Algorithmic Analysis of Nonlinear Hybrid Systems*†

Thomas A. Henzinger
Electrical Engineering & Computer Sciences
University of California
Berkeley, CA, USA
tah@eecs.berkeley.edu

Pei-Hsin Ho
Strategic CAD Labs
Intel Corporation
Hillsboro, OR, USA
pho@ichips.intel.com

Howard Wong-Toi
Cadence Berkeley Labs
Cadence Design Systems
Berkeley, CA, USA
howard@cadence.com

Abstract. Hybrid systems are digital real-time systems that are embedded in analog environments. Model-checking tools are available for the automatic analysis of *linear hybrid automata*, whose environment variables are subject to piecewise-constant polyhedral differential inclusions. In most embedded systems, however, the environment variables have differential inclusions that vary with the values of the variables, *e.g.* $\dot{x} = x$. Such inclusions are prohibited in the linear hybrid automaton model. We present two methods for translating nonlinear hybrid systems into linear hybrid automata. Properties of the nonlinear systems can then be inferred from the automatic analysis of the translated linear hybrid automata.

The first method, called *clock translation*, replaces constraints on nonlinear variables by constraints on clock variables. The clock translation is efficient but has limited applicability. The second method, called *linear phase-portrait approximation*, conservatively overapproximates the phase portrait of a hybrid automaton using piecewise-constant polyhedral differential inclusions. Both methods are sound for safety properties; that is, if we establish a safety property of the translated linear system, we may conclude that the original nonlinear system satisfies the property. When applicable, the clock translation is also complete for safety properties; that is, the original system and the translated system satisfy the same safety properties. The phase-portrait approximation method is not complete for safety properties, but it is asymptotically complete; intuitively, for every safety property, and for every relaxed nonlinear system arbitrarily close to the original, if the relaxed system satisfies the safety property, then there is a linear phase-portrait approximation that also satisfies the property.

We illustrate both methods by using HyTech—a symbolic model checker for linear hybrid automata—to automatically check properties of a nonlinear temperature controller and of a predator-prey ecology.

*This research was supported in part by the Office of Naval Research Young Investigator award N00014-95-1-0520, by the National Science Foundation CAREER award CCR-9501708, by the National Science Foundation grant CCR-9504469, by the Air Force Office of Scientific Research contract F49620-93-1-0056, by the Army Research Office MURI grant DAAH-04-96-1-0341, by the Advanced Research Projects Agency grant NAG2-892, and by the Semiconductor Research Corporation contract 95-DC-324.036.

†An extended version of this paper appeared in the *IEEE Transactions on Automatic Control* **43** (special issue on Hybrid Control Systems), 1998, pp. 540–554. Preliminary reports on this work appeared in Thomas A. Henzinger and Pei-Hsin Ho, "Algorithmic analysis of nonlinear hybrid systems," *Proceedings of the Seventh International Conference on Computer-aided Verification* (CAV), *Lecture Notes in Computer Science* **939**, Springer-Verlag, 1995, pp. 225–238, and in Thomas A. Henzinger and Howard Wong-Toi, "Linear phase-portrait approximations for nonlinear hybrid systems," *Hybrid Systems III: Verification and Control*, *Lecture Notes in Computer Science* **1066**, Springer-Verlag, 1996, pp. 377–388.

1 Introduction

Hybrid systems combine discrete and continuous dynamics. The analysis of hybrid systems, therefore, requires techniques from both computer science and control theory. From computer science, we have the model of *hybrid automata*, which combines discrete control graphs with continuously evolving variables [ACH+95]. A hybrid automaton exhibits two kinds of state changes: discrete jump transitions occur instantaneously, and continuous flow transitions occur while time elapses. We have algorithmic techniques for checking certain properties, such as safety, for *linear hybrid automata*, whose transitions are subject to linearity restrictions: for each jump, the possible source and target values of the variables are constrained by linear inequalities; for each flow, the possible values of the variables during the flow are constrained by linear inequalities on the variables, and the possible derivatives of the variables during the flow are constrained by linear inequalities on the derivatives [AHH96]. It is important to realize that the definition of linearity used here is more restrictive than in systems theory. For instance, linear hybrid automata cannot represent constraints of the form $\dot{x} = x$, which relate the derivative of x with the value of x. Model-checking methods for linear hybrid automata have been implemented in HyTech [HHW95b], and used to verify distributed real-time protocols [HH95b, HW95, HHW95a]. This paper extends the model-checking approach to the analysis of nonlinear hybrid systems, by reduction to the linear problem.

For automata, the verification problem for safety properties reduces to the emptiness problem, *i.e.* whether there exists a trajectory from an initial state to a final state. Every hybrid automaton defines an infinite-state transition system with jump transitions and flow transitions. Checking the emptiness of a hybrid automaton, then, involves computing the successors (or predecessors) of state sets in the underlying transition system. The widest class of hybrid automata for which we know how to compute flow successors reasonably efficiently is that of linear hybrid automata. We therefore propose the following methodology for analyzing a nonlinear hybrid automaton A. First, we attempt to translate the constraints on each nonlinear variable x of A into constraints on a clock variable, which is a variable with the constant derivative 1. Intuitively, the translation is possible if x is reinitialized with every jump that causes changes in the constraints that govern the flow of x, and at all times during a flow the value of x is completely determined by the initial value and the elapsed time. For nonlinear variables that do not satisfy these requirements, in a second step, we overapproximate the set of possible flow vectors using linear constraints. The resulting linear hybrid automaton B, whose emptiness can be checked by HyTech, has the same or strictly more trajectories than A. So if B is empty, then A is also empty. If, however, B is nonempty, we can draw no conclusion about the emptiness of A, and must refine our approximation.

Formally, the hybrid automaton that results from translating the constraints on a nonlinear variable into clock constraints is timed bisimilar to the original automaton, *i.e.* the translation preserves all properties of interest [Hen96]. The method, called *clock translation*, is detailed in Section 3 and illustrated on a simple temperature controller. The hybrid automaton that results from overapproximating the set of possible flow vectors time simulates the original automaton. While the approximate automaton may satisfy strictly fewer safety properties than the original automaton, we show that at the cost of increasing the discrete complexity of the approximation, every hybrid automaton can be approximated arbitrarily closely. The method, called *linear phase-portrait approximation*, is detailed in Section 4 and demonstrated on a predator-prey ecology. Both methods complement each other and both may be required for the successful algorithmic analysis of a nonlinear hybrid system: the clock translation, while efficient, may not be applicable; and the linear phase-portrait approximation, while always applicable, may produce an approximate automaton that does not satisfy the desired property, and increasing the accuracy of the approximation may

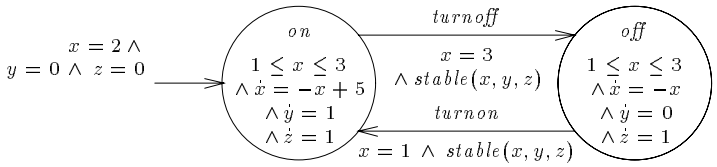


Figure 1: A thermostat

be too expensive.

Related work

Phase portraits have been studied extensively in the literature on dynamical systems [HS74]. Typically, researchers concentrate on nontrivial properties of continuous dynamics, such as stability and convergence. Our work differs in two respects. First, we consider *products* of nondeterministic dynamical systems with discrete control graphs. Second, our goal is to analyze and derive simple properties of such systems *automatically*. In computer science, the technique of deriving system properties using conservative approximations is called *abstract interpretation*. In [Hal93, HRP94, HH95c, OSY94, PV95], abstract interpretation techniques are used for improving the efficiency of analyzing linear hybrid systems, whereas here we approximate nonlinear hybrid systems. In [HH95a, PBV96], restricted cases of linear phase-portrait approximations for nonlinear hybrid systems are considered.

2 Hybrid Automata

Hybrid automata are a mathematical model for systems with both discrete and continuous components. Informally, a hybrid automaton consists of a finite set X of real-valued variables and a labeled multigraph (V, E). The edges E are used to model discrete jumps. They are labeled with constraints on the values of X before and after jumps. The vertices V are used to model continuous flows. They are labeled with constraints on the values and derivatives of X during flows. The state of the automaton changes either instantaneously when a discrete jump occurs or, while time elapses, through a continuous flow.

2.1 Syntax

Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a set of real-valued variables. Let *RelOps* be the set $\{<, \le, =, \ge, >\}$ of binary relational operators. An *atomic predicate* over Y is a predicate of the form $f(y_1, y_2, \ldots, y_n) \sim c$, for a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$, a relational operator $\sim \in$ *RelOps*, and a real constant $c \in \mathbb{R}$. A *predicate* over Y is a positive boolean combination of atomic predicates over Y. A *valuation* over Y is a point $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ in the n-dimensional real space $\mathbb{R}^n$, or equivalently, a function that maps each variable $y_i \in Y$ to a value $a_i \in \mathbb{R}^n$. We write $\mathbf{a}(y_i)$ to refer to the value a_i of the variable y_i in the valuation $\mathbf{a}$. For a predicate φ and a valuation $\mathbf{a}$ over Y, we write $\varphi[Y := \mathbf{a}]$ for the truth value obtained by evaluating φ with the constant a_i replacing in φ all occurrences of the variable y_i, for each $i \in \{1, \ldots, n\}$. Every predicate φ over Y defines a set $[\![\varphi]\!] \subseteq \mathbb{R}^n$ of valuations such that $\mathbf{a} \in [\![\varphi]\!]$ iff $\varphi[Y := \mathbf{a}]$ is true. If $[\![\varphi]\!]$ is a convex set, then φ is called a convex predicate.

A *hybrid automaton* is a system $A = (X, V, \mathit{flow}, E, \mathit{jump}, \Sigma, \mathit{event}, \mathit{init}, \mathit{final})$ consisting of the following components:

Variables A finite ordered set $X = \{x_1, x_2, \ldots, x_n\}$ of real-valued *variables*. For example, the thermostat automaton in Figure 1 has three variables, x, y, and z, where x models the temperature, y models the amount of time spent in control mode *on*, and z models the total elapsed time.

Control modes A finite set V of *control modes*. For example, the thermostat automaton has two control modes, *on* and *off*, where *on* models the heater being turned on, and *off* models the heater being turned off.

A *state* $(v, \mathbf{a}, \dot{\mathbf{a}})$ consists of a control mode $v \in V$, a valuation $\mathbf{a} \in \mathbb{R}^n$ over the set X of variables, and a valuation $\dot{\mathbf{a}} \in \mathbb{R}^n$ over the set $\dot{X}$ of variables, where $\dot{X} = \{\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n\}$. The dotted variable $\dot{x}$ represents the first derivative of x with respect to time, *i.e.* $\dot{x} = dx/dt$. Intuitively, a state describes a control mode, a point, and a flow tangent at the point.

Flow conditions A labeling function *flow* that assigns a *flow condition* to each control mode $v \in V$. The flow condition $\mathit{flow}(v)$ is a predicate over $X \cup \dot{X}$. While the automaton control is in control mode v, the variables change along differentiable trajectories for which the values of the variables and their first derivatives satisfy the flow condition. For example, the control mode *on* of the thermostat automaton has the flow condition $1 \leq x \leq 3 \wedge \dot{x} = -x+5 \wedge \dot{y} = 1 \wedge \dot{z} = 1$. Hence, while the heater is turned on, the temperature x follows the differential equation $\dot{x} = -x + 5$. The variable y measures the accumulated amount of time that the heater is active. The variable z measures the total elapsed time; such a variable, with constant derivative 1, is called a clock.

The state $(v, \mathbf{a}, \dot{\mathbf{a}})$ is *admissible* if $(\mathbf{a}, \dot{\mathbf{a}}) \in [\![\mathit{flow}(v)]\!]$. The *invariant condition* for the control mode v is the predicate $\mathit{inv}(v) = (\exists \dot{X}.\, \mathit{flow}(v))$ over X. The automaton control may reside in a control mode only while the invariant condition holds. Thus invariant conditions can be used to force progress out of a control mode. For example, the control mode *on* of the thermostat automaton has the invariant condition $1 \leq x \leq 3$. Hence, the heater can be turned on only while the temperature is in the range $[1, 3]$, and it must be turned off at the latest when the rising temperature hits 3.

Control switches A finite multiset E of *control switches*. Each control switch (v, v') identifies a source control mode $v \in V$ and a target control mode $v' \in V$. For example, the thermostat automaton has two control switches, (*on*, *off*) and (*off*, *on*).

Jump conditions A labeling function *jump* that assigns a *jump condition* to each control switch $e \in E$. The jump condition $\mathit{jump}(e)$ is a predicate over $X \cup \dot{X} \cup X' \cup \dot{X}'$, where $X' = \{x'_1, \ldots, x'_n\}$ and $\dot{X}' = \{\dot{x}'_1, \ldots, \dot{x}'_n\}$. The variable x_i refers to its value before the control switch, and the primed variable x'_i refers to the value of x_i after the control switch. The variable $\dot{x}_i$ refers to the first derivative of x_i before the control switch, and $\dot{x}'_i$ refers to the derivative of x_i after the control switch. Thus the jump conditions relate the values of the variables before a control switch with those after (allowing, for example, the assertion that trajectories are continuous across control switches), and also relate the tangents before the control switch with those after (allowing, for example, the assertion that trajectories are differentiable across control switches). For convenience, we use the special predicate *stable* to indicate that certain primed variables have the same values as their unprimed counterparts,

e.g. $\mathit{stable}(x, \dot{y})$ denotes $x = x' \wedge \dot{y} = \dot{y}'$. For example, in the thermostat automaton, the control switch $(\mathit{on}, \mathit{off})$ has the jump condition $x = 3 \wedge \mathit{stable}(x, y, z)$. The conjunct $x = 3$ asserts that the heater can be turned off only when the temperature is 3. The conjunct $\mathit{stable}(x, y, z)$ asserts that the target point of the jump is the same as the source point.

