : p(x) < 0\}$ and $\{x : p(x) = 0\}$ where $p(x)$ is a polynomial. The definable flows in this structure are semialgebraic. Therefore, the o-minimal hybrid systems corresponding to this structure are hybrid systems H where all sets and flows are semialgebraic.

$(\mathbb{R}, <, +, -, \cdot, \{\hat{f}\}, 0, 1)$. In order to describe the definable sets in this structure, we need the notions of *semianalytic* and *subanalytic sets*. We provide below an informal definition of these notions. For precise definitions and properties the reader is referred to [13]. We say that a subset S of $\mathbb{R}^n$ is semianalytic in $\mathbb{R}^n$ if for every $x \in \mathbb{R}^n$ there exists a neighborhood U of x such that $U \cap S$ is a boolean combination of sets of the form $\{x : f(x) < 0\}$ and $\{x : f(x) = 0\}$ where f is an analytic function on U. Roughly speaking, a local description of a semianalytic set is analogous to that of a semialgebraic set with analytic functions replacing polynomials. A subset S of $\mathbb{R}^n$ is subanalytic in $\mathbb{R}^n$, if it is locally the image of a relatively compact semianalytic set T under an analytic map (defined on $\bar{T}$). A subset S of $\mathbb{R}^n$ is finitely subanalytic if its image under the map $p : \mathbb{R}^n \longrightarrow [-1, 1]^n$ given by

$$p(x_1, \ldots, x_n) = \left(\frac{x_1}{\sqrt{1 + x_1^2}}, \ldots, \frac{x_n}{\sqrt{1 + x_n^2}} \right)$$

is subanalytic. The finitely subanalytic sets in $\mathbb{R}^n$ are definable in this structure.

Even though polynomial flows are definable in this structure, since the functions $\hat{f}$ are zero outside a compact set, these functions cannot be used to define complete flows. However, the Pre operator corresponding to some periodic flows may still be definable. Consider for example, a hybrid system H whose vector fields are diagonalizable linear vector fields with purely imaginary eigenvalues and all relevant sets are definable in this structure. Since the restriction of sin on $[-\pi, \pi]$ is definable, the Pre operator corresponding to F is definable. This leads to the following corollary of Theorem 5.4, which generalizes to $\mathbb{R}^n$ the planar results in [17, 43, 47].

Corollary 5.5. *Let H be a hybrid system for which all relevant sets (guards, invariants, initial conditions) are finitely subanalytic and all vector fields are diagonalizable linear vector fields with purely imaginary eigenvalues. Let Σ be a finite collection of finitely subanalytic sets. Then the transition system $T_{H,\Sigma}$ has a finite bisimulation quotient.*

$(\mathbb{R}, <, +, -, \cdot, e^x, \{\hat{f}\}, 0, 1)$. This structure, which extends $(\mathbb{R}, <, +, -, \cdot, \{\hat{f}\}, 0, 1)$ by the exponential function, besides enriching the class of definable sets, allows us to capture new classes of definable flows. In particular, the flows of linear vector fields with real eigenvalues are definable. The following corollary is then an immediate consequence of Theorem 5.4.

Corollary 5.6. *Let H be a hybrid system for which all relevant sets are finitely subanalytic and all vector fields are of one of the following two forms:*

- *linear vector fields with real eigenvalues*
- *diagonalizable linear vector fields with purely imaginary eigenvalues*

Let Σ be a finite collection of finitely subanalytic sets. Then the transition system $T_{H,\Sigma}$ has a finite bisimulation quotient.

The above theorem extends the planar results in [43] to $\mathbb{R}^n$. Note that relaxations of Corollary 5.6 would allow spiraling, linear vector fields which are not definable in this structure. As was shown by Example 5.1, such systems, in general, do not admit finite bisimulations. This shows that even though the conditions of Theorem 5.4 are sufficient, they are very tight sufficient conditions.

The above results are existential and show that a finite bisimulations *exist* for the above classes of o-minimal hybrid systems. That means that the bisimulation algorithm will terminate. To show decidability, we must also show that the bisimulation algorithm is computable, which means that there is an effective procedure to compute the Pre operator. This can be achieved for various classes of o-minimal hybrid systems, by posing each step of the bisimulation algorithm as a quantifier elimination problem in the structure $(\mathbb{R}, +, -, \cdot, <, 0, 1)$. The proof then consists of showing that for semi-algebraic sets $A \subset \mathbb{R}^n$, the task of computing the preimage $Pre(A)$ under the flow of such linear systems, reduces to quantifier elimination in $(\mathbb{R}, +, -, \cdot, <, 0, 1)$ by a sequence of definable variable substitutions which eliminate the exponential terms.

