Decidability of hybrid systems with linear and nonlinear differential inclusions [*]

M. Broucke and P. Varaiya
Department of Electrical Engineering and Computer Science
University of California, Berkeley CA 94720
mire, varaiya@eclair.eecs.berkeley.edu

January 21, 1997

1 Introduction

Hybrid systems are dynamic systems consisting of a two-level architecture in which the upper level is a finite automaton and the lower level is a continuous dynamical system. Models and architectures for hybrid systems have been proposed by a number of research groups [11, 12, 4, 13].

Applications for the hybrid modeling approach arise in control of systems that cannot easily be controlled through a single conventional control law, including switched systems; large scale systems and systems with multiple communicating control agents; systems with multiple control objectives; and systems modeling the interaction of (digital) computer processes and continuous plant dynamics. A critical area of investigation is analytical and algorithmic tools for the verification of qualitative properties of the trajectories of hybrid systems. The decidability problem is to determine for a given class of hybrid systems whether a property such as safety or fairness is verifiable by an algorithm that terminates in a finite number of steps.

Early results on decidability for hybrid systems were for restricted classes of models, beginning with the result in [2, 3] for timed automata that the reachability problem is PSPACE-hard. This result was extended to initialized multi-rate timed automata (where the value of a clock is initialized whenever the dynamics for that clock change) by a state transformation [4, 5]. [1] showed that hybrid automata with rectangular differential inclusions are equivalent to multi-rate automata. [15, 6] showed that a hybrid automaton with rectangular differential inclusions could more directly be transformed to a multi-rate automaton, so that the reachability problem is decidable. Kesten, et. al. [10] introduced a class of hybrid automata called integration graphs in which clocks that can change dynamics between lo-

[*] This research is supported by the California Department of Transportation through the California PATH program.

cations are not tested within loops. It was shown that the reachability problem is decidable for the case of a single timer and the case of a single test involving integrators.

Undecidability results are built around transformations of classes of automata to an automaton that can replicate nondeterministic 2-counter machines. It is known that the halting problem for these systems in undecidable. For example, a two-rate timed system that is not initialized can model a nondeterministic 2-counter machine. The main results on undecidability are summarized in the paper [6].

A more detailed account of results on decidability and undecidability can be found in the review articles [15, 6, 8, 9].

In this paper we extend the decidability result of [1]. The techniques and results presented are close to those of [7]; here we focus on the problem of hybrid systems with inclusion dynamics. We consider the following class of hybrid systems. The continuous dynamics are governed by linear differential inclusions of the form $\dot{x} \in [L, U]x$ where L and U are diagonal $n \times n$ matrices. The clock values are compared to constants in the guards, and the assignments of clocks during a transition are to constant, possibly nondeterministic values. The second class we consider consists of hybrid automata with nonlinear differential inclusions where the ith clock component follows an inclusion of the form $\dot{x}_i \in [l_i, u_i]f_i(x_i)$.

Modeling the continuous dynamics using inclusions is well-based. The inclusion can be thought of as an abstraction of behaviors resulting from a continuous control system. If, for example, the control is a regulator it may not be necessary to model the continuous dynamics in detail in a verification of the combined discrete-continuous system: the behavior of the continuous system has been made predictable by control. The inclusion approach follows with the philosophy of hybrid modeling that allows for non-unique solutions from one initial condition. The loosening of precision in defining the behavior of the system is accomodated by the verification tools for non-deterministic finite automata.

The paper is organized as follows. We first review some terminology and definitions needed for the rest of the paper. The third section develops the main results for decidability of the reach set problem for hybrid systems with linear differential inclusions, robustness of the reach set calculation, and decidability of the reach set problem for hybrid systems with nonlinear differential inclusions.

2 Preliminaries

Notation $(\mathbf{R}^+)$ $\mathbf{R}$ is the set of (non-negative) reals and $(\mathbf{Z}^+)$ $\mathbf{Z}$ is the set of (non-negative) integers. For $x \in \mathbf{R}^n$, we write x_i for the ith component of x. $\overline{\sigma}$ refers to a finite sequence of events $\sigma(i) \in \Sigma$ where Σ is a set of events. $s \xrightarrow{\overline{\sigma}} t$ means there is an event sequence $\overline{\sigma}$ such that final state t can be reached from initial state s. If ψ is a set of transition relations, then we write ψ^* for the transitive closure of ψ. Also, denote $s \xrightarrow{\overline{\sigma}} t$ by $(s, \overline{\sigma}, t) \in \psi^*$.

Finite automata A finite automaton is a system $A = (L, \Sigma, \psi, I)$ where L is a finite set of discrete locations, Σ is a finite set of events, $I \subseteq L$ is a set of initial locations, and $\psi \subset L \times \Sigma \times L$ is a set of transition relations. Associated with each transition between locations l and l' is an event $\sigma \in \Sigma$.

The semantics of automaton A are defined on sequences of events $\sigma \in \Sigma$. If Σ^* is the set of all finite sequences of events, then we say the *language* of a finite automata is the set of sequences over Σ accepted by A:

$$\mathcal{L}(A) = \{\overline{\sigma} \in \Sigma^* \mid (l, \overline{\sigma}, l') \in \psi^*\} \ .$$

A *run* of a finite automata A over a sequence $\overline{\sigma} \in \Sigma^*$ is a sequence of locations visited by the automaton as it reads $\overline{\sigma}$

$$l(1) \xrightarrow{\sigma(1)} l(2) \xrightarrow{\sigma(2)} l(3) \xrightarrow{\sigma(3)} \cdots l(m-1) \xrightarrow{\sigma(m-1)} l(m) \ .$$

Timed automata A timed automaton is a system $T = (Q, \Sigma, \psi, I)$ where $Q = L \times \mathbf{R}^n$ and $\psi \subset L \times \Sigma \times L \times G \times J$. Associated with each location is a set of n clocks with valuations, $x \in \mathbf{R}^n$. The state of a timed automaton is $q = (l, x) \in Q$. $I \subseteq Q$ is a set of initial locations and clock valuations. The clocks record the elapsed time; that is, $\dot{x}_i = 1$. Clock values can be reset upon taking a transition whenever an enabling condition on that transition is satisfied. Thus, each edge of the automaton has an enabling condition or guard, $\delta \in G$, where G is the set of all enabling conditions. An edge can reset clock values by a reset condition or jump, $\lambda \in J$, where J is the set of all reset conditions. Edges are labeled $(q, \sigma, q', \delta, \lambda)$ where q and q' are the originating and final locations and clock valuations, respectively.

