ALGORITHMIC ANALYSIS OF NONLINEAR HYBRID SYSTEMS

by

Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi

Memorandum No. UCB/ERL M96/60

15 October 1996

ALGORITHMIC ANALYSIS OF NONLINEAR HYBRID SYSTEMS

by

Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi

Memorandum No. UCB/ERL M96/60

15 October 1996

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

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
Intel Development 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 systems, whose environment variables are subject to polyhedral differential inclusions. Most embedded systems, however, operate in *nonlinear* environments. We present two methods for translating nonlinear hybrid systems into linear hybrid systems. Properties of the nonlinear systems can be inferred from the automatic analysis of the translated linear systems using existing model-checking tools. The first method, the *clock translation*, replaces nonlinear variables by clock variables. It is only applicable when the nonlinear variables are solvable. The second method, *linear phase-portrait approximation*, conservatively overapproximates the automaton's phase-portrait using envelopes that are defined by 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. 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 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 use HyTech — a symbolic model checker for linear hybrid systems — to automatically verify a nonlinear temperature controller using the clock translation and linear phase-portrait approximations, and we automatically compute population bounds for a predator-prey ecology using linear phase-portrait approximations. We also identify the class of *pseudo-linear* hybrid automata, for which linear phase-portrait approximations can be generated automatically.

1 Introduction

Hybrid systems combine discrete and continuous dynamics. Their analysis requires techniques from both computer science and control theory. Computer scientists typically model hybrid systems as

*This research was supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grants CCR-9200794 and CCR-9504469, by the AFOSR contract F49620-93-1-0056, and by the ARPA grant NAG2-892.

†Preliminary reports of this work appeared in [HH95a] and [HWT96].

discrete programs that react to continuous variables undergoing simple dynamics, whereas control theorists typically study complex behaviors of continuous variables within simple control loops. From the field of computer science, there are algorithmic techniques for checking certain properties, such as emptiness, of *linear hybrid automata* [ACHH93, ACH+95]. These automata have linearity restrictions on discrete jumps (linear inequalities between sources and targets of jumps) and continuous flows (differential inequalities of the form $\mathbf{A}\dot{\mathbf{x}} \geq \mathbf{b}$) of variables. The model-checking algorithms [HNSY94, AHH96] have been implemented in HyTech [HHWT95a, HHWT95b], and used to verify distributed real-time protocols [HH95b, HW95]. 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 directly model continuous flows of the form $\dot{x} = x$. This paper extends the model-checking approach to the analysis of certain nonlinear hybrid systems, by reduction to model-checking of linear hybrid approximations. The automaton B is an *approximation* of the automaton A if B is empty implies A is empty.

A hybrid automaton defines an infinite-state transition system. Since the verification of safety properties for hybrid automata can be reduced to emptiness checking, we concentrate on checking the emptiness of automata, *i.e.* whether there is a path from an initial state to a final state. Checking emptiness of a hybrid automaton involves computing weakest preconditions (or strongest postconditions) in the underlying transition system. The widest class of systems for which we know how to compute weakest preconditions reasonably efficiently is that of rational linear hybrid automata. We therefore propose the following methodology for analyzing a nonlinear hybrid automaton A. First, we attempt to translate each nonlinear variable x into a (linear) clock variable. The clock translation of x is only possible when the behavior of x can be solved. Second, we obtain a rational linear phase-portrait approximation B for the automaton resulting from the first step. Third, we apply the symbolic model-checking tool HyTech to B. The automaton B is always an approximation of A. However, it is possible that the approximations are not sufficiently accurate, and B is nonempty even when A is empty. In this case, we need to refine our approximation.

Step 1. Clock translation

The *clock translation* [HH95a] replaces nonlinear variables with a *clock*, a special kind of linear variable whose slope is always 1. The translation can only be applied to the nonlinear variable x when all constraints and assignments involving x can be translated into constraints and assignments of a clock that measures the time between significant events. More specifically, we require the value of x to be solvable in closed form as a function of the latest assignment to x and the time that has elapsed since that assignment. If the clock translation of the automaton A yields the automaton B, then the underlying transition systems are timed bisimilar. The clock translation B is therefore an approximation of A. Moreover, the clock translation is *complete*, in that whenever A is empty, so is its clock translation B. If it is possible to apply the clock translation to all nonlinear variables, and all linear variables of the original automaton are rectangular, then the resulting automaton is a rectangular automaton. Because the emptiness of rectangular automata is decidable, it follows that we obtain a new class of decidable hybrid automata.

Step 2. Linear phase-portrait approximation

Because the clock translation is complete, and involves only a constant blow-up in the number of control modes, we prefer it where possible. However, many typical control environments follow unsolvable differential equations, and often the clock translation cannot be applied. The *phase-portrait approximation method* [HWT96] approximates an automaton by relaxing all constraints

defining its transition system. In this way, it enlarges the set of trajectories. In particular, linear phase-portrait approximations use piecewise-linear envelopes to approximate the variables' flow conditions. The approximation method may be viewed as involving two steps. First, each control mode of a nonlinear hybrid automaton is split into several copies, each of which corresponds to a region of the partitioned state space. Second, within each new control mode, the nonlinear dynamics is replaced with a linear dynamics, which contains all flow tangents of the nonlinear system occurring at that control mode. The phase-portrait approximation method is not complete. We provide an error analysis that shows that linear phase-portrait approximations may be made arbitrarily close to the original automaton, at a cost of increasing the number of control modes. The approximation method is demonstrated on a simple predator-prey ecology, for which a control strategy for altering the ecology to maintain population bounds is verified. We also introduce the class of *pseudo-linear hybrid automata*. For a nonlinear system, it is generally nontrivial to obtain a linear phase-portrait approximation, because some understanding of the solution curves of nonlinear differential equations is required. However, linear phase-portrait approximations can be automatically generated for pseudo-linear hybrid automata.

Related work

Phase portraits have been studied extensively in the literature on dynamical systems [HS74, Arn83]. Typically, researchers concentrate on the complex dynamics of a system, and are able to prove complex properties, such as stability and convergence. Our work differs in two respects. First, we consider products of nondeterministic dynamical systems with discrete transition structures [Hen95]. Second, our goal is to analyze and derive simple properties of such systems *automatically*. In computer science, the technique of deriving system properties using approximations is called *abstract interpretation*. In [Hal93, HRP94, HH95c], abstract interpretation techniques are used to provide linear approximations of linear hybrid systems, whereas here we approximate nonlinear hybrid systems. The work of Puri et al. [PBV96] is closely related to our phase-portrait approximations. They consider approximations of a single differential inclusion, providing error bounds and invariant sets, but do not consider a discrete transition structure. They also consider incomplete approximations using timed automata [PV95]. Olivero et al. [OSY94] study incomplete approximations of linear hybrid automata with timed automata. In [HH95a], we presented an approximation method called the *rate translation*, which is but a special case of the phase-portrait approximation method presented here.

Outline

In the next section, we define nonlinear hybrid automata, and linear hybrid automata, a subclass that can be algorithmically analyzed. The clock translation method appears in Section 3. Section 4 describes linear phase-portrait approximations.

2 Hybrid Automata

We define *hybrid automata*, used to model systems consisting of mixed 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 represent discrete jumps and are labeled with guarded assignments to X. The vertices V represent continuous flows and are labeled with differential inequalities on the derivatives of X. The state of the automaton changes either instantaneously when a discrete jump occurs or, while time elapses, through continuous flows.

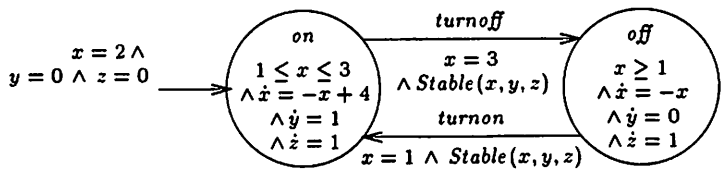

Figure 1: A thermostat

2.1 Syntax

A *hybrid automaton* $A = (X, V, \mathit{flow}, E, \mathit{jump}, \Sigma, \mathit{event}, \mathit{init}, \mathit{final})$ consists 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 uses the 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. A *valuation* over X is a point $(a_1, a_2, \ldots, a_n)$ in the n-dimensional real space $\mathbb{R}^n$, or equivalently, a function that maps each variable x_i to its value a_i. For a valuation $\mathbf{x}$, we sometimes write $\mathbf{x}(x_i)$ or $\mathbf{x}_i$ to refer to the value of x_i in $\mathbf{x}$. An atomic predicate over X is a predicate of the form $f(x_1, x_2, \ldots, x_n) \sim c$, for a real-valued algebraic function $f : \mathbb{R}^n \to \mathbb{R}$, a relation symbol $\sim \in \mathit{RelOps} = \{<, \le, =, \ge, >\}$, and a real constant $c \in \mathbb{R}$. A *predicate* is an arbitrary combination of disjunctions and conjunctions of atomic predicates. Each predicate ϕ over X defines a set $[\![\phi]\!] \subseteq \mathbb{R}^n$ of valuations such that $\mathbf{x} \in [\![\phi]\!]$ iff $\phi[X := \mathbf{x}]$ is true.[1]

Control modes A finite set V of vertices called *control modes*. For example, the thermostat automaton has two control modes, *on* and *off*. A *state* $(v, \mathbf{x}, \dot{\mathbf{x}})$ consists of a control mode $v \in V$, a valuation $\mathbf{x} \in \mathbb{R}^n$ over the set X of variables, and a valuation $\dot{\mathbf{x}} \in \mathbb{R}^n$ over the set $\dot{X}$, 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 tangent for continuous trajectories passing through the point. A *region* is a set of states.

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 control of A is in control mode v, the variables change along differentiable trajectories whose first derivatives satisfy the flow condition. For example, the control mode *on* in the thermostat automaton has flow condition $x \le 3 \wedge \dot{x} = -x + 4 \wedge \dot{y} = 1 \wedge \dot{z} = 1$. For each control mode v, let the *differential inclusion for* v be the function $f_v : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ defined by $f_v(\mathbf{x}) = \{\dot{\mathbf{x}} \mid (\mathbf{x}, \dot{\mathbf{x}}) \in [\![\mathit{flow}(v)]\!]\}$. The state $(v, \mathbf{x}, \dot{\mathbf{x}})$ is *admissible* if $(\mathbf{x}, \dot{\mathbf{x}}) \in [\![\mathit{flow}(v)]\!]$. We infer from the flow conditions a function *inv* that assigns to each location a predicate over X. We define $\mathit{inv}(v) = \exists \dot{\mathbf{x}}.\mathit{flow}(v)[X, \dot{X} := \mathbf{x}, \dot{\mathbf{x}}]$, the *invariant* on X while the automaton control resides in control mode v.

Control switches A finite multiset E of edges called *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, $(\mathit{on}, \mathit{off})$ and $(\mathit{off}, \mathit{on})$.

[1] The expression $\phi[X := \mathbf{x}]$ denotes the predicate ϕ evaluated with each variable in X substituted with its value in $\mathbf{x}$.

Jump conditions A labeling function *jump* that assigns a *jump condition* to each control switch $e \in E$. The jump condition $jump(e)$ 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 the modeling of guard conditions and assignments), and also relate the tangents before the control switch with those after (allowing, for example, the assertion that trajectories are differentiable across control switches). When writing jump conditions in the figures of this paper, we use the predicate *Stable* to indicate a set of primed variables which must have the same values as their unprimed counterparts, *i.e.* $Stable(x, \dot{y})$ denotes $x = x' \wedge \dot{y} = \dot{y}'$. For example, in the thermostat automaton, $jump(on, off)$ is $x = 3 \wedge Stable(x, y, z, \dot{z})$.

