Hybrid Automata:
An Algorithmic Approach to the Specification and Verification of Hybrid Systems[1]

Rajeev Alur[2] Costas Courcoubetis[3] Thomas A. Henzinger[4] Pei-Hsin Ho[3]

Abstract

We introduce the framework of *hybrid automata* as a model and specification language for hybrid systems. Hybrid automata can be viewed as a generalization of timed automata, in which the behavior of variables is governed in each state by a set of differential equations. We show that many of the examples considered in the workshop can be defined by hybrid automata. While the reachability problem is undecidable even for very restricted classes of hybrid automata, we present two semidecision procedures for verifying safety properties of *piecewise-linear* hybrid automata, in which all variables change at constant rates. The two procedures are based, respectively, on minimizing and computing fixpoints on generally infinite state spaces. We show that if the procedures terminate, then they give correct answers. We then demonstrate that for many of the typical workshop examples, the procedures do terminate and thus provide an automatic way for verifying their properties.

1 Introduction

More and more real-life processes, from elevators to aircraft, are controlled by programs. These *reactive* programs are embedded in continuously changing environments and must react to environment changes in real time. Obviously, correctness is of vital importance for reactive programs. Yet traditional program verification methods allow us, at best, to approximate continuously changing environments by discrete sampling. A generalized formal model for computing systems is needed to faithfully represent both discrete and continuous processes within a unified framework. Hybrid automata present such a framework.

A *hybrid* system consists of a discrete program within an analog environment. Hybrid automata are generalized finite-state machines for modeling hybrid systems. As usual, the discrete transitions of a program are modeled by a change of the program counter, which ranges over a finite set of control locations. In addition, we allow for the possibility that the global state of a system changes continuously with time according to the laws of physics. For each control location, the continuous activities of the environment are governed by a set of differential equations. We label each location

[1]An abbreviated version of this paper appeared in *Hybrid Systems* (R.L. Grossman, A. Nerode, A.P. Ravn, H. Rischel, eds.), *Lecture Notes in Computer Science* **736**, Springer-Verlag, 1993, pp. 209–229.

[2]AT&T Bell Laboratories, Murray Hill, NJ 07974.

[3]Institute of Computer Science, FORTH, and Department of Computer Science, University of Crete, Heraklion, Greece. Supported in part by the BRA ESPRIT project REACT.

[4]Department of Computer Science, Cornell University, Ithaca, NY 14853. Supported in part by the National Science Foundation under grant CCR-9200794 and by the United States Air Force Office of Scientific Research under contract F49620-93-1-0056.

also with an invariant condition that must hold while the control resides at the location, and each transition is labeled with a guarded set of assignments. This model for hybrid systems is inspired by the phase transition systems of [MMP92] and [NSY92], and can be viewed as a generalization of timed automata [AD90].

The current paper pursues three objectives. First, hybrid automata are defined and their suitability for specification is demonstrated through some paradigmatic examples. Second, the verification problem for hybrid automata is studied and shown to be intrinsically difficult even under severe restrictions. Third, and most importantly, we successfully verify interesting properties of truly hybrid system behaviors. We note that Nicollin et al. have independently developed an approach similar to ours [NSOY].

For verification purposes, we restrict ourselves to *linear* hybrid automata. In a linear hybrid automaton, for each variable the rate of change with time is constant — though this constant may vary from location to location — and the terms involved in the invariants, guards, and assignments are required to be linear. An interesting special case of a linear hybrid automaton is a *timed automaton* [AD90]. In a timed automaton each continuously changing variable is an accurate clock whose rate of change with time is always 1. Furthermore, in a timed automaton all terms involved in assignments are constants, and all invariants and guards only involve comparisons of clock values with constants. Even though the reachability problem for linear hybrid automata is undecidable, it is PSPACE-complete for timed automata. In this paper, we show that some of the algorithms for the analysis of timed automata can be extended to obtain semidecision procedures for solving the verification problem for linear hybrid automata. In particular, we consider the fixpoint computation method presented in [HNSY92] and the minimization procedure for timed automata presented in [ACH$^+$92]. Both methods perform a reachability analysis over the infinite state space of a timed automaton by computing with sets of states. We show that the primitive steps of the two algorithms can be performed relatively easily even in case of linear hybrid automata and, thus, both methods can be generalized. The crucial observation is that each set of states computed by the algorithms is definable by a linear formula; that is, it is a union of convex polyhedra. However, as we move from timed automata to linear hybrid automata, the termination of the two procedures is no longer guaranteed.

Both methods we consider can be used to prove invariant properties of linear hybrid systems. We illustrate these methods on three examples, and in each case the procedures terminate. The first example involves a water level monitor. It is a truly hybrid system, since the water level increases and decreases continuously in phases. We show how to prove that the water level always remains within the specified bounds. The second example proves the mutual exclusion property of a real-time mutual exclusion protocol. Earlier algorithmic methods based on timed automata fail when the bounds on the various delays are not known. We show how to perform a symbolic analysis so as to deduce constraints between the various bounds. Our third example involves leakage in a gas burner. This is an example of a so-called *integrator system* in which we are required to prove a bound on the ratio of two durations.

2 Modeling Hybrid Systems

We define a formal model and a specification language for hybrid systems.

2.1 Hybrid traces

An *interval* is a nonempty convex subset of the nonnegative real line R^+. Intervals may be open, halfopen, or closed; bounded or unbounded. The left end-point of an interval I is denoted by l_I and the right end-point, for bounded I, is denoted by r_I. Two intervals I_1 and I_2 are *adjacent* if (1) $r_{I_1} = l_{I_2}$, and (2) either I_1 is right-open and I_2 is left-closed, or I_1 is right-closed and I_2 is left-open. An *interval sequence* $I_0 I_1 I_2 \ldots$ is a finite or infinite sequence of intervals that partitions R^+:

1. Any two neighboring intervals I_i and I_{i+1} are adjacent.

2. For all $t \in \mathsf{R}^+$, there is some interval I_i with $t \in I_i$.

In particular, I_0 is left-closed and $l_{I_0} = 0$. The last interval of any finite interval sequence is unbounded. The interval sequence $\overline{I}_1$ *refines* the interval sequence $\overline{I}_2$ if $\overline{I}_1$ is obtained from $\overline{I}_2$ by splitting some intervals. We henceforth identify an interval sequence $\overline{I}$ with its refinement closure $\{\overline{J} \mid \overline{J} \text{ refines } \overline{I}\}$. Clearly, for any finite set $\mathcal{I}$ of interval sequences there is an interval sequence $\bigcap \mathcal{I}$ that refines all sequences in $\mathcal{I}$.