Enabling conditions are formulas generated by the grammar:

$$\delta := x_i \leq c_i | x_i \geq c_i | x_i < c_i | x_i > c_i | \delta_1 \wedge \delta_2 | \delta_1 \vee \delta_2 \quad \text{where} \quad c \in \mathbf{Z}$$

The reset condition $\lambda \in J$ initializes components of x when a transition is taken. The reset $\lambda_i = [a_i, b_i]$ initializes x_i nondeterministically to a value between a_i and b_i, where $a_i, b_i \in \mathbf{Z}$. When a clock is not reset $\lambda_i = id$.

The semantics of a timed automaton T are defined on timed sequences of events. A *timed sequence* is a pair $(\overline{\sigma}, \overline{t})$ where $\overline{\sigma} \in \Sigma^*$ and $\overline{t} = t(1), t(2), ...$ is a time sequence that satisfies $t(k) \in \mathbf{R}^+$ and $t(k) < t(k+1)$. We say the *language* of a finite automata is the set of timed sequences over $\Sigma \times \mathbf{R}^+$ accepted by T:

$$\mathcal{L}(T) = \{(\overline{\sigma}, \overline{t}) \mid (q, \overline{\sigma}, q') \in \psi^*\} \ .$$

where we abuse notation and drop reference to the enabling condition and reset of transitions in ψ. A *run* of a timed automata T over a timed sequence $(\overline{\sigma}, \overline{t})$ is a sequence of states $\overline{q}$ visited by the automaton and if $q(k) = (l(k), x(k))$, there is an edge with event $\sigma(k)$ and enabling condition $\delta(k)$ which evaluates to true for clock values $x(k) = x(k-1) + t(k) - t(k-1)$. $x(k)$ are reset by $\lambda(k)$ at $t(k)$ after the transition is taken.

Decidability of timed automata was shown by Alur and Dill [2] by forming equivalence classes in the space of continuous clocks. The equivalence classes partition R^n into a finite number of regions. The enabling conditions can be defined for positive or negative values of the clocks, and the clocks can be reset to a positive or negative integer value. All the clocks have rate $+1$.

we review some notions of relationships between automata that will be useful in the sequel.

We say timed automata $T_1 = (Q_1, \Sigma_1, \psi_1, I_1)$ and $T_2 = (Q_2, \Sigma_2, \psi_2, I_2)$ are *isomorphic* if there exists a bijection $f : Q_1 \rightarrow Q_2$ such that for $\sigma_1 \in \Sigma_1$ there exists $\sigma_2 \in \Sigma_2$, such that (1) $f(I_1) = I_2$, and (2) for all states $q, q' \in Q_1$, $(q, \sigma_1, q') \in \psi_1$ iff $(f(q), \sigma_2, f(q')) \in \psi_2$.

Given timed automata $T_1 = (Q_1, \Sigma_1, \psi_1, I_1)$ and $T_2 = (Q_2, \Sigma_2, \psi_2, I_2)$, we say that T_2 *simulates* T_1 with relation $R \subset Q_1 \times Q_2$, if $(p, q) \in R$ and $p \xrightarrow{\sigma_1} p'$, with $\sigma_1 \in \Sigma_1$, imply that there exists $q' \in Q_2$ and $\sigma_2 \in \Sigma_2$ such that $q \xrightarrow{\sigma_2} q'$ and $(p', q') \in R$.

Given timed automata $T_1 = (Q_1, \Sigma_1, \psi_1, I_1)$ and $T_2 = (Q_2, \Sigma_2, \psi_2, I_2)$, we say $R \subset Q_1 \times Q_2$ is a *bisimulation* if T_2 simulates T_1 with R and T_1 simulates T_2 with $R^{-1} \in Q_2 \times Q_1$ where $R^{-1} = \{(q, p) | (p, q) \in R\}$.

For the sake of completeness, we state the following well-known result of [2].
Theorem 1 Let T be a timed automaton with n clocks. There exists a finite automaton U which is a bisimulation for T.

Finally, we define a class of initialized timed automata called PN (positive/negative) timed automata.

PN timed automata A PN timed automata is a system $T = (Q, \Sigma, D, \psi, I)$. $Q = L \times \mathbf{R}^n$, and L, $\mathbf{R}^n$, Σ, and I are the same as for timed automata. As before, associated with each edge is an event $\sigma \in \Sigma$, an enabling condition $\delta \in G$, and a reset condition $\lambda \in J$. What is new is a set $D \subset \mathbf{Z}^n$ of continuous state rates. A rate assignment $d \in D$ is associated with each location such that $\dot{x}_i = d_i$, where the components d_i take only one of three values $d_i = \{-1, 0, 1\}$. We require that $\lambda_i \neq id$ if d_i changes upon transitioning from one location to another; therefore, PN timed automata are initialized.

Lemma 2 Let $T = (Q, \Sigma, D, \psi, I)$ be a PN timed automaton. There exists a finite automaton U which is a bisimulation for T.

Proof PN timed automata are initialized multirate timed automata which are isomorphic to finite automata. [4, 5]. □

3 Hybrid systems

A hybrid automaton is a system $H = (Q, \Sigma, D, \psi, I)$, where $Q = L \times \mathbf{R}^n$, L is a finite set of discrete locations, Σ is a set of events, $\psi \subset L \times \Sigma \times L \times G \times J$ is the set of transition relations, and $I \subseteq Q$ is the set of initial locations. ψ labels edges of the automaton by $(l, \sigma, l', \delta, \lambda) \in \psi$. The edge is from location l to l', $\delta \in G$ is an enabling condition, and $\lambda \in J$ is a reset condition. The only change from PN timed automata is that D now defines a set of differential inclusions, with one inclusion associated with each component of the continuous state.

As before, enabling conditions are associated with the edges between discrete locations of the hybrid automaton and they are of the form:

$$\delta := x_i \leq c | x_i \geq c | x_i < c | x_i > c | \delta_1 \wedge \delta_2 | \delta_1 \vee \delta_2 \quad \text{where } c \in \mathbf{Z}$$

Likewise, reset conditions are defined on the edges between discrete states for some components of the continuous state. The reset is of the form $\lambda_i = [r_i, s_i]$ or $\lambda_i = id$ for $r_i, s_i \in \mathbf{Z}$. When λ_i is the identity relation, x_i is not reset. When $\lambda_i = [r_i, s_i]$, x_i is reset non-deterministically to a value between r_i and s_i.

We require $\lambda_i \neq id$ whenever the inclusion for x_i changes in a transition from l to l'.

The *reach set* of a hybrid automaton H, written $Reach_H(Q_0)$, $Q_0 \subseteq Q$, is the set of states that can be reached in any run starting from Q_0.