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 $e \in E$. 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 $init(v)$ is a predicate over $X \cup \dot{X}$. Control of an automaton may start in control mode v when $init(v)$ is true. A state $(v, \mathbf{x}, \dot{\mathbf{x}})$ is *initial* if it is admissible and $(\mathbf{x}, \dot{\mathbf{x}}) \in [\![init(v)]\!]$. 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 control mode *on* with $x = 2 \wedge y = 0 \wedge z = 0$.

Final conditions A labeling function *final* that assigns a *final condition* to each control mode $v \in V$. The final condition $final(v)$ is a predicate over $X \cup \dot{X}$. A state $(v, \mathbf{x}, \dot{\mathbf{x}})$ is *final* if it is admissible and $(\mathbf{x}, \dot{\mathbf{x}}) \in [\![final(v)]\!]$. Safety properties can be expressed as invariants on the set of reachable states. The complement of the invariant represents the set of violation, or error, states for a system. The system is correct with respect to a safety property if no violation states are reachable. We use final conditions to define the set of the violation states for a safety property. For example, for the thermostat automaton, consider the property that the heater is activated (*i.e.* control of the automaton resides in control mode *on*) no more than 60% of the first 60 time units. Correctness with respect to this property is expressed through the final conditions $final(on) = final(off) = y \geq 0.6z \wedge z = 60$.

Remark. The definition used here differs from previous definitions in the literature. First, we augment the notion of a state with a vector over $\dot{X}$, providing the flow tangent at a given point. Jump conditions also express constraints over $\dot{X}$. This addition enables us to model changes (or absence of changes) in the first derivatives when making control switches. Information about higher-order derivatives may also be encoded by explicitly introducing additional variables, *e.g.* with the variable u such that the flow conditions imply $\dot{x} = u$, we may use $\dot{u}$ to refer to the second derivative of x. Second, we omit explicit *invariant* conditions over X that specify the allowable values for the variables within a given control mode. Invariant conditions are implicit in the flow conditions. Third, we add final conditions so that correctness with respect to a safety property can be conveniently expressed as automaton emptiness.

2.2 Semantics

We provide semantics for hybrid automata in terms of labeled timed transition systems. A *labeled timed transition system* $T = (S, I, \mathcal{L}, \rightarrow, F)$ consists of a (possibly infinite) set S of states, a subset $I \subseteq S$ of *initial states*, a subset $F \subseteq S$ of *final states*, a set $\mathcal{L}$ of transition labels (including the special *silent-transition label* τ), for each label $\sigma \in \mathcal{L}$ a binary *jump relation* $\xrightarrow{\sigma} \subseteq S^2$ over the state space, and for each real $\delta \in \mathbb{R}_{\geq 0}$, a binary *flow relation* $\xrightarrow{\delta} \subseteq S^2$ over the state space. Each triple $s \xrightarrow{e} s'$ is called a *transition.*

Let A be a hybrid automaton with n variables. The *state space* $S_A \subseteq V \times \mathbb{R}^n \times \mathbb{R}^n$ of A is the set of admissible states. There are two kinds of transitions: jump transitions and flow transitions. In jump transitions, the control mode of the automaton and the continuous variables change instantaneously, in accordance with a control switch $e \in E$ and its jump condition $jump(e)$. Flow transitions model the continuous change of variables over time in accordance to the flow condition associated with the current control mode of the automaton. The automaton's control mode remains fixed.

Formally, for each event $\sigma \in \Sigma$, the binary *jump relation* $\xrightarrow{\sigma}$ on the admissible states is defined by $(v, \mathbf{x}, \dot{\mathbf{x}}) \xrightarrow{\sigma} (v', \mathbf{x}', \dot{\mathbf{x}}')$ iff there exists a control switch $e = (v, v')$ such that $event(e) = \sigma$ and $jump(e)[X, \dot{X}, X', \dot{X}' := \mathbf{x}, \dot{\mathbf{x}}, \mathbf{x}', \dot{\mathbf{x}}']$ is true. For each real $\delta \in \mathbb{R}_{\geq 0}$, we define the binary *flow relation* $\xrightarrow{\delta}$ on the admissible states by $(v, \mathbf{x}, \dot{\mathbf{x}}) \xrightarrow{\delta} (v', \mathbf{x}', \dot{\mathbf{x}}')$ iff $v = v'$, and either (1) $\delta = 0$ and $\mathbf{x} = \mathbf{x}'$ and $\dot{\mathbf{x}} = \dot{\mathbf{x}}'$, or (2) $\delta > 0$ and there exists a differentiable function $\rho : [0, \delta] \rightarrow \mathbb{R}^n$ such that the following conditions hold:

1. the endpoints of the transition match those of ρ, *i.e.* $\rho(0) = \mathbf{x}$, $\dot{\rho}(0) = \dot{\mathbf{x}}$, $\rho(\delta) = \mathbf{x}'$, and $\dot{\rho}(\delta) = \dot{\mathbf{x}}'$, where $\dot{\rho}$ is the first derivative of ρ with respect to time, and

2. the flow condition is satisfied, *i.e.* for all $t \in [0, \delta]$, $(\rho(t), \dot{\rho}(t)) \in [\![flow(v)]\!]$.

The function ρ is referred to as a *witness* trajectory for $(v, \mathbf{x}, \dot{\mathbf{x}}) \xrightarrow{\delta} (v', \mathbf{x}', \dot{\mathbf{x}}')$.

We define the *transition relation* $\rightarrow_A$ on the state space S_A of the hybrid automaton A to be $\bigcup_{\sigma \in \Sigma} \xrightarrow{\sigma} \cup \bigcup_{\delta \in \mathbb{R}_{\geq 0}} \xrightarrow{\delta}$. Given a hybrid automaton A, we define the infinite-state labeled transition system $[\![A]\!] = \langle S_A, I_A, \mathcal{L}_A, \rightarrow_A, F_A \rangle$ by

- the infinite state space S_A of admissible states,
- the set I_A of initial states $\{(v, \mathbf{x}, \dot{\mathbf{x}}) \mid (\mathbf{x}, \dot{\mathbf{x}}) \in [\![init(v)]\!]\}$,
- the set of transition labels $\mathcal{L}_A = \Sigma_A$,
- the labeled transition relation $\rightarrow_A$, and
- the set F_A of final states $\{(v, \mathbf{x}, \dot{\mathbf{x}}) \mid (\mathbf{x}, \dot{\mathbf{x}}) \in [\![final(v)]\!]\}$.

A *trajectory* of A is a finite path $s_0 \xrightarrow{m_0} s_1 \xrightarrow{m_1} \cdots \xrightarrow{m_{k-1}} s_k$ in $[\![A]\!]$ such that $s_0 \in I_A$, and for all $0 \leq i \leq k-1$, $s_i \xrightarrow{m_i} s_{i+1}$. The state s_k is referred to as the end state of the trajectory. A state is *reachable* if there is a trajectory for which it is the end state. The set $reach(A)$ is the set of *reachable states* of A. The hybrid automaton is *empty* if no reachable state is final.

2.3 Simulation and Bisimilarity

We define the concepts of simulation and bisimularity within our framework. They will be used to establish the completeness of the clock translation and the asymptotic completeness of linear

phase-portrait approximations. For each non-silent label $\sigma \in \mathcal{L} \setminus \{\tau\}$, we define the labeled *stutter-closed jump relation* $\stackrel{\sigma}{\rightarrow\!\!\!\!\rightarrow} \subseteq S \times S$ by $s \stackrel{\sigma}{\rightarrow\!\!\!\!\rightarrow} s'$ iff there exists a sequence of states $s_1, \ldots, s_k$ such that $s = s_1$ and $s_1 \stackrel{\tau}{\rightarrow} s_2 \stackrel{\tau}{\rightarrow} \cdots \stackrel{\tau}{\rightarrow} s_k \stackrel{\sigma}{\rightarrow} s'$. For each $\delta \in \mathbb{R}_{\geq 0}$, we define the *stutter-closed flow relation* $\stackrel{\delta}{\rightarrow\!\!\!\!\rightarrow}$ by $s \stackrel{\delta}{\rightarrow\!\!\!\!\rightarrow} s'$ iff there exists a (possibly empty) *finite* sequence of states $s_1, \ldots, s_{2k-1}$ and constants $\delta_1, \ldots, \delta_k \in \mathbb{R}_{\geq 0}$ such that $s \stackrel{\tau}{\rightarrow} s_1 \stackrel{\delta_1}{\rightarrow} s_2 \cdots \stackrel{\tau}{\rightarrow} s_{2k-1} \stackrel{\delta_k}{\rightarrow} s'$ and $\Sigma_{i=1}^{k} \delta_k = \delta$. The binary relation $\succeq \subseteq S_1 \times S_2$ is a *simulation* of T_2 by T_1 if the following four conditions hold:

- $\mathcal{L}_1 = \mathcal{L}_2$, and
- for every state $s_1 \in S_1$ and $s_2 \in S_2$, if $s_1 \succeq s_2$, then for each label $m \in \mathcal{L}_2 \setminus \{\tau\} \cup \mathbb{R}_{\geq 0}$, if $s_2 \stackrel{m}{\rightarrow\!\!\!\!\rightarrow} s_2'$, then there exists a state s_1' such that $s_1 \stackrel{m}{\rightarrow\!\!\!\!\rightarrow} s_1'$ and $s_1' \succeq s_2'$, and
- for every initial state $s_2 \in I_2$ of B, there exists an initial state $s_1 \in I_1$ of A such that $s_1 \succeq s_2$, and
- for every final state $s_2 \in F_2$ of B, and for every state $s_1 \in S_1$ of A, if $s_1 \succeq s_2$, then $s_1 \in F_1$.

All τ transitions are silent, so this corresponds to the notion of *weak* simulation of Milner [Mil89]. The labeled transition system T_1 simulates T_2, denoted $T_1 \succeq_{sim} T_2$, if there exists a simulation $\succeq$ of T_2 by T_1. The hybrid automaton A *simulates* B if $[\![A]\!] \succeq_{sim} [\![B]\!]$. Let $\equiv$ be a binary relation over $S_1 \times S_2$. Then define $\equiv^{-1}$ to be the binary relation over $S_2 \times S_1$ such that $(s_2, s_1) \in \equiv^{-1}$ iff $(s_1, s_2) \in \equiv$. The binary relation $\equiv \subseteq S_{T_1} \times S_{T_2}$ is a *bisimulation* between T_1 and T_2 if it is a simulation of T_2 by T_1 and $\equiv^{-1}$ is a simulation between T_1 by T_2. The labeled transition systems T_1 and T_2 are *bisimilar*, denoted $T_1 \equiv_{bis} T_2$, if there exists a bisimulation $\equiv$ between T_1 and T_2. The two hybrid automata A and B are *bisimilar* if $[\![A]\!] \equiv_{bis} [\![B]\!]$.

Proposition 2.1 *Let A and B be hybrid automata.*

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

Remark. The notions of simulation and bisimilarity are unnecessarily strong conditions for emptiness checking. However, they are useful for model-checking more general classes of properties.

2.4 Control mode splitting

We often find it useful to split the control modes of a hybrid automaton in order to enable more accurate approximations over the split control modes. A *flow split* $\mathcal{P}$ is a function mapping each control mode v to a finite set $\{flow_1, flow_2, \ldots, flow_k\}$ of k predicates over $X \cup \dot{X}$, such that there exists a finite open cover $\mathcal{O}$ of $flow(v)$ such that $\mathcal{P}(v) = \{flow(v) \cap O_i \mid O_i \in \mathcal{O}\}$. The definition implies that the disjunction $\bigvee_{i=1}^{k} flow_i$ is equivalent to $flow(v)$, and also that splitting of the flow condition does not prohibit flow transitions. Whenever there is a flow transition of the original automaton originating at state s, there is also a flow transition of the split automaton originating from a state derived from s for one of the flow conditions $flow_i$. Let $A = (X_A, V_A, flow_A, E_A, jump_A, \Sigma_A, event_A, init_A, final_A)$ be a hybrid automaton. Applying the flow split $\mathcal{P}$ to the hybrid automaton A yields the following *flow-split automaton* $\mathcal{P}(A) = (X, V, flow, E, jump, \Sigma, event, init, final)$:

- $X = X_A$.

