Reachability Verification for Hybrid Automata

Thomas A. Henzinger[1*] and Vlad Rusu[2**]

[1] EECS Department, University of California, Berkeley, CA
tah@eecs.berkeley.edu
[2] SRI International, Computer Science Laboratory, Menlo Park, CA
rusu@csl.sri.com

Abstract. We study the reachability problem for hybrid automata. Automatic approaches, which attempt to construct the reachable region by symbolic execution, often do not terminate. In these cases, we require the user to guess the reachable region, and we use a theorem prover (Pvs) to verify the guess. We classify hybrid automata according to the theory in which their reachable region can be defined finitely. This is the theory in which the prover needs to operate in order to verify the guess. The approach is interesting, because an appropriate guess can often be deduced by extrapolating from the first few steps of symbolic execution.

Keywords: hybrid automata, reachability verification, theorem proving.

1 Introduction

Hybrid automata are a specification and verification model for hybrid systems [ACH$^+$95], systems that involve mixed continuous and discrete evolutions of variables. The problem that underlies the safety verification for hybrid automata is *reachability*: can an unsafe state be reached from an initial state by executing the system? The traditional approach to reachability attempts to compute the set of reachable states by successive approximation, starting from the set of initial states and repeatedly adding new reachable states. This computation can be automated and is guaranteed to converge in some special cases [KPSY93, AD94, ACH$^+$95, HKPV95, RR96], for which the reachability problem is decidable. In general, however, this approach, which we call *reachability construction*, may not be automatable or may not converge.

It is for this reason that in this paper, we pursue a different approach, called *reachability verification*. In reachability verification, the user guesses the set of reachable states, and then a theorem prover is applied to verify the guess. A guess has the form of a logical formula, which is true exactly for the states that are guessed to be reachable. We classify hybrid automata as to what logical

* This research was supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grant CCR-9504469, by the AFOSR contract F49620-93-1-0056, by the ARO MURI grant DAAH-04-96-1-0341, by the ARPA grant NAG2-892, and by the SRC contract 95-DC-324.036.

** Supported by Lavoisier grant of the French Foreign Affairs Ministry and by SRI.

theory suffices to define the set of reachable states. The formula to be guessed must lie in this theory, and the verification part amounts to a proof in this theory. Hence, the simpler the theory, the more constrained the guess and the easier the verification. In some cases—for example, the case of *additive-inductive* hybrid automata, where the set of reachable states is definable in a decidable subtheory of $(\mathbb{R}, \mathbb{N}, +, \leq)$— the verification part is often completely automatic. The reachability verification approach is interesting because when successive approximation does not converge, a suitable guess can often be found by studying and extrapolating the first few iterations of successive approximation. In this way, some automatic heuristics can be developed to aid the guessing part.

The rest of the paper is organized as follows. In Section 2, we present the hybrid automaton model, the reachability construction method, and the reachability verification method. We restrict our attention to *linear* hybrid automata, for which reachability construction can be automated and has been implemented in verification tools such as HyTech [AHH96]. In Section 3, we classify linear hybrid automata according to the theory in which the set of reachable states is definable. For example, all linear hybrid automata for which reachability construction converges are *polyhedral*, as their reachable region can be defined in $(\mathbb{R}, +, \leq)$. We give examples of linear hybrid automata whose reachable regions are quite simple yet non-polyhedral (e.g., additive-inductive), as well as examples of linear hybrid automata whose reachable regions are quite complex (e.g., most naturally expressed using trigonometric functions). We also present a restricted subclass of additive-inductive automata for which the reachable region can be computed algorithmically, even though reachability construction does not necessarily terminate. Finally, in Section 4 we describe an embedding of hybrid automata into the theorem prover Pvs [ORR+96], and apply the reachability verification method to some well-known examples for which reachability construction fails.

2 Linear Hybrid Automata and Reachability Analysis

Hybrid automata [ACH+95] are finite automata enriched with a finite set of real-valued variables. In each location, the variables evolve continuously according to differential *activities*, as long as the location's *invariant* remains true; then, when a transition *guard* becomes true, the control may proceed to another location, and *reset* some of the variables to new values. We restrict our attention to a simple class of hybrid automata, allowing only straight-line activities and resets of variables to zero. More general feature can be approximated in the simpler framework, with additional locations, transitions, and variables [HHWT98].

Below (Figure 1) is an example of a linear hybrid automaton. It has the three locations s_1, s_2, s_3 and the three variables x, y, z. The automaton starts at location s_1 with variable x set to 0 and variables y, z set to 1, and control can remain at location s_1 while the invariant $x \leq y$ is true. Here, x increases with slope 1 ($\dot{x} = 1$) and y remains constant at 1 ($\dot{y} = 0$). Thus, control can stay at s_1 for at most 1 time unit, until x reaches 1. When this condition becomes true, control leaves s_1 by taking a transition. Here, the only available transition is the one that leads to s_2, which is enabled when $x = y$. Then, control goes to location s_2, where x decreases ($\dot{x} = -1$), and stays there until x reaches 0. When this happens, the transition from s_2 to s_3 is enabled and control goes to s_3, by assigning variable z to 0 in the process. The process continues likewise at location s_3.