3.1 Hybrid automata with linear differential inclusions

Now we will extend our definition of hybrid automata to a class with a linear differential inclusion for the continuous state. We consider a hybrid automata $H = (Q, \Sigma, D, \psi, I)$ where, within each location l, the continuous dynamics satisfy an inclusion $\dot{x} \in [L, U]x \in D$. $L, U \in \mathbf{R}^{n \times n}$ are diagonal matrices so that the inclusion for x_i is

$$\dot{x}_i \in [l_i, u_i]x_i$$

and $u_i \geq l_i$.

Our objective is to find a decidable bisimulation for a linear hybrid automaton. Several steps are needed to transform to a decidable class of automata. The trajectories generated by H form a "funnel" with a rectangular cross-section. The ith coordinate of the extremal trajectories have derivatives equal to the extreme values l_i or u_i. Thus these coordinates are made up of segments of increasing or decreasing exponentials.

The exponential segments may be initialized with a positive or negative reset value for each clock component. Referring to Figure 1, the essential features of the automaton that will be transformed for each location l are the reset conditions on transitions entering the location, the enabling conditions on the transitions leaving the location, and the inclusion in each location which will dictate how the reset and enabling conditions are modified.

The steps of the transformation are:

1. Convert automaton H of Figure 1 to one with positive trajectories, automaton P of Figure 2.

2. Convert automaton P with positive trajectories and linear inclusions to one with an augmented continuous state space and linear differential equations, automaton A of Figure 4.

3. Convert automaton A to a PN timed automaton T.

Summary of transformation We consider each of the four steps above. We start with a linear hybrid automaton.

Step 1 Within each location the trajectories of each component is either positive or negative, accordingly as its initial value is positive or negative. If necessary, we use a "change of variables" $x_i \to -x_i$ to make the initial condition positive. We need to keep track of the change of variable, and that is done by adding a "discrete" state (equivalently, by "splitting" the location into several locations). Call the new automaton P.

Step 2 In each location, the continuous state of P satisfies an inclusion of the form:

$$\dot{x}_i \in [l_i, u_i]x_i.$$

Convert P to an automaton L with $2n$ states. To state x_i of P is associated a pair of states y_{2i-1}, y_{2i} of L which satisfy the linear differential equations

$$\dot{y}_{2i-1} = l_i y_{2i-1}, \quad \dot{y}_{2i} = u_i y_{2i}.$$

The reach set of P is a rectangle whose $2n$ vertices are given by the reach set of L.

Step 3 Convert L to a PN automaton T as follows. Within each location, each component y_j of a trajectory of L is an exponential. With an appropriate choice of α_j the nonlinear change of variables, $\alpha_j \log y_j \to z_j$, the z_j trajectories have a constant slope of -1, 0 or 1. This is the PN automaton T, except that the guards in T need no longer be rational because of the nonlinear transformation.

Step 4 Change the guards of H so that the guards of T become rational. This can be done with arbitrarily small changes. Call the resulting hybrid automaton H'. Using Theorem 7 it follows that H' is a bisimulation of H.

Automaton P The first step is to convert all negative valued components of x to positive values; in other words, if any reset condition sets a component of x to a negative value, we will transform it to a positive value. We define two variables $h_i = sgn(x_i)$ and $w_i = |x_i|$ so that $x_i = h_i w_i$ and construct an automaton P shown in Figure 2. It consists of the same discrete locations and transitions as the original automaton but has continuous states $w_i \geq 0$ and includes an additional discrete state variable $h \in \mathbf{Z}^n$ that records the sign of the continuous components of the original automaton, so that the new discrete state is $l' = (l, h)$. Thus, if the original automaton has m discrete states, the new automaton has $m3^n$ discrete states. Automaton P has only positive valued exponential trajectories.

The variable h is assigned upon entering a discrete location by recording the sign of the trajectory in the original automaton after a reset occurs. For a reset of the form $x_i := [r_i, s_i]$ with $r_i, s_i > 0$, $h_i = 1$. If $r_i, s_i < 0$, then $h_i = -1$. If there is no reset on x_i then $h_i = id$. Finally, if $r_i < 0$ and $s_i > 0$ for any component of x, the transition can be split into three transitions. The first transition includes all negative resets $x_i := [r_i, s_i]$, $r_i, s_i < 0$ and for any reset with $r_i < 0$ and $s_i > 0$, a new reset $x_i := [r_i, 0)$. The second transition includes for any reset with $r_i < 0$ and $s_i > 0$, a new reset $x_i := 0$. The third transition consists of all resets $x_i := [r_i, s_i]$, $r_i, s_i > 0$ and for any reset with $r_i < 0$ and $s_i > 0$, a new reset $x_i := (0, s_i]$. Thus, for all i, the first transition assigns $h_i = -1$, the second assigns $h_i = 0$, and the third assigns $h_i = 1$. This case is shown in Figure 3. If there is no reset for x_i, then $h_i = id$ in each of the three transitions.

Enabling conditions of P are of the form $w_i \in [a_i', b_i']$ where $a_i' = a_i$ if $h_i = 0$ or $h_i = 1$, and $a_i' = -a_i$ if $h_i = -1$. A similar transformation applies to b_i'.

Lemma 4 Automaton H and automaton P are isomorphic.

Proof Automaton H and automaton P have identical discrete locations and transitions, and the tranformation of the continuous states is described by the bijection $x_i = h_i w_i$. The transformation of the resets and enabling conditions is explained above. □

A consequence of Lemma 4 is that if the bijection from automaton H to automaton P is f then $Reach_P(f(Q_0)) = f(Reach_H(Q_0))$.

Automaton A In the second step we consider the automaton P with positive trajectories and follow the approach of [6] where an automaton with differential inclusions for the continuous dynamics is converted to an automaton with linear dynamics. Thus, we form a new automaton A with an augmented continuous state space in $\mathbf{R}^{2n}$ such that corresponding to each component x_i there are two components, y_{2i-1} and y_{2i} in automaton A, whose dynamics are specified by

$$\begin{aligned} \dot{y}_{2i-1} &= l_i y_{2i-1} \\ \dot{y}_{2i} &= u_i y_{2i} \ . \end{aligned}$$

See Figure 4. y_{2i-1} and y_{2i} satisfy $y_{2i-1} \leq x_i \leq y_{2i}$. Further, the reach sets are related as follows. When automaton P reaches (l, w), automaton A reaches $\{l\} \times [y_1, y_2] \times ... \times [y_{2n-1}, y_{2n}]$.