- $V = \{(v, \phi) \mid v \in V_A \text{ and } \phi \in \mathcal{P}(v)\}$.
- For every control mode $(v, \phi) \in V$, $\mathit{flow}(v, \phi) = \phi$.
- $E = E_1 \cup E_2$, where $E_1 = \{((v, \phi), (v', \phi')) \mid (v, v') \in E_A, \phi \in \mathcal{P}(v), \text{ and } \phi' \in \mathcal{P}(v')\}$ and $E_2 = \{((v, \phi), (v, \phi')) \mid \phi, \phi' \in \mathcal{P}(v)\}$. Intuitively, control switches in E_1 are inherited from control switches of E_A and control switches in E_2 are *silent control switches* enabling control to pass freely across copies of the same control mode.
- For all control switches $e = ((v, \phi), (v', \phi')) \in E_1$, $\mathit{jump}((v, \phi), (v', \phi')) = \mathit{jump}_A(v, v')$, *i.e.* the jump condition is inherited from the corresponding control switch (v, v') of A. For all control switches $e = ((v, \phi), (v, \phi')) \in E_2$, $\mathit{jump}((v, \phi), (v, \phi')) = \mathit{Stable}(X, \dot{X})$, *i.e.* the jump condition requires the state to remain unchanged.
- $\Sigma = \Sigma_A$.
- For every control switch $\epsilon = ((v, \phi), (v', \phi')) \in E_1$, $\mathit{event}((v, \phi), (v', \phi')) = \mathit{event}_A(v, v')$, and for every control switch $\epsilon = ((v, \phi), (v, \phi')) \in E_2$, $\mathit{event}((v, \phi), (v, \phi')) = \tau$.
- For every control mode $(v, \phi) \in V$, $\mathit{init}(v, \phi) = \mathit{init}_A(v)$.
- For every control mode $(v, \phi) \in V$, $\mathit{final}(v, \phi) = \mathit{final}_A(v)$.

We define the function $\pi_1 : S_{\mathcal{P}(A)} \to S_A$ by $\pi_1((v, \phi), \mathbf{x}, \dot{\mathbf{x}}) = (v, \mathbf{x}, \dot{\mathbf{x}})$. The function is extended to sets of states in the natural way.

The hybrid automaton A is *splittable* if for all flow splits $\mathcal{P}$, $\mathcal{P}(A)$ and A are bisimilar. It may appear to be the case that for some flow split $\mathcal{P}$, the flow-split automaton $\mathcal{P}(A)$ does not simulate A, since there may be a flow transition in $[\![A]\!]$ for which there is no simulating sequence of flow transitions interleaved with silent transitions in $[\![\mathcal{P}(A)]\!]$. This anomaly could arise if the only witness trajectory for the flow transition $s \xrightarrow{\delta} s'$ in $[\![A]\!]$ has unbounded variability and cannot be mimicked by any finite sequence in $\mathcal{P}(A)$. The following theorem states that 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 claim that the relation $\equiv \subseteq S_A \times S_{\mathcal{P}(A)}$, defined by $s_1 \equiv s_2$ iff $\pi_1(s_2) = s_1$, is a bisimulation.

First consider simulation of $\mathcal{P}(A)$ by A. Suppose that $s_1 \xrightarrow{\delta_1}_{\mathcal{P}(A)} s_2 \xrightarrow{\tau} s_3 \xrightarrow{\delta_2}_{\mathcal{P}(A)} s_4$. By construction of the jump conditions for silent control mode switches, the witness trajectories for $s_1 \xrightarrow{\delta_1} s_2$ and $s_3 \xrightarrow{\delta_2} s_4$ can be concatenated into a witness trajectory for $\pi_1(s_1) \xrightarrow{\delta_1+\delta_2} \pi_1(s_4)$ since the right derivative at s_2 matches the left derivative at s_3. An inductive argument shows that transitions of the form $s \xrightarrow{\delta}_{\mathcal{P}(A)} s'$ can be simulated in A. It is straightforward to see that jump transitions are also simulated, since control mode switches in $\mathcal{P}(A)$ are directly inherited from A.

Consider now the simulation of A by $\mathcal{P}(A)$. Again, it is not hard to see that jump transitions are simulated. For flow transitions, we prove that if $s \xrightarrow{\delta}_A s'$, then there exist states s_2 and s_2' of $\mathcal{P}(A)$ such that $s_2 \xrightarrow{\delta} s_2'$ and $\pi_1(s_2) = s$ and $\pi_1(s_2') = s'$. It suffices to consider the case where $s \xrightarrow{\delta}_A s'$, since sequences of transitions can be concatenated. Suppose that $s = (v, \mathbf{x}, \dot{\mathbf{x}})$ and $s' = (v, \mathbf{x}', \dot{\mathbf{x}}')$. Let $\rho : [0, \delta] \to \mathbb{R}^n$ be a witness trajectory for $s \xrightarrow{\delta}_A s'$. Let $\mathcal{O}$ be the open cover from which the set of flow conditions $\mathcal{P}(v)$ is derived. For each $t \in (0, \delta)$, let B_t be an open ball containing $\rho(t)$

and entirely contained in some open set $O \in \mathcal{O}$. Such a ball exists since $\mathcal{O}$ is an open cover of $\mathit{flow}(v)$ and all points in the range of ρ satisfy $\mathit{flow}(v)$. Since ρ is continuous, it follows that the set $\rho^{-1}(B_t)$ is open, and hence includes an open interval I_t containing t. Thus the set of intervals $\{I_t \mid t \in (0, \delta)\}$ 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 I_t. 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 witness trajectories between the points, where each witness trajectory lies entirely within some open set from the cover $\mathcal{O}$. ■

Remark. Our definition of flow splits precludes the splitting of flow conditions into closed flow conditions. The above theorem still holds if the definition of a flow split need not require the cover to be open, but instead that it be flow preserving. A cover $\mathcal{F}$ for the flow condition $\mathit{flow}(v)$ is *flow-preserving* if for all w satisfying $\mathit{flow}(v)$, there exists an open ball B containing w such that $B \cap [\![\mathit{flow}(v)]\!] = \bigcup\{B \cap F \mid F \in \mathcal{F} \text{ and } w \in F\}$. In particular, this relaxation allows the flow condition $1 \leq x \leq 3$ to be split into $1 \leq 2$ and $2 \leq x \leq 3$, but not into $1 \leq 2$ and $2 < x \leq 3$.

2.5 Linear hybrid automata

Our goal is to approximate hybrid automata by rational linear hybrid automata, since they form a subclass of hybrid automata that can be analyzed effectively. A *linear expression* over a set X of variables is a linear combination $\sum_{i=1}^{k} \alpha_i x_i$ of variables with real-valued coefficients $\alpha_i \in \mathbb{R}$. A linear expression is *rational* if all its coefficients are rational. A variable x of a hybrid automaton A is *(rational and) linear* if the following four conditions hold:

1. for all $v \in V$, all occurrences of x in the initial condition $\mathit{init}(v)$ and the final condition $\mathit{final}(v)$ are in (rational) linear expressions,
2. for all $v \in V$, all occurrences of x and $\dot{x}$ in the flow condition $\mathit{flow}(v)$ are in (rational) linear expressions,
3. for all $v \in V$, the flow condition $\mathit{flow}(v)$ can be decomposed into the form $\varphi_x \wedge \varphi_{\dot{x}}$ where φ_x is a predicate over X and $\varphi_{\dot{x}}$ is a predicate over $\dot{X}$, and
4. for all $e \in E$, all occurrences of x, $\dot{x}$, x', and $\dot{x}'$ in the jump condition $\mathit{jump}(e)$ are in (rational) linear expressions.

Since every linear set has a finite convex cover, it is convenient for us, without loss of generality, to define linear hybrid automata as having only convex predicates. A predicate ϕ is *convex* if $[\![\phi]\!]$ is a convex set. A hybrid automaton A is a *(rational) linear hybrid automaton* if

1. all variables of A are (rational and) linear,
2. for all control modes $v \in V$, the initial condition $\mathit{init}(v)$, the final condition $\mathit{final}(v)$, and the flow condition $\mathit{flow}(v)$ are all convex, and
3. for all control switches $e \in E$, the jump condition $\mathit{jump}(e)$ is convex.

In flow conditions, linear dependencies between the rates of variables can be expressed, although the flow field must be independent of the *values* of the variables in X. For example, the linear hybrid automata may have the predicate $x \geq y \wedge \dot{x} \leq 3\dot{y} + 2$ as a flow condition, but not the predicate $x \leq \dot{x}$.

Let A be a hybrid automaton. We define the functions $post : 2^{S_A} \to 2^{S_A}$ by $post(W') = \{s_2 \mid \exists s_1 \in W.s_1 \to_A s_2\}$, and $pre : 2^{S_A} \to 2^{S_A}$ by $pre(W) = \{s_1 \mid \exists s_2 \in W.s_1 \to_A s_2\}$. The set of reachable states of A is the set $\bigcup_{i=0}^{\infty} post^i(W_0)$, where W_0 is the set of initial states of A. A *linear inequality* over a set X is an inequality between linear expressions over X. A *linear predicate* is a disjunction of conjunctions of linear inequalities. A region W is *linear* if there exists a linear predicate ϕ such that $W = [\![\phi]\!]$.

Theorem 2.3 *Let A be a hybrid automaton. If W is a linear region of A, then $post(W)$ and $pre(W)$ are computable linear regions.*

Proof Sketch. The proof of the theorem is similar to that for previous definitions of hybrid automata in the literature [AHH96], where the notion of state does not include a valuation over $\dot{X}$. Details are omitted here.

We need to compute the successors via jump transitions and flow transitions. The proof for jump transitions is analogous to that for the definition of state appearing in [AHH96]—one can treat the variables in $\dot{X}$ on the same basis as those in X.

Consider flow transitions originating from the state $s = (v, \mathbf{x}, \dot{\mathbf{x}})$ where the flow condition of v is $\varphi = \varphi_x \wedge \varphi_{\dot{x}}$ for the linear predicates φ_x over X and $\varphi_{\dot{x}}$ over $\dot{X}$. We need to compute not only the reachable valuations over X, but also the flow tangents obtained at those valuations. Let $post_{\varphi}^{\geq 0}(s)$ be the set $\{s' \mid \exists \delta \in \mathbb{R}_{>0}.s \xrightarrow{\delta} s'\}$. It suffices to show how to compute $post_{\varphi}^{\geq 0}(s)$. Let $\widetilde{post}_{\varphi} : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ be defined by $\widetilde{post}_{\varphi}(\mathbf{y}) = \{\mathbf{z} \mid \varphi_x(\mathbf{y}) \wedge \varphi_x(\mathbf{z}) \wedge \exists \delta \in \mathbb{R}_{\geq 0}.\varphi_{\dot{x}}[\dot{X} := (\mathbf{z} - \mathbf{y})/\delta]\}$. Computing the function $\widetilde{post}$ corresponds to computing reachable valuations, independent of the reachable flow tangents at those valuations. Observe that any reachable valuation $\mathbf{y}$ over X can be reached with a flow tangent $\dot{\mathbf{y}}$ provided there is another reachable valuation $\mathbf{y}'$ over X such that the line between $\mathbf{y}'$ and $\mathbf{y}$ is in the direction $\dot{\mathbf{y}}$. Thus

$$\begin{aligned} post_{\varphi}^{\geq 0}(s) \;=\; \{(v, \mathbf{z}, \dot{\mathbf{z}}) \mid\; & flow(v)[X, \dot{X} := \mathbf{y}, \dot{\mathbf{y}}] \text{ and} \\ & flow(v)[X, \dot{X} := \mathbf{z}, \dot{\mathbf{z}}] \text{ and} \\ & \mathbf{z} \in \widetilde{post}_{\varphi}(\mathbf{y}) \text{ and} \\ & \exists k \in \mathbb{R}_{>0}.\mathbf{y} + k\dot{\mathbf{y}} \in \widetilde{post}_{\varphi}(\mathbf{y}) \text{ and} \\ & \exists m \in \mathbb{R}_{>0}.\mathbf{z} - m\dot{\mathbf{z}} \in \widetilde{post}_{\varphi}(\mathbf{y})\} \end{aligned}$$

This region is linear and computable. ■

If a fixpoint is reached when iterating the *post* function from the initial states, we gain a linear representation of the set of reachable states.