Let V be a finite set of real-valued variables. A *state* is an interpretation of all variables in V. We write Σ for the set of states. A *trace* is a function from R^+ to Σ. Equivalently, a trace τ is a collection of functions $\tau(x)$ from R^+ to R, one for each variable $x \in V$. We say that the trace τ has property P if all of its constituent functions $\tau(x)$, for $x \in V$, have property P. We will use the following properties of functions:

- A function $f \colon \mathsf{R}^+ \to \mathsf{R}$ is *piecewise smooth* if there exists an interval sequence $\overline{I}^f = I_0 I_1 I_2 \ldots$ and a sequence $f_0 f_1 f_2 \ldots$ of C^∞-functions such that the restriction of f to each interval I_i coincides with the restriction of f_i to I_i. Each restriction of f to an interval I_i is called a *phase* of f. The phases of a piecewise-smooth trace τ are the restrictions of τ to the intervals of the sequence $\overline{I}^\tau = \bigcap \{\overline{I}^{\tau(x)} \mid x \in V\}$.

- A piecewise-smooth function $f \colon \mathsf{R}^+ \to \mathsf{R}$ is *piecewise linear* if each phase of f is linear.

- A piecewise-linear function $f \colon \mathsf{R}^+ \to \mathsf{R}$ is a *step function* if the slope of all phases of f is 1.

- A piecewise-linear function $f \colon \mathsf{R}^+ \to \mathsf{R}$ is a *clock function* if the slope of all phases of f is 1.

- A piecewise-linear function $f : \mathsf{R}^+ \to \mathsf{R}$ is a *skewed-clock function* if there is some constant $k \in \mathsf{R}$ such that the slope of all phases of f is k.

- A piecewise-linear function $f : \mathsf{R}^+ \to \mathsf{R}$ is an *integrator function* if the slope of each phase of f is either 0 or 1.

A *timed trace* τ is a trace each of whose constituent functions $\tau(x)$, for $x \in V$, is either a step function or a clock function. A set $\mathcal{S}$ of traces is *fusion-closed* if for all traces $\tau_1, \tau_2 \in \mathcal{S}$ and all $t_1, t_2 \in \mathsf{R}^+$, if $\tau_1(t_1) = \tau_2(t_2)$, then $\tau \in \mathcal{S}$ for the trace τ with $\tau(t) = \tau_1(t)$ for all $t \leq t_1$ and $\tau(t) = \tau_2(t + t_2 - t_1)$ for all $t > t_1$.

2.2 Hybrid automata

We model a hybrid system as fusion-closed set $\mathcal{S}$ of piecewise-smooth traces. Each trace $\tau \in \mathcal{S}$ represents a possible behavior of the system over real time. The piecewise smoothness of τ ensures that in any bounded interval of time, there are only finitely many discontinuous state changes.

The fusion closure of $\mathcal{S}$ ensures that each state contains all information necessary to determine the future evolution of the system.

We define sets of traces by graphs whose edges represent discrete transitions and whose vertices represent continuous activities. A *hybrid system* $A = (V_D, Q, \mu_1, \mu_2, \mu_3)$ is given by six components:

- A finite set V_D of real-valued *data variables*. A *data state* is an interpretation of all variables in V_D. We write Σ_D for the set of data states.

- A finite set Q of vertices called *locations*. We use the variable $pc \notin V_D$ as a *control variable* that ranges over the set Q of locations (properly encoded in R), and let $V = \{pc\} \cup V_D$. Thus, $\Sigma = Q \times \Sigma_D$; that is, a (system) state is a pair (ℓ, σ) consisting of a location $\ell \in Q$ and a data state $\sigma \in \Sigma_D$.

- A labeling function μ_1 that assigns to each location in Q a set of possible *activities*. Each activity is a C^∞-function from R^+ to Σ_D.

- A labeling function μ_2 that assigns to each location $\ell \in Q$ an *exception* set $\mu_2(\ell) \subseteq \Sigma_D$. The system control must leave location ℓ before an exception $\mu_2(\ell)$ occurs. The complement $\Sigma_D - \mu_2(\ell)$ is called the *invariant* of the location ℓ.

- A labeling function μ_3 that assigns to each pair $e \in Q^2$ of locations a *transition relation* $\mu_3(e) \subseteq \Sigma_D^2$. We require that for all locations $\ell \in Q$ and all data states $\sigma \in \Sigma_D$, $(\sigma, \sigma) \in \mu_3(\ell, \ell)$. The state (σ', ℓ') is called a *successor* of the state (ℓ, σ) iff $(\sigma, \sigma') \in \mu_3(\ell, \ell')$.

At any time instant, the state of a hybrid system specifies a control location and values for all data variables. The state can change in two ways: (1) by an instantaneous transition that changes the entire state according to the successor relation, and (2) by elapse of time that changes only the values of data variables in a continuous manner according to the activities of the current location. The exceptions of a hybrid system enforce the progress of the underlying discrete transition system: some transition must be taken before an exception occurs. Typical exceptions are timeouts and sensor readings that trigger a discrete state change.

Formally, a *run* of the hybrid system A is a finite or infinite sequence

$$\rho: \quad \mapsto_{\sigma_0} \ (\ell_0, I_0, f_0) \ {}_{\sigma_0'} \!\rightarrow_{\sigma_1} \ (\ell_1, I_1, f_1) \ {}_{\sigma_1'} \!\rightarrow_{\sigma_2} \ (\ell_2, I_2, f_2) \ {}_{\sigma_2'} \!\rightarrow \ \cdots \qquad (\dagger)$$

of data states $\sigma_i, \sigma_i' \in \Sigma_D$, locations $\ell_i \in Q$, intervals I_i, and activities f_i such that

1. for all $i \geq 0$, the state $(\ell_{i+1}, \sigma_{i+1})$ is a successor of the state (ℓ_i, σ_i');

2. $I_0 I_1 I_2 \ldots$ is an interval sequence;

3. for all $i \geq 0$, the activity f_i is in $\mu_1(\ell_i)$, and (1) $f_i(0) = \sigma_i$ and $f_i(r_{I_i} - l_{I_i}) = \sigma_i'$, and (2) for all $t \in I_i$, $f_i(t - l_{I_i}) \notin \mu_2(\ell_i)$.

Each run ρ of A uniquely determines a trace τ_ρ: for all $i \geq 0$ and $t \in I_i$, let $\tau(t) = (\ell_i, f_i(t - l_{I_i}))$. Observe that, for all $i > 0$, if I_i is left-closed, then the state at time l_{I_i}, that is, at the time of transition from state σ_{i-1}' to state σ_i, is defined to be σ_i. On the other hand, if I_i is left-open, then the state at time l_{I_i} is defined to be σ_{i-1}'.

By $\mathcal{S}_A$ we denote the set of all traces τ_ρ that correspond to runs ρ of the system A. The set $\mathcal{S}_A$ is fusion-closed, because at any time instant during a run, the configuration of the system is completely determined by the location in which the control resides and the values of all data variables.

Linear hybrid systems

A *linear term* α over a set of variables V is a linear combination of the variables in V with rational coefficients. A *linear formula* ϕ over V is a boolean combination of inequalities between linear terms over V.