The reset condition $w_i := [r'_i, s'_i]$ is transformed to $y_{2i-1} := r'_i$ and $y_{2i} := s'_i$. To obtain the new enabling conditions for automaton A consider an enabling condition of the form $a'_i \leq w_i \leq b'_i$. If w_i of automaton P satisfies $w_i \geq a'_i$, then in automaton A, $y_{2i} \geq a'_i$. When the transition is taken and $y_{2i-1} \leq a'_i$, it is necessary to reset the lower trajectory y_{2i-1} to a'_i. Thus, the enabling condition $w_i \geq a'_i$ becomes

$$(y_{2i} \geq a'_i \wedge ((y_{2i-1} \leq a'_i) \longrightarrow (y_{2i-1} := a'_i)) \tag{1}$$

Similarly, if $w_i \leq b'_i$, then $y_{2i-1} \leq b'_i$ and if $y_{2i} \geq b'_i$, it must be reset to b'_i, so the enabling condition becomes

$$(y_{2i-1} \leq b'_i \wedge ((y_{2i} \geq b'_i) \longrightarrow (y_{2i} := b'_i)). \tag{2}$$

The next theorem will show that there is a simulation relation between automaton A and automaton P and this relation will allow us to relate the reach sets of the two types of automata.

First we need a definition and a fact. If H is a hybrid automaton, then the reverse system is H^{-1} where if $(l, \sigma, l', \delta, \lambda) \in \psi_H$ then $(l', \sigma, l, \delta, \lambda) \in \psi_{H^{-1}}$ and $D_{-H} = -D_H$, that is all rates are reversed.

Fact 4 If A and B are automata, and B simulates A with simulation relation $S \subset Q_A \times Q_B$ and A^{-1} simulates B^{-1} with S^{-1}, and $Q_0 \subset Q_B$, then $Reach_A(S^{-1}(Q_0)) = S^{-1}(Reach_B(Q_0))$.

The next theorem gives the required relation between automaton A and automaton P, whose proof is found in [16].

Theorem 5 Let $P = (Q, \Sigma, D, \psi, I)$ be a hybrid automaton with linear differential inclusions $\dot{w} \in [L, U]w \in D$ where $L, U \in \mathbf{R}^{n \times n}$ are diagonal matrices and $w_i(t) \geq 0 \ \forall t$. There exists an automaton A with linear dynamics that simulates P with relation S^{-1} and P^{-1} simulates A^{-1} with relation S. The simulation relation is $S \subset (L, \mathbf{R}^n) \times (L, \mathbf{R}^{2n})$

$$S = \{((l, y), (l, w)) \mid y_{2i-1} \leq w_i \leq y_{2i}\} \ .$$

Automaton T The last step of our procedure is to apply a transformation to the variables y of A to obtain a PN timed automaton T. Consider the upper trajectory formed by y_{2i}. Define a new state variable $z_{2i} = y_{2i} + v_{2i}$ and pick the feedback v_{2i} to satisfy $\dot{v}_{2i} = -u_i y_{2i} + d_i$. Then,

$$\dot{z}_{2i} = \dot{y}_{2i} + \dot{v}_{2i} = d_i \ .$$

In the z state space the linear dynamics have been transformed to constant rates: $\dot{z}_{2i-1} = c_i$ and $\dot{z}_{2i} = d_i$ and we may pick $c_i, d_i \in \mathbf{R}$, $c_i \leq d_i$. If we apply this technique to all continuous state components, we obtain a new automaton T with constant rate continuous states.

Note that the transformation from y to z variables can be achieved more directly by applying the bijection:

$$z_{2i-1} = \begin{cases} \frac{c_i}{l_i} \ln y_{2i-1} & \text{if} \quad l_i \neq 0 \\ y_{2i-1} & \text{if} \quad l_i = 0 \end{cases} \tag{3}$$

$$z_{2i} = \begin{cases} \frac{d_i}{u_i} \ln y_{2i} & \text{if} \quad u_i \neq 0 \\ y_{2i} & \text{if} \quad u_i = 0 \end{cases} \tag{4}$$

Next we must transform the y enabling conditions to z coordinates. Consider $y_{2i} \geq a_i'$. Let $y_{2i}^o > 0$ be the initial condition for y_{2i} in a discrete location l. Suppose first that $a_i' > 0$. If $u_i > 0$ we can find a time when y_{2i} first reaches a_i' after which the enabling condition is satisfied, i.e., $t \geq \frac{1}{u_i} \ln \frac{a_i'}{y_{2i}^o}$. At that time the upper component z_{2i} will have reached

$$z_{2i} = d_i t + z_{2i}^o = \frac{d_i}{u_i} \ln |a_i'| - \frac{d_i}{u_i} \ln |y_{2i}^o| + z_{2i}^o \ . \tag{5}$$

To eliminate the dependence on initial conditions, pick $z_{2i}^o = \frac{d_i}{u_i} \ln |y_{2i}^o|$. If we pick a $d_i > 0$, then the enabling condition becomes $z_{2i} \geq \frac{d_i}{u_i} \ln |a_i'|$.

If $u_i < 0$ then we are interested in the first time that y_{2i} no longer satisfies $y_{2i} \geq a_i'$. This translates to the condition $t \leq \frac{1}{u_i} \ln \frac{a_i'}{y_{2i}^o}$. Note that if $a_i' > y_{2i}^o$ the condition is never satisfied, as expected. Now if we pick initial conditions as before and select $d_i < 0$ the enabling condition is the same as before: $z_{2i} \geq \frac{d_i}{u_i} \ln |a_i'|$.

The third case, when $a_i' > 0$ and $u_i = 0$ corresponds to the transformation $z_{2i} = y_{2i}$; thus the enabling condition becomes $z_{2i} \geq a_i'$. When $a_i' \leq 0$ the enabling condition is always satisfied because y_{2i} is positive-valued, so the enabling condition is *true*.

Finally, we must consider what happens when $y_{2i}^o = 0$. This case arises whenever a transition is taken that sets $h_i = 0$ and $y_{2i} \geq a_i'$ is either always true or always false. We transform the enabling condition by setting $u_i = 0$, which implies $z_{2i} = y_{2i}$; thus, $z_{2i} \geq a_i'$ is the transformed enabling condition.