Corollary 2.4 *The emptiness problem for linear hybrid automata is co-recursively enumerable.* ■

Algorithmic analysis techniques for rational linear hybrid automata have been implemented in tools such as HyTech [HHWT95a] and Polka [HRP94].

3 Clock Translation

Given a solvable nonlinear hybrid automaton A, we construct a linear hybrid automaton A_x—the *clock translation* of A—by replacing each nonlinear variable x with a clock t_x. A *clock* is a linear variable whose slope is always 1, *i.e.* $\dot{t}_x = 1$ is implied by all flow conditions. The idea behind the translation is that the clock t_x carries enough information about the value of the variable x. If t_x records the time elapsed since the value of x was last changed by a control switch, and the value

of that last change is recorded, and the solution curve of the nonlinear variable can be uniquely solved, then the current value of x can be determined from the value of t_x. All atomic predicates involving x are replaced by atomic predicates on t_x.

3.1 Conditions for solvability

The clock translation can only be applied to the nonlinear variable x in automaton A when the predicates in A do not compare x to other variables, and the differential equations describing the flow of x are independent of other variables and solvable.

Our goal is to translate the nonlinear variable x into the clock t_x, and translate predicates involving x into predicates involving t_x. We require the translated predicates to be true whenever the original predicates are. Before giving the formal definitions, we intuitively describe the conditions we require for the variable x to be solvable. First, we require that x be simple, *i.e.* independent of the other variables in initial, final, and jump conditions. We also require independence of its flow condition. 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 have a unique solution curve 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 predicates such as $x \geq c$, for some real $c \in \mathbb{R}$, by the value of t_x, we require solution curves for 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 $t_x \geq m$ where m is the time it takes for the unique solution curve for x to progress from b to c. Fourth, we require that at any instant, we know the initial value of the solution curve x is currently following as well as the time that has elapsed. For example, suppose x is initially 1 and subject to a flow condition that implies $\dot{x} = x$. Suppose control of the automaton can switch to another control mode v' when the variable y equals 2. If the flow condition for control mode v' also implies $\dot{x} = x$, then the value of x can be determined from the time elapsed from the instant when x had value 1, *i.e.* $x = e^{t_x}$, regardless of when the control switch to location v' took place. However, if $\mathit{flow}(v')$ differs from $\mathit{flow}(v)$, *e.g.* it implies $\dot{x} = -x + 4$, then it is no longer possible to determine the value of x. Intuitively, we say that the variable x is definite for the control switch $e \in E$ if the jump condition $\mathit{jump}(e)$ implies $x' = c$ for some real $c \in \mathbb{R}$. We require control switches to be definite for x if the flow conditions for the source and target locations differ.

3.2 Solvable variables

We now provide formal definitions for the concepts introduced above. An atomic predicate is *simple* if it is of the form $x \sim c$, $x' \sim c$, $x' = x$, or $\dot{x}' = \dot{x}$, where $x \in X$, $\sim \in \mathit{RelOps}$, and $c \in \mathbb{R}$. A predicate ϕ is *simple* for x if every occurrence of x in ϕ is in a simple atomic predicate. The variable x is *simple* in the automaton A if every invariant, initial condition, final condition, and jump condition is simple for x. The variable x is *flow independent* in control mode v if $\mathit{flow}(v)$ is of the form $\mathit{flow}_{v,x} \wedge \mathit{flow}_{v,Y}$ for some predicates $\mathit{flow}_{v,x}$ over $\{x, \dot{x}\}$ and $\mathit{flow}_{v,Y}$ over $Y \cup \dot{Y}$, where $Y = X \setminus \{x\}$. In this case, we say that x has the *independent flow condition* $\mathit{flow}_{v,x}$ in control mode v. For example, the variable x is flow independent in the flow condition $\dot{x} = 4x^2 \wedge \dot{y} = \dot{z} \wedge w \leq \dot{y}$, whereas the variables w, y, and z are not. The variable x is *uniquely determined* in location v if for all initial values $x_0 \in \mathbb{R}$, the initial-value problem "$\dot{y}(t) = f_{v,x}(y(t)); y(0) = x_0$" has a unique algebraic solution $g_{v,x_0}(t)$, and that solution is strictly monotone.

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 *initial value* $c \in \mathbb{R}$. The variable x is definite for the control switch

$e \in E$ of A if the jump condition $jump(e)$ implies $x' = c$, for some *arrival value* $c \in \mathbb{R}$.

The variable x of A is *solvable* if the following four conditions hold:

1. the variable x is simple,
2. for all control modes $v \in V$ of A, x is flow independent and uniquely determined in v,
3. for all control modes of $v \in V$, x is initially definite for v,
4. for all control switches $e = (v, v') \in E$ of A, if x is not definite for e, then $flow_{v,x} = flow_{v',x}$ and $jump(e)$ implies $x' = x$.

The hybrid automaton A is *solvable* if all its nonlinear variables are solvable.

Example 3.1 The thermostat automaton of Figure 1 is solvable, since its only nonlinear variable x is solvable. ■

Remark. The strict monotonicity condition can be relaxed in various ways, at the expense of complicating the proofs. The key property we require is that for each initial-value problem $\mathcal{I}$ that arises in the automaton, the unique solution $g : \mathbb{R}_{\geq 0} \to \mathbb{R}$ of $\mathcal{I}$ is such that for each constant c that appears in an atomic predicate of the form $x \sim c$ or $x' \sim c$, if there exists a $t \geq 0$ such that $g(t) = c$, then t is unique.

Remark. There is a straightforward condition that implies strict monotonicity of solution curves for a nonlinear variable. Suppose x is flow independent in control mode v with deterministic flow function f. Then every algebraic solution of f is monotone, since it cannot be both decreasing and increasing at any point. If for all reals $d \in \mathbb{R}$, we have that $f(d) \neq 0$, then all its algebraic solutions are strictly monotone, since their derivatives are never 0.

3.3 The clock translation algorithm

The clock translation algorithm applies only to solvable hybrid automata, and it yields a linear hybrid automaton. For each solvable nonlinear variable x, let $c \in \mathbb{R}$ be a *starting value* for x if there exists an initial state $(v, \mathbf{x}, \dot{\mathbf{x}})$ of A in which x has value c, or c is the arrival value of x for some definite control switch[2]. Let $Start(x) = \{c_1, \ldots, c_n\}$, with $c_1 < \cdots < c_n$, be the set of all starting values for x. For each solvable nonlinear variable x, the construction proceeds in two steps:

1. Each control mode v of A is split into a collection $(v, c_1), \ldots, (v, c_n)$ 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 control mode (v, c_i) is $x(t_x)$, where $x(t)$ is the solution of the initial-value problem "$\dot{y}(t) = flow_{v,x}(y(t))$; $y(0) = c_i$".
2. All initial and final conditions, and jump conditions are translated from conditions on x to conditions on t_x, and the variable x is discarded.

[2]For simplicity, we consider a global set of starting values, rather than parametrizing by control modes.

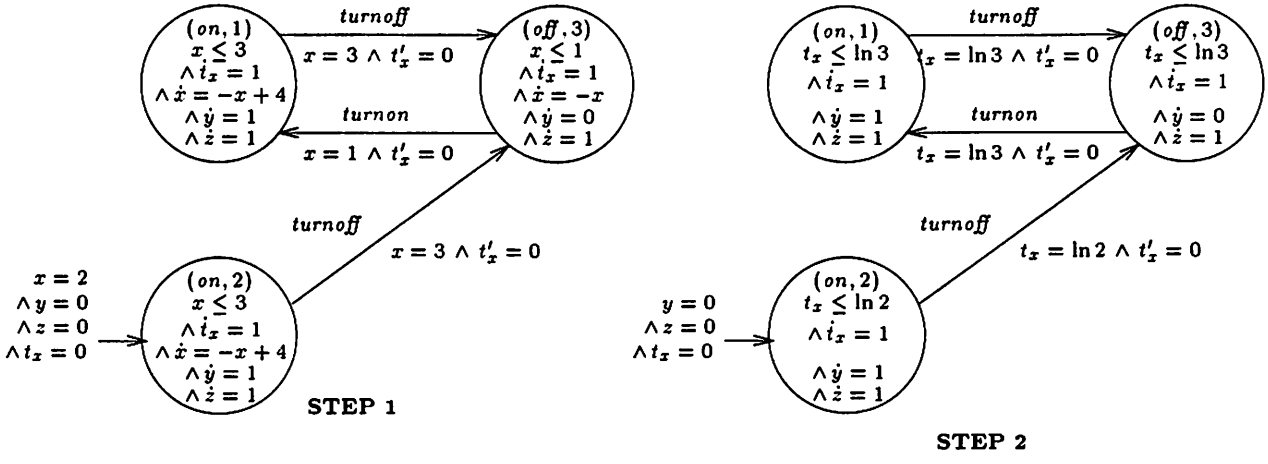

Figure 2: Clock translation of the thermostat automaton

Step 1. Splitting control modes and control switches

In this step, we create a new automaton A_1 derived from an automaton A by splitting each control mode into a set of control modes, one for each starting value. Thus the control modes of A_1 are $V_A \times \mathit{Start}(x)$. Each new control mode (v, c_i) has flow condition $\mathit{flow}_{A_1}(v, c_i) = \mathit{flow}_A(v) \wedge \dot{t}_x = 1$, reflecting the behavior of the clock t_x. The new control mode (v, c_i) has the initial condition $\mathit{init}_{A_1}(v, c_i) = \mathit{init}_A(v) \wedge t_x = 0$ if $\mathit{init}_A(v)$ implies $x = c_i$, and $\mathit{init}_{A_1}(v, c_i) = \mathit{false}$ otherwise. The final condition of the control mode (v, c_i) is $\mathit{final}_{A_1}(v, c_i) = \mathit{final}_A(v)$. For each indefinite control switch $e = (v, v')$ of A, the automaton A_1 includes all control switches of the form $((v, c_i), (v', c_i))$ with jump condition $\mathit{jump}_A(e) \wedge t_x = t'_x$. For each definite control switch $e = (v, v')$ of A with the arrival value c_j, A_1 includes all control switches of the form $((v, c_i), (v', c_j))$ with jump condition $\mathit{jump}_A(e) \wedge t'_x = 0$.