Events A finite set Σ of *events* including the *silent event* τ, and a labeling function *event* that assigns an event in Σ to every control switch $e \in E$. Control switches labeled by τ are called *silent*. For convenience, we require that $\mathit{jump}(e) = \mathit{stable}(X, \dot{X})$ for all silent control switches e. Although not done here, the events can be used to define the parallel composition of hybrid automata [AHH96].

Initial conditions A labeling function *init* that assigns an *initial condition* to each control mode $v \in V$. The initial condition $\mathit{init}(v)$ is a predicate over $X \cup \dot{X}$. The automaton control may start in the control mode v when $\mathit{init}(v)$ holds. The state $(v, \mathbf{a}, \dot{\mathbf{a}})$ is *initial* if it is admissible and $(\mathbf{a}, \dot{\mathbf{a}}) \in [\![\mathit{init}(v)]\!]$. In the graphical representation of automata, initial conditions appear as labels on incoming arrows without a source control mode, and initial conditions of the form *false* are not depicted. For example, all initial states of the thermostat automaton are in the control mode *on* with $x = 2 \wedge y = 0 \wedge z = 0$. Hence, initially the heater is turned on and the temperature is 2.

Final conditions A labeling function *final* that assigns a *final condition* to each control mode $v \in V$. The final condition $\mathit{final}(v)$ is a predicate over $X \cup \dot{X}$. The state $(v, \mathbf{a}, \dot{\mathbf{a}})$ is *final* if it is admissible and $(\mathbf{a}, \dot{\mathbf{a}}) \in [\![\mathit{final}(v)]\!]$. We use final conditions to specify the unsafe, or error, states of a system. Then, the system is safe if, when started in an initial state, no final state can be reached. For example, for the thermostat automaton, consider the safety property that the heater is active less than 50% of the first 60 time units. The corresponding unsafe states are specified by the final conditions $\mathit{final}(\mathit{on}) = \mathit{final}(\mathit{off}) = (z = 60 \wedge y \geq z/2)$.

Remark. The definition of hybrid automata used here differs from previous definitions in the literature [ACH+95, Hen96]. In a minor change, we add final conditions so that system safety can be conveniently expressed as automaton emptiness. In a major change, we augment the notion of a state with a vector over $\dot{X}$, providing the flow tangent at the given point, we augment jump conditions with constraints over $\dot{X}$, and we make invariant conditions implicit within flow conditions. This enables us to model changes (or the absence of changes) in the first derivatives when a control switch occurs. Information about higher-order derivatives can be encoded by explicitly introducing additional variables, *e.g.* with the flow condition $\dot{x} = u$, we may use $\dot{u}$ to refer to the second derivative of x.

2.2 Timed transition-system semantics

We provide a formal semantics for hybrid automata in terms of timed transition systems. Let $\mathbb{R}_{\geq 0}$ denote the set of nonnegative reals. A *timed transition system* $T = (Q, Q^I, Q^F, \Sigma, \rightarrow)$ consists of a (possibly infinite) set Q of states, a subset $Q^I \subseteq Q$ of *initial states*, a subset $Q^F \subseteq Q$ of *final states*, a set Σ of *transition labels* (including the special *silent transition label* τ), and a *transition relation* $\rightarrow \subseteq Q \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times Q$. Each triple $(s, m, s') \in \rightarrow$ is called a *transition*, and denoted $s \xrightarrow{m} s'$. There are two kinds of transitions: *jump transitions* are of the form $s \xrightarrow{\sigma} s'$, for a transition label $\sigma \in \Sigma$, and *flow transitions* are of the form $s \xrightarrow{\delta} s'$, for a nonnegative real $\delta \in \mathbb{R}_{\geq 0}$ which is called the *duration* of the flow transition. Jump transitions of the form $s \xrightarrow{\tau} s'$ are called *silent*.

Every hybrid automaton A defines a timed transition system $T_A = (Q, Q^I, Q^F, \Sigma, \rightarrow)$ with the following components:

- Q is the set of admissible states of A.
- Q^I is the set of initial states of A.
- Q^F is the set of final states of A.
- Σ is the set of events of A.
- The transition relation $\rightarrow = \bigcup_{\sigma \in \Sigma} \stackrel{\sigma}{\rightarrow} \cup \bigcup_{\delta \in \mathbb{R}_{\geq 0}} \stackrel{\delta}{\rightarrow}$, where the jump transitions and the flow transitions are defined as follows.

In jump transitions, the control mode of the automaton and the values and derivatives of the variables change instantaneously, in accordance with a control switch and its jump condition. Formally, for each event $\sigma \in \Sigma$, the binary *jump relation* $\stackrel{\sigma}{\rightarrow}$ on the admissible states is defined by $(v, \mathbf{a}, \dot{\mathbf{a}}) \stackrel{\sigma}{\rightarrow} (v', \mathbf{a}', \dot{\mathbf{a}}')$ iff there exists a control switch $e = (v, v')$ of A such that $\mathit{event}(e) = \sigma$ and $\mathit{jump}(e)[X, \dot{X}, X', \dot{X}' := \mathbf{a}, \dot{\mathbf{a}}, \mathbf{a}', \dot{\mathbf{a}}']$ is true. The control switch e is referred to as a *witness* for the jump transition $(v, \mathbf{a}, \dot{\mathbf{a}}) \stackrel{\sigma}{\rightarrow} (v', \mathbf{a}', \dot{\mathbf{a}}')$.

During flow transitions, the control mode of the automaton stays fixed and the values and derivatives of the variables change over time in accordance with the flow condition of the active control mode. Formally, for each nonnegative real $\delta \in \mathbb{R}_{\geq 0}$, the binary *flow relation* $\stackrel{\delta}{\rightarrow}$ on the admissible states is defined by $(v, \mathbf{a}, \dot{\mathbf{a}}) \stackrel{\delta}{\rightarrow} (v', \mathbf{a}', \dot{\mathbf{a}}')$ iff $v = v'$, and either (1) $\delta = 0$ and $\mathbf{a} = \mathbf{a}'$ and $\dot{\mathbf{a}} = \dot{\mathbf{a}}'$, or (2) $\delta > 0$ and there exists a continuously differentiable function $\rho : [0, \delta] \rightarrow \mathbb{R}^n$ such that the following two conditions hold:

1. The endpoints of the transition match those of ρ, *i.e.* $\rho(0) = \mathbf{a}$, $\dot{\rho}(0) = \dot{\mathbf{a}}$, $\rho(\delta) = \mathbf{a}'$, and $\dot{\rho}(\delta) = \dot{\mathbf{a}}'$, where $\dot{\rho}$ is the first derivative of ρ.
2. The flow condition is satisfied along ρ, *i.e.* $\mathit{flow}(v)[X, \dot{X} := \rho(t), \dot{\rho}(t)]$ is true for all $t \in [0, \delta]$.

The function ρ is referred to as a *witness* for the flow transition $(v, \mathbf{a}, \dot{\mathbf{a}}) \stackrel{\delta}{\rightarrow} (v', \mathbf{a}', \dot{\mathbf{a}}')$. This completes the definition of T_A.

A *trajectory* of A is a finite path $s_0 \stackrel{m_0}{\rightarrow} s_1 \stackrel{m_1}{\rightarrow} \cdots \stackrel{m_{k-1}}{\rightarrow} s_k$ of transitions in T_A, with $k \geq 0$, such that $s_0 \in Q^I$, and $s_i \stackrel{m_i}{\rightarrow} s_{i+1}$ for all $i \in \{0, \ldots, k-1\}$. The state s_k is referred to as the *end state* of the trajectory. The state s of A is *reachable* if there is a trajectory of A with the end state s. We write $\mathit{reach}(A)$ for the set of reachable states of A. The hybrid automaton A is *empty* if no final state of A is reachable, *i.e.* $Q^F \cap \mathit{reach}(A) = \emptyset$. Otherwise A is *nonempty*.

2.3 Time simulation and timed bisimilarity

We define the concepts of time simulation and timed bisimilarity for timed transition systems [Hen96]. In the sense of Milner our simulations are *weak* [Mil89], with silent transitions being invisible. In addition, a flow transition of duration δ cannot be distinguished from two or more consecutive flow transitions whose durations add up to δ. This is captured by closing the timed transition system $T = (Q, Q^I, Q^F, \Sigma, \rightarrow)$ under stuttering. For each non-silent transition label $\sigma \in \Sigma \setminus \{\tau\}$, we define the *stutter-closed jump relation* $\stackrel{\sigma}{\twoheadrightarrow} \subseteq Q^2$ by $s \stackrel{\sigma}{\twoheadrightarrow} s'$ iff there exists a finite sequence $s_0, \ldots, s_k$ of states, with $k \geq 0$, such that $s = s_0$ and $s_0 \stackrel{\tau}{\rightarrow} s_1 \stackrel{\tau}{\rightarrow} \cdots \stackrel{\tau}{\rightarrow} s_k \stackrel{\sigma}{\rightarrow} s'$. For each nonnegative real $\delta \in \mathbb{R}_{\geq 0}$, we define the *stutter-closed flow relation* $\stackrel{\delta}{\twoheadrightarrow} \subseteq Q^2$ by $s \stackrel{\delta}{\twoheadrightarrow} s'$ iff there

exist a finite sequence $s_0, \ldots, s_{2k}$ of states and a finite sequence $\delta_0, \ldots, \delta_k \in \mathbb{R}_{\geq 0}$ of constants, with $k \geq 0$, such that $s = s_0$ and $s_0 \xrightarrow{\delta_0} s_1 \xrightarrow{\tau} s_2 \xrightarrow{\delta_1} \cdots \xrightarrow{\tau} s_{2k} \xrightarrow{\delta_k} s'$ and $\Sigma_{i=0}^{k} \delta_i = \delta$.

Let $T_1 = (Q_1, Q_1^I, Q_1^F, \Sigma, \rightarrow_1)$ and $T_2 = (Q_2, Q_2^I, Q_2^F, \Sigma, \rightarrow_2)$ be two timed transition systems with the same transition labels. The binary relation $\succeq \subseteq Q_1 \times Q_2$ is a *time simulation* of T_2 by T_1 if the following three conditions hold:

1. For all states $s_1 \in Q_1$ and $s_2 \in Q_2$, if $s_1 \succeq s_2$, then for each transition label $m \in \Sigma \setminus \{\tau\} \cup \mathbb{R}_{\geq 0}$, if $s_2 \overset{m}{\twoheadrightarrow}_2 s_2'$, then there exists a state s_1' such that $s_1 \overset{m}{\twoheadrightarrow}_1 s_1'$ and $s_1' \succeq s_2'$.

2. For every initial state $s_2 \in Q_2^I$, there exists an initial state $s_1 \in Q_1^I$ such that $s_1 \succeq s_2$.

3. For every final state $s_2 \in Q_2^F$, and for every state $s_1 \in Q_1$, if $s_1 \succeq s_2$, then $s_1 \in Q_1^F$.

The timed transition system T_1 *time simulates* the timed transition system T_2, denoted $T_1 \succeq_{sim} T_2$, if there exists a time simulation $\succeq$ of T_2 by T_1. The hybrid automaton A *time simulates* the hybrid automaton B if $T_A \succeq_{sim} T_B$.

For a binary relation $\equiv \subseteq Q_1 \times Q_2$, define the inverse $\equiv^{-1}$ to be the binary relation over $Q_2 \times Q_1$ such that $(s_2, s_1) \in \equiv^{-1}$ iff $(s_1, s_2) \in \equiv$. The binary relation $\equiv \subseteq Q_1 \times Q_2$ is a *time bisimulation* between T_1 and T_2 if $\equiv$ is a time simulation of T_2 by T_1 and $\equiv^{-1}$ is a time simulation of T_1 by T_2. The timed transition systems T_1 and T_2 are *timed bisimilar*, denoted $T_1 \equiv_{bis} T_2$, if there exists a time bisimulation $\equiv$ between T_1 and T_2. The two hybrid automata A and B are *timed bisimilar* if $T_A \equiv_{bis} T_B$.

By induction on the length of trajectories, it is easy to check that if A time simulates B, then for every trajectory of B there exists a trajectory of A that follows the same sequence of non-silent events. Therefore, if A is empty, then so is B.

Proposition 2.1 [Par81] *Let A and B be hybrid automata.*

- *If A time simulates B and A is empty, then B is empty.*
- *If A and B are timed bisimilar, then A is empty iff B is empty.*

Remark. The notions of time simulation and timed bisimilarity are unnecessarily strong conditions for emptiness checking. They are, however, useful for checking also more general classes of properties than safety, precisely because they provide such a tight coupling between systems [Hen96].