We can now pick values for the rate d_i following the choices that simplified the enabling conditions in the preceding discussion:

$$d_i = \begin{cases} 1 & u_i > 0 \\ -1 & u_i < 0 \\ 0 & h_i u_i = 0 \end{cases}$$

This selection of rates is almost in the form of a PN timed automaton, except that the rates cannot be assigned a priori because of the dependence on h_i. To remove this dependence, recall that we have partitioned the transitions so that they assign h_i uniquely. Consider transitions with the label $h_i = 0$. We can create a new discrete location l^o which has as input transitions, those transitions of location l, with $h_i = 0$ and with the same output transitions of l. Also, we redefine $u_i = 0$ in location l^o since this does not change the dynamics. With this modification of the automaton, we may assign rates for d_i:

$$d_i = \begin{cases} 1 & u_i > 0 \\ -1 & u_i < 0 \\ 0 & u_i = 0 \end{cases} \tag{6}$$

To summarize, the transformation of enabling condition expressions $y_{2i} \geq a'_i$ is:

$$\begin{array}{lll} (a'_i > 0) \wedge (u_i \neq 0) & \longrightarrow & (z_{2i} \geq \frac{d_i}{u_i} \ln |a'_i|) \\ (a'_i > 0) \wedge (u_i = 0) & \longrightarrow & (z_{2i} \geq a'_i) \\ (a'_i \leq 0) & \longrightarrow & (true) \ . \end{array} \tag{7}$$

After repeating the above procedure for enabling condition expressions $y_{2i-1} \leq b_i$ for the lower component, the transformed expression is:

$$\begin{array}{lll} (b'_i > 0) \wedge (l_i \neq 0) & \longrightarrow & (z_{2i-1} \leq \frac{c_i}{l_i} \ln |b'_i|) \\ (l_i = 0) & \longrightarrow & (z_{2i} \leq b'_i) \ . \end{array} \tag{8}$$

where we do not include the case of $(b'_i \leq 0)$ because it always evaluates to $false$. The rate for z_{2i-1} is selected by:

$$c_i = \begin{cases} 1 & l_i > 0 \\ -1 & l_i < 0 \\ 0 & l_i = 0 \end{cases} \tag{9}$$

after creating new locations l^o with input transitions that set $h_i = 0$ and then setting $l_i = 0$ in l^o.

The resets are transformed according to the bijection from y to z. $y_{2i-1} := r'_i$ becomes

$$\begin{array}{lll} z_{2i-1} & := & \frac{c_i}{l_i} \ln r'_i \quad \text{if } r'_i > 0 \\ z_{2i-1} & := & r'_i \qquad\quad \text{if } r'_i = 0 \text{ or } l_i = 0 \end{array} \tag{10}$$

Similarly, for resets $y_{2i} := s'_i$,

$$\begin{array}{lll} z_{2i} & := & \frac{d_i}{u_i} \ln s'_i \quad \text{if } s'_i > 0 \\ z_{2i} & := & s'_i \qquad\quad \text{if } s'_i = 0 \text{ or } u_i = 0 \end{array} \tag{11}$$

Theorem 6 Automaton A and automaton T are isomorphic.

Proof We construct a bijection from locations and transitions of A to locations and transitions of T: either the locations are identical, or locations in A with $h_i = 0$ can be mapped to a location in T uniquely. The continuous states are related by the bijection of Equation 3. The enabling conditions are related by Equations 7 and 8, and the reset conditions are related by Equations 10 and 11. □

A consequence of Theorem 6 is that we can relate the reach sets of the two automata; namely, if the bijection from automaton A to automaton T is f then $Reach_T(f(Q_0)) = f(Reach_A(Q_0))$.

Theorem 7 Let $H = (Q, \Sigma, D, \psi, I)$ be a linear hybrid automaton with differential inclusions $\dot{x} \in [L, U]x \in D$ where $L, U \in \mathbf{R}^{n \times n}$ are diagonal matrices. There exists a decidable hybrid automaton $H' = (Q, \Sigma, D, \psi', I)$ with Q, Σ, D, and I the same as in H. The enabling conditions δ' of H' can be made arbitrarily close to δ.

Proof The proof relies on invoking the steps of the transformation just described. First, Lemma 4 says we can calculate $Reach_H$ from $Reach_P$. Theorem 5 says we can calculate $Reach_P$ from $Reach_A$. Finally, Theorem 6 says we can calculate $Reach_A$ from $Reach_T$. We want to show that by a perturbation of the enabling conditions of H, P will be a PN timed automaton, and we showed in Lemma 2 that we can obtain a finite computation of $Reach_T$; that is, reachability of T, a PN timed automaton, is decidable.

To construct the adjusted enabling conditions of H', partition the real line in intervals of length $\frac{1}{q}$, $q \in \mathbf{Z}_+$. Considering Equations 7 and 8, we require that the expressions on the right-hand sides are rational, so we can find the smallest $\beta_i \in \mathbf{Z}$ and largest $\alpha_i \in \mathbf{Z}$ such that

$$\frac{\alpha_i}{q} \leq \frac{d_i}{u_i} \ln |a'_i|; \quad \frac{c_i}{l_i} \ln |b'_i| \leq \frac{\beta_i}{q} .$$

We define

$$\begin{aligned} a''_i &= \exp(\frac{\alpha_i u_i}{q d_i}) \\ b''_i &= \exp(\frac{\beta_i l_i}{q c_i}) . \end{aligned}$$

Then given any $\epsilon > 0$

$$\begin{aligned} |a''_i|[1 + o(\epsilon)] &\geq |a'_i| \geq |a''_i| \\ |b''_i|[1 - o(\epsilon)] &\leq |b'_i| \leq |b''_i| \end{aligned}$$

This follows by picking $q = \max\{\frac{u_i}{d_i \epsilon}, \frac{l_i}{c_i \epsilon}\}$ and noting that

$$\begin{aligned} & \frac{\alpha_i + 1}{q} \geq \frac{d_i}{u_i} \ln |a'_i| \geq \frac{\alpha_i}{q} \\ \Longrightarrow \quad & |a''_i| \exp(\frac{u_i}{d_i q}) \geq |a'_i| \geq |a''_i| \\ \Longrightarrow \quad & |a''_i|[1 + o(\epsilon)] \geq |a'_i| \geq |a''_i| . \end{aligned}$$

The same argument applies for b_i''. Thus, if we use a_i'' and b_i'' in Equations 7 and 8 for automaton H', then after transformation to automaton T, the enabling conditions are defined on rationals and the result follows. □

Lemma 8 The set of decidable hybrid automata with linear differential inclusions is dense.

Proof Follows from the fact that the rationals are dense. □

3.2 Robustness

Theorem 7 constructs a modified automaton whose reach set is an approximation of the original automaton's reach set. The modified automaton is an over-approximation in that it has more permissive enabling conditions and thus will allow more trajectories than the original automaton H.