Example 3.2 The thermostat automaton of Figure 1 has only definite control switches. 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 automaton control switches from the initial control mode $(on, 2)$, we remove these three control modes from the clock-translated automaton. The result of Step 1 is shown on the left in Figure 2. ■

Step 2. Updating final conditions, flow conditions, and jump conditions

In this step, we derive a new automaton A_2 from an automaton A_1 resulting from Step 1 above. We eliminate the nonlinear variable x from the initial conditions and flow conditions, and replace predicates involving x with predicates involving t_x in the final conditions and the jump conditions. Let the function $g_{v,c}(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_{v,x}(y(t))$; $y(0) = c$". We first define a transformation function $F_{v,c}$ from simple atomic predicates over x to simple atomic predicates over t_x as follows:

$$F_{v,c}(x \sim k) = \begin{cases} \mathit{true} & \text{if } \beta_{v,c}(k) = \perp \text{ and } c \sim k \\ \mathit{false} & \text{if } \beta_{v,c}(k) = \perp \text{ and } c \not\sim k \\ t_x \; lt(\sim)\, \beta_{v,c}(k) & \text{if } \beta_{v,c}(k) \neq \perp \text{ and } c \sim k \\ t_x \; gt(\sim)\, \beta_{v,c}(k) & \text{if } \beta_{v,c}(k) \neq \perp \text{ and } c \not\sim k \end{cases}$$

where $\sim \in \mathit{RelOps}$, the function $\beta_{v,c} : \mathbb{R} \to \mathbb{R} \cup \{\perp\}$ is defined by $\beta_{v,c}(k) = d$ if $g_{v,c}(d) = k$ and $\beta_{v,c}(k) = \perp$ if there is no solution to the equation $g_{v,c}(t_x) = k$, and $lt : \mathit{RelOps} \to \mathit{RelOps}$ and $gt : \mathit{RelOps} \to \mathit{RelOps}$ are defined by

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

We conduct the following four steps for each control mode. Consider control mode (v, c_i).

1. We remove all atomic predicates that involve the variable x from the initial condition $\mathit{init}(v, c_i)$.
2. We likewise remove all atomic predicates that involve the variables x and x' from the flow condition $\mathit{flow}(v, c_i)$.
3. We translate the final condition of control mode (v, c_i). Suppose that $x \sim k$, for some $\sim \in \mathit{RelOps}$, is an atomic predicate of the final condition $\mathit{final}(v, c_i)$. Replace $x \sim k$ with $F_{v,c_i}(x \sim k)$.
4. We translate the flow condition of control mode (v, c_i). Suppose that $x \sim k$, for some $\sim \in \mathit{RelOps}$, is an atomic predicate of the flow condition $\mathit{inv}(v, c_i)$. If $c_i \not\sim k$, remove control mode (v, c_i), since the flow condition cannot be satisfied. Otherwise, replace $x \sim k$ with $F_{v,c_i}(x \sim k)$.
5. We translate the jump conditions of all control switches that leave the control mode (v, c_i). Consider the control switch $e = (v, v')$. First, we delete all atomic predicates of the form $x' \sim k$ from the jump condition $\mathit{jump}(e)$. Then, replace in the jump condition $\mathit{jump}(e)$, each atomic predicate of the form $x \sim k$, for some $\sim \in \mathit{RelOps}$, with $F_{v,c_i}(x \sim k)$.

Example 3.3 In the thermostat example, we have solutions $g_{on,1}(t) = -3e^{-t} + 4$, $g_{on,2}(t) = -2e^{-t} + 4$, and $g_{off,3}(t) = 3e^{-t}$. Consider the predicate $x = 3$ of the jump condition of the control switch from on_2 to off_3. Since $\ln 2$ is the solution of $-2e^{-t}+4 = 3$, it follows that $x = 3$ iff $t_x = \ln 2$. Hence the predicate $x = 3$ is replaced by $t_x = \ln 2$. The final result of Step 2 is shown on the right in Figure 2. ■

3.4 Bisimilarity

We show that the clock translation is complete for checking the emptiness of solvable automata. Let A be a solvable hybrid automaton, and let A_x be the automaton derived from A by translating a nonlinear variable x into a clock t_x. We show that A and A_x are bisimilar. We define the function

$\alpha_x\ S_{A_x} \to S_A$ such that $\alpha_x((v,c),\mathbf{x}_1,\dot{\mathbf{x}}_1) = (v,\mathbf{x}_2,\dot{\mathbf{x}}_2)$, where the control mode (v,c) is split from the control mode v for the starting value c, the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ agree on all variables except x and t_x, $\mathbf{x}_2(x) = g_{v,c}(\mathbf{x}_1(t_x))$ if the function $g_{v,c}(t)$ is the solution of the initial-value problem "$\dot{y}(t) = f_{v,x}(y(t))$; $y(0) = c$", and vectors $\dot{\mathbf{x}}_1$ and $\dot{\mathbf{x}}_2$ agree on all variables except $\dot{x}$ and $\dot{t}_x$, and $\dot{\mathbf{x}}_2(\dot{x}) = f_{v,x}(g_{v,c}(\mathbf{x}_1(t_x)))$. We define the binary relation $\equiv_x \subseteq S_{A_x} \times S_A$ by $\{(\sigma_1,\sigma_2) \mid \sigma_2 = \alpha_x(\sigma_1)\}$.

Lemma 3.1 *Let A be a hybrid automaton with a solvable nonlinear variable x, and let A_x be the clock translation of A that results from replacing the variable x by the clock t_x. Suppose $((v,c),\mathbf{x}_1,\dot{\mathbf{x}}_1) \equiv_x (v_2,\mathbf{x}_2,\dot{\mathbf{x}}_2)$. Then, for all predicates of the form $x \sim k$ for $\sim \in \{<,\leq,=,\geq,>\}$ and $k \in \mathbb{R}$, $\mathbf{x}_1$ satisfies $F_{v,c}(x \sim k)$ iff $\mathbf{x}_2$ satisfies $x \sim k$.*

Proof. Let $s_1 = ((v,c),\mathbf{x}_1,\dot{\mathbf{x}}_1)$ be a state of A_x and let $s_2 = (v_2,\mathbf{x}_2,\dot{\mathbf{x}}_2)$ be a state of A such that $s_1 \equiv_x s_2$. Let $g_{v,c}(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_{v,x}(y(t)); y(0) = c$," and let $\mathbf{x}_2(x) = k$. Then by the definition of $\equiv_x$ and α_x, we have $g_{v,c}(\mathbf{x}_1(t_x)) = k = \mathbf{x}_2(x)$. We now consider the four cases arising in the definition of $F_{v,c}$ in Subsection 3.3.

1. Assume $\beta_{v,c}(k) = \perp$, and $c \sim k$. In this case, $\sim$ cannot be the equality relation. Since $g_{v,c}(t)$ is monotonic, $c = g_{v,c}(0) \sim k$ for the inequality relation $\sim$, and for all $t \geq 0$, $g_{v,c}(t) \neq k$, it follows that for all $t \geq 0$, $g_{v,c}(t) \sim k$. Hence $\mathbf{x}_2(x) \sim k$, and so $\mathbf{x}_2$ satisfies $x \sim k$ iff $\mathbf{x}_1$ satisfies *true*.

2. Assume $\beta_{v,c}(k) = \perp$, and $c \not\sim k$. Suppose $\sim$ is the equality relation. Then since for all t, $g_{v,c}(t) \neq k$, it follows that $x = k$ is not satisfied in $\mathbf{x}_2$, and also $\mathbf{x}_1$ does not satisfy $F_{v,c}(x = k)$, which is the predicate *false*. If $\sim$ is an inequality, then by continuity, $g_{v,c}(t) \not\sim k$ for all $t \geq 0$.

3. Assume $\beta_{v,c}(k) \neq \perp$, and $c \sim k$. If $\sim$ is the equality relation, then it is clear that $\mathbf{x}_2$ satisfies $x \sim c = k$ iff $\mathbf{x}_1$ satisfies $t_x \sim \beta_{v,c}(k) = \beta_{v,c}(c) = 0$. For inequalities φ of the form $x < k$ or $x > k$, by monotonicity, $\mathbf{x}_2$ satisfies φ exactly when $\mathbf{x}_1$ satisfies $t_x < \beta_{v,c}(k)$, *i.e.* as long as x has not reached the cutoff value k, t_x has not reached the cutoff value $\beta_{v,c}(k)$. Similarly, for inequalities φ of the form $x \leq k$ or $x \geq k$, $\mathbf{x}_2$ satisfies φ exactly when $\mathbf{x}_1$ satisfies $t_x \leq \beta_{v,c}(k)$.

4. Assume $\beta_{v,c}(k) \neq \perp$, and $c \not\sim k$. The proof is analogous to that of Case 3 above. ∎

Lemma 3.2 *Let A be a hybrid automaton with a solvable nonlinear variable x, and let A_x be the clock translation of A that results from replacing the variable x by the clock t_x. Then $\equiv_x$ is a simulation of $[\![A]\!]$ by $[\![A_x]\!]$.*

Proof. Let $s_2 = (v,\mathbf{x}_2,\dot{\mathbf{x}}_2)$ and $s_2' = (v',\mathbf{x}_2',\dot{\mathbf{x}}_2')$ be states of A. Let $s_1 = ((v,c),\mathbf{x}_1,\dot{\mathbf{x}}_1)$ be such that $s_1 \equiv_x s_2$. We show that if $s_2 \xrightarrow{m} s_2'$, then there exists a state s_1' such that $s_1 \xrightarrow{m} s_1'$ and $\alpha_x(s_1') = s_2'$. This is sufficient to show that if $s_2 \xrightarrow{m} s_2'$, then there exists a state s_1' such that $s_1 \xrightarrow{m} s_1'$ and $\alpha_x(s_1') = s_2'$.

Consider first flow transitions. Suppose that $s_2 \xrightarrow{\delta} s_2'$, with witness trajectory $\rho_2 : [0,\delta] \to \mathbb{R}$ having duration $\delta \geq 0$. We will construct a witness trajectory $\rho_1 : [0,\delta] \to \mathbb{R}$ from s_1 to some state s_1' such that $s_1' \equiv_x s_2'$. Let $t_a = \mathbf{x}_1(t_x)$. 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 $\rho_1(t)(t_x) = t_a + t$. Clearly ρ_1 satisfies the flow condition, since t_x is a clock and all other variables are unaffected by the clock translation. By Lemma 3.1 and the clock translation for the invariant, it follows that for all $t \in [0,\delta]$, $\rho_1(t) \in [\![inv(v,c)]\!]$ since $\rho_2(t) \in [\![inv(v)]\!]$. It is straightforward to see that $\rho_1(0) = \mathbf{x}_1$ and $\dot{\rho}_1(0)(t_x) = 1 = \mathbf{x}_1(\dot{t}_x)$. Finally,

we establish that $\alpha_x((v,c),\rho_1(\delta)) = s_2'$. Let $g_{v,c}(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_{v,x}(y(t)); y(0) = c$". By definition of α_x, $\alpha_x((v,c),\rho_1(\delta))(x) = g_{v,c}(\delta + t_a) = \mathbf{x}_2'(x)$, and $\alpha_x((v,c),\rho_1(\delta))(\dot{x}) = f_{v,c}(g_{v,c}(\delta + t_a)) = f_{v,c}(\mathbf{x}_2'(x)) = \mathbf{x}_2'(\dot{x})$. The other variables agree because x is flow independent.

We now consider jump transitions. Suppose $s_2 \xrightarrow{m} s_2'$ where $m = event(e_2)$ for the control switch $e_2 = (v, v')$. We will consider two cases to show the existence of a state $s_1' = ((v', c'), \mathbf{x}_1', \dot{\mathbf{x}}_1') \in S_{A_x}$ such that $s_1' \equiv_x s_2'$ and $s_1 \xrightarrow{m} s_1'$ in $[\![A_x]\!]$.