2.4 Control-mode splitting

We often find it useful to split the control modes of a hybrid automaton in order to obtain simpler or more constrained flow conditions. Let $A = (X_A, V_A, \mathit{flow}_A, E_A, \mathit{jump}_A, \Sigma_A, \mathit{event}_A, \mathit{init}_A, \mathit{final}_A)$ be a hybrid automaton. A *flow split* for A is a function that maps each control mode $v \in V_A$ to a finite set $\{\mathit{flow}_1^v, \ldots, \mathit{flow}_k^v\}$ of predicates over $X \cup \dot{X}$ such that there exists a finite open cover $\mathcal{O}^v = \{O_1^v, \ldots, O_k^v\} \in 2^{2^{\mathbb{R}^n}}$ of $[\![\mathit{flow}(v)]\!]$ with $[\![\mathit{flow}_i^v]\!] = [\![\mathit{flow}(v)]\!] \cap O_i^v$ for each $i \in \{1, \ldots, k\}$. For example, for any $\epsilon > 0$, the function $\mathcal{P}$ defined by $\mathcal{P}(\mathit{on}) = \{1 \leq x < 2 + \epsilon, 2 - \epsilon < x \leq 3\}$ and $\mathcal{P}(\mathit{off}) = \{1 \leq x \leq 3\}$ is a flow split for the thermostat automaton in Figure 1. Since $\mathcal{O}^v$ is a cover of $[\![\mathit{flow}(v)]\!]$, *i.e.* $[\![\mathit{flow}(v)]\!] \subseteq \bigcup \mathcal{O}^v$, the flow condition $\mathit{flow}(v)$ is equivalent to the disjunction $\bigvee_{i=1}^{k} \mathit{flow}_i^v$. Since all sets in $\mathcal{O}^v$ are open, the splitting of the flow condition into disjuncts does not prohibit flow transitions.

Given a flow split, we derive a new hybrid automaton such that for every flow transition of the original automaton, the split automaton has a matching sequence of alternating flow transitions and silent jump transitions. Formally, the application of the flow split $\mathcal{P}$ to the hybrid automaton A yields the hybrid automaton $\mathcal{P}(A) = (X_{\mathcal{P}(A)}, V_{\mathcal{P}(A)}, \mathit{flow}_{\mathcal{P}(A)}, E_{\mathcal{P}(A)}, \mathit{jump}_{\mathcal{P}(A)}, \Sigma_{\mathcal{P}(A)}, \mathit{event}_{\mathcal{P}(A)}, \mathit{init}_{\mathcal{P}(A)}, \mathit{final}_{\mathcal{P}(A)})$ with the following components:

- $X_{\mathcal{P}(A)} = X_A$.
- $V_{\mathcal{P}(A)} = \{(v, \varphi) \mid v \in V_A \text{ and } \varphi \in \mathcal{P}(v)\}$.
- For every control mode $(v, \varphi) \in V_{\mathcal{P}(A)}$, define $\mathit{flow}(v, \varphi) = (\mathit{flow}(v) \wedge \varphi)$.
- $E_{\mathcal{P}(A)} = E_1 \cup E_2$, where $E_1 = \{((v, \varphi), (v', \varphi')) \mid (v, v') \in E_A \text{ and } \varphi \in \mathcal{P}(v) \text{ and } \varphi' \in \mathcal{P}(v')\}$ and $E_2 = \{((v, \varphi), (v, \varphi')) \mid \varphi, \varphi' \in \mathcal{P}(v)\}$. The control switches in E_1 are inherited from control switches of A, and the control switches in E_2 are silent control switches that enable the automaton control to pass freely across copies of the same control mode.
- For every control switch $e = ((v, \varphi), (v', \varphi')) \in E_1$, define $\mathit{jump}_{\mathcal{P}(A)}((v, \varphi), (v', \varphi')) = \mathit{jump}_A(v, v')$, *i.e.* the jump condition is inherited from the corresponding control switch of A. For every control switch $e = ((v, \varphi), (v, \varphi')) \in E_2$, define $\mathit{jump}_{\mathcal{P}(A)}((v, \varphi), (v, \varphi')) = \mathit{stable}(X, \dot{X})$, *i.e.* the jump condition requires the state to remain unchanged.
- $\Sigma_{\mathcal{P}(A)} = \Sigma_A$.
- For every control switch $e = ((v, \varphi), (v', \varphi')) \in E_1$, define $\mathit{event}_{\mathcal{P}(A)}((v, \varphi), (v', \varphi')) = \mathit{event}_A(v, v')$. For every control switch $e \in E_2$, define $\mathit{event}_{\mathcal{P}(A)}(e) = \tau$.
- For every control mode $(v, \varphi) \in V_{\mathcal{P}(A)}$, define $\mathit{init}_{\mathcal{P}(A)}(v, \varphi) = (\mathit{init}_A(v) \wedge \varphi)$.
- For every control mode $(v, \varphi) \in V_{\mathcal{P}(A)}$, define $\mathit{final}_{\mathcal{P}(A)}(v, \varphi) = (\mathit{final}_A(v) \wedge \varphi)$.

We define the projection $\pi : Q_{\mathcal{P}(A)} \to Q_A$ on states by $\pi((v, \varphi), \mathbf{a}, \dot{\mathbf{a}}) = (v, \mathbf{a}, \dot{\mathbf{a}})$.

The hybrid automaton A is *splittable* if for every flow split $\mathcal{P}$ for A, the two automata $\mathcal{P}(A)$ and A are timed bisimilar. It may appear to be possible that for some hybrid automaton A and some flow split $\mathcal{P}$ for A, the automaton $\mathcal{P}(A)$ does not time simulate A, because all witnesses for some flow transition of T_A correspond to infinite sequences of flow transitions and silent jump transitions of $T_{\mathcal{P}(A)}$. We show that, due to our requirement that all flow splits are derived from finite open covers, this scenario cannot occur.

Theorem 2.2 *Every hybrid automaton is splittable.*

Proof. Let A be a hybrid automaton, and $\mathcal{P}$ a flow split for A. We show that the relation $\equiv \subseteq Q_A \times Q_{\mathcal{P}(A)}$, defined by $s_1 \equiv s_2$ iff $\pi(s_2) = s_1$, is a time bisimulation between T_A and $T_{\mathcal{P}(A)}$.

First, consider time simulation of $\mathcal{P}(A)$ by A. Suppose that $s_1 \xrightarrow{\delta_1}_{\mathcal{P}(A)} s_2 \xrightarrow{\tau}_{\mathcal{P}(A)} s_3 \xrightarrow{\delta_2}_{\mathcal{P}(A)} s_4$ for nonnegative reals $\delta_1, \delta_2 \in \mathbb{R}_{\geq 0}$, and states s_1, s_2, s_3, and s_4 of $\mathcal{P}(A)$. The jump conditions for all silent control switches imply that the variables and their first derivatives do not change. Hence, if s_2 and s_3 are derived from the same control mode of A, the witnesses for $s_1 \xrightarrow{\delta_1}_{\mathcal{P}(A)} s_2$ and $s_3 \xrightarrow{\delta_2}_{\mathcal{P}(A)} s_4$ can be concatenated to a witness for $\pi(s_1) \xrightarrow{\delta_1+\delta_2}_A \pi(s_4)$. If s_2 and s_3 are derived from different control modes of A, then $s_2 \xrightarrow{\tau}_A s_3$, and therefore $s_1 \xrightarrow{\delta_1}_A s_2 \xrightarrow{\tau}_A s_3 \xrightarrow{\delta_2}_A s_4$. By induction it follows that stutter-closed flow transitions of the form $s \xrightarrow{\delta}_{\mathcal{P}(A)} s'$, with $\delta \in \mathbb{R}_{\geq 0}$, can be time

simulated in A. For stutter-closed jump transitions of the form $s \stackrel{\sigma}{\twoheadrightarrow}_{\mathcal{P}(A)} s'$, with $\sigma \in \Sigma \setminus \{\tau\}$, there exists a state s'' such that $s \stackrel{0}{\twoheadrightarrow}_{\mathcal{P}(A)} s'' \stackrel{\sigma}{\rightarrow}_{\mathcal{P}(A)} s'$. By the time simulation of flow transitions, we need only show that each jump transition $s'' \stackrel{\sigma}{\rightarrow}_{\mathcal{P}(A)} s'$ can be time simulated in A. This is immediate, because all non-silent control switches in $\mathcal{P}(A)$ are directly inherited from A.

Second, consider the time simulation of A by $\mathcal{P}(A)$. Suppose that $s \stackrel{\delta}{\rightarrow}_A s'$ for a nonnegative real $\delta \in \mathbb{R}_{\geq 0}$, and states s and s' of A with the same control mode, say v. Let $\rho : [0, \delta] \rightarrow \mathbb{R}^n$ be a witness for $s \stackrel{\delta}{\rightarrow}_A s'$. Let $\mathcal{O}^v$ be the finite open cover from which the set $\mathcal{P}(v)$ of flow conditions is derived. Let $\xi : (0, \delta) \rightarrow \mathbb{R}^{2n}$ be the function defined by $\xi(t) = (\rho(t), \dot{\rho}(t))$. Since ρ is continuously differentiable, ξ is continuous. For each $t \in (0, \delta)$, let B_t be an open ball that contains $\xi(t)$ and lies entirely in some open set $O \in \mathcal{O}^v$. Such a ball exists, because $\mathcal{O}^v$ is an open cover of $[\![\mathit{flow}(v)]\!]$ and all points in the range of ξ satisfy $\mathit{flow}(v)$. Since ρ is continuously differentiable, it follows that the set $\xi^{-1}(B_t)$ is open, and hence includes an open interval I_t containing t. Thus the set of intervals $\mathcal{I} = \{I_t \mid t \in (0, \delta)\}$ is an open cover for $[0, \delta]$. The Heine-Borel-Lebesgue theorem states that every open cover of a closed and bounded subset of the space of real numbers has a finite subcover. Hence there is a finite open cover of $[0, \delta]$ consisting of intervals in $\mathcal{I}$. Since the cover is open, we can identify a point in the overlap between each pair of consecutive intervals, and construct a finite sequence of witnesses between the endpoints and the intermediate points, with each witness lying entirely within some set from the cover $\mathcal{O}^v$. It follows that $\hat{s} \stackrel{\delta}{\twoheadrightarrow}_{\mathcal{P}(A)} \hat{s}'$ for some states $\hat{s}$ and $\hat{s}'$ of $\mathcal{P}(A)$ with $\pi(\hat{s}) = s$ and $\pi(\hat{s}') = s'$. By induction we conclude that stutter-closed flow transitions of the form $s \stackrel{\delta}{\twoheadrightarrow}_A s'$ can be time simulated in $\mathcal{P}(A)$. The time simulation of stutter-closed jump transitions is again immediate. ■

This theorem enables the control modes of an automaton to be split in order to meet the conditions for applying the clock translation (see Section 3), and to allow more accurate phase-portrait approximation (see Section 4).

2.5 Linearity

The linear hybrid automata form a subclass of hybrid automata that can be analyzed effectively [AHH96]. A *linear term* over a set Y of variables is a linear combination $\theta = \sum_{i=1}^{k} \alpha_i y_i$ of variables $y_i \in Y$ with real-valued coefficients $\alpha_i \in \mathbb{R}$. The linear term θ has *rational coefficients* if all coefficients α_i are rational. The atomic predicate φ over Y is *(rationally) linear* if φ has the form $\theta \sim c$, for a linear term θ over Y (with rational coefficients), a relation symbol $\sim \in \mathit{RelOps}$, and a real (rational) constant c. The predicate φ over Y is *(rationally) linear* if φ is a positive boolean combination of (rationally) linear atomic predicates over Y. If φ is a (rationally) linear predicate, then $[\![\varphi]\!]$ is called a (rationally) linear set.

Consider the hybrid automaton $A = (X, V, \mathit{flow}, E, \mathit{jump}, \Sigma, \mathit{event}, \mathit{init}, \mathit{final})$. A variable $x \in X$ is *(rationally) linear* if in all flow, jump, initial, and final conditions of A, all occurrences of x and x' are contained within (rationally) linear atomic predicates over $X \cup X'$, and all occurrences of $\dot{x}$ and $\dot{x}'$ are contained within (rationally) linear atomic predicates over $\dot{X} \cup \dot{X}'$. Otherwise x is a *nonlinear* variable. The hybrid automaton A is *(rationally) linear* if all variables in X are (rationally) linear. If A is a (rationally) linear hybrid automaton, then every flow, jump, initial, and final condition of A is a (rationally) linear predicate. Furthermore, for every control mode v of A, the flow condition $\mathit{flow}(v)$ is equivalent to a conjunction of the form $\varphi \wedge \dot{\varphi}$, where φ is a predicate over X and $\dot{\varphi}$ is a predicate over $\dot{X}$. Thus, there may be linear dependencies between the derivatives of variables, but the derivative of a variable cannot depend on the value of a variable: the set of flow tangents

is constant for a given control mode. For example, the flow condition $2x \geq y \wedge \dot{x} \leq 3\dot{y} + 2$ is legal for linear hybrid automata, but the flow condition $\dot{x} = x$ is not.

Let $T_A = (Q, Q^I, Q^F, \Sigma, \rightarrow)$ be the timed transition system defined by the hybrid automaton A. A *region* of A is a set of states of A. The region P is *(rationally) linear* if there exist (rationally) linear predicates φ_v over $X \cup \dot{X}$, one for each control mode $v \in V$, such that $P = \bigcup_{v \in V} \{v\} \times [\![\varphi_v]\!]$. For example, if A is a (rationally) linear hybrid automaton, then the region Q of admissible states, the region Q^I of initial states, and the region Q^F of final states are (rationally) linear. We define the *successor* function $post : 2^Q \rightarrow 2^Q$ on regions by $post(P) = \{s_2 \mid \exists s_1 \in P.\, s_1 \rightarrow s_2\}$, and the *predecessor* function $pre : 2^Q \rightarrow 2^Q$ by $pre(P) = \{s_1 \mid \exists s_2 \in P.\, s_1 \rightarrow s_2\}$.