The hybrid system $A = (V_D, Q, \mu_1, \mu_2, \mu_3)$ is *linear* if its activities, exceptions, and transition relations can be defined by linear expressions over the set V_D of data variables:

1. For all locations $\ell \in Q$, the possible activities are linear functions defined by a set of differential equations of the form $x' = k_x$, one for each data variable $x \in V_D$, where k_x is a rational constant:

   $f \in \mu_1(\ell)$ iff all $t \in \mathsf{R}^+$ and $x \in V_D$, $f(t)(x) = f(0)(x) + k_x \cdot t$.

   We write $\mu_1(\ell, x) = k_x$ to define the activities of the linear hybrid system A.

2. For all locations $\ell \in Q$, the exception is defined by a linear formula ϕ over V_D: $\sigma \in \mu_2(\ell)$ iff $\sigma(\phi)$.

3. For all pairs $e \in Q^2$ of locations, the transition relation $\mu_3(e)$ is defined by a guarded set of assignments
   $$\phi \ \rightarrow \ \{x := \alpha_x \mid x \in V_D\},$$
   where ϕ is a linear formula over V_D and each α_x is either a linear term over V_D or "?":

   $(\sigma, \sigma') \in \mu_3(e)$ iff $\sigma(\phi)$ and for all $x \in V_D$, either $\alpha_x =?$ or $\sigma'(x) = \sigma(\alpha_x)$.

   An assignment of the form $x :=?$ indicates that the value of the variable x is changed nondeterministically to an arbitrary value. We write $\mu_3(e, x)$ for the term α_x.

Various special cases of linear hybrid systems are of particular interest:

- If $\mu_1(\ell, x) = 0$ for each location $\ell \in Q$, then x is a *discrete variable*. Thus a discrete variable changes only when the location of control changes. A *discrete* system is a linear hybrid system all of whose data variables are discrete variables.

- A discrete variable x is a *proposition* if $\mu_3(e, x) \in \{0, 1\}$ for all pairs $e \in Q^2$. If all the data variables are propositions, then a linear hybrid automaton is same as a finite-state system whose states are labeled with propositions.

- If $\mu_1(\ell, x) = 1$ for each location ℓ and $\mu_3(e, x) \in \{0, x\}$ for each pair $e \in Q^2$, then x is a *clock*. Thus the value of a clock variable increases with time uniformly; a transition of the automaton either resets it to 0, or leaves it unchanged. A (finite-state) *timed* system is a linear hybrid system all of whose data variables are propositions and clocks.

- If there is a constant $k \in \mathsf{R}$ such that $\mu_1(\ell, x) = k$ for each location ℓ and $\mu_3(e, x) \in \{0, x\}$ for each pair $e \in Q^2$, then x is a *skewed clock*. Thus a skewed clock is similar to a clock variable except that it changes with time at some (fixed) rate different from 1. A *multirate timed* system is a linear hybrid system all of whose data variables are propositions and skewed clocks. An *n-rate* timed system is a multirate timed system whose skewed clocks proceed at n different rates.

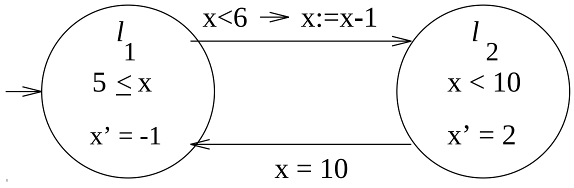

Figure 1: Graphical representation

- If $\mu_1(\ell, x) \in \{0,1\}$ for each location ℓ and $\mu_3(e, x) \in \{0, x\}$ for each pair $e \in Q^2$, then x is an *integrator*. Thus an integrator is like a clock that can be stopped and restarted, and can measure accumulated durations. An *integrator* system is a linear hybrid system all of whose data variables are propositions and integrators.
- A discrete variable is a *parameter* if $\mu_3(e, x) = x$ for all pairs $e \in Q^2$. Thus a parameter is a symbolic constant which can be used, for instance, in the guards of the transitions. For different special types of linear hybrid automata defined above, we can define its parameterized version also. For instance, a *parameterized timed* system is a linear hybrid system all of whose data variables are propositions, parameters, and clocks.

Clearly, if A is linear (discrete; timed; multirate timed; integrator) system, then all traces in $\mathcal{S}_A$ are piecewise linear (step traces; timed traces; skewed-clock traces; integrator traces, respectively).

Graphical representation

Instead of using exceptions, we label locations with their invariants. We suppress location labels of the form $x' = 0$ for activities and *true* for invariants. For transition labels, we suppress the guard *true* and assignments of the form $x := x$. Reflexive transitions with the label *true* are suppressed altogether.

As an example, consider the linear hybrid system of Figure 1 with the single data variable x. This system has two locations, ℓ_1 and ℓ_2. In location ℓ_1, the value of x decreases at a constant rate of 1. The transition from ℓ_1 to ℓ_2 may be taken at any time after the value of x has fallen below 6, and it must be taken before the value of x falls below 5. When the transition is taken, the value of x is instantaneously decreased by 1. Once in location ℓ_2, the rate of x starts to increase at the constant rate of 2. The transition back to location ℓ_1 is taken exactly when the value of x hits 10. Indeed, at the very time instant when $x = 10$, the control of the system is already in location ℓ_1, because location ℓ_2 has the invariant $x < 10$.

Initial and acceptance conditions

We can turn a hybrid system A into an automaton by adding initial and acceptance conditions. The initiality criterion is given by a labeling function μ_4 that assigns to each location $\ell \in Q$ an *initial condition* $\mu_4(\ell) \subseteq \Sigma_D$. The Muller acceptance criterion is given by a collection $F \subseteq 2^Q$ of *acceptance sets* of locations. The run ρ (†) of the *hybrid Muller automaton* (A, μ_4, F) is *accepting* if

4. $\sigma_0 \in \mu_4(\ell_0)$;
5. either ρ is finite with final location ℓ_n and $\{\ell_n\} \in F$, or $\rho_\infty \in F$ for the set ρ_∞ of locations that are visited infinitely often during ρ (i.e, ρ_∞ is the set $\{\ell \mid \ell = \ell_i$ for infinitely many $i \geq 0\}$).

The hybrid automaton (A, μ_4, F) is linear if A is a linear hybrid system and for all locations $\ell \in Q$, the initial condition is defined by a linear formula ϕ over V_D (i.e., $\sigma \in \mu_4(\ell)$ iff $\sigma(\phi)$).