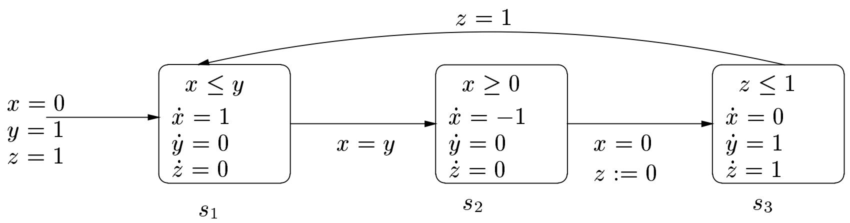

Figure 1. Example of a linear hybrid automaton

Syntax of linear hybrid automata. A convex linear predicate is a system of linear inequalities over given variables. A linear predicate is a finite disjunction of convex linear predicates. A linear hybrid automaton consists of the following elements:

- a finite set $X = \{x_1, x_2, \ldots, x_n\}$ of *variables*;
- a finite set L of *locations*;
- a finite multiset of transitions $T \subseteq L \times L$;
- for each location $l \in L$:
  - an *invariant* $Inv(l)$, which is a convex linear predicate on the variables;
  - an *activity* $Act(l)$, which is a tuple of differentials laws (on law per variable) of the form $\dot{x} = A(l, x)$. Here, $A(l, x)$ is a rational constant, also called the *slope* of variable x at location l;
  - an *initial condition* $Init(l)$, which is a convex linear predicate on the variables;
- for each transition $t \in T$:
  - a *guard* $Guard(t)$, which is a convex linear predicate on the variables;
  - a *reset* $Reset(t)$, which is a set of variables $Reset(t) \subseteq X$.

Semantics. The semantics of hybrid automata builds upon the following preliminary notions. A *valuation* is a function $v : X \to I\!R$ that associates a real number $v(x)$ to each variable $x \in X$. Given a variable valuation v and a linear predicate P over the variables, we say v satisfies P, written $P(v) = true$, if by replacing in P each variable x with its value $v(x)$, one obtains a true statement. In particular, if valuation v satisfies the invariant of location l (respectively, the guard of transition t) we write $Inv(l)(v) = true$ (respectively, $Guard(t)(v) = true$). Given a valuation v and a subset $Y \subseteq X$ of variables, we write $v[Y := 0]$ for the valuation that assigns 0 to all variables in Y, and agrees with v on all variables in $X \setminus Y$. Given a valuation v, a location $l \in L$, and a non-negative real $\tau \in \mathbb{R}^+$, we write $v +_l \tau$ for the valuation that assigns to each variable x in X the value $v(x) + A(l,x) \cdot \tau$, where $A(l,x)$ is the slope of variable x at location l.

The semantic features of a hybrid automaton are the following:

- a *state* is a pair (l, v), where l is a location and v a valuation such that $Inv(l)(v) = true$;
- for a non-negative real $\tau \in \mathbb{R}^+$, there is a *continuous step* of duration τ between two states (l, v) and (l, v') denoted $(l, v) \xrightarrow{\tau} (l, v')$, if $v' = v +_l \tau$;
- for a transition $a = (l, l') \in T$, there is a *discrete step* of label a between two states (l, v) and (l', v') denoted $(l, v) \xrightarrow{a} (l', v')$, if $Guard(t)(v) = true$ and $v' = v[Reset(t) := 0]$;
- a *run* is a finite sequence of continuous and discrete steps $(l_0, v_0) \rightarrow (l_1, v_1) \rightarrow \cdots \rightarrow (l_m, v_m)$ such that the first state (l_0, v_0) is initial; i.e., v_0 satisfies the initial condition $Init(l_0)$.

A state is *reachable* if it coincides with the last state of a run. A *linear region* is a pair $\langle l, P \rangle$, where l is a location and P is a linear predicate on the automaton variables. A state (s, v) *satisfies* the linear region $\langle l, P \rangle$ if $s = l$ and v satisfies P. In this case we also say that the region $\langle l, P \rangle$ *contains* the state (s, v). The *reachability problem* for linear hybrid automata is: given a linear hybrid automaton $\mathcal{A}$ and a set $\mathcal{R}$ of linear regions, is there a reachable state of $\mathcal{A}$ that is contained in some region in $\mathcal{R}$. We discuss below two kinds of approaches to this problem.

Reachability construction [ACH+95]. This method performs a symbolic execution of the hybrid automaton. It consists in successively approximating the reachable region, starting from the initial region, and iterating *successor* operations until the computation converges. There are two kinds of successors.

The *continuous successor* of a region $\langle l, P \rangle$ is the region $\langle l, \vec{P_l} \rangle$ that contains all the states that can be reached from states satisfying $\langle l, P \rangle$, by a single continuous step. The predicate $\vec{P_l}$ is obtained by *extension of P at location l*. Suppose P is a linear predicate over the variables $x_1, \ldots, x_n$, and that variable x_i evolves in location l by the law $\dot{x}_i = k_i$ (for all $i \in \{1, \ldots, n\}$); then, the extension of P at location l is described by the following predicate:

$$\vec{P_l} = \exists \tau \geq 0 . P(x_1 - k_1 \cdot \tau, \ldots, x_n - k_n \cdot \tau) \tag{1}$$

It can be shown that the elimination of the existential quantifier in formula (1) can be performed, and it again produces a linear predicate in variables $x_1, \ldots, x_n$: the continuous successor of a linear region is still a linear region.

The *discrete successor* of a linear region $\langle l, P \rangle$ by a transition $(l, l') \in T$ is the region $\langle l', \vec{P}_{(l,l')} \rangle$ that contains all the states that can be reached from states satisfying $\langle l, P \rangle$, by a single discrete step. The predicate $\vec{P}_{(l,l')}$ is obtained from P by *projection over transition* (l, l'). Suppose that P is a linear predicate over variables $x_1, \ldots, x_n$, and transition (l, l') resets the variables $x_{i_1}, x_{i_2}, \ldots, x_{i_p}$; then, the projection of P over transition (l, l') can be described by the following predicate:

$$\vec{P}_{(l,l')} = (x_{i_1} = 0 \wedge x_{i_2} = 0 \wedge \cdots \wedge x_{i_p} = 0) \wedge \exists x_{i_1}.\exists x_{i_2} \ldots \exists x_{i_p}.P(x_1, \ldots, x_n) \tag{2}$$

It can be shown that the elimination the existential quantifiers in formula (2) can be performed, and it again produces a linear predicate in variables $x_1, \ldots, x_n$. Thus, the discrete successor of a linear region is still a linear region.