1. Assume that $jump(e_2)$ implies $x' = d$, for some $d \in \mathbb{R}$. In this case, according to the clock translation, there is a control switch $e_1 = ((v,c),(v',d))$ in A_x such that $jump(e_1)$ implies $t_x' = 0$ and $event(e_1) = m$. Let $\mathbf{x}_1'$ be such that for all $y \neq t_x$, $\mathbf{x}_1'(y) = \mathbf{x}_2'(y)$ and $\mathbf{x}_1'(t_x) = 0$, and for all $y \neq t_x$, $\dot{\mathbf{x}}_1'(\dot{y}) = \dot{\mathbf{x}}_2'(\dot{y})$ and $\dot{\mathbf{x}}_1'(\dot{t}_x) = 1$. Then by construction of $jump(e_1)$ and Lemma 3.1, $s_1 \xrightarrow{m} ((v', d), \mathbf{x}_1', \dot{\mathbf{x}}_1') = s_1'$. Furthermore, $s_1' \equiv_x s_2'$ since the action on all variables other than x in $\mathbf{x}_1'$ is the same as for $\mathbf{x}_2'$, and $\mathbf{x}_2'(x) = d$ is the initial value of the solution curve for control mode (v', d), and the action on all variables other than $\dot{x}$ in $\dot{\mathbf{x}}_1'$ is the same as for $\dot{\mathbf{x}}_2'$, and $\dot{\mathbf{x}}_2'(\dot{x}) = f_{v,x}(d) = g_{v,d}(\mathbf{x}_1'(t_x))$.

2. Assume that $jump(e_2)$ implies $x' = x$. In this case, according to the clock translation, there is a control switch $e_1 = ((v,c),(v',c))$ in A_x such that $jump(e_1)$ implies $t_x' = t_x$ and $event(e_1) = m$. Let $\mathbf{x}_1'$ be such that for all $y \neq t_x$ $\mathbf{x}_1'(y) = \mathbf{x}_2'(y)$ and $\dot{\mathbf{x}}_1'(\dot{y}) = \dot{\mathbf{x}}_2'(\dot{y})$, and $\mathbf{x}_1'(t_x) = \mathbf{x}_1(t_x)$ and $\dot{\mathbf{x}}_1'(\dot{t}_x) = 1$. Then by construction of $jump(e_1)$ and Lemma 3.1, $s_1 \xrightarrow{m} ((v', c), \mathbf{x}_1', \dot{\mathbf{x}}_1') = s_1'$. Furthermore, $s_1' \equiv_x s_2'$ since the action on all variables other than x and $\dot{x}$ in $(\mathbf{x}_1', \dot{\mathbf{x}}_1')$ is the same as for $(\mathbf{x}_2', \dot{\mathbf{x}}_2')$, and $\mathbf{x}_2'(x) = \mathbf{x}_2(x)$, $\dot{\mathbf{x}}_2'(\dot{x}) = \dot{\mathbf{x}}_2(\dot{x})$, and $\mathbf{x}_1(t_x) = \mathbf{x}_1'(t_x)$.

Finally, we show the conditions on initial and final states hold. Let $s_2 = (v, \mathbf{x}_2, \dot{\mathbf{x}}_2)$ be an initial state of A. Then since control mode v is definite for x, $init_A(v)$ implies $x = c$ for some $c \in \mathbb{R}$. Let $s_1 = ((v,c), \mathbf{x}_1, \dot{\mathbf{x}}_1)$, where $\mathbf{x}_1$ is the same as $\mathbf{x}_2$ over all variables $y \neq t_x$, and $\mathbf{x}_1(t_x) = 0$, and $\dot{\mathbf{x}}_1$ is the same as $\dot{\mathbf{x}}_2$ over all variables $\dot{y} \neq \dot{t}_x$, and $\dot{\mathbf{x}}_1(\dot{t}_x) = 1$. Then $s_1 \equiv_x s_2$ by definition of $\equiv_x$. The clock translation for initial conditions $init(v)$ and Lemma 3.1 imply s_1 is an initial state of A_x as required.

Let s_2 be a final state of A, and suppose that $s_1 \equiv_x s_2$. By Lemma 3.1 and the transformation of final conditions in the clock translation, s_1 is a final state of A_x. ■

Lemma 3.3 *Let A be a hybrid automaton with solvable nonlinear variable x, and let A_x be the clock translation of A that results from replacing the variable x by the clock t_x. Then $\equiv_x^{-1}$ is a simulation of $[\![A_x]\!]$ by $[\![A]\!]$.*

Proof. Let $s_1 = ((v,c), \mathbf{x}_1, \dot{\mathbf{x}}_1)$ and $s_1' = ((v', c'), \mathbf{x}_1', \dot{\mathbf{x}}_1')$ be two states in S_{A_x}. Let $\alpha_x(s_1) = s_2 = (v, \mathbf{x}_2, \dot{\mathbf{x}}_2)$ and $\alpha_x(s_1') = s_2' = (v', \mathbf{x}_2', \dot{\mathbf{x}}_2')$. We need to show that $s_1 \overset{m}{\rightarrow} s_1'$ implies $s_2 \overset{m}{\rightarrow} s_2'$. We do so by showing that $s_1 \xrightarrow{m} s_1'$ implies $s_2 \xrightarrow{m} s_2'$.

We first consider flow transitions. Suppose $s_1 \xrightarrow{\delta} s_1'$ for some $\delta \geq 0$ and witness trajectory $\rho_1 : [0, \delta] \to \mathbb{R}^n$. We need to show that $s_2 \xrightarrow{\delta} s_2'$. By definition of $\xrightarrow{\delta}$, we have that $v = v'$ and $c = c'$. Let the function $g_{v,c}(t)$ be the solution of the initial-value problem "$\dot{y}(t) = f_{v,x}(y(t)); y(0) = c$". Suppose $\mathbf{x}_1(t_x) = t_a$. Then $\mathbf{x}_1'(t_x) = t_a + \delta$. Let ρ_2 be the trajectory such that for all $t \in [0, \delta]$, $\rho_2(t)(y) = \rho_1(t)(y)$ for all variables $y \neq x$, and $\rho_2(t)(x) = g_{v,c}(t + t_a)$. We claim that the trajectory ρ_2 is a witness for $s_2 \xrightarrow{\delta} s_2'$. By definition of α_x, it follows that $\rho_2(0) = \mathbf{x}_2$ and $\rho_2(\delta) = \mathbf{x}_2'$. By Lemma 3.1 and the clock translation for the invariant, for all $t \in [0, \delta]$, $\rho_2(t) \in [\![inv(v)]\!]$ since $\rho_1(t) \in [\![inv(v,c)]\!]$. Finally, since the function $g_{v,c}(t)$ is the solution of the initial-value problem

"$\dot{y}(t) = f_{v,x}(y(t)); y(0) = c$", it follows that $(\rho_2(t), \dot{\rho}_2(t)) \in [\![flow(v,c)]\!]$ since x is flow independent with $flow(v) = flow_{v,X\setminus\{x\}} \wedge x' = f_{v,x}(x)$, and for all variables $y \neq x$, $\rho_2(y) = \rho_1(y)$.

We now consider jump transitions. Suppose $s_1 \xrightarrow{m} s_1'$ where $m = event(e_1)$ for the control switch $e_1 = ((v,c),(v',c'))$. By definition of the clock translation, there is a control switch $e_2 = (v,v')$ of A such that $event(v,v') = m$. Since x is a simple variable, we need only consider the effect of the clock translation on atomic predicates over x since other variables are unaffected by the clock translation and the function α_x. By Lemma 3.1, $\mathbf{x}_2$ satisfies all atomic predicates of the form $x \sim k$ in $jump(\epsilon_2)$. In order to show $s_2 \xrightarrow{m} s_2'$, we now consider atomic predicates involving x' for the following two cases:

1. $jump(\epsilon_2)$ implies $x' = d$, for some $d \in \mathbb{R}$. In this case, according to the clock translation, $jump(e_1)$ implies $t_x' = 0$. Since $t_x = 0$ in $\mathbf{x}_1'$, $x = d$ as required in $\mathbf{x}_2'$.

2. $jump(\epsilon_2)$ implies $x' = x$. In this case, according to the clock translation, $c' = c$ and $jump(\epsilon_1)$ implies $t_x' = t_x$, and so t_x has the same value in $\mathbf{x}_1$ and $\mathbf{x}_1'$. Hence x has the same value in $\mathbf{x}_2$ and $\mathbf{x}_2'$ as required.

Finally, we consider the conditions on initial and final states. Let $s_1 = ((v,c), \mathbf{x}_1, \dot{\mathbf{x}}_1)$ be an initial state of A_x. Then $\mathbf{x}_1(t_x) = 0$. Let $s_2 = (v, \mathbf{x}_2, \dot{\mathbf{x}}_2)$, where $\mathbf{x}_2$ is the same as $\mathbf{x}_1$ over all variables $y \neq x$, and $\mathbf{x}_2(x) = c$, and $\dot{\mathbf{x}}_2$ is the same as $\dot{\mathbf{x}}_1$ over all variables $\dot{y} \neq \dot{x}$, and $\mathbf{x}_2(\dot{x}) = f_{v,x}(c)$. Then $s_1 \equiv_x s_2$ by definition of $\equiv_x$. According to the clock translation, the initial condition $init_A(v)$ implies $x = c$. Then by Lemma 3.1, s_2 is an initial state of A as required. The argument for the condition on final states is analogous to that in the proof of Lemma 3.2. ■

Theorem 3.4 *If A is a solvable hybrid automaton and B is the clock translation of A, then A and B are bisimilar.*

Proof. By Lemmas 3.2 and 3.3, the relation $\equiv_x$ is bisimulation between $[\![A]\!]$ and $[\![A_x]\!]$, where A_x is the automaton that results from replacing the nonlinear variable x with the clock t_x. Since bisimilarity is transitive, if B results from A by replacing several nonlinear variables with clocks, A and B are bisimilar. ■

It follows from Theorem 3.4 that the clock translation is sound and complete. We conclude that for solving the emptiness problem for the solvable nonlinear automaton A, it suffices to solve the emptiness problem for the linear automaton A_x. A solvable hybrid automaton A is *rationally solvable* if its clock translation A_x is rational.

Corollary 3.5 *The emptiness problem is co-recursively enumerable for the class of rationally solvable hybrid automata.* ■

A variable x in a hybrid automaton A is a *rectangular* if it is simple, and for each control mode v, it is flow independent with flow function $f_{v,x}$ of the form $\dot{x} \in [l,u]$ for some $l, u \in \mathbb{R}$. A hybrid automaton is *rectangular* if all its variables are rectangular. The emptiness problem for rectangular automata is decidable. Therefore, when the clock translation results in a rectangular automaton, emptiness is decidable. We thus obtain the first decidability result for a class of nonlinear hybrid automata, whereas all previously published results refer to linear hybrid automata [KPSY93, AD94, HKPV95] only. However, when the clock translation A_x is not rational, the emptiness problem cannot be solved exactly.

Corollary 3.6 *The emptiness problem is decidable for the class of rationally solvable hybrid automata for which all linear variables are rectangular.* ■

4 Linear Phase-Portrait Approximation

A hybrid automaton may be simulated (and therefore approximated) by another automaton by relaxing the initial conditions, the final conditions, the flow conditions, or the jump conditions.

4.1 Phase-portrait approximation

Two hybrid automata are *compatible* if they have the same set of variables, control modes and control switches. Let A and B be compatible hybrid automata. We say A is a *basic phase-portrait approximation* of B if A and B are compatible, and the following four conditions hold:

- for all control modes v, $\mathit{flow}(B)(v)$ implies $\mathit{flow}(A)(v)$, and
- for all control modes v, $\mathit{init}(B)(v)$ implies $\mathit{init}(A)(v)$,
- for all control modes v, $\mathit{final}(B)(v)$ implies $\mathit{final}(A)(v)$, and
- for all control switches e, $\mathit{jump}(B)(e)$ implies $\mathit{jump}(A)(e)$.

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

Observation *If A is a phase-portrait approximation of B, then $A \succeq_{sim} B$.*

Proof. For any flow split $\mathcal{P}$ for B, the identity relation is a simulation of $\mathcal{P}(B)$ by A. ■

An automaton A is a *linear phase-portrait approximation* of B if it is a linear hybrid automaton and it is a phase-portrait approximation of B. Linear phase-portrait approximations can often be obtained by first splitting the control modes using a flow split, and then approximating, for each control mode, the flow field using a convex linear predicate containing the convex hull of the set of flow vectors occurring in its flow field.