Parallel composition

A hybrid system typically consists of many components operating concurrently and coordinating with each other. Such a system can be constructed from the descriptions of its components using a product operation. Let $A_1 = (V_D^1, Q^1, \mu_1^1, \mu_2^1, \mu_3^1)$ and $A_2 = (V_D^2, Q^2, \mu_1^2, \mu_2^2, \mu_3^2)$ be two hybrid systems. The *product* $A_1 \times A_2$ of A_1 and A_2 is the hybrid system $(V_D^1 \cup V_D^2, Q^1 \times Q^2, \mu_1, \mu_2, \mu_3)$ such that

- An activity f belongs to $\mu_1(\ell_1, \ell_2)$ iff the restriction of f to the data variables V_D^1, denoted by $f|_{V_D^1}$, is in $\mu_1^1(\ell_1)$, and $f|_{V_D^2}$ is in $\mu_1^2(\ell_2)$;
- A data state σ over $V_D^1 \cup V_D^2$ is in $\mu_2(\ell_1, \ell_2)$ iff $\sigma|_{V_D^1}$, the projection of σ onto the variables V_D^1, is in $\mu_2^1(\ell_1)$, or $\sigma|_{V_D^2}$ is in $\mu_2^2(\ell_2)$;
- $(\sigma, \sigma') \in \mu_3((\ell_1, \ell_2), (\ell_1', \ell_2'))$ iff $(\sigma|_{V_D^1}, \sigma'|_{V_D^1}) \in \mu_3(\ell_1, \ell_1')$ and $(\sigma|_{V_D^2}, \sigma'|_{V_D^2}) \in \mu_3(\ell_2, \ell_2')$.

It is not hard to see that traces of the product system are precisely those hybrid traces whose projections are traces of the component systems. It follows that the product of two linear hybrid systems is again linear, etc. An accepting run of a product automaton must meet the initial and acceptance conditions of both component automata.

2.3 Examples of hybrid systems

We model a thermostat, a water level monitor, a clock-based mutual-exclusion protocol, and a leaking gas burner as hybrid systems.

Temperature controller

Our first example describes a nonlinear hybrid system. The temperature of a plant is controlled through a thermostat, which continuously senses the temperature and turns a heater on and off [NSY92]. The temperature is governed by differential equations. When the heater is off, the temperature, denoted by the variable x, decreases according to the exponential function $x(t) = \theta e^{-Kt}$, where t is the time, θ is the initial temperature, and K is a constant determined by the plant; when the heater is on, the temperature follows the function $x(t) = \theta e^{-Kt} + h(1 - e^{-Kt})$, where h is a constant that depends on the power of the heater. Suppose that initially the temperature is M degrees and the heater is turned off. We wish to keep the temperature between m and M degrees. The resulting system can be described by the hybrid automaton of Figure 2 (note the representation of the initial condition $x = M$). The automaton has two locations: in location ℓ_0, the heater is turned off; in location ℓ_1, the heater is on.

Water level monitor

Our next example describes a linear system. The water level in a tank is controlled through a monitor, which continuously senses the water level and turns a pump on and off. Unlike the temperature, the water level changes as a piecewise-linear function over time. When the pump is off, the water level, denoted by the variable y, falls by 2 inches per second; when the pump is on, the water level rises by 1 inch per second. Suppose that initially the water level is 1 inch and the

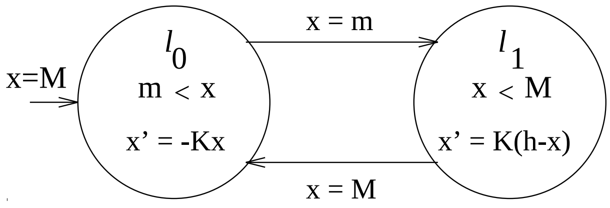

Figure 2: Temperature controller

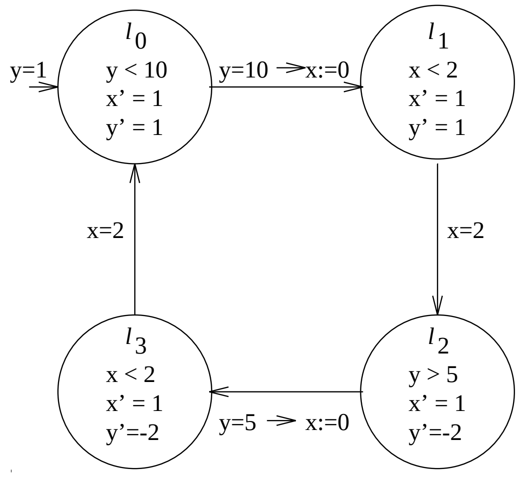

Figure 3: Water level monitor

pump is turned on. We wish to keep the water between 1 and 12 inches. But from the time that the monitor signals to change the status of the pump to the time that the change becomes effective, there is a delay of 2 seconds. Thus the monitor must signal to turn the pump on before the water level falls to 1 inch, and it must signal to turn the pump off before the water level reaches 12 inches.

The linear hybrid automaton of Figure 3 describes a water level monitor that signals whenever the water level passes 5 and 10 inches, respectively. The automaton has four locations: in locations ℓ_0 and ℓ_1, the pump is turned on; in locations ℓ_2 and ℓ_3, the pump is off. The clock x is used to specify the delays: whenever the automaton control is in location ℓ_1 or ℓ_3, the signal to switch the pump off or on, respectively, was sent x seconds ago. In the next section, we will prove that the monitor indeed keeps the water level between 1 and 12 inches.

Mutual-exclusion protocol

This example describes a parameterized multirate timed system. We present a timing-based algorithm that implements mutual exclusion for a distributed system with skewed clocks. Consider an asynchronous shared-memory system that consists of two processes P_1 and P_2 with atomic read and write operations. Each process has a critical section and at each time instant, at most one of the two processes is allowed to be in its critical section. Mutual exclusion is ensured by a version of Fischer's protocol [Lam87], which we describe first in pseudocode. For each process P_i, where $i = 1, 2$:

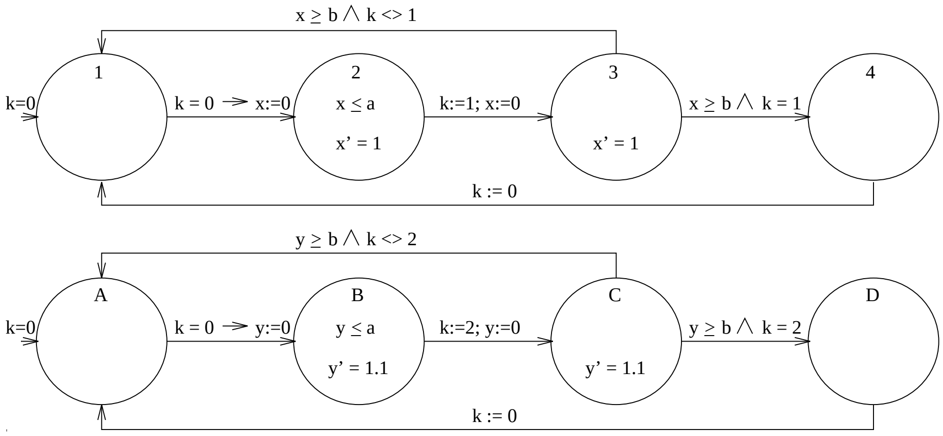

Figure 4: Mutual-exclusion protocol

repeat
    repeat
        await $k = 0$
        $k := i$
        delay b
    until $k = i$
    Critical section
    $k := 0$
forever

The two processes P_1 and P_2 share a variable k and process P_i is allowed to be in its critical section iff $k = i$. Each process has a private clock. The instruction **delay** b delays a process for at least b time units as measured by the process's local clock. Furthermore, each process takes at most a time units, as measured by the process's clock, for a single write access to the shared memory (i.e., for the assignment $k := i$). The values of a and b are the only information we have about the timing behavior of instructions. Clearly, the protocol ensures mutual exclusion only for certain values of a and b. If both private processor clocks proceed at precisely the same rate, then mutual exclusion is guaranteed iff $a < b$.