It is interesting to ask what is the robustness of the reach set calculation to perturbations of the enabling conditions. To make progress on this question, it is helpful to characterize the form of the reach set. Let $S_{[0,t]}(x^0)$ be the set of trajectories ϕ on the interval $[0,t]$ starting from x^0, and let $R(x^0,t)$ be the reach set for a time transition starting from the continuous state x^0. That is, $R(x^0,t) = \{\phi \mid \phi \in S_{[0,t]}(x^0)\}$. Each component ϕ_i satisfies

$$x_i^0 \exp(l_i t) \leq \phi_i(t) \leq x_i^0 \exp(u_i t)$$

where $\phi_i(0) = x_i^0$, so we can write

$$R(x^0,t) = \{F(t)x^0 \mid \exp(l_i t) \leq F_i(t) \leq \exp(u_i t)\},$$

where $F(t)$ is a diagonal $n \times n$ matrix with diagonal elements $F_i(t)$.

Now consider hybrid trajectories. Define the set of hybrid trajectories on an interval $[0,t]$ to be $H_{[0,t]}(x^0)$. As before, $Reach(x^0,t)$ is the set of states that can be reached from the initial state $q^0 = l^0 \times x^0$. We write $\pi \in H_{[0,t]}$ a hybrid trajectory as

$$\pi : (l^0, \phi^0, I^0), ..., (l^m, \phi^m, I^m)$$

where $I^i = [t_i, t_{i+1}]$ is the time interval of the ith phase, $\tau_i = t_{i+1} - t_i$ is the duration, and $\phi^i : I^i \rightarrow R^n, \quad \phi^i \in S_{[t_i,t_{i+1}]}(x^i)$.

The kth phase (l^k, ϕ^k, I^k) of a hybrid trajectory consists of three components:
1) initial condition: ϕ^{k-1},
2) time transition: $R(\phi^{k-1}, \tau_k)$,
3) enabling condition and reset, forming the graph $E_k : G_k \times \lambda_k$, where G_k is the enabling condition and λ_k is the reset which can be represented in general form as the map $\lambda_k(x) = A_k x + B_k$ where A_k is an $n \times n$ matrix and $B_k \subset R^n$ is convex. We can write the kth phase in the form of a discrete update of the continuous states:

$$\begin{aligned} \phi^k &= A_k[R(\phi^{k-1}, \tau_k) \cap G_k] + B_k \\ &= \Phi^k(\phi^{k-1}, \tau_k). \end{aligned} \tag{12}$$

where $R(\phi^{k-1}, \tau_k) = F(\tau_k)\phi^{k-1}$, as defined above. Thus, the reach set at time t is:

$$\begin{aligned} Reach(x^0, t) &= \bigcup_{\sum_{k=1}^{m} \tau_k = t} \Phi^m(\ldots \Phi^2(\Phi^1(x^0, \tau_1), \tau_2), \ldots \tau_m) \\ &= \bigcup_{\sum_{k=1}^{m} \tau_k = t} (\phi^m \circ \ldots \circ \phi^1)(x^0) \end{aligned}$$

where we take the union over hybrid trajectories on $[0, t]$.

Now the approximate reach set is obtained by small perturbations of the enabling regions G_k. The outer reach set approximation is obtained from Equation (12) be replacing G_k by $G_k^\epsilon = G_k + B_\epsilon$, where B_ϵ is an ϵ-ball, and it is denoted by $Reach^\epsilon(x^0, t)$. The main result for the outer reach set approximation is that under certain conditions, as ϵ goes to zero, the outer reach set approximation will approach the original reach set. The necessary conditions and result are summarized in the following theorem, whose proof can be found in [14].

Theorem 9 Assume that the enabling regions are closed, the initial region X^0 is compact, and there are at most a finite number of steps over a finite time interval. Then the outer reach set approximation approachs the original reach set; that is, as $\epsilon \to 0$,

$$\bigcap_{\epsilon > 0} Reach^\epsilon(X^0, t) = Reach(X^0, t).$$

We consider next the inner reach set approximation which is obtained from Equation (12) by replacing G_k by $G_{k\delta} = \{x \mid d(x, S^c) > \delta\}$. The inner reach set approximation is denoted $Reach_\delta(X^0, t)$. It is clear that $\cup_{\delta>0} Reach_\delta(X^0, t) \subset Reach(X^0, t)$. If $x \in Reach(X^0, t)$ it can be shown that every neighborhood of x contains a point of $\cup_{\delta>0} Reach_\delta(X^0, t)$ so x is a limit point. Therefore, $x \in cl(\cup_{\delta>0} Reach_\delta(X^0, t))$. The main result for the inner reach set approximation is the following.

Theorem 10 Assume that the enabling regions are closed, the initial region X^0 is compact, and there are at most a finite number of steps over a finite time interval. Then the inner reach set approximation will approach the interior of the original reach set; that is

$$cl(\bigcup_{\delta > 0} Reach_\delta(X^0, t)) = Reach(X^0, t).$$

3.3 Hybrid automata with nonlinear differential inclusions

We consider nonlinear hybrid automata in which the ith state component x_i follows an inclusion of the form $\dot{x}_i \in [l_i, u_i] f_i(x_i)$, with $l_i \le u_i$, $l_i, u_i \in \mathbf{R}$, and

$$f_i(x_i) := 1 \mid x_i \mid f_i(x_i).$$

We will use a transformation technique analogous to the linear case to show that reachability for this class of nonlinear hybrid automata is decidable.

First, we will consider the ith clock component of location l of automaton H. We assume $f_i(x_i)$ has a finite number of zeros, so that we can identify a finite number of regions of f_i, labeled ρ_{ik}, ordered from left to right, and corresponding to the positive, negative, and zero-valued segments of f_i. Each ρ_{ik} identifies a range of values of x_i, i.e. $\rho_{ik} = [\alpha_{ik}, \beta_{ik}]$, $\alpha_{ik}, \beta_{ik} \in \mathbf{R}$. ρ_{ik} can be closed such as $\rho_{ik} = [-1, -1]$ or open, such as $\rho_{ik} = (-1, 0)$. We also identify the sign of each region: $h_{ik} = \{sgn(f_i(x_i)) \mid x_i \in \rho_{ik}\}$.

The procedure, as before, is to convert automaton H with inclusions to automaton P with clock components labeled by the regions defined above. Then convert automaton P to automaton A with linear differential equations. Finally, convert automaton A to PN timed automaton T.

We first construct automaton P. Suppose there are n clocks and each f_i has k_i regions. For each location l of automaton H, we create $\Pi_{i=1}^n k_i$ locations in automaton P. Define $h = [h_1 \ldots h_n]'$ with $h_i = sgn(f_i(x_i))$ Observe that in the linear case we kept track of the sign of the trajectory which could not change without a reset, because of the linear dynamics. Here we record invariant regions ρ_{ik} of the components of the trajectories and the clock components only change regions through a reset.