Theorem 2.3 *Let A be a (rationally) linear hybrid automaton. If P is a (rationally) linear region of A, then $post(P)$ and $pre(P)$ are also (rationally) linear regions of A. Moreover, for every rationally linear predicate φ, we can effectively construct rationally linear predicates φ_{post} and φ_{pre} such that $[\![\varphi_{post}]\!] = post([\![\varphi]\!])$ and $[\![\varphi_{pre}]\!] = pre([\![\varphi]\!])$.*

Proof sketch. The proof of the theorem is similar to the proof of the analogous theorem for previous definitions of hybrid automata, where states do not include flow tangents [AHH96]. Here we consider only the *post* operator, and omit many details. It suffices to show that the successor region of a single (rational) state is (rationally) linear. We compute the successor states via jump transitions and the successor states via flow transitions separately; the former can be computed as in [AHH96], by treating the variables in $\dot{X}$ like those in X.

So consider flow transitions with the source state $s = (v, \mathbf{a}, \dot{\mathbf{a}})$. We assume that the flow condition of v has the form $\varphi \wedge \dot{\varphi}$ for convex linear predicates φ over X and $\dot{\varphi}$ over $\dot{X}$; nonconvex flow conditions can be treated as in [AHH96], by splitting into convex parts. We need to compute the possible target points $\mathbf{b}$, and the possible flow tangents $\dot{\mathbf{b}}$ for these points, such that $s \xrightarrow{\delta} (v, \mathbf{b}, \dot{\mathbf{b}})$ for some $\delta \in \mathbb{R}_{\geq 0}$. If $\delta = 0$, then $\mathbf{b} = \mathbf{a}$ and $\dot{\mathbf{b}} = \dot{\mathbf{a}}$. For the case $\delta > 0$, we compute the region $post_{>0}(s) = \{s' \mid \exists \delta \in \mathbb{R}_{>0}.\, s \xrightarrow{\delta} s'\}$, where $\mathbb{R}_{>0}$ is the set of positive reals. Since $\mathit{flow}(v)$ is convex, for flow transitions of positive duration, any target point $\mathbf{b}$ can be reached with flow tangent $\dot{\mathbf{b}}$ iff there is a different target point $\mathbf{b}'$ such that the straight line from $\mathbf{b}'$ to $\mathbf{b}$ has direction $\dot{\mathbf{b}}$. Define $\widetilde{post} : \mathbb{R}^n \rightarrow 2^{\mathbb{R}^n}$ by $\widetilde{post}(\mathbf{a}) = \{\mathbf{b} \mid \varphi(\mathbf{a}) \wedge \varphi(\mathbf{b}) \wedge (\mathbf{b} = \mathbf{a} \vee (\exists \delta \in \mathbb{R}_{>0}.\, \dot{\varphi}[\dot{X} := (\mathbf{b} - \mathbf{a})/\delta]))\}$. Then

$$
\begin{aligned}
post_{>0}(v, \mathbf{a}, \dot{\mathbf{a}}) \;=\; \{(v, \mathbf{b}, \dot{\mathbf{b}}) \mid\; & (\mathbf{a}, \dot{\mathbf{a}}) \in [\![\mathit{flow}(v)]\!] \text{ and} \\
& (\mathbf{b}, \dot{\mathbf{b}}) \in [\![\mathit{flow}(v)]\!] \text{ and} \\
& \mathbf{b} \in \widetilde{post}(\mathbf{a}) \text{ and} \\
& \exists k_1 \in \mathbb{R}_{>0}.\, (\mathbf{a} + k_1 \dot{\mathbf{a}}) \in \widetilde{post}(\mathbf{a}) \text{ and} \\
& \exists k_2 \in \mathbb{R}_{>0}.\, (\mathbf{b} - k_2 \dot{\mathbf{b}}) \in \widetilde{post}(\mathbf{a})\}.
\end{aligned}
$$

This region is linear. If the flow condition of v is rationally linear, then for rational vectors $\mathbf{a}$ and $\dot{\mathbf{a}}$, the region $post_{>0}(v, \mathbf{a}, \dot{\mathbf{a}})$ is rationally linear. ■

The function that results from composing the *post* operator i times is denoted $post^i$. Then, the region of reachable states of the hybrid automaton A is $reach(A) = \bigcup_{i=0}^{\infty} post^i(Q^I)$. Even if A is rationally linear, there may not be a rationally linear representation for $reach(A)$ because of the infinite union. If, however, a state s of A is reachable, then $s \in post^i(Q^I)$ for some nonnegative integer i. It follows that for rationally linear hybrid automata, there is an effective procedure that terminates if a final state is reachable, but may not necessarily terminate if no final state is reachable.

Corollary 2.4 *The nonemptiness problem for rationally linear hybrid automata ("Given a rationally linear hybrid automaton A, is A nonempty?") is recursively enumerable.* ■

Algorithmic analysis techniques for rationally linear hybrid automata have been implemented in tools such as HyTech [HHW95b] and Polka [HRP94]. In particular, when applied to an unsafe system, the emptiness-checking procedures of these tools are guaranteed to detect a violation of the safety property. Experiments to date show that for many real-world examples, there is a nonnegative integer k such that all reachable states can be reached within k jump and flow transitions, *i.e.* $post^{k+1}(Q^I) \subseteq \bigcup_{i=0}^{k} post^i(Q^I)$. In these cases, the safety checks terminate also when applied to a safe system.

Remark. The safety of a rationally linear hybrid automaton can be checked, alternatively, by iterating the *pre* operator, starting from the region Q^F of final states. The *pre* operator is also useful for checking more general classes of properties than safety [AHH96].

3 Clock Translation

For certain nonlinear hybrid automata, we can construct timed bisimilar linear hybrid automata, by replacing all nonlinear variables with clocks. For a hybrid automaton A, the variable t is a *clock* if t is linear and all flow conditions of A imply $\dot{t} = 1$. We can replace a nonlinear variable x by a clock t_x if at all times the value of x can be determined uniquely from the value of t_x. This is the case if t_x measures the time that has elapsed since the value of x was last changed by a control switch, if the value of x after that change is recorded, and if x has followed a unique flow since that change. The variables that can be replaced by clocks are called solvable.

3.1 Solvability

Before giving formal definitions, we intuitively describe the conditions we require for a variable x to be solvable. First, we require that x be independent of the other variables in flow, jump, initial, and final conditions. Thus we need not consider how to translate relationships between x and the other variables into relationships between t_x and the other variables. Second, we require x to follow a unique flow from any starting point. This restriction enables us to determine the value of x if we know its initial value and the elapsed time. Third, in order to determine the truth of atomic predicates such as $x \geq c$, for any constant $c \in \mathbb{R}$, from the value of t_x, we require the flows of x to be strictly monotone. For example, if x has initial value b, strictly less than c, and is strictly increasing, we know that $x \geq c$ is true iff $t_x \geq a$ where a is the time it takes for the unique flow of x to progress from b to c. Fourth, we require that at all times we know the initial value of the current flow of x and the elapsed time. For example, suppose that x is 1 when the automaton control enters the control mode v, and the flow condition of v implies $\dot{x} = x$. Suppose that the automaton control switches to another control mode v' when the variable y equals 2. If the flow condition of v' also implies $\dot{x} = x$, then the value of x can be determined from the time that has elapsed since the control entered v, by $x = e^{t_x}$, regardless of when the control switched to v'. However, if $\mathit{flow}(v')$ differs from $\mathit{flow}(v)$ and the value of x at the control switch is not known, then it is no longer possible to determine the current value of x. Thus, we say that the variable x is definite for the control switch e if the jump condition of e implies $x' = c$ for some constant $c \in \mathbb{R}$, and we require control switches to be definite for x whenever the flow conditions for the source and target modes differ.

We now formalize these concepts. An atomic predicate is *simple* if it has the form $x \sim c$ or $x' = c$ or $x' = x$, where x is a variable, $\sim \in \mathit{RelOps}$ is a relational operator, and $c \in \mathbb{R}$ is a constant. The predicate φ is *simple for* x if all occurrences of x and x' in φ are contained within simple

atomic predicates, and $\dot{x}$ and $\dot{x}'$ do not occur in φ. The variable x is *independent* in the hybrid automaton A if the following conditions hold:

- Every jump, initial, and final condition of A is simple for x.
- For every control mode v of A, the flow condition $\mathit{flow}(v)$ has the form $(\dot{x} = f_x^v(x)) \wedge \mathit{inv}_x \wedge \mathit{flow}_Y$ for a function $f_x^v : \mathbb{R} \to \mathbb{R}$, a simple predicate inv_x over $\{x\}$, and a predicate flow_Y over $Y \cup \dot{Y}$, where $Y = X \setminus \{x\}$. The function f_x^v is called the *flow function* for x in the control mode v.

The independent variable x is *monotonically determined* in the control mode v of A if for all reals $c \in \mathbb{R}$, the initial-value problem "$\dot{y}(t) = f_x^v(y(t)); y(0) = c$" has a unique continuous solution $g(t)$, and that solution is strictly monotone (this is the case, for instance, if $f_x^v(y) \neq 0$ for all $y \in \mathbb{R}$). The variable x is *initially definite* for the control mode v if the initial condition $\mathit{init}(v)$ is either *false* or implies $x = c$, for some constant $c \in \mathbb{R}$. The constant c is called the *initial value* of x for v. The variable x is *definite* for the control switch e of A if the jump condition $\mathit{jump}(e)$ implies $x' = d$, for some constant $d \in \mathbb{R}$. The constant d is called the *arrival value* of x for e.

The nonlinear variable x of the hybrid automaton A is *solvable* if the following three conditions hold:

1. The variable x is independent.
2. For all control modes v of A, the variable x is initially definite and monotonically determined in v.
3. For all control switches $e = (v, v')$ of A, if the variable x is not definite for e, then v and v' have the same flow functions for x, *i.e.* $f_x^v = f_x^{v'}$, and the jump condition $\mathit{jump}(e)$ implies $x' = x$.

The hybrid automaton A is *solvable* if all nonlinear variables of A are solvable. For example, the thermostat automaton of Figure 1 is solvable, since x is the only nonlinear variable and x is solvable.

Remark. While our requirements for solvability are convenient and easily checkable, they are, of course, unnecessarily strong and can be relaxed in various ways. For instance, the strict monotonicity of solutions can be replaced by the requirement that the unique solution $g(t)$ of the initial-value problem is such that for each constant $c \in \mathbb{R}$ that appears in an atomic predicate of the form $x \sim c$ or $x' \sim c$ in certain invariant and jump conditions, if there exists a time $t \geq 0$ with $g(t) = c$, then t is unique.

3.2 The clock-translation algorithm

The clock-translation algorithm replaces a solvable variable x of a hybrid automaton A by a clock t_x such that the resulting hybrid automaton is timed bisimilar to A. In this way, if A is solvable, then all nonlinear variables of A can be replaced one by one, and the final result is a linear hybrid automaton that is timed bisimilar to A.

Consider a solvable variable x of the hybrid automaton $A = (X_A, V_A, \mathit{flow}_A, E_A, \mathit{jump}_A, \Sigma_A, \mathit{event}_A, \mathit{init}_A, \mathit{final}_A)$. The constant $c \in \mathbb{R}$ is a *starting value* for $x \in X_A$ if there exists a control mode $v \in V_A$ such that c is the initial value of x for v, or there exists a control switch $e \in E_A$ such that c is the arrival value of x for v. Let $\mathit{Start}_A(x) = \{c_1, \ldots, c_k\}$, with $c_1 < \cdots < c_k$, be the set of starting values for x in A. The clock-translation algorithm proceeds in two steps:

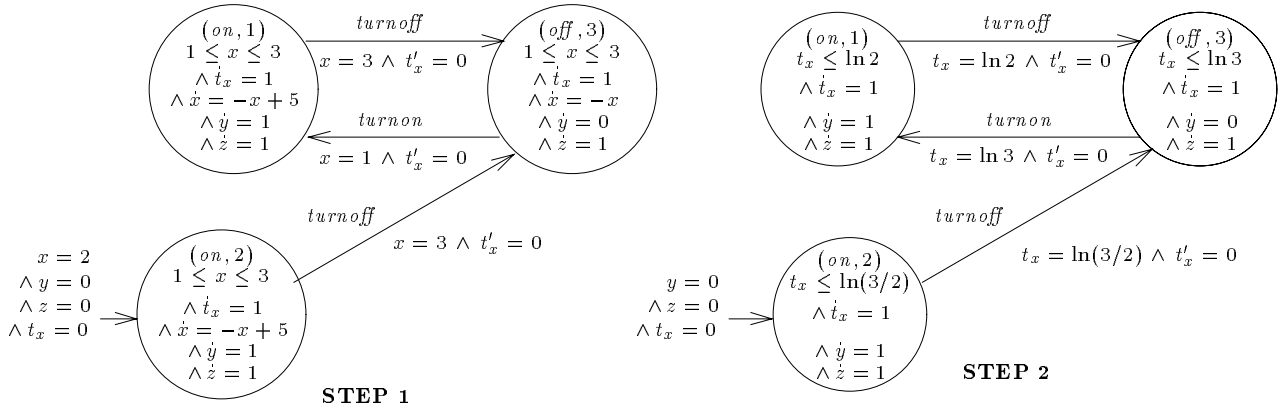


Figure 2: Clock translation of the thermostat automaton

1. Each control mode v of A is split into a collection $(v, c_1), \ldots, (v, c_k)$ of control modes, one for each starting value c_i of x. We then add the clock t_x such that the value of x in the control mode (v, c_i) is $g(t_x)$, where $g(t)$ is the unique solution of the initial-value problem "$\dot{y}(t) = f_x^v(y(t))$; $y(0) = c_i$."

2. For all invariant, jump, and final conditions of A, each atomic subformula over $\{x\}$ is replaced by an atomic predicate over $\{t_x\}$. Then the variable x can be discarded.

Remark. For simplicity, we consider a global set of starting values for each variable. If the starting values are parametrized by control modes, in Step 1 the control modes can be split more selectively.