To make the example more interesting, we assume that the two private clocks of the processes P_1 and P_2 proceed at different rates, namely, the local clock of P_2 is 1.1 times faster than the clock of P_1. The resulting system can be modeled by the product of the two hybrid automata presented in Figure 4.

Each of the two automata models one process, with the two critical sections being represented by the locations 4 and D. The private clocks of the processes P_1 and P_2 determine the rate of change of the two skewed-clock variables x and y, respectively. In the next section, we will prove that mutual exclusion is guaranteed if $a = 2$ and $b = 3$: in this case, it will never happen that the control of P_1 is in location 4 while the control of P_2 is in location D.

Leaking gas burner

Now we consider an integrator system. In [CHR91], the duration calculus is used to prove that a gas burner does not leak excessively. It is known that (1) any leakage can be detected and stopped within 1 second and (2) the gas burner will not leak for 30 seconds after a leakage has been stopped. We wish to prove that the accumulated time of leakage is at most one twentieth of the time in any

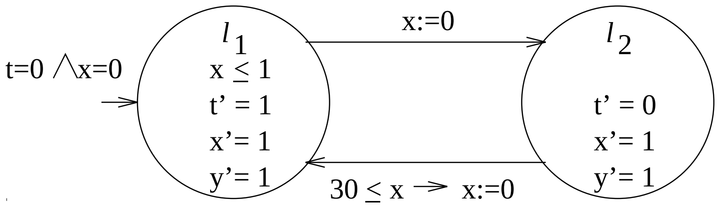

Figure 5: Leaking gas burner

interval of at least 60 seconds. The system is modeled by the hybrid automaton in Figure 5. The automaton has two locations: in location ℓ_1, the gas burner leaks; ℓ_2 is the nonleaking location. The integrator t records the cumulative leakage time; that is, the accumulated amount of time that the system has spent in location ℓ_1. The clock x records the time the system has spent in the current location; it is used to specify the properties (1) and (2). The clock y records the total elapsed time. In the next section, we will prove that $y \geq 60 \rightarrow 20t \leq y$ is an invariant of the system.

2.4 Undecidability of verification

The design of verification algorithms for hybrid systems is impaired by the fact that the emptiness problem ("Does a hybrid system have a run?") is undecidable already for very restricted classes of systems. On the positive side, the emptiness problem for timed automata (only propositions and clocks) is PSPACE-complete [AD90]. On the negative side, the emptiness problem is undecidable for asynchronous timed systems (propositions and skewed clocks that proceed at different rates) and for integrator systems (propositions and integrators).

To obtain strong undecidability results, we restrict the classes of multirate timed systems and integrator systems further. A linear hybrid system is *simple* if all linear atoms in exceptions and transition guards are of the form $x \leq k$, and all assignments are of the form $x := x$ or $x := k$, for $x \in V_D$ and $k \in \mathsf{Z}$. In particular, for n-rate timed systems the simplicity condition prohibits the comparison of clocks with different rates.

Theorem 1 *The emptiness problem is undecidable for 2-rate timed systems and for simple integrator systems.*

Proof. The first part of the theorem follows from the undecidability of the halting problem for nondeterministic 2-counter machines (NCMs). Given any two distinct clock rates, a 2-rate timed system can encode the computations of an NCM. Suppose we have three "accurate" clocks of rate 1 and two skewed clocks x_1 and x_2 of rate 2. Then we can encode the values of two counters in the i-th machine configuration by the values of x_1 and x_2 at accurate time i: the counter value n is encoded by the clock value $1/2^n$.

The accurate clock y is reset whenever it reaches 1 and thus marks intervals of length 1. It is obvious how a counter can be initialized to 0 and tested for being 0. Hence it remains to be shown how a counter can be incremented and decremented. To increment the counter represented by the skewed clock x from time i to time $i+1$, start an accurate clock z with x in the interval $[i-1, i)$ and reset z when it reaches 1; then nondeterministically reset x in the interval $[i, i+1)$ and test $x = z$ at time $i+1$. To decrement the counter represented by the skewed clock x from time i to time $i+1$, nondeterministically start an accurate clock z in the interval $[i-1, i)$ and test $x = z$ at time i; when z reaches 1 in the interval $[i, i+1)$, reset x. Given an NCM M, we can so construct a 2-rate timed system that has a run iff M halts. (Indeed, using acceptance conditions, we can construct a 2-rate timed automaton that has a run iff a counter is 0 infinitely often along some run of M; this shows that the emptiness problem is Σ_1^1-complete for 2-rate timed automata [HPS83].)

The second part of the theorem follows from an undecidability result for timed systems with memory cells [Čerāns]. ∎

We point out that the emptiness problem is decidable for simple n-rate timed automata. This is because any simple asynchronous timed automaton can be transformed into a timed automaton by (1) factoring into 1-process timed automata, (2) scaling all 1-process timed automata to the same clock rate, and (3) constructing the product. An analogous result holds for real-time temporal logics [WME92].

3 Verification Procedures

Consider a linear hybrid system $A = (V_D, Q, \mu_1, \mu_2, \mu_3)$. Given a linear formula ϕ over V_D, we wish to determine whether ϕ is an *invariant* of A:

"Is ϕ true in all states that occur on some trace of A?"

Recall that $\Sigma = Q \times \Sigma_D$ is the state space of A. We define the following reachability relations between states:

- *Time step.* For all locations $\ell \in Q$ and data variables $x \in V_D$, let $\mu_1(\ell, x) = k_x$ and let $\mu_2(\ell) = \phi$. For all data states $\sigma \in \Sigma_D$ and nonnegative reals $\delta \in \mathsf{R}^+$, define $(\sigma + \delta) \in \Sigma_D$ to be the data state that assigns to each data variable $x \in V_D$ the value $\sigma(x) + \delta \cdot k_x$. Then $\Rightarrow_l \subseteq \Sigma^2$ is the smallest relation such that

  if $\neg(\sigma + \delta')(\phi)$ for all $0 \leq \delta' < \delta$, then $(\ell, \sigma) \Rightarrow_l (\ell, \sigma + \delta)$;

  $\Rightarrow_r \subseteq \Sigma^2$ is the smallest relation such that

  if $\neg(\sigma + \delta')(\phi)$ for all $0 < \delta' \leq \delta$, then $(\ell, \sigma) \Rightarrow_r (\ell, \sigma + \delta)$;

  and $\Rightarrow_{lr} \subseteq \Sigma^2$ is the smallest relation such that

  if $\neg(\sigma + \delta')(\phi)$ for all $0 \leq \delta' \leq \delta$, then $(\ell, \sigma) \Rightarrow_{lr} (\ell, \sigma + \delta)$.