Reachability construction consists in iterating the following *Post* procedure:

Input: set A of linear regions.
Output: set B of linear regions, initially empty.
For each linear region $\langle l, P \rangle$ in the set A, for each transition (l, l') with origin l:

- let P_1 be the *intersection* of P with the *guard* of transition (l, l')
- let P_2 be the *projection* of P_1 over transition (l, l')
- let P_3 be the *intersection* of P_2 with the *invariant* of l'
- let P_4 be the *extension* of P_3 at state l'
- let P_5 be the *intersection* of P_4 with the *invariant* of l'
- add $\langle l', P_5 \rangle$ to set B.

We denote by $Post^k(I)$ the set of regions obtained by applying k times the *Post* operation to the set of initial regions $I = \{\langle l, \overrightarrow{Init}(l)_l \wedge Inv(l) \rangle | l \in L\}$, and by $Post^*(I)$ the countably infinite union $\bigvee_{k \in \mathbb{N}} Post^k(I)$. This is also called the *reachable region*, and it represents all the reachable states of the hybrid automaton. Once $Post^*(I)$ is computed, the reachability problem for a set of linear regions $\mathcal{R}$ can be solved by checking if the intersection $Post^*(I) \cap \mathcal{R}$ is non-empty.

We call *reachability construction* the process of computing the sequence I, $Post(I)$, $Post^2(I) \ldots$ of sets of regions. If, for some $k \in \mathbb{N}$, it is the case that $Post^{k+1}(I) \subseteq Post^k(I)$, then reachability construction terminates in finitely many steps, and $Post^*(I) = Post^k(I)$. This does not happen in general for linear hybrid automata [HKPV95]. Some subclasses for which reachability construction terminates have been identified, such as timed automata, initialized rectangular hybrid automata[1],and others [KPSY93, AD94, ACH+95, HKPV95, RR96]. For these classes, the reachability problem is decidable. Reachability construction is the procedure implemented in symbolic model-checking tools like HyTech [AHH96].

[1] For these classes, termination is achieved by slightly modifying the automaton.

Reachability verification. We define a new approach to the reachability problem, called the *reachability verification* method. This method can succeed in cases when reachability construction fails. Reachability verification consists of two steps: first, to *guess* the reachable region; second, to *verify* that the guess is correct. In many cases (some of which are presented in Sections 3 and 4), a suitable guess can be found using the simple heuristic described below, and the verification can be performed by induction, using a theorem prover.

It appears that when reachability construction does not terminate, the reachable region of a hybrid automaton can still behave in a regular manner. As an example, consider the hybrid automaton in Figure 1. By studying the reachability construction over a few iterations (performed in this situation by the tool HyTech), it can be seen that the reachable region is described by the following set of regions:

$$\begin{aligned} &\{\langle s_1\ , \exists i \in \mathbb{N}.(i \geq 1 \land x \leq i \land y = i \land z = 1)\rangle, \\ &\ \langle s_2\ , \exists i \in \mathbb{N}.(i \geq 1 \land x \leq i \land y = i \land z = 1)\rangle, \\ &\ \langle s_3\ , \exists i \in \mathbb{N}.(i \geq 1 \land x = 0 \land y = z + i \land i \leq y \leq i + 1)\rangle\} \end{aligned} \tag{3}$$

The above expression involves a quantifier over a new natural-number variable i. Thus, a simple heuristic to guess the reachable region is to observe a few iterations of reachability construction, and to search for a reachable region of the form $\exists i_1 \in \mathbb{N} \ldots \exists i_q \in \mathbb{N}.\mathcal{R}(i_1, \ldots, i_q)$; that is, the reachable region involves some new natural-number variables $i_1, \ldots, i_q$ in addition to the automaton variables.

A typical situation is to guess a reachable region written using only one natural-number variable j, which represents the number of iterations of the *Post* procedure. In this case, we call the guessed region *directly inductive*, and proving that the guess is correct amounts to prove that for all $j \in \mathbb{N}$, $Post^j(I) = \mathcal{R}(j)$. This can be performed by induction using a theorem prover. In particular, we need to show the two following proof obligations: $\mathcal{R}(0) = \{\vec{Init}(l)_l \land Inv(l) \mid l \in L\}$ for the base step, and $\forall j \in \mathbb{N}.Post(\mathcal{R}(j)) = \mathcal{R}(j+1)$ for the induction step. As we shall see in Section 3, these proof obligations can often be discharged automatically.

In other situations the guessed region may not be directly inductive, but it can be made so by introducing new variables and constraints. For instance, the reachable region defined by expression (3) is not directly inductive, since the natural-number variable i does not represent the number of iterations. But this region becomes directly inductive by adding the constraints $j = 3i$, $j = 3i + 1$, and $j = 3i + 2$ respectively to the three regions in the set (3). That is, we define the sets of regions $\mathcal{R}(j) = \{\langle s_1, \exists i \in \mathbb{N}.(j = 3i \land i \geq 1 \land x \leq i \land y = i \land z = 1)\rangle$, $\langle s_2, \exists i \in \mathbb{N}.(j = 3i + 1 \land i \geq 1 \land x \leq i \land y = i \land z = 1)\rangle, \langle s_3, \exists i \in \mathbb{N}.(j = 3i + 2 \land i \geq 1 \land x = 0 \land y = z + i \land i \leq y \leq i + 1)\rangle\}$ and now the "guess" $\exists j \in \mathbb{N}.\mathcal{R}(j)$ is directly inductive, with j representing the number of iterations.