Theorem 5.7 (Hybrid Systems with Linear Differential Equations [45])**.** *Consider the class of o-minimal hybrid system H where*

- *for each $\ell \in V$ and edges $e \in E$, the sets $Inv(\ell)$, $Init(\ell)$, and $Guard(e)$ are semialgebraic with rational coefficients, and*
- *for all $\ell \in V$, $F(\ell, x) = A_\ell x$, where $A_\ell \in \mathbb{Q}^{n \times n}$, and either*
  - *A_ℓ is nilpotent; or,*
  - *A_ℓ is diagonalizable and has real, rational eigenvalues; or,*
  - *A_ℓ has purely imaginary eigenvalues $i\omega$, with ω rational, and its real Jordan form is block diagonal with 2×2 blocks.*

Then CTL and LTL model checking for this class of hybrid systems is decidable.

As an immediate consequence, the reachability problem is also decidable for the above classes of hybrid systems. Theorem 5.7 can be extended to include linear hybrid systems where in each discrete state the dynamics are of the form $\dot{x} = Ax + Bu$ for various types of inputs..

Theorem 5.8 (Hybrid Systems with Linear Control Systems [46]). *Consider the class of o-minimal hybrid system H where*

- *for each $\ell \in V$ and edges $e \in E$, the sets $Inv(\ell)$, $Init(\ell)$, and $Guard(e)$ are semialgebraic with rational coefficients, and*
- *for all $\ell \in V$, $F(\ell, x) = A_\ell x + B_\ell u$, where $A_\ell \in \mathbb{Q}^{n\times n}$, $B_\ell \in \mathbb{Q}^{n\times k}$, and either*
  - *A_ℓ is nilpotent, and each entry of u is a polynomial in t; or,*
  - *A_ℓ is diagonalizable, has real rational eigenvalues, and each entry of u is of the form $e^{\mu t}$ with μ rational, and not an eigenvalue of A_ℓ; or,*
  - *A_ℓ has purely imaginary eigenvalues of the form $i\omega$ with ω rational, and the entries in the input u are of the form $\sin(\alpha t)$ or $\cos(\alpha t)$ with α rational, and $\alpha \neq \pm\omega$,*

Then CTL and LTL model checking for this class of hybrid systems H is decidable.

The above results remain valid if the inputs are allowed to be rational, linear combinations of the functions of the corresponding type: exponentials in case of real eigenvalues, and sinusoidal in the case of imaginary eigenvalues. In all cases the same *resonance* restrictions apply on the parameters μ and α.

6. Conclusions

In this paper, we have considered the algorithmic analysis of hybrid systems by the process of abstraction. We have presented a unified collection of results where finite, property preserving abstractions of hybrid systems are possible. Given the known undecidability barriers, we showed that discrete abstractions of hybrid systems are possible when either the continuous or the discrete dynamics are restricted.

In cases where discrete abstractions with *equivalent* properties cannot be constructed, abstractions whose properties are *sufficient* to check can be useful. This approach is taken in [18, 21, 34, 30, 60, 61, 63, 67, 70], where reachable sets of differential equations are over- or under-approximated. This line of work often allows us to verify instances of hybrid systems even if they belong to undecidable classes. The construction of tight approximations along with the tradeoff between complexity and precision is of great importance and should be pursued further. Research along this direction will expand the scope and applicability of computational tools, like Kronos and HyTech. This is

needed before they can be applied on large scale, hybrid systems with complicated discrete and continuous dynamics.

Acknowledgments. The authors would like to thank the reviewers for their detailed comments. This work was supported by the DARPA grant F33615-98-C-3614, the DARPA/NASA grant NAG2-1214, the ARO MURI grant DAAH-04-96-1-0341, the DARPA/ITO MARS program, the NSF CAREER awards CCR-9501708 and CCR97-34115, and a Sloan faculty fellowship.