Any edge from location l' to l of H must have an edge from l' to each of the $\Pi_{i=1}^n k_i$ locations of P. Correspondingly, each reset $x_i := [r_i, s_i]$ is split into $\sigma = k_i$ resets:

$$\begin{array}{lll} x_i & := & [r_i^1, s_i^1], \quad [r_i^1, s_i^1] \subseteq h_{i1} \\ \vdots & & \\ x_i & := & [r_i^\sigma, s_i^\sigma], \quad [r_i^\sigma, s_i^\sigma] \subseteq h_{i\sigma} \end{array}$$

The edges are now labeled with the modified resets. That is, if the resets corresponding to one of the new edges are $x_i := [r_i^j, s_i^j]$, $i = 1, \ldots n$, then $h_i = sgn(f_i(x_i))$ with $x_i \in \rho_{ij}$. Thus, we have ensured that within the new location, the invariant regions encoded in ρ are fixed. Automaton P, in summary, has an augmented set of locations, modified resets $x_i := [r_i', s_i']$ for each edge, a vector h associated with each location, inclusions $\dot{x}_i \in [l_i, u_i] f_i(x_i)$, and the same enabling conditions $x_i \in [a_i, b_i]$ as for automaton H.

The second step is to transform the nonlinear inclusions to nonlinear differential equations. The resulting automaton A has differential equations

$$\begin{array}{rcl} \dot{y}_{2i-1} & = & l_i f(y_{2i-1}) \\ \dot{y}_{2i} & = & u_i f(y_{2i}) \end{array}$$

The reset condition $x_i := [r_i', s_i']$ is transformed to $y_{2i-1} := r_i'$ and $y_{2i} := s_i'$, as before. The enabling conditions are given in Equations (1) and (2) with $a_i' = a_i$ and $b_i' = b_i$.

The last step is to transform automaton A to automaton T with linear dynamics. We consider first the upper trajectory of component i, formed by y_{2i}. As before, define a new state variable $z_{2i} = y_{2i} + v_{2i}$ and pick the feedback v_{2i} to satisfy $\dot{v}_{2i} = -u_i f(y_{si}) + d_i$. Then $\dot{z}_{2i} = d_i$.

Consider the transformation of the enabling condition $y_{2i} \geq a_i$. Here we make use of the fact that the location records the invariant region of each clock component and retains the sign of the derivative in the vector h. The transformed enabling condition has three cases. Case (a) is

$$\text{if}(y_{2i} \in \rho_{ik} \wedge a_i \in \rho_{ij}, j < k) \rightarrow (\text{true})$$

in which y_{2i} stays in a region whose values are larger than a_i. Case (b) is

$$\text{if}(y_{2i} \in \rho_{ik} \wedge a_i \in \rho_{ij}, j > k) \rightarrow (\text{false})$$

in which y_{2i} stays in a region whose values are smaller than a_i. The last case is when $y_{2i} \in \rho_{ik}$ and $a_i \in \rho_{ik}$. In this case, there are three subcases. First, if $h_{ik}u_i > 0$ the ith clock has positive derivative. Define the indefinite integral

$$F(w) = \int \frac{dw}{u_i f_i(w)}.$$

Then, $y_{2i} = a_i$ when $t = F(a_i) - F(y^0_{2i}$. Then

$$\begin{aligned} z_{2i}(t) &= d_i t + z^0_{2i} \\ &= d_i(F(a_i) - F(y^0_{2i})) + z^0_{2i}. \end{aligned}$$

We pick $z^0_{2i} = d_i F(y^0_{2i})$ to cancel initial conditions. Then we obtain the transformed enabling condition

$$z_{2i} \geq F(a_i)$$

where we choose $d_i = 1$. The transformed reset condition for this case is $z_{2i} := F(r_i)$.

The second subcase is when $h_{ik}u_i < 0$ and the ith clock has negative derivative. One can check that the correct transformed enabling condition is $z_{2i} \geq -F(a_i)$ and the transformed reset condition is $z_{2i} := -F(r_i)$. Finally, the last subcase is when $\dot{y}_{2i} = 0$ and the enabling condition and reset condition are unchanged.

The transformation for the lower trajectory y_{2i-1} follows in a similar manner as above.

The steps outlined above lead to the follow result.

Theorem 11 Let $H = (Q, \Sigma, D, \psi, I)$ be a nonlinear hybrid automaton with differential inclusions $\dot{x}_i \in [l_i, u_i] f_i(x_i)$ with $l_i \leq u_i$, $l_i, u_i \in \mathbf{R}$ and $f : \mathbf{R}^n \rightarrow \mathbf{R}^n$. There exists a decidable hybrid automaton $H' = (Q, \Sigma, D, \psi', I)$ with Q, Σ, D, and I the same as in H. The enabling conditions δ' of H' can be made arbitrarily close to δ.

Acknowledgements The authors are grateful to Anuj Puri and Tom Henzinger for their helpful suggestions on this problem.

References

[1] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In D.L. Dill, ed., *Computer-Aided Verification*, LNCS 818, pp. 95-104, Springer-Verlag, 1994.

[2] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *"Proc. 17th ICALP: Automata, Languages and Programming*, LNCS 443, Springer-Verlag, 1990.

[3] R. Alur, D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, no. 126, pp. 183-235, 1994.

[4] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho. Hybrid Automaton: An algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems I*, LNCS 736, pp. 209-229, Springer-Verlag, 1993.

[5] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, no. 138, pp. 3-34, 1995.

[6] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proc. 27th Annual Symp. Theory of Computing Science*, pp. 373-382, ACM Press, 1995.

[7] T. Henzinger and P. Ho. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, ed., *Computer Aided Verification*, LNCS 939, pp. 225-238, Springer-Verlag, 1995.

[8] T. Henzinger. Hybrid automata with finite bisimulations. In *"Proc. 22nd ICALP: Automata, Languages and Programming*, LNCS 944, pp. 324-335, Springer-Verlag, 1995.

[9] T. Henzinger. The theory of hybrid automata. In *Proc. 11th IEEE Symposium on Logic in Computer Science*, pp. 278-292, New Brunswick, NJ, 1996.

[10] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems I*, LNCS 736, pp. 179-208, Springer-Verlag, 1993.

[11] A. Nerode and W. Kohn. Multiple agent hybrid control architecture. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems I*, LNCS 736, pp. 297-316, Springer-Verlag, 1993.

[12] A. Nerode and W. Kohn. Models for hybrid systems: automata, topologies, controllability, observability. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems I*, LNCS 736, pp. 317-356, Springer-Verlag, 1993.

[13] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, eds., *Hybrid Systems I*, LNCS 736, pp. 149-178, Springer-Verlag, 1993.