Finally, even in situations when the guess is not (or cannot be transformed into) directly inductive, a useful approach is to prove that it is an *invariant* of the system. This can often be done automatically and it is often enough in practice for proving safety properties. We present sample proofs in Section 4.

We now give a classification of hybrid automata according to the theory in which their reachable region can be written finitely. The less expressive this theory, the less interactive theorem proving is needed for doing reachability verification.

3 Reachable Region Classification

The first class that we define contains in particular all the hybrid automata for which reachability construction terminates.

Definition 3.1 (polyhedral hybrid automata). A linear hybrid automaton is *polyhedral* if its reachable region can be expressed as a set of pairs $\{\langle l, P_l\rangle \mid \in L\}$ such that for each location $l \in L$, P_l is a formula of the theory[2] $(\mathbb{R}, +, \leq)$. □

We say a linear hybrid automaton is *finitely constructible* if its reachability construction terminates: i.e., for some $k \in \mathbb{N}$, $Post^*(I) = Post^k(I)$. While all finitely constructible hybrid automata are polyhedral, the converse is not true: it is easy to construct a hybrid automaton such that for all $k \in \mathbb{N}$, $Post^k(I)$ is the closed interval $[0, k]$; thus, the reachable region $Post^*(I)$ is the interval $[0, \infty)$, but reachability construction does not converge in finitely many steps. The class of finitely constructible hybrid automata includes the timed automata [AD94] and the initialized rectangular hybrid automata [HKPV95] (with some minor modifications to force the reachability construction to terminate) as well as some other restricted classes [KPSY93, RR96].

Definition 3.2 (additive-inductive hybrid automata). A linear hybrid automaton is *additive-inductive* if its reachable region can be expressed as a set of pairs $\{\langle l, P_l\rangle \mid l \in L\}$ such that for each location $l \in L$, P_l is a formula of the theory $(\mathbb{R}, \mathbb{N}, +, \leq)$ in which *all natural-number variables are outermost existentially quantified.* □

For instance, the hybrid automaton in Figure 1 is additive-inductive: we have seen that its reachable region (3) involves the real variables x, y, z and the natural-number variable i, which is outermost existentially quantified.

Proposition 3.3. *The class of polyhedral hybrid automata is strictly included in the class of additive-inductive hybrid automata.*

Proof. The inclusion is obvious (since any formula of $(\mathbb{R}, +, \leq)$ is also a formula of $(\mathbb{R}, \mathbb{N}, +, \leq)$). Let us show that the inclusion is strict. For this, consider the hybrid automaton in Figure 1. We have seen that it is additive-inductive, and let us suppose it is finitely constructible, thus polyhedral by a previous observation. Then, formula (4) $\exists i \in \mathbb{N}.(i \geq 1 \wedge x \leq i \wedge y = i \wedge z = 1)$ can be also expressed in the theory $(\mathbb{R}, +, \leq)$; that is, the set of triples (x, y, z) satisfying (4) constitute a finite union of convex polyhedra $P_1, \ldots, P_N$ in $\mathbb{R}^3$. Since (4) is the countably

[2] Whenever we define a logical theory, we allow (unless explicitly restricted) arbitrary first-order quantification and boolean connectives.

infinite union $\bigvee_{i \in \mathbb{N}} (i \geq 1 \wedge x \leq i \wedge y = i \wedge z = 1)$, it follows that at least one of the convex polyhedra P_j coincides with the union of infinitely many polyhedra of the form (5) $(x \leq i \wedge y = i \wedge z = 1)$. This is not possible, because the union of polyhedra of the form (5) is not convex (they are all disjoint). $\square$

Suppose the user can guess a reachable region like in Definition 3.2 (using the simple heuristic of extrapolating from the first few reachability steps) and that furthermore the guess is *directly inductive* (cf. end of Section 2). Then, verifying that the guess is correct can be done by induction in a completely automatic manner. Indeed, both the base and the inductive steps of the proof require computing the extension and projection operations (cf. equations (1), (2) of Section 2) for formulas of the theory $(\mathbb{R}, \mathbb{N}, +, \leq)$. This amounts to proving finitely many implications of the form $\forall \mathbf{x} \in \mathbb{R}^n . \forall \mathbf{i} \in \mathbb{N}^m . \exists y \in \mathbb{R} . \varphi(\mathbf{x}, \mathbf{i}, y) \Rightarrow \psi(\mathbf{x}, \mathbf{i}, y)$. Proving such an implication can be done automatically, by eliminating the existential quantifiers on the real variables using the Fourier-Motzkin algorithm [Zie95] (transforming the universal quantifiers into existential ones by taking the negation of the formula whenever necessary). At the end we are left with a formula of Presburger arithmetic, which is decidable.

In the situation where the guessed region is not directly inductive, one can still attempt make it directly inductive as indicated in Section 2, by introducing new variables (one of which represents the iteration number) and new constraints connecting the existing and the new variables. Finally, even when a guess is not directly inductive, it can be useful (as an invariant of the system) to prove safety properties. We demonstrate these approaches in Section 4 on some well-known examples.

We now define a class of linear hybrid automata whose reachable region can be defined in terms of natural and real numbers, using addition *and* multiplication. Consider the theory $(\mathbb{R}, \mathbb{N}, +, \cdot_{\mathbb{N} \times \mathbb{N}}, \cdot_{\mathbb{N} \times \mathbb{R}}, \leq)$ of reals and naturals with multiplication between naturals $\cdot_{\mathbb{N} \times \mathbb{N}}$, multiplication between naturals and reals $\cdot_{\mathbb{N} \times \mathbb{R}}$, and inequality. Any formula in this theory is a boolean combination of *linear* inequalities in the real variables, with *polynomial* coefficients in the natural-number variables; for example, $(n^3 - 1) \cdot x + m \cdot y + n \geq 0$, where x, y are real variables and m, n are natural-number variables.