  In other words, $\Rightarrow_{lr} = \Rightarrow_l \cap \Rightarrow_r$.

- *Transition step.* The relation $\triangleright \subseteq \Sigma^2$ is the smallest relation such that for all pairs $(\ell, \ell') \in Q^2$ and all data states $\sigma_1, \sigma_2 \in \Sigma_D$,

  if $(\sigma_1, \sigma_2) \in \mu_3(\ell, \ell')$, then $(\ell, \sigma_1) \triangleright (\ell', \sigma_2)$.

  Note that the relation $\triangleright$ is reflexive.

- *Single step.* For all states $\sigma_1, \sigma_2 \in \Sigma$,

  $\sigma_1 \Rightarrow \sigma_2$ iff there exist $\sigma_1', \sigma_2' \in \Sigma$ with either $\sigma_1 \Rightarrow_{lr} \sigma_1' \triangleright \sigma_2' \Rightarrow_r \sigma_2$ or $\sigma_1 \Rightarrow_l \sigma_1' \triangleright \sigma_2' \Rightarrow_{lr} \sigma_2$.

- The *reachability relation* $\Rightarrow^* \subseteq \Sigma^2$ is the transitive closure of the single-step relation $\Rightarrow$.

The single-step reachability relation can be extended to sets of states. For a state $\sigma \in \Sigma$ and a set $R \subseteq \Sigma$ of states, let $\sigma \Rightarrow R$ iff $\sigma \Rightarrow \sigma'$ for some $\sigma' \in R$; for two sets $R_1, R_2 \subseteq \Sigma$ of states, define $R_1 \Rightarrow R_2$ iff $\sigma \Rightarrow R_2$ for some $\sigma \in R_1$. Again, $\Rightarrow^*$ denotes the transitive closure of $\Rightarrow$.

Given two sets $R_i, R_f \subseteq \Sigma$ of states, we wish to find out if the reachability relation $R_i \Rightarrow^* R_f$ holds. A solution to this reachability problem allows the verification of safety properties of the hybrid system A. Suppose the initial condition is given by the labeling function μ_4, then take R_i to be the set defined by $(\ell, \sigma) \in R_i$ iff $\sigma \in \mu_4(\ell)$. To check whether a linear formula ϕ is an invariant of A, we consider the set R_f of "bad" states: $\sigma \in R_f$ iff $\neg\phi(\sigma)$. Now ϕ is an invariant iff the reachability relation $R_i \Rightarrow^* R_f$ does not hold.

From the undecidability of the emptiness problem, it follows that the reachability problem is undecidable for linear hybrid automata. As the state space Σ of A is generally infinite, we will attempt to work on a quotient of the state-transition graph $(\Sigma, \Rightarrow)$. Our method will succeed only if there is a finite quotient of the state space in which states are identified whenever they are "equivalent" with respect to the given reachability problem (R_i, R_f). This problem can be attacked in many different ways:

- We can choose from two definitions of state equivalence. We can move "forward" from the initial set R_i and identify two states whenever they can be reached from R_i by the same sequence of single steps. Alternatively, we can move "backward" from the final set R_f and identify two states whenever they can reach R_f by the same sequence of single steps.

- Working forward from R_i (backward from R_f), we can choose to add one equivalence class of states at a time until either the current set intersects with R_f (or R_i, respectively) or no new states can be added. We refer to this category of verification methods as *fixpoint* methods, because the computation can be viewed as the iterative approximation of a fixpoint that defines the class of reachable states.

  Alternatively, we can start with an initial partition of the state space and refine it until it respects the equivalence relation, and thus can be used for checking reachability. The verification methods in this category are called *minimization* methods, because the computation can be viewed as constructing a bisimulation relation, namely, the minimal (coarsest) state partition that respects single-step reachability.

In this paper, we present one instance of the fixpoint computation approach and one instance of the minimization approach. Both procedures rely on the same set of primitive operations.

3.1 Fixpoint computation

We define a backward fixpoint computation procedure that solves the reachability problem (R_i, R_f) provided it terminates. The procedure starts with the set $R_{cur} = R_f$ and repeatedly adds states from which any state in R_{cur} can be reached. The procedure terminates with the answer YES (indicating that R_f is reachable from R_i) if at some stage an initial state in R_i is added, and it terminates with the answer NO if no new states can be added. The procedure may, of course, not terminate at all; it is a semidecision procedure for the reachability problem of linear hybrid systems.

Backward fixpoint computation:

$R_{old} := \emptyset$; $R_{cur} := R_f$
while $R_{cur} \cap R_i = \emptyset$ and $R_{cur} \not\subseteq R_{old}$ **do**
$R := pre(R_{cur})$
$R_{old} := R_{old} \cup R_{cur}$; $R_{cur} := R$
od
return $R_{cur} \cap R_i \neq \emptyset$.

The crucial step is the computation of the state set

$$pre(R) \ = \ \{\sigma \in \Sigma \mid \sigma \Rightarrow R\}.$$

This computation is possible due to the fact that all state sets encountered by the procedure are definable by linear formulas, and hence, for two state sets R and R', the problems of deciding whether $R \subseteq R'$ and whether $R \cap R' = \emptyset$ have algorithmic solutions. A *data region* $R_D \subseteq \Sigma$ is a set of data states of the form $\{\sigma \in \Sigma_D \mid \sigma(\phi)\}$, for some linear formula ϕ over V_D. A *region* $(\ell, R_D) \subseteq \Sigma$ consists of a location $\ell \in Q$ and a data region R_D. The union $R \subseteq \Sigma$ of regions (ℓ, R_D^ℓ), one for each location $\ell \in Q$, is called a *region family*. The following central lemma ensures that for linear hybrid systems, all required state sets are computable:

Lemma 1 *If $R \subseteq \Sigma$ is a region family, then so is $pre(R)$.*

Proof. It suffices to show that if R_D is a data region, then so is the set

$$pre_e(R_D) \ = \ \{\sigma \in \Sigma_D \mid (\ell_1, \sigma) \Rightarrow (\ell_2, R_D)\},$$

for each pair $e = (\ell_1, \ell_2)$ of locations. Let ψ be the linear formula that defines R_D. We construct a linear formula $pre_e(\psi)$ that defines the set $pre_e(R_D)$. If ψ contains n variables, we can think of ψ as defining a set of points in n-dimensional space. This set is an n-dimensional polyhedron whose bounding hyperplanes are linear functions with rational coefficients.