Appendix A. First-Order Logic

A *language* is a set of symbols separated into three groups: relations, functions and constants. The sets $\mathcal{L}_0 = \{<, +, -, 0, 1\}$, $\mathcal{L}_R = \{<, +, -, \cdot, 0, 1\}$, and $\mathcal{L}_{\exp} = \{<, +, -, \cdot, 0, 1, \exp\}$ are examples of languages where $<$ (less than) is the relation, $+$ (plus), $-$ (minus), $\cdot$ (product) and exp (exponentiation) are the functions, and 0 (zero) and 1 (one) are the constants.

Consider a countable collection of variables $W = \{x, y, z, x_0, x_1, \dots\}$. The set of *terms* of a language is inductively defined as follows. A term θ is a variable, a constant, or $F(\theta_1, \dots, \theta_m)$, where F is a m-ary function and θ_i, $i = 1, \dots, m$ are terms. For instance, $x - 2y + 3$ and $x + yz^2 - 1$ are terms of $\mathcal{L}_0$ and $\mathcal{L}_R$, respectively. In other words, terms of $\mathcal{L}_0$ are linear expressions and terms of $\mathcal{L}_R$ are polynomials with integer coefficients. Notice that integers are the only numbers allowed in expressions (they can be obtained by repeatedly adding the constant 1).

The *atomic formulas* of a language are of the form $\theta_1 = \theta_2$, or $R(\theta_1, \dots, \theta_n)$, where θ_i, $i = 1, \dots, n$ are terms and R is an n-ary relation. For example, $xy > 0$ and $x^2 + 1 = 0$ are atomic formulas of $\mathcal{L}_R$. The set of *(first-order) formulas* is recursively defined as follows. An atomic formula ϕ is a formula, and if ϕ_1 and ϕ_2 are formulas and x is a variable, then $\phi_1 \wedge \phi_2$, $\neg\phi_1$, $\forall x : \phi_1$ or $\exists x : \phi_1$ are formulas. Examples of $\mathcal{L}_R$-formulas are $\forall x\ \forall y : xy > 0$, $\exists x : x^2 - 2 = 0$, and $\exists w : xw^2 + yw + z = 0$. The occurrence of a variable in a formula is *free* if it is not inside the scope of a quantifier; otherwise, it is *bound*. For example, in the formula $\exists w : xw^2 + yw + z = 0$. x, y, and z are free and w is bound. We often write $\phi(x_1, \dots, x_n)$ to indicate that $x_1, \dots, x_n$ are the free variables of the formula ϕ. A *sentence* of $\mathcal{L}_R$ is a formula with no free variables. The formula $\forall x\ \forall y : xy > 0$ is a sentence whereas $\exists w : xw^2 + yw + z = 0$ is not.

A (model-theoretic) *structure* over a set S of a language consists of a non-empty set S and an interpretation of the relations, functions and constants. For example, $(\mathbb{R}, <, +, -, \cdot, 0, 1)$ and $(\mathbb{Q}, <, +, -, \cdot, 0, 1)$, are *structures* of $\mathcal{L}_R$ over $\mathbb{R}$ and $\mathbb{Q}$ respectively, with the usual interpretation of all the symbols. A set $Y \subseteq S^n$ is *definable* if there exists a formula $\phi(x_1, \dots, x_n)$ such that $Y = \{(a_1, \dots, a_n) \in S^n \mid \phi(a_1, \dots, a_n)\}$. For example, over $\mathbb{R}$, the formula $x^2 - 2 = 0$ defines the set $\{\sqrt{2}, -\sqrt{2}\}$. A set is *definable with parameters in C*, if each $c \in C$ is a constant. For example,

$x^2 - \pi = 0$ defines the set $\{\sqrt{\pi}, -\sqrt{\pi}\}$ over $\mathbb{R}$, using π as a parameter. If a language $\mathcal{L}$ is interpreted over $\mathbb{R}$ and $C = \mathbb{R}$, we simply say that a set is definable with parameters (without mentioning C).

References

[1] *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, 43(4), April 1998.

[2] *Automatica, Special Issue on Hybrid Systems*, 35(3), March 1999.

[3] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[4] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

[5] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[6] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee. Modular specification of hybrid systems in CHARON. In N. Lynch and B. H. Krogh, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*. Springer Verlag, 2000.