Definition 3.4 (multiplicative-inductive hybrid automata). A linear hybrid automaton is *multiplicative-inductive* if its reachable region can be expressed as a set of pairs $\{\langle l, P_l \rangle \mid l \in L\}$ such that for all location $l \in L$, P_l is a formula of the theory $(\mathbb{R}, \mathbb{N}, +, \cdot_{\mathbb{N} \times \mathbb{N}}, \cdot_{\mathbb{N} \times \mathbb{R}}, \leq)$ with *all the natural-number variables outermost existentially quantified.* $\square$

The linear hybrid automaton[3] in Figure 2 is multiplicative inductive: it can be shown easily that the reachable region at location s_1 is defined by the formula (6) $\exists n \in \mathbb{N} . (n \geq 1 \wedge x = 1 \wedge n \cdot y = 1 \wedge v = 0 \wedge u = 0)$, where x, y, u, v are real variables, and n is a natural-number variable.

[3] In Figure 2, activities $\dot{x} = \dot{y} = \dot{u} = \dot{v} = 0$ at all locations are not represented.

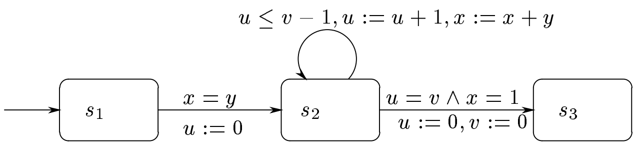

Figure 2. Multiplicative-inductive hybrid automaton

Proposition 3.5. *The class of additive-inductive hybrid automata is strictly included in the class of multiplicative-inductive hybrid automata.*

Proof. The proof of this proposition is based on the following observations. Given two predicates φ and ψ on the real variables $x_1, \ldots, x_n$, we identify φ and ψ with the sets of points in $\mathbb{R}^n$ that they respectively define. We define the *maximal distance* $\Delta(\varphi, \psi)$ between φ and ψ as follows: if φ or ψ are empty then $\Delta(\varphi, \psi)$ is a special value $\perp$; otherwise, $\Delta(\varphi, \psi)$ is the lowest upper bound of the set of distances in $\mathbb{R}^n$ between a point satisfying φ and a point satisfying ψ.

Consider now an additive-inductive hybrid automaton, a location l of the automaton, and the formula $\exists i_1 \in \mathbb{N} \ldots \exists i_q \in \mathbb{N}.\varphi(x_1, \ldots x_n, i_1 \ldots i_q)$ that defines the reachable region of the automaton at location l. Without restricting the generality, it is possible to suppose that formula φ is a convex linear predicate in variables $x_1, \ldots, x_n, i_1, \ldots, i_q$. We define a sequence $(\varphi_m)_{m \geq 1}$ of linear predicates by the relation $\varphi_m(x_1, \ldots, x_n) = \varphi(x_1, \ldots x_n, m \ldots, m)$; i.e., the sequence of predicates $(\varphi_m)_{m \geq 1}$ is obtained by replacing in formula φ all integer variables by the value m. Thus, any predicate in the sequence $(\varphi_m)_{m \geq 1}$ is a convex linear predicate on $x_1, \ldots, x_n$; that is, any predicate φ_m is a convex polyhedron in $\mathbb{R}^n$.

We now define the sequence $(\Delta_m)_{m \geq 1}$ by $\Delta_m = \Delta(\varphi_m, \varphi_{m+1})$ for all $m \geq 1$. We show that the sequence $(\Delta_m)_{m \geq 1}$ can behave in one of three possible manners. In the first case, there are infinitely many polyhedra φ_m that are empty and thus for infinitely many $m \geq 1$, $\Delta_m = \perp$. Otherwise, there exists an index $M \geq 1$ such that for all $m \geq M$, all polyhedra φ_m are non-empty. Then it can be shown that for all $m \geq M$, each vertex of φ_{m+1} is obtained from some vertex of φ_m by translation by some constant vector $w \in \mathbb{R}^n$. The vector w depends on the vertex but not on the index m. If all such vectors w are 0, then we have the second case: for all $m \geq M$, the polyhedra φ_m are equal, and hence the sequence $(\Delta_m)_{m \geq M}$ is constant. Otherwise, at least one vector w is not 0 and we have the third case: for all $m \geq M$, $\Delta_m \geq |w| > 0$ (where $|w|$ denotes the length of vector w).

Consider now the hybrid automaton in Figure 2 and suppose that it is additive-inductive. We have seen that the formula (6) $\exists i \in \mathbb{N}.(i \geq 1 \wedge x = 1 \wedge i \cdot y = 1 \wedge v = 0 \wedge u = 0)$ represents the reachable region of this hybrid automaton at location s_3. We apply the previous constructions: we obtain the sequence of predicates $\varphi_m = (x = 1 \wedge m \cdot y = 1 \wedge v = 0 \wedge u = 0)$ and the sequence of distances $\Delta_m = 1/m(m+1)$, for all $m \geq 1$. The last sequence is strictly decreasing and converges to 0. But we have seen that this cannot be the case for a sequence $(\Delta_m)_{m \geq 1}$ obtained (as described above) from the reachable region of an additive-inductive hybrid automaton. Hence, the multiplicative-inductive hybrid automaton in Figure 2 is not additive-inductive. $\square$