[14] A. Puri, M. Broucke, and P. Varaiya. On the trajectories of hybrid systems. Presented at *Int. Conf. Hybrid Systems*, Cornell University, Ithaca, NY, October 1996.

[15] A. Puri, and P. Varaiya. Decidable hybrid systems. *Mathematical and Computer Modeling*, vol. 23, no. 11-12, pp. 191-202, June 1996.

[16] A. Puri. Theory of hybrid systems and discrete event systems. University of California, Berkeley, 1995.

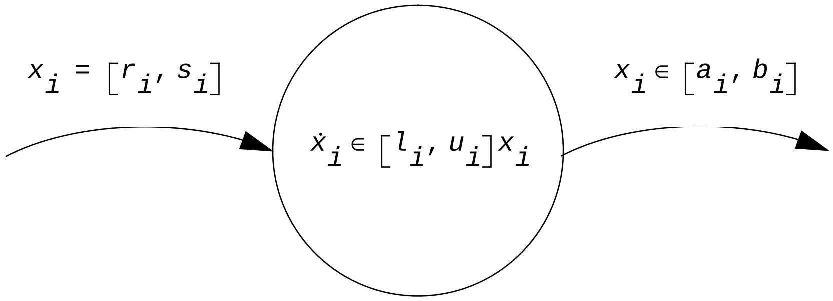

Figure 1: Automata H: hybrid system with linear differential inclusions.

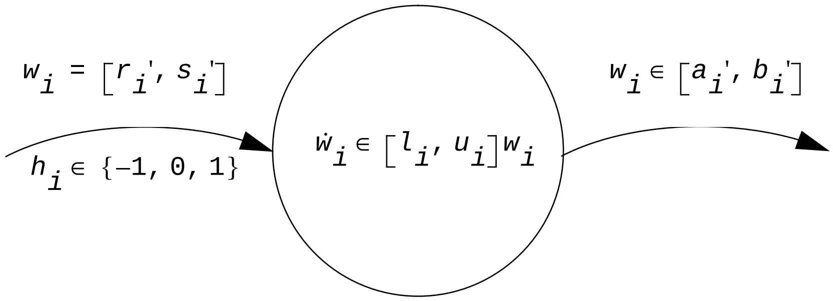

Figure 2: Automaton P: hybrid system with linear differential inclusions and positive-valued trajectories.

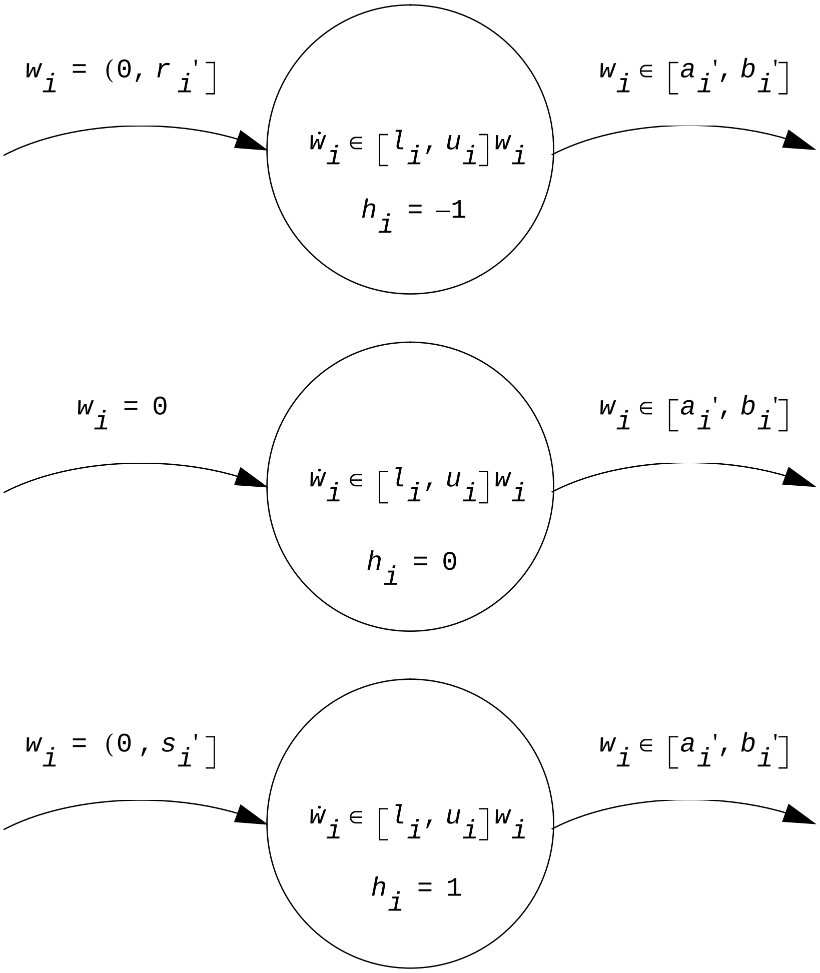

Figure 3: Automata P: hybrid system with linear differential inclusions.

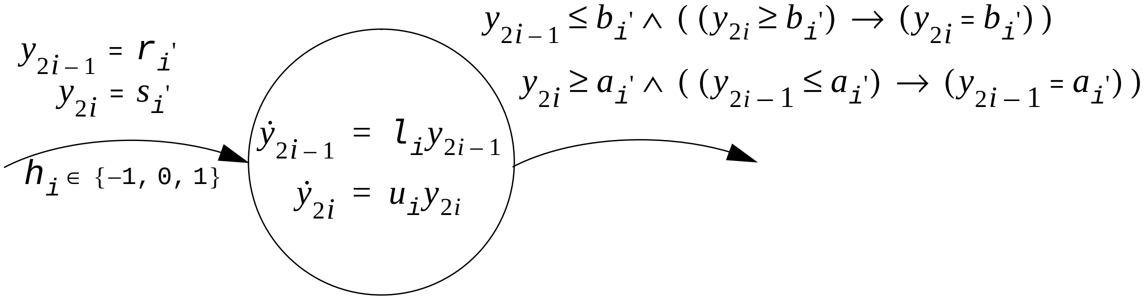
$y_{2i-1} = r_i'$
$y_{2i} = s_i'$
$h_i \in \{-1, 0, 1\}$
$\dot{y}_{2i-1} = l_i y_{2i-1}$
$\dot{y}_{2i} = u_i y_{2i}$
$y_{2i-1} \leq b_i' \wedge ((y_{2i} \geq b_i') \rightarrow (y_{2i} = b_i'))$
$y_{2i} \geq a_i' \wedge ((y_{2i-1} \leq a_i') \rightarrow (y_{2i-1} = a_i'))$

Figure 4: Automaton A: hybrid system with linear clock rates.