[7] R. Alur, T.A. Henzinger, and E.D. Sontag, editors. *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[8] R. Alur and R.P. Kurshan. Timing analysis in COSPAN. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 220–231. Springer-Verlag, 1996.

[9] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[10] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[11] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective controller synthesis of switching controllers for linear systems. *Proceedings of the IEEE*. This issue.

[12] A. Balluchi, L. Benvenuti, M. DiBenedetto, C. Pinello, and A. Sangiovanni-Vincentelli. Hybrid control in automotive applications. *Proceedings of the IEEE*. This issue.

[13] E. Bierstone and P.D. Milman. Semianalytic and subanalytic sets. *Inst. Hautes Études Sci. Publ. Math.*, 67:5–42, 1988.

[14] A. Bouajjani, J.-C. Fernandez, and N. Halbwachs. Minimal model generation. In R.P. Kurshan and E.M. Clarke, editors, *CAV 90: Computer-aided Verification*, Lecture Notes in Computer Science 531, pages 197–203. Springer-Verlag, 1990.

[15] M.C. Browne, E.M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.

[16] P.E. Caines and Y.J. Wei. Hierarchical hybrid control systems: A lattice theoretic formulation. *IEEE Transactions on Automatic Control: Special Issue on Hybrid Systems*, 43(4):501–508, April 1998.

[17] K. Cerans and J. Viksna. Deciding reachability for planar multi-polynomial systems. In R. Alur, T. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 389–400. Springer Verlag, Berlin, Germany, 1996.

[18] A. Chutinam and B. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In F. Vaandrager and J. H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*. Springer Verlag, 1999.

[19] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Workshop on Logic of Programs*, Lecture Notes in Computer Science 131, pages 52–71. Springer-Verlag, 1981.

[20] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for the static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual Symposium on Principles of Programming Languages*. ACM Press, 1977.

[21] T. Dang and O. Maler. Reachability analysis via face lifting. In T. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 96–109. Springer Verlag, Berlin, 1998.

[22] J. Davoren. Topologies, continuity, and bisimulations. *Theoretical Informatics and Applications*, 33:357–381, 1999.

[23] J. Davoren and A. Nerode. Logics for hybrid systems. *Proceedings of the IEEE*. This issue.

[24] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer-Verlag, 1996.

[25] R. DeCarlo, M. Branicky, and S. Pettersson. Perspectives and results on the stability of hybrid systems. *Proceedings of the IEEE*. This issue.

[26] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[27] H.B. Enderton. *A mathematical Introduction to Logic*. Academic Press, 1972.

[28] S. Engell, S. Kowalewski, C. Schulz, and O. Stursberg. Simulation, analysis and optimization of continuous-discrete interactions in chemical processing plants. *Proceedings of the IEEE*. This issue.

[29] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors. *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[30] T. Henzinger, P. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):540–554, April 1998.

[31] T. Henzinger and S. Sastry, editors. *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[32] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

[33] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[34] T.A. Henzinger and P.-H. Ho. A note on abstract-interpretation strategies for hybrid automata. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 252–264. Springer-Verlag, 1995.

[35] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, volume 1019 of *Lecture Notes in Computer Science 1019*, pages 41–71. Springer-Verlag, 1995.

[36] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):540–554, 1998.

[37] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.

[38] T.A. Henzinger and R. Majumdar. Symbolic model checking for rectangular hybrid systems. In S. Graf, editor, *TACAS 00: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science. Springer-Verlag, 2000.

[39] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[40] R. Horowitz and P. Varaiya. Design of an automated highway system. *Proceedings of the IEEE*. This issue.

[41] P.C. Kanellakis and S.A. Smolka. CCS expressions, finite-state processes, and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.

[42] X. Koutsoukos, P. Antsaklis, J.Stiver, and M. Lemmon. Supervisory control of hybrid systems. *Proceedings of the IEEE*. This issue.

[43] G. Lafferriere, G. J. Pappas, and S. Sastry. Hybrid systems with finite bisimulations. In P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors, *Hybrid Systems V*, Lecture Notes in Computer Science. Springer Verlag, New York, 1998.

[44] G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, 2000.

[45] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 137–151. Springer Verlag, 1999.