Reachability verification can still be applied to multiplicative-inductive hybrid automata, provided the user guesses the reachable region. For instance, consider the hybrid automaton in Figure 2, whose initial region I is defined by location s_1. We apply reachability verification: we guess the reachable region at location s_3 to be formula (6) above (using the heuristic of observing the first steps of reachability construction). This guess is furthermore *directly inductive* (cf. end of Section 2): to prove that the guess is correct, we show by induction that for all $k \geq 1$, the region $Post^k(I)$ at location s_3 is described by the formula $(x = 1 \wedge k \cdot y = 1 \wedge v = 0 \wedge u = 0)$. However, unlike the case of additive-inductive hybrid automata, this proof can only be partially automated. Indeed, the extension and projection operations (equations (1), (2) of Section 2) can be computed automatically for predicates in $(\mathbb{R}, \mathbb{N}, +, \cdot_{\mathbb{N}\times\mathbb{N}}, \cdot_{\mathbb{N}\times\mathbb{R}} \leq)$: these operations require eliminating the existential quantifiers on the real variables, which can be done using a generalization of the Fourier-Motzkin algorithm [BR97]. But after the quantifier elimination, we are left to decide a first-order formula of the (undecidable) theory $(\mathbb{N}, +, \cdot, \leq)$. This last formula has to be dealt with by theorem proving. So, the verification process is more involved than in the case of additive-inductive hybrid automata.

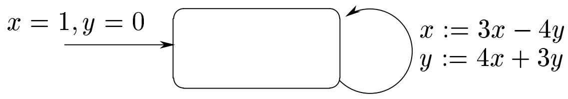

Figure 3. Hybrid automaton with exponential/trigonometric reachable region

While the theory of natural numbers with addition, multiplication and order is extremely expressive for encoding purposes, there exist linear hybrid automata whose reachable regions are most naturally expressed in terms of other operations, like exponentials and trigonometric functions. Consider the hybrid automaton in Figure 3. The transition sets the variables to new values[4] that we denote x', y'. Let $\theta \in \mathbb{R}$ be such that $5\cos\theta = 3$. Then, we have $x' = 5(x\cos\theta - y\sin\theta), y' = 5(x\sin\theta + y\cos\theta)$. Interpreted as a vector operation, the previous relations just say that vector $[x', y']$ has a length 5 times greater than vector $[x, y]$, and that $[x', y']$ is rotated by angle θ from $[x, y]$. Thus, the reachable region is defined by formula $\exists n \in \mathbb{N}.\exists\theta \in \mathbb{R}.(x = 5^n \cos n\theta \wedge y = 5^n \sin n\theta \wedge 5\cos\theta = 3)$. This would suggest that reachable regions need quite expressive theories in order to be expressed finitely. However, it is easy to show that the previous region can be encoded in the first-order theory of integers with multiplication: let $code(x, y)$ be an encoding function of pairs of integers as natural numbers, and consider the natural numbers of the form (7): $2^{code(x_1,y_1)} \cdot 3^{code(x_2,y_2)} \cdot \ldots \cdot p_n^{code(x_n,y_n)}$. Here, p_n is the n-th prime number, and x_n, y_n are the terms of the sequence defined by $x_1 = 1, y_1 = 0$, and the transition relation of the automaton. Clearly, the fact that (x_n, y_n) is in the reachable region is encoded by the existence of natural numbers of the form (7), which can be described in the theory $(\mathbb{N}, +, \cdot, \leq)$.

[4] The linear assignments can be simulated by appropriate slopes, tests, and resets.

Finally, we mention a restricted subclass of linear hybrid automata for which the reachable region can be computed algorithmically, even though reachability construction does not necessarily terminate. Some well-known examples of hybrid automata (like the ones we discuss in Section 4) are in this class. We say a hybrid automaton is *time-predictable* if for each location l' and each pair of transitions (l, l') and (l, l'') with destination (resp. with origin) l, there exists an interval of $\mathbb{R}^+$ such that transition (s', s'') can be fired at any moment within the given interval, after the firing of transition (s, s'). We say a hybrid automaton is *without nested cycles* if its graph is equivalent to a regular expression (on the transition names) without nested $*$ operations. We have proved[5] that time-predictable hybrid automata without nested cycles are additive-inductive but not polyhedral (cf. Definitions 3.1, 3.2), and that their reachable region can be computed algorithmically, by a procedure different from reachability construction. This shows that there exist hybrid automata for which the reachability problem is decidable, even though reachability construction does not terminate.

4 Hybrid Automata in PVS

We outline the modeling of hybrid automata and reachability verification in PVS [ORR+96]. First we specify a theory `polyhedra[n]` of n-dimensional polyhedra (parametric in the dimension $n \in \mathbb{N}$). It contains essentially the definitions of extension, projection (formulas (1), (2) of Section 2), and intersection operations on polyhedra. Writing such first-order predicates in the higher-order PVS specification language is straightforward. Then, we write another theory that is specific to the particular hybrid automaton to be analyzed (containing the definition of the automaton features: states, transitions, activities, invariants, guards, and resets). This second theory uses (imports) the theory `polyhedra[n]`, instantiating n with the number of variables of the hybrid automaton. Finally, in a third theory called `symbolic-analysis` we specify the types and operations of reachability analysis (independent of any particular hybrid automaton): the `region` type (record of state and polyhedron), the continuous and discrete successors of a region, and a `post` predicate on regions, according to the definition of the *Post* operation (cf. Section 2):

```
region : TYPE = [# thestate: state, thepoly: poly #]

continuous(r1:region) : region =
      (#
         thestate:= thestate(r1),
         thepoly:= intersection(extend(thepoly(r1),
                   slope(thestate(r1))),invar(thestate(r1)))
       #)
```

[5] The proof is not presented here due to lack of space.

```
discrete (r1:region, t:trans) : region =
      (#
        thestate := dest(t),
        thepoly:= intersection(project(reset(t),
                              intersection(thepoly(r1),guard(t))),
                                           invar(dest(t)))
      #)

post(R1,R2:setof[region]) : bool =
                            FORALL (r2:region): member (r2,R2)
                            IMPLIES EXISTS(r1:region,t:trans):
                            member(r1,R1) AND orig(t)=thestate(r1)
                            AND r2 = continuous(discrete(r1,t))
```

To prove statements about the reachable region $Post^*(I)$, we use induction and the predicate `post`. We now describe the application of reachability verification to examples of hybrid systems modeled by additive-inductive hybrid automata.