First let us extend the time-step and transition-step relations to linear formulas. For any linear formula ψ and location ℓ of A:

$$\triangleright(e, \psi) \ = \ \{\sigma \in \Sigma_D \mid \exists \sigma' \in \Sigma_D. (\psi(\sigma') \wedge (\ell_1, \sigma) \triangleright (\ell_2, \sigma'))\},$$

$$\rightarrow(\ell, \psi) \ = \ \{\sigma \in \Sigma_D \mid \exists \sigma' \in \Sigma_D. (\psi(\sigma') \wedge (\ell, \sigma) \rightarrow (\ell, \sigma'))\}$$

for $\rightarrow \in \{\Rightarrow_l, \Rightarrow_r, \Rightarrow_{lr}\}$. Then the linear formula $pre_e(\psi)$ is the following disjunction:

$$pre_e(\psi) \ = \Rightarrow_{lr} (\ell_1, \triangleright(e, \Rightarrow_r (\ell_2, \psi))) \vee \Rightarrow_l (\ell_1, \triangleright(e, \Rightarrow_{lr} (\ell_2, \psi))).$$

The transition-step relation $\triangleright$ can be computed by substitution. Let ϕ be the guard of $\mu_3(e)$ and for all $x \in V_D$, let $\mu_3(e, x) = \alpha_x$. Then:

$$\triangleright(e, \psi) \ = \ \phi \wedge (\psi[x := \alpha_x]),$$

where the linear formula $\psi[x := \alpha_x]$ is obtained by replacing all occurrences of x in ψ with α_x.

The time-step relations can be computed by quantifier elimination. For all locations ℓ of A, let $\mu_2(\ell) = \phi_\ell$. If $\mu_1(\ell, x) = k_x$ for all $x \in V_D$, then the linear formulas $\psi +_\ell \delta$ and $\psi -_\ell \delta$ result from ψ by replacing all occurrences of x with $x + k_x \cdot \delta$ or $x - k_x \cdot \delta$, respectively. Then:

$$\Rightarrow_r (\ell, \psi) \ = \ (\exists \delta \geq 0. (\psi \wedge \neg\phi_\ell) +_\ell \delta \ \wedge \forall 0 < \epsilon < \delta. \neg\phi_\ell +_\ell \epsilon),$$

$$\Rightarrow_l (\ell, \psi) \;=\; (\psi \wedge \neg\phi_\ell) \vee (\exists \delta > 0.\, \psi +_\ell \delta \wedge \forall 0 \leq \epsilon < \delta.\, \neg\phi_\ell +_\ell \epsilon),$$

and $\Rightarrow_{lr} (\ell, \psi) = \neg\phi_\ell \wedge \Rightarrow_r (\ell, \psi)$. It remains to be shown how the quantifiers can be eliminated from these formulas. We first convert all quantifiers into existential form and translate all quantifier-free subformulas into disjunctive normal form. Since existential quantifiers distribute over disjunction, it suffices construct a linear formula over V that is equivalent to the formula $\exists \delta \in \mathsf{R}.\, \varphi$, where φ is a conjunction of linear inequalities over $V \cup \{\delta\}$. Note that the formula $\exists \delta.\, \varphi$ defines a *convex* rational polyhedron. To eliminate the existential quantifier, (1) solve all inequalities for δ and (2) construct the conjunction of all δ-free inequalities that are implied by transitivity. ■

As for timed systems [HNSY92], the fixpoint method can be extended to check properties of linear hybrid systems that are specified in real-time extensions of branching-time logics such as CTL.

3.2 Minimization

Let π be a partition of the state space Σ into regions. A region $R \in \pi$ is *stable* if

$$\forall R' \in \pi.\, (R \Rightarrow R' \text{ implies } \forall \sigma \in R.\, \sigma \Rightarrow R').$$

The partition π is a *bisimulation* if every region of π is stable. The partition π *respects* R_f if for every region $R \in \pi$, either $R \subseteq R_f$ or $R \cap R_f = \emptyset$. Observe that if a partition π that respects R_f is a bisimulation, then it can be used for reachability analysis: to see if R_f is reachable from R_i, check if there exists a path from some π-region R_1 such that $R_1 \cap R_i \neq \emptyset$ to some π-region R_2 such that $R_2 \subseteq R_f$. Our objective is to construct the coarsest bisimulation provided it is finite. For this purpose, we can adopt algorithms for performing a simultaneous reachability and minimization analysis of transition systems [BFH90, LY92].

The minimization procedure of [BFH90] is given below. Let $\pi_0 = \{(\ell, \mathit{true}) \mid \ell \in Q\}$ be the initial partition of Σ into regions — one region per location. The initial partition is refined into $\pi_1 = \pi_0 \cap \{R_f, \Sigma - R_f\}$ so that it respects R_f.

Minimization:

> $\pi := \pi_1$; $\alpha := \{R \mid R \cap R_i \neq \emptyset\}$; $\beta := \emptyset$
> **while** $\alpha \neq \beta$ **do**
>   choose $R \in (\alpha - \beta)$
>   let $\alpha' := \mathit{split}(R, \pi)$
>   **if** $\alpha' = \{R\}$ **then**
> $\beta := \beta \cup \{R\}$
> $\alpha := \alpha \cup \{R' \in \pi \mid R \Rightarrow R'\}$
>   **else**
> $\alpha := \alpha - \{R\}$
>   **if** $\exists R' \in \alpha'$ such that $R' \cap R_i \neq \emptyset$ **then** $\alpha := \alpha \cup \{R'\}$ **fi**
> $\beta := \beta - \{R' \in \pi \mid R' \Rightarrow R\}$
> $\pi := (\pi - \{R\}) \cup \alpha'$
>   **fi**
> **od**
> **return** there is $R \in \alpha$ such that $R \subseteq R_f$.

Starting from π_1, the procedure selects a region R and checks if R is stable with respect to the current partition; if not, then R is split into smaller regions. Additional book-keeping is needed to record which regions are reachable from the region containing the initial state. In the following procedure, π is the current partition, α is the set of π-regions that have been found reachable from (the region of) the initial state, and β is the set of π-regions that have been found stable with respect to π. The function $split(R, \pi)$ splits the π-region R into subregions that are "more" stable with respect to π: $split(R, \pi) := \{R', R - R'\}$ if there is some region $R'' \in \pi$ such that the region $R' = pre(R'') \cap R$ is a proper subset of R, and $split(R, \pi) := \{R\}$ otherwise. Since the operation *split* is computed using *pre*, all state sets encountered by the minimization procedure are again definable by linear formulas. The procedure terminates if the coarsest bisimulation has only a finite number of equivalence classes.

If the minimization procedure terminates, we obtain a finite bisimulation of Σ with respect to $\Rightarrow$. As with timed automata [ACH+92], the resulting reachability graph can be used to solve also the emptiness problem for A, even in presence of acceptance conditions, and for model checking branching-time properties. The minimization procedure may be replaced by the more efficient procedure presented in [LY92], which can also be implemented using the primitive *pre*.