As a special case of linear phase-portrait approximation, notice that linear hybrid automata need to be approximated by *rational* linear hybrid automata before being input into the model-checking tool HyTech. We advocate taking rational phase-portrait approximations of linear hybrid automata by rounding up or down irrational constants.

Example 4.1 The constraint $t_x = \ln 2$ in the clock translation of the thermostat automaton in Figure 2 can be replaced by the rational predicate $69/100 \leq t_x \leq 70/100$, since $\ln 2$ is approximately equal to 0.693. Similarly, we can approximate $t_x = \ln 3$ with $109/100 \leq t_x \leq 110/100$. Suppose we are interested in showing that in its first 60 seconds of operation, the heater is not on for more than 60% of the time. The final condition for each location is then $z = 60 \wedge y \geq 3/5z$. HyTech verifies the property for the rational linear phase-portrait approximation of the clock translation described above. HyTech also determines that after exactly 60 time units, the thermostat has been in control mode *on* between $(753/15)\% \approx 50.2\%$ and $(760/15)\% = 50.7\%$ of the time. ■

Example 4.2 To demonstrate the linear phase-portrait approximation technique, we suppose that we directly approximate the thermostat automaton of Figure 1 without first performing a clock translation. Figure 3 depicts a linear phase-portrait approximation for the flow split $\mathcal{P}_1$ dereived from the cover with predicates $1 \leq x \leq 2$ and $2 \leq x \leq 3$ for control mode *on*, and the predicates $1 \leq x \leq 2$, $2 \leq x \leq 3$, and $x \geq 3$ for control mode *off*. However, HyTech does not verify

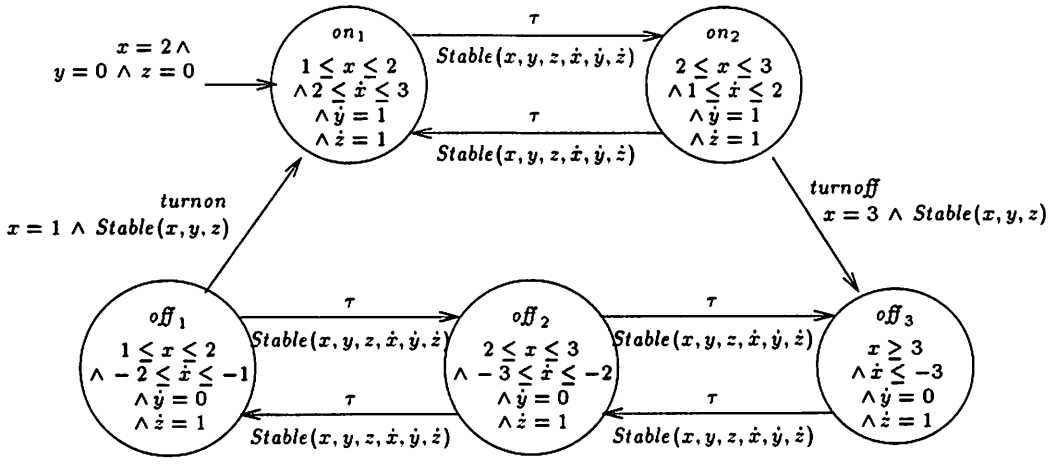

Figure 3: Linear phase-portrait approximation of the thermostat automaton

the property that the heater is on for less than 60% of the time during its first 60 time units of operation. This approximation is much coarser than in the previous example, with the time the heater is on ranging from $\approx 35.6\%$ to $\approx 64.2\%$. The approximation can be tightened using a finer flow split. For example, consider the flow split $\mathcal{P}_2$ that additionally splits the control mode on_2 in Figure 3 according to the condition $x \le 2.5$ and splits the control mode off_1 according to the condition $x \le 1.5$, *i.e.* the flow split derived from the predicates $1 \le x \le 2$, $2 \le x \le 2.5$ and $2.5 \le x \le 3$ for control mode *on*, and the predicates $1 \le x \le 1.5$, $1.5 \le x \le 2$, $2 \le x \le 3$, and $x \ge 3$ for control mode *off*. Using HyTech to analyze the automaton with the flow split $\mathcal{P}_2$ shows that the heater is on between 40.7% and 59.2% of the time, and the property we are interested in is now verified.

The finer the flow split, the tighter the approximation, but the greater the computational cost. Using flow split $\mathcal{P}_2$, computation time is also twice as long (10.6s versus 5.5s on a Sun Sparcstation 5) as for $\mathcal{P}_1$. By contrast, HyTech required only 2.5s to generate the much better bounds for the clock-translated automaton in Example 4.1. This example demonstrates the benefits of using the clock translation where possible. ■

4.2 Example: predator-prey systems

We demonstrate the use of phase-portrait approximations on nonlinear systems modeling the population growth of two interacting species [Lot20, HS74]. We show that several interesting properties of the system can be discovered automatically through algorithmic analysis.

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

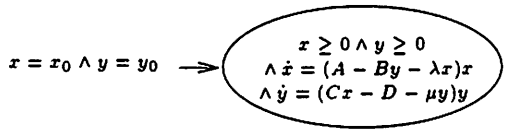

Figure 4: Predator-prey hybrid automaton

between its actual per capita food supply and a basic per capita food supply required to maintain its population. The population of the prey is subject to two competing forces. First, the population may grow because there is a constant food supply available: the prey's population would increase without bound in the absence of predators. Furthermore, we assume this rate of increase would be 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. Thus the rate of increase for the variable x is given by the equation:

$$\dot{x} = Ax - Bxy$$

The flow conditions describing the entire system may be rewritten as:

$$\dot{x} = (A - By)x \quad (1)$$
$$\dot{y} = (Cx - D)y \quad (2)$$

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. Assuming these negative growth factors are proportional to the species population and its difference from the threshold population leads to the Volterra-Lotka predator-prey equations [Lot20, HS74]

$$\dot{x} = (A - By - \lambda x)x \quad (3)$$
$$\dot{y} = (Cx - D - \mu y)y \quad (4)$$

where A, B, C, D, λ, and μ are all positive real-valued constants. The automaton for the system appears in Figure 4. By examining the flow field determined by the flow condition above, we partition the state space with lines where either $\dot{x}$ or $\dot{y}$ have value 0, *i.e.* along the coordinate axes, and along the lines $L : A - By - \lambda x = 0$ and $M : Cx - D - \mu y = 0$. If the lines L and M do not intersect in the upper right quadrant, then the phase portrait of the system looks like that of Figure 5.

Linear phase-portrait approximation

In the region R, to the right of the line M in Figure 5, we infer tighter constraints on $\dot{x}$ and $\dot{y}$ than their signs. The values taken by the function $\xi(x, y) = \dot{y}/\dot{x}$ in the region R determine the flow vectors in R, since $\dot{x}$ is nonpositive and $\dot{y}$ nonnegative. The absolute value of $\xi(x, y)$ is bounded above by any 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, since $D + \mu y$ is always positive. Since the lines L and M do not intersect in the positive quadrant of $\mathbb{R}^2$, we know that $A/\lambda < D/C$, and hence that $A - \lambda D/C < 0$. Because x is no less than D/C in R, we infer that $A - \lambda x < 0$, and hence that $A - \lambda x - By < -By$. We may therefore take Byx for Min. We conclude that $\xi(x, y)$

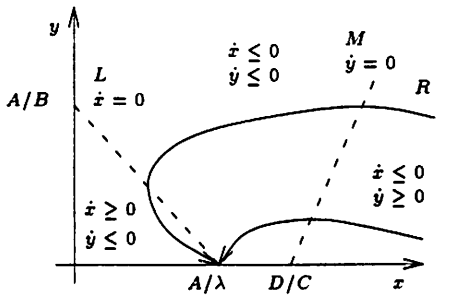

Figure 5: Phase portrait for pred-prey populations: L, M non-intersecting

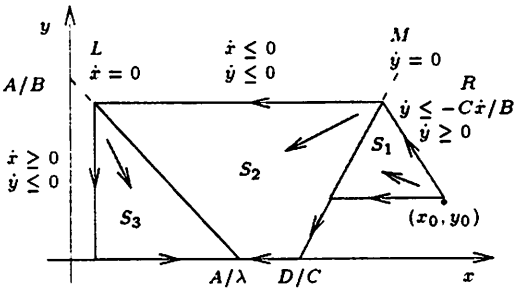

Figure 6: Reachability using linear phase-portrait approximation

is bounded below by $-Cxy/(Bxy) = -C/B$, and thus that all flow vectors in R have direction between $(-B, C)$ and $(-1, 0)$, *i.e.* they all satisfy the flow condition $\dot{y} \geq 0 \wedge \dot{y} \leq -C\dot{x}/B$.

The automaton for a linear phase-portrait approximation appears in Figure 7, where the predicate *Stable*ϵ is shorthand for *Stable*$\epsilon(x, y, \dot{x}, \dot{y})$. The layout of the control modes matches the partitioning of the state space as shown in Figure 5. The implicit invariant constraint $x \geq 0 \wedge y \geq 0$ has been omitted from all invariants. The constraint M refers to all valuations on line M, *i.e.* all valuations where $Cx - D - \mu y = 0$. The constraint $M^{\geq}$ refers to all valuations at, or to the right of, the line M, *i.e.* $M^{\geq}$ is $Cx \geq D + \mu y$. Similarly $M^{\leq}$ is $Cx \leq D + \mu y$, L is $A - By - \lambda x = 0$, $L^{\leq}$ is $\lambda x \leq A - By$, and $L^{\geq}$ is $\lambda x \geq A - By$.

Computing bounds on the population growth

The phase-portrait approximation above 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 the initial populations, x_0 and y_0 say, lie in the region R of the state space. The set of time-step successors of the state (x_0, y_0) is obtained by following all flow vectors in the cone indicated in Figure 6. 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 S_3, reachability analysis terminates. The maximum value of y among the reachable states is $(By_0 + Cx_0 - D)/(B + \mu)$. For

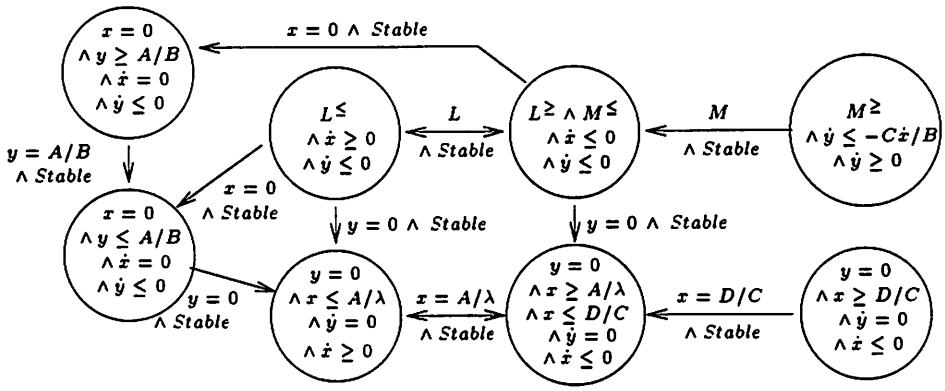

Figure 7: Phase-portrait approximation for predator-prey system

example, using the equations $\dot{x} = (2000 - y - 5x)x$ and $\dot{y} = (4x - 2600 - 4y)y$, and initial population vector $(900, 150)$, we can use HyTech to obtain a bound of 230 on the predator population y.

It can be shown that when the lines L and M intersect in the upper right quadrant, reachability analysis on a phase-portrait approximation demonstrates that the populations are bounded in this case too.