The leaking gas burner. The hybrid automaton in Figure 4 models a leaking gas burner [CHR91]: location s_1 (resp. s_2) stands for the leaking (resp. the non-leaking) state of the system; variable x is used to control the time spent in each state, variable y is a global clock, and variable z measures the total time spent by the gas burner in the leaking state. A design requirement for the leaking

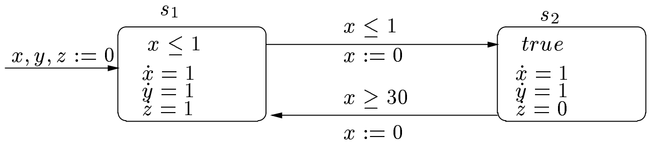

Figure 4. Leaking gas burner automaton

gas burner is that in any interval of time of at least 60 seconds, the leaking time does not exceed 5% of the total time. This can be expressed by the fact that linear predicate $y \geq 60 \Rightarrow 20z \leq y$ is an invariant of the system (i.e., true in all reachable states). The specification in PVS of this example includes the theories `polyhedra[n]` with n instantiated by 3 (the number of variables of the automaton), and `symbolic-analysis` for the reachability analysis of the system. The system itself (hybrid automaton in Figure 4) is specified in a theory `leaking-gas-burner`, that contains the description of the automaton: locations with invariants and differential laws, and transitions called `s1_to_s2` and `s2_to_s1`, with their guards and variables to reset. The reachability construction does not terminate[6] but by studying the first few iterations, one can guess

[6] Although *backwards* reachability construction terminates in this case.

that the reachable region is described by the following set of linear regions (from which it can be seen that the hybrid automaton is additive-inductive):

$$\{\langle s_1, 0 \le x \le 1 \wedge x = y = z \vee \exists i \in I\!N.(i \ge 1 \wedge 0 \le x \le 1 \wedge 0 \le z - x \le i \wedge 30i + z \le y)\rangle, \\ \langle s_2, 0 \le z \le 1 \wedge y = x + z \wedge x \ge 0 \vee \exists i \in I\!N.(i \ge 1 \wedge 0 \le x \wedge 0 \le z \le i + 1 \wedge 30i + x + z \le y)\rangle\}$$

However, this guess is not directly inductive (cf. end of Section 2) because the natural-number variable i does not represent the number of iterations. It is possible to make the guess directly inductive, by introducing a new natural-number variable j and two new constraints $j = 2i$, $j = 2i + 1$. More precisely, we define the sets of regions $\mathcal{R}(j)$ such that for all $j \ge 2$, $\mathcal{R}(j)$ is equal to:

$$\{\langle s_1, \exists i \in I\!N.(i \ge 1 \wedge j = 2i \wedge 0 \le x \le 1 \wedge 0 \le z - x \le i \wedge 30i + z \le y)\rangle, \\ \langle s_2, \exists i \in I\!N.(i \ge 1 \wedge j = 2i + 1 \wedge 0 \le x \wedge 0 \le z \le i + 1 \wedge 30i + x + z \le y)\rangle\}$$

Furthermore, $\mathcal{R}(0)$ equals $\{\langle s_1, 0 \le x \le 1 \wedge x = y = z\rangle, \langle s_2, false\rangle\}$ and $\mathcal{R}(1)$ equals $\{\langle s_1, false\rangle, \langle s_2, 0 \le z \le 1 \wedge y = x + z \wedge x \ge 0\rangle\}$. Now, the new "guess" $\exists j \in I\!N.\mathcal{R}(j)$ is directly inductive (with j representing the number of iterations). We prove by induction on j that $Post^j(I) = \mathcal{R}(j)$, for all $j \in I\!N$. This means that $Post^*(I) = \exists j \in I\!N.\mathcal{R}(j)$; i.e., the guess of the reachable region is correct.

Finally, to prove the design requirement of the gas burner $y \ge 60 \Rightarrow 20z \le y$, we prove that it is implied by $Post^*(I)$. Except for some details (like the expansions of the definitions for `continuous`, `discrete`, `post` etc), PVS can do all the proofs automatically, using its built-in decision procedures.

The reactor temperature controller. This example is taken from [JLHM91]. It is a variant of the nuclear reactor temperature control problem, in which non-linear evolutions are approximated by piecewise-linear functions [HHWT98]. The reactor automaton (cf. Figure 5) has three locations: in the *no_rod* location,

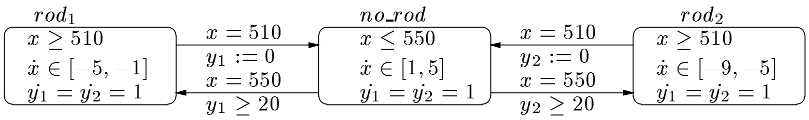

Figure 5. Reactor temperature control automaton

the temperature x increases according to the law $\dot{x} \in [1, 5]$, and control can stay in location *no_rod* as long as the temperature does not exceed 550. When the temperature reaches 550, the reactor uses one of two cooling rods, and the control goes to a location where temperature decreases, according to law $\dot{x} \in [-5, -1]$ or $\dot{x} \in [-9, -5]$, depending on the cooling rod that is used. When the temperature falls to 510, the rod is removed and the reactor goes back to the *no_rod* location. After a rod has been used, it cannot be used again before 20 time units. This is specified using two clocks y_1 and y_2: when the control leaves the location rod_i (that is, rod i is removed from the reactor) the clock variable y_i is reset, and the next entry to location rod_i is guarded by the condition $y_i \ge 20$. A design requirement for the temperature control system is that the temperature never reaches the upper limit ($x = 550$) in the *no_rod* location of the automaton with both rods unavailable ($y_1 < 20$ and $y_2 < 20$). The reachability construction from the initial region (location *no_rod*, variables $x = 510$, $y_1 = y_2 = 20$) does not terminate. However, the reachable region behaves in a regular manner; by studying the output of the model checker HyTech, it can be guessed that the reachable region for location *no_rod* (the location that interests us) has the form:

$$\begin{array}{l}(x \leq 550) \wedge [(y_1 = y_2 \wedge x \geq y_1 + 490 \wedge x \leq 5y_1 + 410) \vee \\ \exists i \in I\!N.(x \geq y_1 + 510 \wedge x \leq 5y_1 + 510 \wedge y_2 \geq y_1 + 36 + 28i \wedge y_2 \leq y_1 + 100 + 80i) \vee \\ \exists i \in I\!N.(x \geq y_1 + 510 \wedge x \leq 5y_1 + 510 \wedge y_2 \geq y_1 + 16 + 28i \wedge y_2 \leq y_1 + 80(1+i)) \vee \\ \exists i \in I\!N.(x \geq y_2 + 510 \wedge x \leq 5y_2 + 510 \wedge 9y_1 \geq 9y_2 + 112 + 220i \wedge y_1 \leq y_2 + 48(i+2)) \vee \\ \exists i \in I\!N.(x \geq y_2 + 510 \wedge x \leq 5y_2 + 510 \wedge 9y_1 \geq 9y_2 + 292 + 220i \wedge y_1 \leq y_2 + 68 + 48i)].\end{array}$$

We prove in PVS that the above predicate is an invariant at location *no_rod* of the automaton. For this, we show that our guess $\mathcal{R}$ satisfies $I \subseteq R$ and $Post(\mathcal{R}) \subseteq \mathcal{R}$. This is enough for proving the design requirement: indeed, the above predicate implies the negation of the 'dangerous' region $x = 550 \wedge y_1 < 20 \wedge y_2 < 20$, so the design requirement is met. Except for details like definition expansion, these proofs are completely automatic in PVS.

5 Conclusion

We have presented a new approach to the reachability problem of hybrid automata. The idea is to guess the form of the reachable region and to use theorem proving for verifying that the guess is correct. We have classified hybrid automata according to the theory in which their reachable region can be written finitely. In this classification, we have identified the *additive-inductive* and *multiplicative-inductive* hybrid automata, for which the guess can be done using a simple heuristic and the verification by induction. We have presented some applications using the prover PVS. In the future, we plan to automate the method as much as possible (including automated guess heuristics and adapted strategies for the PVS proofs) for being able to cope with larger examples.

Related work. [BW94] exploit the regularity of cycles on a discrete model (automata with counters). Their approach is fully automatic but it is limited to linear operations on the variables that are idempotent. [BBR97] present a similar approach for a restricted class of hybrid automata (there is a fixed interval of time between transitions), but their method is fully automatic. Abstract interpretation of hybrid automata [HPR94] would automatically recognize the regularities of polyhedra and detect an invariant which, in general, is only an over-approximation of the actually reachable states. Finally, [VH96] describe an approach based on stepwise refinement for the verification of hybrid systems, where PVS is used to prove the correctness of each refinement step.

Acknowledgments. Thanks to Natarajan Shankar, Luca de Alfaro, Peter Habermehl, and the anonymous reviewers of the Hybrid Systems workshop for useful comments and suggestions.

References

ACH+95. 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.

AD94. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

AHH96. R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

BBR97. B. Boigelot, L. Bronne, and S. Rassart. An improved reachability analysis method for strongly linear hybrid systems. In *Proc. of the 9th Conference on Computer-Aided Verification, CAV'97*, LNCS 1254, pages 167–178. Springer-Verlag, 1997.

BR97. A. Burgueño and V. Rusu. Task-system analysis using slope-parametric hybrid automata. In *Proc. of the 3rd Conference on Parallel Processing, Euro-Par'97*, LNCS 1300, pages 1262–1273. Springer-Verlag, 1997.

BW94. B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Proc. of the 6th Conference on Computer-Aided Verification, CAV'94*, LNCS 818, pages 55–67. Springer-Verlag, 1994.

CHR91. Z. Chaochen, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40:269–276, 1991.

HHWT98. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 1998. To appear.

HKPV95. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proc. of the 27th Annual ACM Symposium on Theory of Computing, STOC'95*, pages 373–382, 1995.

HPR94. N. Halbwachs, Y.-E. Proy, and P. Raymond. Verification of linear hybrid systems by means of convex approximations. In *Proc. of the 1st Static Analysis Symposium, SAS'94*, LNCS 864, pages 223–237. Springer-Verlag, 1994.

JLHM91. M. Jaffe, N. Levenson, M. Heimdahl, and B. Melhart. Software requirements analysis for real-time process-control systems. *IEEE Transactions on Software Engineering*, 17(3):241–258, 1991.

KPSY93. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In *Proc. of the 1st Workshop on Theory of Hybrid Systems*, LNCS 736, pages 179–208. Springer-Verlag, 1993.

ORR+96. S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. Pvs: Combining specification, proof checking, and model checking. In *Proc. of the 8th Conference on Computer-Aided Verification, CAV '96*, LNCS 1102, pages 411–414. Springer-Verlag, 1996.

RR96. O. Roux and V. Rusu. Uniformity for the decidability of hybrid automata. In *Proc. of the 3rd Static Analysis Symposium, SAS'96*, LNCS 1145, pages 301–316. Springer-Verlag, 1996.

VH96. Jan Vitt and Josef Hooman. Assertional specification and verification using Pvs of the steam boiler control system. In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, LNCS 1165, pages 453–472. Springer-Verlag, 1996.

Zie95. G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.