Step 1: adding the clock t_x

The result of this step is the hybrid automaton $B = (X_B, V_B, \mathit{flow}_B, E_B, \mathit{jump}_B, \Sigma_B, \mathit{event}_B, \mathit{init}_B, \mathit{final}_B)$ with the following components:

- The variables of B are $X_B = X_A \cup \{t_x\}$, and the events of B are $\Sigma_B = \Sigma_A$.

- The control modes of B are $V_B = V_A \times \mathit{Start}_A(x)$. Each control mode $v_i = (v, c_i)$ of B has the flow condition $\mathit{flow}_B(v_i) = (\mathit{flow}_A(v) \wedge (\dot{t}_x = 1))$, reflecting the fact that the new variable t_x is a clock. The control mode v_i has the initial condition $\mathit{init}_B(v_i) = (\mathit{init}_A(v) \wedge (t_x = 0))$ if $\mathit{init}_A(v)$ implies $x = c_i$, and $\mathit{init}_B(v_i) = \mathit{false}$ otherwise. The control mode v_i has the final condition $\mathit{final}_B(v_i) = \mathit{final}_A(v)$.

- For each control switch $e = (v, v')$ of A for which x is not definite, the automaton B has, for each $c_i \in \mathit{Start}_A(x)$, a control switch of the form $e_i = ((v, c_i), (v', c_i))$ with the jump condition $\mathit{jump}_B(e_i) = (\mathit{jump}_A(e) \wedge (t_x = t'_x))$ and the event label $\mathit{event}_B(e_i) = \mathit{event}_A(e)$. For each control switch $e = (v, v')$ of A for which x is definite with arrival value c_j, the automaton B has, for each $c_i \in \mathit{Start}_A(x)$, a control switch of the form $e_{i,j} = ((v, c_i), (v', c_j))$ with the jump condition $\mathit{jump}_B(e_{ij}) = (\mathit{jump}_A(e) \wedge (t'_x = 0))$ and the event label $\mathit{event}_B(e_{ij}) = \mathit{event}_A(e)$.

Example 3.1 We apply Step 1 to the variable x of the thermostat automaton from Figure 1. All control switches are definite for x. The starting values of x are 1, 2, and 3, so we split both control modes *on* and *off* into three control modes each. Since the control modes $(on, 3)$, $(off, 1)$, and $(off, 2)$ are not reachable by a sequence of control switches from the initial control mode $(on, 2)$, we omit these three control modes from the clock-translated automaton. The result of Step 1 is shown on the left in Figure 2. ■

Step 2: replacing the conditions on x by conditions on t_x

Let $g_c(t)$ be the unique solution of the initial-value problem "$\dot{y}(t) = f_x^v(y(t)); y(0) = c$," for $c \in \mathbb{R}$. Since $g_c(t)$ is strictly monotone, for each $d \in \mathbb{R}$ there is at most one $t \in \mathbb{R}_{\geq 0}$ such that $g_c(t) = d$. Let $g_c^{-1}(d) = t$ if $g_c(t) = d$, and $g_c^{-1}(d) = \perp$ if $g_c(t) \neq d$ for all $t \in \mathbb{R}_{\geq 0}$. The transformation function α_c from simple atomic predicates over $\{x\}$ to simple atomic predicates over $\{t_x\}$ is defined as follows:

$$\alpha_c(x \sim d) \;=\; \begin{cases} \mathit{true} & \text{if } g_c^{-1}(d) = \perp \text{ and } c \sim d, \\ \mathit{false} & \text{if } g_c^{-1}(d) = \perp \text{ and } c \not\sim d, \\ t_x \; lt(\sim)\; g_c^{-1}(d) & \text{if } g_c^{-1}(d) \neq \perp \text{ and } c \sim d, \\ t_x \; gt(\sim)\; g_c^{-1}(d) & \text{if } g_c^{-1}(d) \neq \perp \text{ and } c \not\sim d, \end{cases}$$

where $\sim \in \mathit{RelOps}$ is a relational operator, and $lt : \mathit{RelOps} \to \mathit{RelOps}$ and $gt : \mathit{RelOps} \to \mathit{RelOps}$ are defined by the following table:

op	$lt(op)$	$gt(op)$
$<$	$<$	$>$
$\leq$	$\leq$	$\geq$
$=$	$=$	$=$
$\geq$	$\leq$	$\geq$
$>$	$<$	$>$

Predicates using the $lt(\sim)$ operators correspond to constraints on how long the $x \sim d$ predicate will remain true. We conduct the following four steps for each control mode $v_i = (v, c_i)$ of the hybrid automaton B:

1. In the flow condition of v_i, replace by *true* each atomic predicate that contains the variable $\dot{x}$. Then replace each atomic predicate of the form $x \sim d$, for $\sim \in \mathit{RelOps}$, by $\alpha_{c_i}(x \sim d)$ if $c_i \sim d$, and by *false* otherwise (in the latter case, the control mode v_i may be removed).

2. For each control switch e of B with the source v_i, in the jump condition of e, replace by *true* each atomic predicate that contains the variable x', and replace each atomic predicate of the form $x \sim d$, for $\sim \in \mathit{RelOps}$, by $\alpha_{c_i}(x \sim d)$.

3. In the initial condition of v_i, replace by *true* each atomic predicate that contains the variable x.

4. In the final condition of v_i, replace each atomic predicate of the form $x \sim d$, for $\sim \in \mathit{RelOps}$, by $\alpha_{c_i}(x \sim d)$.

The resulting hybrid automaton C is called the *clock translation* of A with respect to x.

Example 3.2 In the thermostat example, we have the solutions $g_1(t) = -4e^{-t} + 5$ and $g_2(t) = -3e^{-t} + 5$ for x in the control mode *on*, and the solution $g_3(t) = 3e^{-t}$ for x in the control mode *off*. Consider the atomic predicate $x = 3$ of the jump condition of the control switch from $(on, 2)$

to $(\mathit{off}, 3)$. Since $-3e^{-t} + 5 = 3$ implies $t = \ln(3/2)$, it follows that $x = 3$ iff $t_x = \ln(3/2)$. Hence the atomic predicate $x = 3$ is replaced by $t_x = \ln(3/2)$. The final result of Step 2 is shown on the right in Figure 2. ■

3.3 Correctness

Let x be a solvable variable of the hybrid automaton A, and let C be the clock translation of A with respect to x. We show that A and C are timed bisimilar. Let Q_A be the set of admissible states of A, and let Q_C be the set of admissible states of C. Define $\beta : Q_C \to Q_A$ such that $\beta((v, c), \mathbf{a}_1, \dot{\mathbf{a}}_1) = (v, \mathbf{a}_2, \dot{\mathbf{a}}_2)$, where the valuations $\mathbf{a}_1$ and $\mathbf{a}_2$ agree on all variables except x and t_x, the valuations $\dot{\mathbf{a}}_1$ and $\dot{\mathbf{a}}_2$ agree on all variables except $\dot{x}$ and $\dot{t}_x$, and $\mathbf{a}_2(x) = g_c(\mathbf{a}_1(t_x))$ and $\dot{\mathbf{a}}_2(\dot{x}) = f_x^v(g_c(\mathbf{a}_1(t_x)))$ for the solution $g_c(t)$ of the initial-value problem "$\dot{y}(t) = f_x^v(y(t)); y(0) = c$." Define $\equiv_\beta \subseteq Q_C \times Q_A$ by $s_1 \equiv_\beta s_2$ iff $s_2 = \beta(s_1)$. We prove that $\equiv_\beta$ is a time bisimulation between the hybrid automata C and A.

Lemma 3.1 *If* $((v_1, c), \mathbf{a}_1, \dot{\mathbf{a}}_1) \equiv_\beta (v_2, \mathbf{a}_2, \dot{\mathbf{a}}_2)$, *then* $\mathbf{a}_1 \in [\![\alpha_c(x \sim d)]\!]$ *iff* $\mathbf{a}_2 \in [\![x \sim d]\!]$.

Proof. Let $s_1 = ((v, c), \mathbf{a}_1, \dot{\mathbf{a}}_1)$ be an admissible state of C, and let $s_2 = (v, \mathbf{a}_2, \dot{\mathbf{a}}_2)$ be an admissible state of A such that $s_1 \equiv_\beta s_2$. Let $g_c(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_x^v(y(t)); y(0) = c$." Recall that $g_c(t)$ is continuous and strictly monotone. Let $d \in \mathbb{R}$ be any real. We consider the four cases that arise from the definition of α_c:

- Assume that $g_c^{-1}(d) = \bot$ and $c \sim d$. Since $g_c(t) \not\sim d$ for all $t \geq 0$, and $g_c(0) = c$, in this case $\sim$ cannot be the equality relation. By the continuity of $g_c(t)$, we have $g_c(t) \sim d$ for all $t \geq 0$. Hence $\mathbf{a}_2(x) = g_c(\mathbf{a}_1(t_x)) \sim d$.

- Assume that $g_c^{-1}(d) = \bot$ and $c \not\sim d$. If $\sim$ is the equality relation, $g_c^{-1}(d) = \bot$ implies $\mathbf{a}_2(x) = g_c(\mathbf{a}_1(t_x)) \neq d$. If $\sim$ is an inequality, then by continuity, $g_c(t) \not\sim d$ for all $t \geq 0$, which implies $\mathbf{a}_2(x) = g_c(\mathbf{a}_1(t_x)) \not\sim d$.

- Assume that $g_c^{-1}(d) \neq \bot$ and $c \sim d$. If $\sim$ is the equality relation, then $g_c(0) = c = d$, and $g_c(t) \neq d$ for all $t > 0$. Therefore $\mathbf{a}_2(x) = d$ iff $g_c(\mathbf{a}_1(t_x)) = d$ iff $\mathbf{a}_1(t_x) = 0$ iff $\mathbf{a}_1(t_x) = g_c^{-1}(d)$. For inequalities φ of the form $x < d$ or $x > d$, by the strict monotonicity of $g_c(t)$, we have $\mathbf{a}_2 \in [\![\varphi]\!]$ iff $\mathbf{a}_1 \in [\![t_x < g_c^{-1}(d)]\!]$, because x reaches the cutoff value d precisely when t_x reaches the cutoff value $g_c^{-1}(d)$. Similarly, for inequalities φ of the form $x \leq d$ or $x \geq d$, we have $\mathbf{a}_2 \in [\![\varphi]\!]$ iff $\mathbf{a}_1 \in [\![t_x \leq g_c^{-1}(d)]\!]$.

- Assume that $g_c^{-1}(d) \neq \bot$ and $c \not\sim d$. The proof is analogous to the previous case. ■

Lemma 3.2 $\equiv_\beta$ *is a time simulation of* T_A *by* T_C.

Proof. Let $s_2 = (v, \mathbf{a}_2, \dot{\mathbf{a}}_2)$ and $s_2' = (v', \mathbf{a}_2', \dot{\mathbf{a}}_2')$ be two admissible states of A such that $s_2 \xrightarrow{m}_A s_2'$. Let $s_1 = ((v, c), \mathbf{a}_1, \dot{\mathbf{a}}_1)$ be an admissible state of C such that $s_1 \equiv_\beta s_2$. We show that there exists an admissible state s_1' of C such that $s_1 \xrightarrow{m}_C s_1'$ and $s_1' \equiv_\beta s_2'$.

First, consider flow transitions. Suppose that $s_2 \xrightarrow{\delta}_A s_2'$ has the duration $\delta \geq 0$ and the witness $\rho_2 : [0, \delta] \to \mathbb{R}^n$. We construct a witness $\rho_1 : [0, \delta] \to \mathbb{R}^n$ for a flow transition of C originating from s_1 and having duration δ. Let $t_0 = \mathbf{a}_1(t_x)$. For all $t \in [0, \delta]$, define $\rho_1(t)$ such that $\rho_1(t)(y) = \rho_2(t)(y)$ for $y \neq t_x$, and $\rho_1(t)(t_x) = t_0 + t$. We show that $((v, c), \rho_1(t), \dot{\rho}_1(t)) \equiv_\beta (v, \rho_2(t), \dot{\rho}_2(t))$ for all $t \in [0, \delta]$. Let $g_c(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_x^v(y(t)); y(0) = c$." Since $((v, c), \rho_1(0), \dot{\rho}_1(0)) \equiv_\beta (v, \rho_2(0), \dot{\rho}_2(0))$, we have $\rho_2(0)(x) = g_c(\rho_1(0)(t_x)) = g_c(t_0)$. Therefore

for all $t \in [0,\delta]$, we have $\rho_2(t)(x) = g_c(t_0+t) = g_c(\rho_1(t)(t_x))$ and $\dot{\rho}_2(t)(\dot{x}) = f_c^v(g_c(t_0+t)) = f_c^v(g_c(\rho_1(t)(t_x)))$. It remains to be shown that ρ_1 witnesses a flow transition of C, *i.e.* the flow condition of (v,c) is satisfied along ρ_1. This follows from the construction of $inv_C(v,c)$, which ensures, by Lemma 3.1, that for all $t \in [0,\delta]$, if $((v,c),\rho_1(t),\dot{\rho}_1(t)) \equiv_\beta (v,\rho_2(t),\dot{\rho}_2(t))$ and $\rho_2(t) \in [\![inv_A(v)]\!]$, then $\rho_1(t) \in [\![inv_C(v,c)]\!]$.