Iterated approximations

In general, bounds on the reachable region can be used to generate better phase-portrait approximations. Let φ be a predicate such that $[\![\varphi]\!]$ is a superset of the reachable states of a hybrid automaton A. The *restriction* of A to φ is the automaton $A|_\varphi$ that differs from A only in its flow conditions, where for all control modes v, $flow_{A|_\varphi}(v) = flow_A(v) \wedge \varphi$. The automaton $A|_\varphi$ is an approximation of A. It may be possible to find tighter linear phase-portrait approximations for $A|_\varphi$ than for A, since the flow condition has been restricted, and need not contain as many flow tangents as in A.

In our predator-prey example, it can be shown that in the rightmost region R of Figure 6, the absolute value of $\xi(x, y)$, the flow tangent at (x, y), is bounded above by $Cy/(\lambda x + By - A)$. Let Z be a bounded subset of R. Let y_{max} be an upper bound for y over all valuations in Z, and x_{min} (resp. y_{min}) be a lower bound for x (resp. y) over Z. It follows $|f(x, y)| \leq Cy_{max}/(\lambda x_{min} + By_{min} - A)$, provided $(\lambda x_{min} + By_{min} - A) \geq 0$. Previously, we showed how reachability using the automaton in Figure 7 leads to the region S_1 in R, from which we infer bounds of $y_{max} = 230$, $y_{min} = 150$, and $x_{min} = 800$. We can therefore replace the flow condition $\dot{y} \leq -C\dot{x}/B$ in region S_1 with $\dot{y} \leq -92\dot{x}/215$. Recomputing now shows that a proper subset of region S_1 is actually reachable. In particular, we obtain a tighter bound of $y_{max} = 55250/307 \approx 180$. We could iterate this procedure, gaining successively lower values of y_{max}, and more restrictive flow conditions.

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 [HS74]. 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 is unwise to increase the prey population to its maximal allowable value, since 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 the tool 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 lead to excessively large predator populations, since the approximation yields more reachable states than the true system.

4.3 Error analysis

Linear phase-portrait approximations are not complete. However, under certain conditions, a hybrid automaton may be approximated arbitrarily closely by choosing a sufficiently fine splitting of its flow conditions.

Approximation operators

Let $\mathcal{H}$ be the class of hybrid automata. An *approximation operator* is a map $\gamma : \mathbb{R}_{\geq 0} \times \mathcal{H} \to 2^{\mathcal{H}}$. The approximation operator γ is *sound* if for all hybrid automata A. and for all $\delta \geq 0$, $B \in \gamma(\delta, A)$ implies B is an approximation of A. In this case, proving emptiness of B also proves emptiness of A. We define a notion of asymptotic completeness. We define the infinity metric $dist : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ by $dist(\mathbf{x}, \mathbf{y}) = max_{1 \leq i \leq n}\{|\mathbf{x}_i - \mathbf{y}_i|\}$. The distance $dist(\mathbf{x}, \mathbf{y})$ between points $\mathbf{x}$ and $\mathbf{y}$ is the maximal componentwise separation. Let ϕ be a predicate over $\mathbb{R}^n$. The *ε-relaxation* of ϕ, for $\varepsilon \geq 0$, is the predicate ϕ^ε such that for all $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} \in [\![\phi^\varepsilon]\!]$ iff there exists $\mathbf{y} \in \mathbb{R}^n$ such that $\mathbf{y} \in [\![\phi]\!]$ and $dist(\mathbf{x}, \mathbf{y}) \leq \varepsilon$. Given a hybrid automaton A, let the *ε-relaxed* automaton A^ε, for $\varepsilon \geq 0$, be the automaton obtained from A by replacing all flow conditions, initial and accepting conditions, and jump conditions with their ε-relaxations. If the hybrid automaton A models a system with sensors and actuators, then A^ε models the same system where measurement errors of the sensors and actuators are bounded by ε, and the flow and jump transitions are subject to a modeling error of ε. The approximation operator is *asymptotically complete* if for all hybrid automata $A \in \mathcal{H}$, and for all $\varepsilon > 0$, there exists a $\delta > 0$ such that for all $B \in \gamma(\delta, A)$ $A^\varepsilon \succeq B$. In particular, asymptotic completeness implies that for a hybrid automaton A, if automata very close to A are empty, then the approximating automata for a sufficiently small δ are also empty.

Asymptotic completeness

Asymptotic completeness holds even when linear phase-portrait approximations are of a very restricted form, namely when the flow conditions provide independent lower and upper bounds on the rates of each variable [HH95a]. A predicate ϕ is *rectangular* if $[\![\phi]\!]$ is rectangular, *i.e.* if $[\![\phi]\!]$ is of the form $\prod_{i=1..n} I_i$ where each I_i is an interval over $\mathbb{R}$. A linear phase-portrait approximation is a *rectangular phase-portrait approximation* if every initial condition, final condition, jump condition, and flow condition is rectangular. The *rectangular phase-portrait translation* $rect(A)$ of A is the minimal rectangular phase-portrait approximation of A with respect to the order $\succeq$. The minimal approximation exists since the minimal rectangular predicate implied by a predicate is well-defined: take the projection onto the x axis to obtain the interval for x.

Example 4.3 The linear phase-portrait approximation in Figure 3 is the rectangular phase-portrait approximation for the given flow split. ■

A predicate φ is of *width* δ if, for all valuations $x_1, x_2 \in [\![\phi]\!]$, $dist(x_1, x_2) \leq \delta$. A flow split $\mathcal{P}$ is of *width* δ if each predicate ϕ occurring in $\mathcal{P}$ is of width δ.

Proposition 4.1 *The approximation operator* γ *defined by* $\gamma(\delta, A) = \{rect(\mathcal{P}(A)) \mid \mathcal{P}$ *is of width* $\delta\}$ *is asymptotically complete.*

Proof. For a given ε, choose $\delta \in \mathbb{R}_{>0}$ such that $\delta < \varepsilon$. Let $\mathcal{P}$ be a flow split of width δ. We show that the predicates of $rect(\mathcal{P}(A))$ imply the corresponding predicates of A^ε, and hence $A^\varepsilon \succeq_{sim} rect(\mathcal{P}(A))$.

For example, let s be an initial state of $rect(\mathcal{P}(A))$ with control mode (v, φ). Since the value of each variable x in s is the projection of some valuation satisfying φ, and φ is of width δ, there exists a state s' within distance δ of s that satisfies φ. Thus the state s satisfies φ^ε, and hence $\pi_1(s)$ is an initial state of A^ε since φ implies $init_A(v)$ and thus φ^ε implies $init_A(v)^\varepsilon$, which is $init_{A^\varepsilon}(v)$. Similarly, for all control modes (v, φ) of $rect(\mathcal{P}(A))$, the final condition $final(v, \varphi)$ implies $final_{A^\varepsilon}(v)$, and $flow(v, \varphi)$ implies $flow_{A^\varepsilon}(v)$. Analogously, for all control switches $e \in E$ directly inherited from A, the jump condition $jump_{rect(\mathcal{P}(A))}(e)$ implies $jump_{A^\varepsilon}(e)$. Silent control switches between copies of a control mode need not be directly simulated. ■

It is easy to see that asymptotic completeness also holds for the approximation operator that yields linear phase-portrait approximations that are simulated by the rectangular phase-portrait approximations. In practice, rectangular approximations are often easier to compute (since we need only bound the rate of the derivative for each variable). Linear phase-portrait approximations, however, are sometimes more accurate, as seen in the predator-prey example.

4.4 Example: pseudo-linear automata

It is not obvious how to compute useful linear phase-portrait approximations of an arbitrary hybrid automaton, since it may involve partially solving a set of differential inequalities. However, we describe a restricted class of hybrid automata for which approximations may be found automatically. We assume every flow condition is given as a finite conjunction of inequalities. A hybrid automaton is *pseudo-linear* if (1) for every flow condition, substituting every occurrence of a variable $x \in X$ with an arbitrary real-valued constant results in a convex linear predicate over $\dot{X}$, (2) for every control mode v, every operator in the flow condition $flow(v)$ is monotonic in every argument over the domain $inv(v)$, and (3) all initial, final, and jump conditions are linear. Many automata can be made to satisfy Condition 2 through flow splitting.

Given a pseudo-linear hybrid automaton A, the following transformation yields a linear phase-portrait approximation $\lambda(A)$. Every instance of a variable x_i in a flow condition is replaced with either x_i^{min} or x_i^{max} depending on its *force* in the inequality. The variable x_i is replaced with a constant value that guarantees the transformed inequality is satisfied whenever the original is. The value x_i^{min} is a lower bound on the values x_i may take along any curve satisfying the flow condition and the control mode's invariant. Similarly, x_i^{max} is an upper bound on x_i. For simplicity, we consider only bounded automata[3], where bounds may be inferred from the invariants. Without loss of generality, we assume every inequality in every flow condition asserts its left-hand side is less than its right-hand side. An operator is *positive*, with respect to an invariant, in its i-th argument if it is monotonically nondecreasing in its i-th argument, over the domain of the invariant. Otherwise, it is *negative*. Both of the following conditions affect the force of an instance of the variable x_i in a flow condition for a given control mode.

[3]The generalization to unbounded automata is straightforward.

1. the instance occurs on the right-hand side of the inequality.

2. the force of the instance of the variable occurs within the scope of an even number of negative operators.

If an even number of the above conditions is true, then the instance of x_i in A has positive force, and is replaced with x_i^{max} in $\lambda(A)$. Otherwise, it is replaced with x_i^{min}. For example, x and y occur positively, whereas z occurs negatively, in the inequality $\dot{y}e^x - y^3 \leq \dot{x} + 1/z^2$.

Theorem 4.2 *Given a pseudo-linear hybrid automaton A, the automaton $\lambda(A)$ is a linear phase-portrait approximation of A.* ■

Example 4.4 The thermostat automaton of Figure 1 is pseudo-linear. In control mode *on*, the flow condition $\dot{x} = -x + 4 \wedge \dot{y} = 1 \wedge \dot{z} = 1$ is replaced by $\dot{x} \geq 1 \wedge \dot{y} = 1 \wedge \dot{z} = 1$, since $-x+4 \leq \dot{x} \leq -x+4$ is transformed to $-3+4 \leq \dot{x} \leq -\infty+4$. The flow condition for control mode *off* is transformed to $\dot{x} \leq -1 \wedge \dot{y} = 0 \wedge \dot{z} = 1$. ■

Example 4.5 The predator-prey systems of the previous section are also pseudo-linear. However, the phase-portrait approximation we obtain using the algorithmic technique above gives no additional constraints than the signs of the derivatives of x and y in the control mode with invariant $M^{\geq}$. This demonstrates the limitation of automatic methods: often a more careful, handwritten analysis can provide tighter approximations. To gain the maximal benefits of computer-aided analysis, one often needs to draw on an understanding of nonlinear equations governing continuous behavior. ■

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.

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, Lecture Notes in Computer Science 736, pages 209–229. Springer-Verlag, 1993.

[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.

[Arn83] V. I. Arnol'd. *Geometric Methods in the Theory of Ordinary Differential Equations.* Springer-Verlag, New York, 1983.

[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.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata. Languages, and Programming*, Lecture Notes in Computer Science 944, pages 324–335. Springer-Verlag, 1995.

[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.

[HHWT95a] 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.

[HHWT95b] 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.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[Ho95] P.-H. Ho. *Automatic Analysis of Hybrid Systems.* PhD thesis, Department of Computer Science, Cornell University, 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.

[HWT96] T. A. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 377–388. Springer-Verlag, 1996.

[KPSY93] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems*, Lecture Notes in Computer Science 736, pages 179–208. Springer-Verlag, 1993.

[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.

[PBV96] A. Puri, V. Borkar, and P. Varaiya. ϵ-approximation of differential inclusions. In *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.