[46] G. Lafferriere, G. J. Pappas, and S. Yovine. Reachability computation for linear hybrid systems. In *Proceedings of the 14th IFAC World Congress*, volume E, pages 7–12, Beijing, P.R. China, July 1999.

[47] G. Lafferriere, G.J. Pappas, and S. Sastry. Subanalytic stratifications and bisimulations. In T. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 205–220. Springer Verlag, Berlin, 1998.

[48] K. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Springer International Journal of Software Tools for Technology Transfer*, 1(1), 1997.

[49] C. Livadas, J. Lygeros, and N. Lynch. High-level modeling and analysis of tcas. *Proceedings of the IEEE*. This issue.

[50] J. Lygeros, D.N. Godbole, and S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Transactions on Automatic Control*, 43(4):522–539, April 1998.

[51] J. Lygeros, G. J. Pappas, and S. Sastry. An approach to the verification of the Center-TRACON Automation System. In T. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 289–304. Springer Verlag, Berlin, 1998.

[52] O. Maler, editor. *Hybrid and Real-Time Systems*, volume 1201 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[53] O. Maler and S. Yovine. Hardware timing verification using KRONOS. In *Proceedings of 7th Conference on Computer-based Systems and Software Engineering*, 1996.

[54] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.

[55] N. H. McClamroch and I. Kolmanovsky. Hybrid controller design for linear and nonlinear systems. *Proceedings of the IEEE*. This issue.

[56] C. Miller and P. Speissegger. Expansions of the real line by open sets: o-minimality and open cores. *Fund. Math.*, 162:193–208, 1999.

[57] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[58] X. Nicolin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid automata. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 149–178. Springer-Verlag, 1993.

[59] A. Ohata, E. Gassenfeit, K. Butts, and B. Krogh. Automotive engine systems: A need for hybrid systems simulation and analysis. *Proceedings of the IEEE*. This issue.

[60] A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In *Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 81–94. Springer-Verlag, July 1994.

[61] G. J. Pappas and S. Sastry. Towards continuous abstractions of dynamical and control systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*, pages 329–341. Springer Verlag, Berlin, Germany, 1997.

[62] G.J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. In *Proceedings of the 37th IEEE Conference in Decision and Control*. Tampa, FL, December 1998.

[63] G.J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Transactions on Automatic Control*, June 2000. To appear.

[64] D. Pepyne and C. Cassandras. Hybrid systems in manufacturing. *Proceedings of the IEEE*. This issue.

[65] A. Pnueli. The temporal logic of programs. In IEEE Computer Society Press, editor, *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[66] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.

[67] A. Puri, V. Borkar, and P. Varaiya. ϵ-approximation of differential inclusions. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, 1996.

[68] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In *Computer Aided Verification*, pages 95–104, 1994.

[69] J. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In M. Dezani-Ciancaglini and U. Montanari, editors, *Fifth International Symposium on Programming*, Lecture Notes in Computer Science 137, pages 337–351. Springer-Verlag, 1981.

[70] J. Raisch and S.D. O'Young. Discrete approximations and supervisory control of continuous systems. *IEEE Transactions on Automatic Control: Special Issue on Hybrid Systems*, 43(4):569–573, April 1998.

[71] M. Song, T-J. Tarn, and N. Xi. Intelligent scheduling, planning and control: Analytical integration of hybrid systems. *Proceedings of the IEEE*. This issue.

[72] A. Tarski. *A decision method for elementary algebra and geometry*. University of California Press, second edition, 1951.

[73] C. Tomlin, J. Lygeros, and S. Sastry. Controller design of hybrid systems. *Proceedings of the IEEE*. This issue.

[74] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A study in muti-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.

[75] F.W. Vaandrager and J.H. van Schuppen, editors. *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[76] D. van Dalen. *Logic and Structure*. Springer-Verlag, third edition, 1994.

[77] L. van den Dries. *Tame Topology and o-minimal structures*. Cambridge University Press, 1998.

[78] L. van den Dries and C. Miller. On the real exponential field with restricted analytic functions. *Israel Journal of Mathematics*, 85:19–56, 1994.

[79] P. Varaiya. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38(2):195–207, 1993.

[80] A. J. Wilkie. Model completeness results for expansions of the ordered field of real numbers by restricted pfaffian functions and the exponential function. *Journal of the American Mathematical Society*, 9(4):1051–1094, Oct 1996.