Second, consider jump transitions. Suppose that $s_2 \xrightarrow{\sigma}_A s_2'$ has as witness the control switch $e_2 = (v,v')$ of A. We consider two cases:

1. Assume that $jump_A(e_2)$ implies $x' = d$ for some real $d \in \mathbb{R}$. In this case, there exists a control switch $e_1 = ((v,c),(v',d))$ of C derived from e_2 such that $jump_C(e_1)$ implies $t_x' = 0$ and $event_C(e_1) = \sigma$. Define $\mathbf{a}_1'$ such that $\mathbf{a}_1'(y) = \mathbf{a}_2'(y)$ for $y \neq t_x$, and $\mathbf{a}_1'(t_x) = 0$. Define $\dot{\mathbf{a}}_1'$ such that $\dot{\mathbf{a}}_1'(\dot{y}) = \dot{\mathbf{a}}_2'(\dot{y})$ for $\dot{y} \neq \dot{t}_x$, and $\dot{\mathbf{a}}_1'(\dot{t}_x) = 1$. Define $s_1' = ((v',d),\mathbf{a}_1',\dot{\mathbf{a}}_1')$. We show that $s_1' \equiv_\beta s_2'$. Let $g_d(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_x^{v'}(y(t)); y(0) = d$." Since $(\mathbf{a}_2,\dot{\mathbf{a}}_2,\mathbf{a}_2',\dot{\mathbf{a}}_2') \in [\![jump_A(e_2)]\!]$, we have $\mathbf{a}_2'(x) = d = g_d(0) = g_d(\mathbf{a}_1'(t_x))$ and then since $(\mathbf{a}_2',\dot{\mathbf{a}}_2') \in [\![flow_A(v')]\!]$, we have $\dot{\mathbf{a}}_2'(\dot{x}) = f_x^{v'}(\mathbf{a}_2'(x)) = f_x^{v'}(g_d(\mathbf{a}_1'(t_x)))$. Admissibility of s_1' follows from $s_1' \equiv_\beta s_2'$ and Lemma 3.1. It remains to be shown that e_1 witnesses the jump transition $s_1 \xrightarrow{\sigma}_C s_1'$ of C. This follows from the construction of $jump_C(e_1)$, which ensures, by Lemma 3.1, that $s_1 \equiv_\beta s_2$ and $s_1' \equiv_\beta s_2'$ and $(\mathbf{a}_2,\dot{\mathbf{a}}_2,\mathbf{a}_2',\dot{\mathbf{a}}_2') \in [\![jump_A(e_2)]\!]$ imply $(\mathbf{a}_1,\dot{\mathbf{a}}_1,\mathbf{a}_1',\dot{\mathbf{a}}_1') \in [\![jump_C(e_1)]\!]$.

2. Assume that $jump_A(e_2)$ implies $x' = x$. In this case, there exists a control switch $e_1 = ((v,c),(v',c))$ of C derived from e_2 such that $jump_C(e_1)$ implies $t_x' = t_x$ and $event_C(e_1) = \sigma$. Define $\mathbf{a}_1'$ such that $\mathbf{a}_1'(y) = \mathbf{a}_2'(y)$ for all $y \neq t_x$, and $\mathbf{a}_1'(t_x) = \mathbf{a}_1(t_x)$. Define $\dot{\mathbf{a}}_1'$ such that $\dot{\mathbf{a}}_1'(\dot{y}) = \dot{\mathbf{a}}_2'(\dot{y})$ for all $\dot{y} \neq \dot{t}_x$, and $\dot{\mathbf{a}}_1'(\dot{t}_x) = 1$. Define $s_1' = ((v',c),\mathbf{a}_1',\dot{\mathbf{a}}_1')$. We show that $s_1' \equiv_\beta s_2'$. Let $g_c(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_x^v(y(t)); y(0) = c$." Since $(\mathbf{a}_2,\dot{\mathbf{a}}_2,\mathbf{a}_2',\dot{\mathbf{a}}_2') \in [\![jump_A(e_2)]\!]$, we have $\mathbf{a}_2'(x) = \mathbf{a}_2(x)$. Thus, since $s_1 \equiv_\beta s_2$, we have $\mathbf{a}_2'(x) = \mathbf{a}_2(x) = g_c(\mathbf{a}_1(t_x)) = g_c(\mathbf{a}_1'(t_x))$ and then since $(\mathbf{a}_2',\dot{\mathbf{a}}_2') \in [\![flow_A(v')]\!]$, we have $\dot{\mathbf{a}}_2'(\dot{x}) = f_x^{v'}(\mathbf{a}_2'(x)) = f_x^{v'}(g_c(\mathbf{a}_1'(t_x)))$. Similar to the previous case, it can be shown that s_1' is admissible and that e_1 witnesses the jump transition $s_1 \xrightarrow{\sigma}_C s_1'$ of C.

Finally, we need to consider the initial and final states. Let $s_2 = (v,\mathbf{a}_2,\dot{\mathbf{a}}_2)$ be an initial state of A. Since x is initially definite for v, the initial condition $init_A(v)$ implies $x = d$ for some $d \in \mathbb{R}$. Analogously to Case 1 for jump transitions, we can find an initial state s_1 of C such that $s_1 \equiv_\beta s_2$. If s_2 is a final state of A and $s_1 \equiv_\beta s_2$, then the construction of $final_C$ ensures, by Lemma 3.1, that s_1 is a final state of C. ■

Lemma 3.3 *$\equiv_\beta^{-1}$ is a time simulation of T_C by T_A.*

Proof. Let $s_1 = ((v,c),\mathbf{a}_1,\dot{\mathbf{a}}_1)$ and $s_1' = ((v',c'),\mathbf{a}_1',\dot{\mathbf{a}}_1')$ be two admissible states of C. Let $\beta(s_1) = s_2 = (v,\mathbf{a}_2,\dot{\mathbf{a}}_2)$ and $\beta(s_1') = s_2' = (v',\mathbf{a}_2',\dot{\mathbf{a}}_2')$. We show that $s_1 \xrightarrow{m}_C s_1'$ implies $s_2 \xrightarrow{m}_A s_2'$.

First, consider flow transitions. Suppose that $s_1 \xrightarrow{\delta}_C s_1'$ for some duration $\delta \geq 0$ and witness $\rho_1 : [0,\delta] \to \mathbb{R}^n$. In this case $v' = v$ and $c' = c$. Let $g_c(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_x^v(y(t)); y(0) = c$." Let $t_0 = \mathbf{a}_1(t_x)$. Then $\mathbf{a}_1'(t_x) = t_0 + \delta$. Define $\rho_2 : [0,\delta] \to \mathbb{R}^n$ such that for all $t \in [0,\delta]$, we have $\rho_2(t)(y) = \rho_1(t)(y)$ for all $y \neq x$, and $\rho_2(t)(x) = g_c(t_0+t)$. By the definition of β, it follows that $\beta((v,c),\rho_1(t),\dot{\rho}_1(t)) = (v,\rho_2(t),\dot{\rho}_2(t))$ for all $t \in [0,\delta]$. We claim that ρ_2 is a witness for $s_2 \xrightarrow{\delta}_A s_2'$. This follows from the construction of $inv_C(v,c)$, which ensures, by Lemma 3.1, that for all $t \in [0,\delta]$, since $\rho_1(t) \in [\![inv_A(v)]\!]$ also $\rho_2(t) \in [\![inv_C(v,c)]\!]$.

Second, consider jump transitions. Suppose that $s_1 \xrightarrow{\sigma}_C s_1'$ has as witness the control switch $e_1 = ((v,c),(v',c'))$ of C. Then there is a control switch $e_2 = (v,v')$ of A from which e_1 is derived

such that $event_A(e_2) = \sigma$. We claim that e_2 is a witness for $s_2 \xrightarrow{\sigma}_A s_2'$. Since $jump_A(e_2)$ is simple for x, we need to consider only the atomic subformulas of $jump_A(e_2)$ that contain x or x'. The atomic subformulas of the form $x \sim d$ are covered by Lemma 3.1. There are two types of atomic subformulas that contain x':

1. If $jump_A(e_2)$ implies $x' = d$, for some $d \in \mathbb{R}$, then $jump_C(e_1)$ implies $t_x' = 0$. Since $\mathbf{a}_1'(t_x) = 0$, we have $\mathbf{a}_2'(x) = d$ as required.

2. If $jump_A(e_2)$ implies $x' = x$, then $c' = c$ and $jump_C(e_1)$ implies $t_x' = t_x$. Since $\mathbf{a}_1'(t_x) = \mathbf{a}_1(t_x)$, we have $\mathbf{a}_2'(x) = \mathbf{a}_2(x)$ as required.

The conditions on the initial and final states follow from similar considerations. ■

Theorem 3.4 *If x is a solvable variable of the hybrid automaton A, and C is the clock translation of A with respect to x, then A and C are timed bisimilar.*

Let A be a solvable hybrid automaton with the nonlinear variables $x_1, \ldots, x_k$. Let $A_0 = A$ and for $i \in \{1, \ldots, k\}$, let A_i be the clock translation of A_{i-1} with respect to x_i. The linear hybrid automaton $C_A = A_k$ is called the *clock linearization* of A. If C_A is rationally linear, then the hybrid automaton A is called *rationally solvable*. By Theorem 3.4 and the transitivity of timed bisimilarity, it follows that A and C_A are timed bisimilar. By Proposition 2.1, it follows that A is nonempty iff C_A is nonempty.

Corollary 3.5 *The nonemptiness problem for rationally solvable hybrid automata is recursively enumerable.*

Remark. The nonemptiness problem is known to be recursive for certain classes of rationally linear hybrid automata, such as timed automata and initialized rectangular automata [AD94, HKPV95]. For each such class, we can formulate a corresponding decidability result for the nonemptiness problem of nonlinear hybrid automata whose clock linearizations fall into the class.

4 Linear Phase-Portrait Approximation

Since the clock translation applies only to solvable hybrid automata, it is desirable to have a theory of conservative approximations for linearizing a wider class of systems. Moreover, often the clock linearization of a nonlinear hybrid automaton is not rationally linear, and needs to be approximated using rational coefficients before analysis with HyTech is possible. This, for example, is the case for the thermostat automaton of Figure 1, whose clock linearization is linear but not rationally linear (see Figure 2).

We advocate the use of linear phase-portrait approximations. Essentially, for each control mode of a hybrid automaton, the state space is partitioned into linear regions, and within each region, the flow field is overapproximated using linear sets of flow vectors. The approximations may be obtained manually, leveraging techniques from dynamics theory, or in some cases automatically, when lower and upper bounds on derivatives can be obtained from bounds on the values of the variables [HW96a]. The approximations can be made arbitrarily accurate by approximating over suitably small regions of the state space. Furthermore, initial approximations may be successively refined with the help of automated analysis, as demonstrated in Subsection 4.2.

4.1 Phase-portrait approximations

A hybrid automaton is time simulated (and therefore approximated) by any hybrid automaton that results from relaxing flow, jump, initial, and/or final conditions. Formally, the hybrid automaton $A = (X_A, V_A, \mathit{flow}_A, E_A, \mathit{jump}_A, \Sigma_A, \mathit{event}_A, \mathit{init}_A, \mathit{final}_A)$ is a *basic phase-portrait approximation* of the hybrid automaton $B = (X_B, V_B, \mathit{flow}_B, E_B, \mathit{jump}_B, \Sigma_B, \mathit{event}_B, \mathit{init}_B, \mathit{final}_B)$ if the following conditions hold:

- $X_B = X_A$ and $V_B = V_A$ and $E_B = E_A$ and $\Sigma_B = \Sigma_A$ and $\mathit{event}_B = \mathit{event}_A$.
- For all control modes v, the predicate $\mathit{flow}_B(v)$ implies the predicate $\mathit{flow}_A(v)$, the predicate $\mathit{init}_B(v)$ implies the predicate $\mathit{init}_A(v)$, and the predicate $\mathit{final}_B(v)$ implies the predicate $\mathit{final}_A(v)$.
- For all control switches e, the predicate $\mathit{jump}_B(e)$ implies the predicate $\mathit{jump}_A(e)$.

The hybrid automaton A is a *phase-portrait approximation* of the hybrid automaton B if there exists a flow split $\mathcal{P}$ for B such that A is a basic phase-portrait approximation of $\mathcal{P}(B)$.

Proposition 4.1 *Let A and B be hybrid automata. If A is a phase-portrait approximation of B, then A time simulates B.*

Proof. Suppose that A is a basic phase-portrait approximation of $\mathcal{P}(B)$, for some flow split $\mathcal{P}$ of B. Then the identity relation on the admissible states of $\mathcal{P}(B)$ is a time simulation of $\mathcal{P}(B)$ by A. The proposition follows by Theorem 2.2 and the transitivity of time simulation. ■

If A is a phase-portrait approximation of B, by Proposition 2.1 it follows that, if A is empty, then B is empty. Hence phase-portrait approximations provide necessary criteria for nonemptiness. The tool HyTech can be applied only to phase-portrait approximations that are rationally linear. The hybrid automaton A is a *(rationally) linear phase-portrait approximation* of the hybrid automaton B if A is both (rationally) linear and a phase-portrait approximation of B. Rationally linear phase-portrait approximations are typically obtained by first splitting the control modes using a flow split, and then overapproximating, for each control mode v, the flow condition $\mathit{flow}(v)$ by a convex rationally linear predicate φ so that $[\![\varphi]\!]$ contains the convex hull of $[\![\mathit{flow}(v)]\!]$.

Example 4.1 Suppose that we want to prove that within first 60 time units of operation of the thermostat automaton from Figure 1, the heater is active less than 50% of the time. For this purpose, we replace the constraint $t_x = \ln 2$ in the clock linearization of the thermostat automaton (Figure 2) by the rationally linear predicate $69/100 \leq t_x \leq 70/100$, because $\ln 2$ is approximately equal to 0.693. Similarly, we overapproximate $t_x = \ln 3$ by $109/100 \leq t_x \leq 110/100$. Then HyTech automatically verifies the safety property. Indeed, HyTech determines that after 60 time units, the thermostat has been in control mode *on* between $(2317/60)\% \approx 38.6\%$ and $(2351/60)\% = 39.2\%$ of the time. These bounds are tight for the approximate automaton, but they can be tightened further for the original automaton by refining the approximation. ■

Example 4.2 Suppose that we directly approximate the thermostat automaton of Figure 1 without first performing a clock translation. As before, the goal is to prove that the heater is acive for less than 50% of the first 60 time units. We use the flow split $\mathcal{P}_1$ with the predicates $1 \leq x \leq 2$ and $2 \leq x \leq 3$ for the control mode *on*, and the predicates $1 \leq x \leq 2$ and $2 \leq x \leq 3$ for the control mode *off*. Figure 3 depicts the resulting rationally linear phase-portrait approximation of

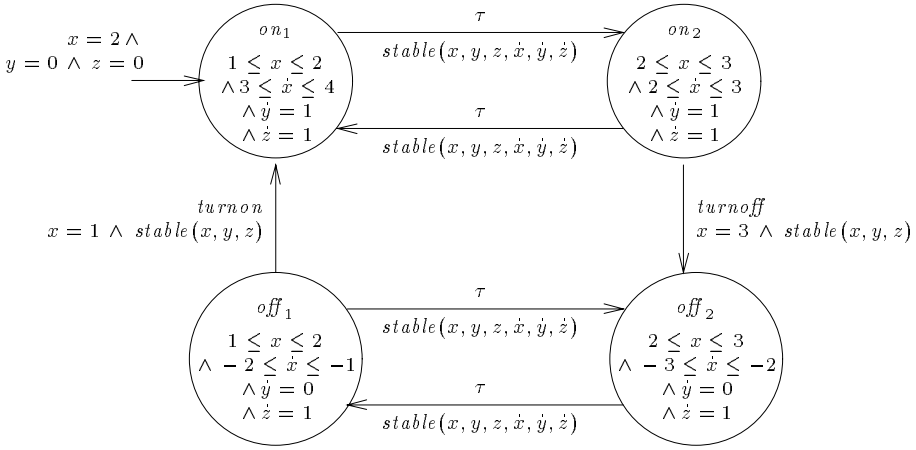


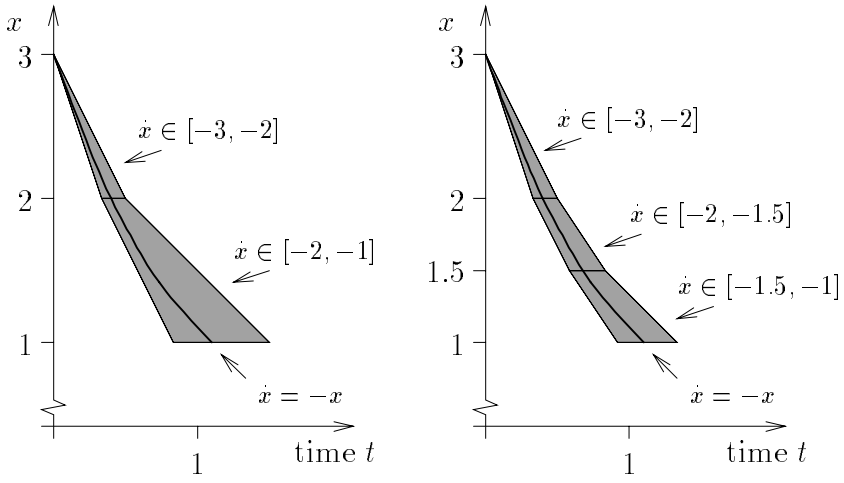
Figure 3: Linear phase-portrait approximation of the thermostat automaton

the thermostat automaton. While the proof of Theorem 2.2 requires that all flow splits are derived from open covers, it is easy to see that overlapping closed covers suffice for this example. All flow transitions that pass through the dividing point $x = 2$ in the original automaton are mimicked in the approximate automaton by a flow transition up to the point $x = 2$, followed by a silent transition between control modes, followed by a flow transition originating at $x = 2$. This approximation is too coarse: HyTech reveals that for this approximation, the active time of the heater ranges from $\approx 27.8\%$ to $\approx 50.0\%$.

The approximation can be tightened by using a finer flow split. For instance, consider the flow split $\mathcal{P}_2$ that splits the control mode off_1 of Figure 3, and is derived from the predicates $1 \le x \le 2$ and $2 \le x \le 3$ for the control mode *on*, and the predicates $1 \le x \le 1.5$, $1.5 \le x \le 2$, and $2 \le x \le 3$ for the control mode *off*. Figure 4 depicts the increased accuracy of the resulting approximation for computing time successors of the state $(off, x = 3)$. Automatic analysis with HyTech now shows that the heater is active between $\approx 30.7\%$ and $\approx 48.1\%$ of the time, which implies the safety property of interest. The finer the flow split, the tighter the approximation, but the greater the computational cost. Using flow split $\mathcal{P}_2$, the computation time of HyTech is longer than for $\mathcal{P}_1$ (6.4 seconds versus 5.3 seconds of CPU time on a Sun Sparcstation 5). By contrast, HyTech requires only 2.3 seconds to generate the much better bounds for the approximated clock linearization of Example 4.1. This demonstrates the benefits of using the clock-translation algorithm where possible. ■

4.2 Example: predator-prey systems

We illustrate the use of linear phase-portrait approximations on nonlinear systems modeling the population growth of two interacting species. We show that several interesting properties of the system can be discovered automatically through a combination of deductive reasoning and algorithmic analysis.



(a) using flow split $\mathcal{P}_1$ (b) using flow split $\mathcal{P}_2$

Figure 4: Tighter approximation via finer flow splits

A predator-prey ecology with limited growth

Much of our exposition defining predator-prey systems is derived from Chapter 12 of [HS74]. One species is the *predator*, whose population is modeled by the variable y, and the other its *prey*, modeled using the variable x. The prey forms the entire food supply for the predator, and we assume that the per-capita food supply for the predator at any instant of time is proportional to the number of prey. The growth of the predator population is proportional to the difference between its actual per-capita food supply and a basic per-capita food supply required to maintain the predator population. The population of the prey is subject to two competing forces. First, the prey population may grow because there is a constant food supply available, and may increase without bound in the absence of predators. Furthermore, we assume the rate of increase is proportional to the number of prey. Second, the predators consume the prey at a rate that is proportional to the number of predators and to the number of prey. This gives us the following equations:

$$\begin{aligned} \dot{x} &= (A - By)x \\ \dot{y} &= (Cx - D)y \end{aligned}$$

for positive real-valued constants A, B, C, and D. No population really has the potential to increase without bound. There are social phenomena, such as overcrowding, spread of disease, and pollution, that imply that most populations will experience negative growth once they exceed a threshold limiting population. We assume that these negative growth factors are proportional to the species population and its difference from the threshold population. This leads to the Volterra-Lotka predator-prey equations [Lot20]:

$$\begin{aligned} \dot{x} &= (A - By - \lambda x)x \\ \dot{y} &= (Cx - D - \mu y)y \end{aligned}$$

where A, B, C, D, λ, and μ are all positive real-valued constants. Assuming that the initial prey

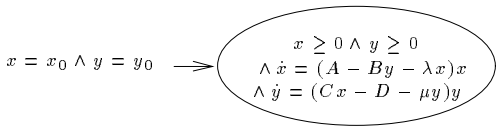


Figure 5: Predator-prey hybrid automaton

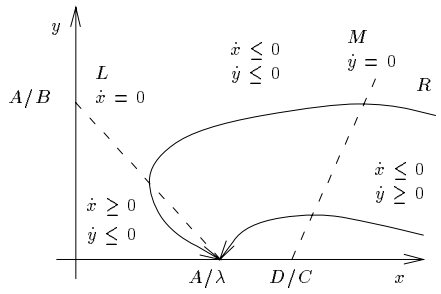


Figure 6: Phase portrait for predator-prey populations (L, M nonintersecting)

population is x_0 and the initial predator population is y_0, the resulting hybrid automaton is shown in Figure 5. Both x and y are unsolvable nonlinear variables.

Linear phase-portrait approximation

We consider the case that the two lines $L = (A - By - \lambda x = 0)$ and $M = (Cx - D - \mu y = 0)$ do not intersect in the quadrant $\mathbb{R}^2_{>0}$. In this case, the phase portrait of the predator-prey system looks as shown in Figure 6. Using the coordinate axes and the two lines L and M, we split the state space into linear regions. Within each region, we can infer the signs of $\dot{x}$ and $\dot{y}$ as shown in Figure 6. For the region R, to the right of the line M, we can infer also a linear constraint that relates the derivatives of x and y. Since $\dot{x}$ is nonpositive and $\dot{y}$ nonnegative in R, the directions of the flow vectors in R are determined by the function $\xi(x, y) = \dot{y}/\dot{x}$. The absolute value of $\xi(x, y)$ is bounded above by any ratio max/min, where max is an upper bound on the value of $\dot{y}$ in R and min is a lower bound on the absolute value of $\dot{x}$. We can take Cxy for max, because $D + \mu y$ is always positive. Since the lines L and M do not intersect in $\mathbb{R}^2_{>0}$, we know that $A/\lambda < D/C$, and hence $A - \lambda D/C < 0$. Since x is no less than D/C in R, we infer that $A - \lambda x < 0$, and hence $A - \lambda x - By < -By$. We may therefore take Byx for min. We conclude that $\xi(x, y)$ is bounded below by $-Cxy/(Bxy) = -C/B$. It follows that all flow vectors in R have a direction between $(-B, C)$ and $(-1, 0)$, *i.e.* they satisfy the flow condition $\dot{y} \geq 0 \wedge \dot{y} \leq -C\dot{x}/B$.

The hybrid automaton that represents the resulting linear phase-portrait approximation is shown in Figure 7. The layout of the control modes matches the partitioning of the state space as shown in Figure 6. The predicate *stable* is shorthand for $stable(x, y, \dot{x}, \dot{y})$. The implicit invariant constraint $x \geq 0 \wedge y \geq 0$ has been omitted from all flow conditions. The constraint M refers to all valuations on the line M, *i.e.* all valuations with $Cx - D - \mu y = 0$. The constraint $M^{\geq}$ refers to all valuations at, or to the right of, the line M, *i.e.* $M^{\geq}$ stands for $Cx \geq D + \mu y$. Similarly, let $M^{\leq}$

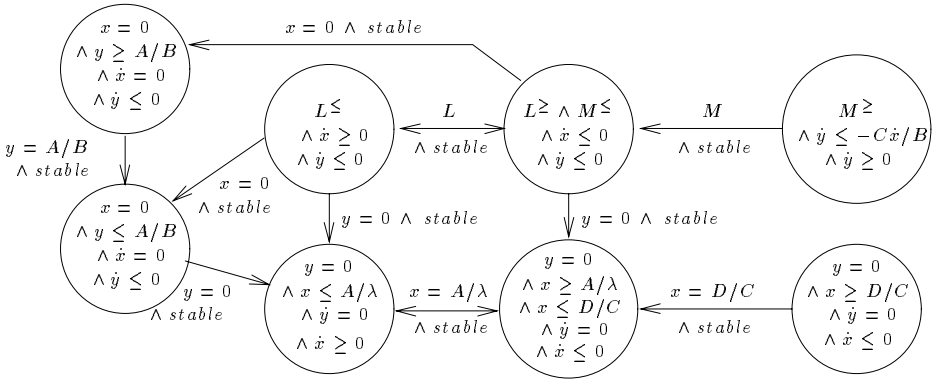


Figure 7: Linear phase-portrait approximation of the predator-prey automaton

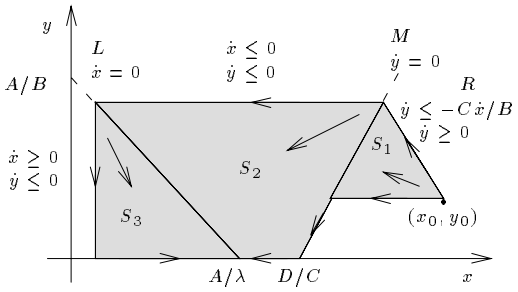


Figure 8: Reachability computation for the linear phase-portrait approximation

stand for $Cx \leq D + \mu y$, let L stand for $A - By - \lambda x = 0$, let $L^{\leq}$ stand for $\lambda x \leq A - By$, and let $L^{\geq}$ stand for $\lambda x \geq A - By$. If A, B, C, D, λ, and μ are all rational constants, then the phase-portrait approximation is rationally linear.

Computing bounds on the population growth

The linear phase-portrait approximation can be used to compute, for given starting populations, bounds on the populations of both species. In particular, this shows that the populations are indeed bounded. For example, suppose that the initial populations x_0 and y_0 lie in the rightmost region R of the state space. The time successors of the state (x_0, y_0) are obtained by following all flow vectors in the cone indicated in Figure 8. First, the states in region S_1 are reached. Control may then pass to the control mode corresponding to the central region in the partition, where both $\dot{x}$ and $\dot{y}$ are nonpositive. After adding the states in region S_2, and then the states in S_3, the computation of reachable states terminates. The maximum value of y among the reachable states is $(By_0 + Cx_0 - D)/(B + \mu)$. For example, given the equations $\dot{x} = (2000 - y - 5x)x$ and $\dot{y} = (4x - 2600 - 4y)y$, and the initial population vector $(900, 150)$, HyTech computes a bound of 230 on the predator population y.

Bounds on the region of reachable states can often be used to construct better phase-portrait

approximations. Let φ be a predicate such that $[\![\varphi]\!]$ contains all reachable states of the hybrid automaton A. The *restriction* of A to φ is the hybrid automaton $A|_\varphi$ that differs from A only in its flow conditions: for all control modes v, let $\mathit{flow}_{A|_\varphi}(v) = (\mathit{flow}_A(v) \wedge \varphi)$. The two automata $A|_\varphi$ and A are timed bisimilar (via the identity relation). Restriction is useful, because it may be possible to find tighter linear phase-portrait approximations for $A|_\varphi$ than for A, because the phase portrait of $A|_\varphi$ may contain fewer flow vectors than the phase portrait of A.

In the predator-prey example, it can be shown that in the rightmost region R, the absolute value of $\xi(x, y)$ is bounded above by $Cy/(\lambda x + By - A)$. Let R' be a bounded subset of R. Let y_{max} be an upper bound for y over all valuations in R', and let x_{min} (resp. y_{min}) be a lower bound for x (resp. y) over R'. It follows that $|\xi(x, y)| \leq Cy_{max}/(\lambda x_{min} + By_{min} - A)$, provided that $(\lambda x_{min} + By_{min} - A) \geq 0$.

Previously, we showed how reachability computation for the automaton of Figure 7 leads to the region S_1 in R, from which we can infer the bounds $y_{max} = 230$, $y_{min} = 150$, and $x_{min} = 800$. We therefore replace the flow condition $\dot{y} \leq -C\dot{x}/B$ of the region S_1 by $\dot{y} \leq -92\dot{x}/215$. Recomputation now shows that only a proper subset of the region S_1 is reachable. In particular, we obtain the tighter bound of $y_{max} = 55250/307 \approx 180$. If we iterate this procedure, we gain successively lower values of y_{max}, more restrictive flow conditions, and more accurate approximations of the set of reachable states.

Controlling the ecology

Standard analysis techniques can be used to show that the predator population always tends toward 0, while the prey population tends to A/λ. Suppose, however, that we wish to keep the predator population above a nontrivial minimal value, or more generally, that the populations need to be controlled so that they remain within given lower and upper bounds. Assume that the prey population can be accurately measured, but that the predator population is unobservable. Our control strategy consists of monitoring the prey population, and releasing a fixed number k of additional prey into the system whenever it reaches its minimal allowable value. In general, it may be unwise to increase the prey population to its maximal allowable value, because the abundance of prey may cause the predator population to grow too large. For the ecology above, we require the predator population to lie within the range $[100, 350]$, and the prey population within $[800, 1100]$. Using HyTech, we can verify that the bounds are successfully maintained whenever $k \leq 200$. For larger values of k, the phase-portrait approximation admits trajectories where the predator population exceeds the upper bound of 350. Note, however, that this does not imply that all values of k greater than 200 may lead to excessively large predator populations, because the approximation has more reachable states than the true system.

4.3 Error analysis

Given a hybrid automaton A, some linear phase-portrait approximations of A are closer to A than others. The closer the approximation of A, the more safety properties of A can be verified by analyzing the approximation. We show that using linear phase-portrait approximation, A can be approximated arbitrarily closely by choosing a sufficiently fine flow split.

Proximity can be defined via the infinity metric $\mathit{dist} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, where $\mathit{dist}(\mathbf{a}, \mathbf{b}) = \mathit{max}_{1 \leq i \leq n} |a_i - b_i|$, *i.e.* the distance $\mathit{dist}(\mathbf{a}, \mathbf{b})$ between two points $\mathbf{a}$ and $\mathbf{b}$ is the maximal componentwise separation. The predicate ψ is an *ε-relaxation* of the predicate φ, for a nonnegative real $\varepsilon \in \mathbb{R}_{\geq 0}$, if $[\![\varphi]\!] \subseteq [\![\psi]\!]$ and for all valuations $\mathbf{a} \in [\![\psi]\!]$, there exists a valuation $\mathbf{b} \in [\![\varphi]\!]$ such that $\mathit{dist}(\mathbf{a}, \mathbf{b}) \leq \varepsilon$. The hybrid automaton B is an *ε-relaxation* of the hybrid automaton A, for $\varepsilon \in \mathbb{R}_{\geq 0}$,

if B results from A by replacing all flow, initial, final, and jump conditions with ε-relaxations. If A models a system with sensors and actuators, then the ε-relaxations of A model the same system under the assumption that the sensors and actuators may have measurement errors, and the flow transitions are subject to modeling errors, with all errors being bounded by ε. Clearly, every ε-relaxation of A, for $\varepsilon \geq 0$, is a phase-portrait approximation of A.

An *approximation operator* γ for hybrid automata is a function that maps each hybrid automaton A to a set $\gamma(A)$ of hybrid automata—the *γ-approximations* of A—such that for all $B \in \gamma(A)$, the automaton B time simulates A. For example, the *(rationally linear) phase-portrait approximation operator* maps every hybrid automaton A to the set of (rationally linear) phase-portrait approximations of A. For an approximation operator γ, if $B \in \gamma(A)$ is empty, then A is also empty, *i.e.* every safety property of B is also satisfied by A. The converse, however, is not necessarily true; hence approximation is a sound but not complete proof technique for verifying safety properties.

Given an approximation operator γ for hybrid automata, and a hybrid automaton A, the γ-approximation $B \in \gamma(A)$ is *ε-close* to A, for a nonnegative real $\varepsilon \in \mathbb{R}_{\geq 0}$, if some ε-relaxation of A time simulates B. We write $\gamma_\varepsilon(A)$ for the set of ε-close γ-approximations of A. Then, if $B \in \gamma_\varepsilon(A)$ is nonempty, some ε-relaxation of A is also nonempty, *i.e.* every safety violation of B corresponds to a safety violation of an automaton that lies within distance ε from A. The approximation operator γ is *asymptotically complete* if for all hybrid automata A and all positive reals $\varepsilon > 0$, the set $\gamma_\varepsilon(A)$ of ε-close γ-approximations is nonempty. Asymptotic completeness ensures that for every hybrid automaton A, if some automaton arbitrarily close to, but different from, A satisfies a safety property, then there is an approximation of A that also satisfies the property.

We show that already a restricted form of linear phase-portrait approximations are asymptotically complete, namely, when all automaton constraints are overapproximated using independent, rational lower and upper bounds on the values and derivatives of each variable [HH95a]. The predicate φ is *rectangular* if $[\![\varphi]\!]$ has the form $\prod_{i=1}^{n} I_i$, where each I_i is a (possibly unbounded) interval over $\mathbb{R}$. The predicate φ is *rationally rectangular* if the endpoints of the interval I_i are rational, for each $i \in \{1, \ldots, n\}$. The hybrid automaton A is *(rationally) rectangular* if all flow, jump, initial, and final conditions of A are (rationally) rectangular. Clearly, every (rationally) rectangular hybrid automaton is (rationally) linear. The *(rationally) rectangular phase-portrait approximation operator* maps every hybrid automaton A to the set of phase-portrait approximations of A that are (rationally) rectangular. For example, the automaton of Figure 3 is a rationally rectangular phase-portrait approximation of the thermostat automaton.

Theorem 4.2 *The rationally rectangular phase-portrait approximation operator for hybrid automata is asymptotically complete.*

Proof. Let A be a hybrid automaton, and let $\varepsilon > 0$ be a positive real. We construct a rationally rectangular phase-portrait approximation that is ε-close to A in two steps: first we construct a (possibly irrational) rectangular phase-portrait approximation B that is $\varepsilon/2$-close to A, and then we approximate all predicates of B by rationally rectangular predicates.

Given a predicate φ over the set $X = \{x_1, \ldots, x_n\}$ of variables, let I_{φ,x_i}, for $i \in \{1, \ldots, n\}$, be the infimal (possibly unbounded) interval of the reals that contains the projection of $[\![\varphi]\!]$ onto the x_i axis. The *rectangularization* of φ is the rectangular predicate ψ over X such that $[\![\psi]\!] = \prod_{i=1}^{n} I_{\varphi,x_i}$. The predicate φ has *diameter* $\delta \in \mathbb{R}_{\geq 0}$ if $\mathit{dist}(\mathbf{a}, \mathbf{b}) \leq \delta$ for all valuations $\mathbf{a}, \mathbf{b} \in [\![\varphi]\!]$. Let $\mathcal{P}$ be a flow split for A that maps every control mode to a set of predicates, each with diameter $\varepsilon/2$. Define B to be the hybrid automaton that results from replacing all flow, initial, final, and jump condition of $\mathcal{P}(A)$ by their rectangularizations. Then the flow, jump, initial, and final conditions of B all have diameter $\varepsilon/2$. Hence, for all control modes v of B, for every valuation $(\mathbf{a}, \dot{\mathbf{a}}) \in \mathit{flow}_B(v)$, there

exists a valuation $(\mathbf{b}, \dot{\mathbf{b}}) \in [\![\mathit{flow}_{\mathcal{P}(A)}(v)]\!]$ with $\mathit{dist}((\mathbf{a}, \dot{\mathbf{a}}), (\mathbf{b}, \dot{\mathbf{b}})) \leq \varepsilon/2$. Similarly, every valuation satisfying the jump, initial, and final conditions of B is within $\varepsilon/2$ of some valuation satisfying the corresponding condition of $\mathcal{P}(A)$, (and therefore of A). It follows that the rectangular hybrid automaton B is time simulated by an $\epsilon/2$-relaxation of A.

We derive the rationally rectangular hybrid automaton C from B by replacing every rectangular predicate φ in B with a rationally rectangular predicate that is an $\epsilon/2$-relaxation of φ. In particular, if $[\![\varphi]\!] = \prod_{i=1}^{n} I_i$, then φ is replaced by the rationally rectangular predicate ψ with $[\![\psi]\!] = \prod_{i=1}^{n} I'_i$, where for each $i \in \{1, \ldots, n\}$, we have $I_i \subseteq I'_i$ and the endpoints of the interval I'_i result from shifting up, or down, by at most $\varepsilon/2$ the endpoints of the interval I_i to some rationals. Then, every valuation satisfying the flow, jump, initial, and final conditions of C is within $\varepsilon/2$ of some valuation satisfying the corresponding condition of B, and therefore within ε of some valuation satisfying the corresponding condition of A. It follows that some ε-relaxation of A time simulates C. ■

In practice, rectangular phase-portrait approximations are often easier to compute than nonrectangular phase-portrait approximations (because we need only compute projections for all variables). Nonrectangular linear phase-portrait approximations, however, are sometimes more accurate, as seen in the predator-prey example.

5 Conclusion

We presented a methodology that enables the algorithmic analysis of nonlinear hybrid systems via translation to linear hybrid automata. Two transformation steps may be utilized. The first step, the clock translation, should be applied whenever possible. It is efficient, both sound and complete for proving safety properties, but applies only to a restricted class of variables. Linear phase-portrait approximation can be used to remove any remaining nonlinearities. It is sound, but not complete, for proving safety properties, and it may cause a substantial blow-up of the state space. Linear phase-portrait approximation, however, is applicable to all hybrid systems, supports the successive refinement of approximations, and in many cases can be automated. The combined methodology has been successfully applied to the benchmark industrial steam-boiler specification [HW96b], and to an electronic automotive suspension-control system developed by BMW [SMF97].

Acknowledgement. We thank Peter Kopke for numerous helpful 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.L. 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.

[Hal93] N. Halbwachs. Delay analysis in synchronous programs. In C. Courcoubetis, editor, *CAV 93: Computer-aided Verification*, Lecture Notes in Computer Science 697, pages 333–346. Springer-Verlag, 1993.

[Hen96] T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[HH95a] T.A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 225–238. Springer-Verlag, 1995.

[HH95b] T.A. Henzinger and P.-H. Ho. HyTech: The Cornell Hybrid Technology Tool. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 265–293. Springer-Verlag, 1995.

[HH95c] T.A. Henzinger and P.-H. Ho. A note on abstract-interpretation strategies for hybrid automata. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 252–264. Springer-Verlag, 1995.

[HHW95a] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: the next generation. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 56–65. IEEE Computer Society Press, 1995.

[HHW95b] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1019, pages 41–71. Springer-Verlag, 1995.

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

[HRP94] N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of linear hybrid systems by means of convex approximation. In B. LeCharlier, editor, *SAS 94: Static Analysis Symposium*, Lecture Notes in Computer Science 864, pages 223–237. Springer-Verlag, 1994.

[HS74] M.W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, 1974.

[HW95] P.-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, Lecture Notes in Computer Science 939, pages 381–394. Springer-Verlag, 1995.

[HW96a] T.A. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 377–388. Springer-Verlag, 1996.

[HW96b] T.A. Henzinger and H. Wong-Toi. Using HyTech to synthesize control parameters for a steam boiler. In J.-R. Abrial, E. Börger, and H. Langmaack, editors, *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, Lecture Notes in Computer Science 1165, pages 265–282. Springer-Verlag, 1996.

[Lot20] A.J. Lotka. Analytical note on certain rhythmic relations in organic systems. *Proceedings of the National Academy of Sciences of the United States of America*, 6:410–415, 1920.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[OSY94] A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In D.L. Dill, editor, *CAV 94: Computer-aided Verification*, Lecture Notes in Computer Science 818, pages 81–94. Springer-Verlag, 1994.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings of 5th GI conference*, pages 167–183. Springer-Verlag, 1981. LNCS 104.

[PBV96] A. Puri, V. Borkar, and P. Varaiya. ε-approximation of differential inclusions. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 362–376. Springer-Verlag, 1996.

[PV95] A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 259–369. Springer-Verlag, 1995.

[SMF97] T. Stauner, O. Müller, and M. Fuchs. Using HyTech to verify an automotive control system. In O. Maler, editor, *HART 97: Hybrid and Real-Time Systems*, Lecture Notes in Computer Science 1201, pages 139–153. Springer-Verlag, 1997.