3.3 Verification examples

In the following, we demonstrate that both the fixpoint computation procedure and the minimization procedure terminate in many cases.

Minimization: water level monitor

Let A be the hybrid automaton defined in Figure 3. We use the minimization procedure to prove that the formula $1 \le y \le 12$ is an invariant of A. It follows that the water level monitor keeps the water level between 1 and 12 inches.

By $\langle\psi\rangle$, for a linear formula ψ over V, we denote the set of all states (ℓ, σ) such that σ satisfies $\psi[pc := \ell]$. Let the set R_i of initial states be so defined by

$$R_i \;=\; \langle pc = 0 \wedge x = 0 \wedge y = 1\rangle$$

and let the set R_f of "bad" states be defined by

$$R_f \;=\; \langle y < 1 \vee y > 12\rangle.$$

The initial partition is $\pi_0 = \{(\ell, true) \mid \ell \in \{\ell_0, \ell_1, \ell_2, \ell_3\}\}$. We next partition each region of the initial partition into "good" and "bad" states: $\pi_1 = \{$

$$\begin{array}{llll} C_{00} = & \langle pc = 0 \wedge 1 \le y \le 12\rangle, & C_{01} = & \langle pc = 0 \wedge (y < 1 \vee y > 12)\rangle, \\ C_{10} = & \langle pc = 1 \wedge 1 \le y \le 12\rangle, & C_{11} = & \langle pc = 1 \wedge (y < 1 \vee y > 12)\rangle, \\ C_{20} = & \langle pc = 2 \wedge 1 \le y \le 12\rangle, & C_{21} = & \langle pc = 2 \wedge (y < 1 \vee y > 12)\rangle, \\ C_{30} = & \langle pc = 3 \wedge 1 \le y \le 12\rangle, & C_{31} = & \langle pc = 3 \wedge (y < 1 \vee y > 12)\rangle\}. \end{array}$$

The bad states are in the regions C_{i1}, for $i \in \{0, 1, 2, 3\}$. Since the initial region R_i is contained in C_{00}, let $\alpha = \{C_{00}\}$. Considering $R = C_{00} \in \alpha$, we find that $split(C_{00}, \pi_1) = \{$

$$\begin{array}{lll} C_{000} & = & \langle pc = 0 \wedge 1 \le y < 10\rangle, \\ C_{001} & = & \langle pc = 0 \wedge 10 \le y \le 12\rangle\}. \end{array}$$

Therefore, $\pi_2 = \{C_{000}, C_{001}, C_{01}, C_{10}, C_{11}, C_{20}, C_{21}, C_{30}, C_{31}\}$. Now $R_i \subseteq C_{000}$, so take $\alpha = \{C_{000}\}$ and $\beta = \emptyset$. Considering $R = C_{000}$, we find that it is stable with respect to π_2. Thus $\alpha = \alpha \cup \{R' \in$

$\pi \mid R \Rightarrow R'\} = \{C_{000}, C_{001}, C_{10}\}$ and $\beta = \{C_{000}\}$. Since $R = C_{001}$ is also stable in π_2 and is not reaching any new states not in α, α remains the same and $\beta = \{C_{000}, C_{001}\}$. However, considering $R = C_{10}$, we obtain $split(C_{10}, \pi_2) = \{$

$$\begin{aligned} C_{100} &= \langle pc = 1 \wedge 0 \leq x < 2 \wedge 1 \leq y \leq 12\rangle, \\ C_{101} &= \langle pc = 1 \wedge x \geq 2 \wedge 1 \leq y \leq 12\rangle\}. \end{aligned}$$

These two regions together with the regions in π_2, except for C_{10}, constitute π_3. The new β is obtained by removing $\{R' \in \pi \mid R' \Rightarrow R\} = C_{000}$ from the old β. The new α becomes $\{C_{000}, C_{001}\}$. Now $R = C_{000}$ is stable in π_3. Hence $\alpha = \{C_{000}, C_{001}, C_{100}\}$ and $\beta = \{C_{000}, C_{001}\}$. Since $R = C_{100}$ is stable in π_3, we have $\alpha = \{C_{000}, C_{001}, C_{100}, C_{101}, C_{20}\}$ and $\beta = \{C_{000}, C_{001}, C_{100}\}$. $R = C_{101}$ is also stable in π_3, so $\beta = \{C_{000}, C_{001}, C_{100}, C_{101}\}$ and α remains unchanged. Considering $R = C_{20}$, we obtain $split(C_{20}, \pi_3) = \{$

$$\begin{aligned} C_{200} &= \langle pc = 2 \wedge 5 < y \leq 12\rangle, \\ C_{201} &= \langle pc = 2 \wedge 1 \leq y \leq 5\rangle\}. \end{aligned}$$

Now π_4 contains C_{200} and C_{201}, and thus C_{100} must be reconsidered. It is split into $split(C_{100}, \pi_4) = \{$

$$\begin{aligned} C_{1000} &= \langle pc = 1 \wedge 0 \leq x < 2 \wedge 3 < y \leq 12 \wedge 3 < y - x \leq 12\rangle, \\ C_{1001} &= \langle pc = 1 \wedge 0 \leq x < 2 \wedge 1 \leq y \leq 5 \wedge 1 \leq y - x \leq 3\rangle\}. \end{aligned}$$

Thus π_5 contains C_{1000} and C_{1001}. After finding that C_{000}, C_{1000} and C_{200} all are stable, we finally have $\alpha = \{C_{000}, C_{001}, C_{1000}, C_{200}, C_{201}, C_{30}\}$ and $\beta = \{C_{000}, C_{001}, C_{1000}, C_{200}\}$. So let $R = C_{201}$. It is stable, so $\beta = \beta \cup \{C_{200}\}$ and α does not change. Then $R = C_{30}$ is partitioned into $\{$

$$\begin{aligned} C_{300} &= \langle pc = 3 \wedge 0 \leq x < 2 \wedge 1 \leq y \leq 12\rangle, \\ C_{301} &= \langle pc = 3 \wedge x \geq 2 \wedge 1 \leq y \leq 12\rangle\}. \end{aligned}$$

C_{200} has to be considered again. It is stable with respect to the current partition. Then $R = C_{300}$ is considered and $split(C_{300}, \pi_6) = \{$

$$\begin{aligned} C_{3000} &= \langle pc = 3 \wedge 0 \leq x < 2 \wedge 1 \leq y \leq 12 \wedge 5 \leq y + 2x < 14\rangle, \\ C_{3001} &= \langle pc = 3 \wedge 0 \leq x < 2 \wedge 1 \leq y < 5 \wedge 1 \leq y + 2x < 5\rangle\}. \end{aligned}$$

We must consider C_{200} again. It turns out that it is still stable. After considering $R = C_{3000}$, we have $\beta = \{C_{000}, C_{001}, C_{1000}, C_{200}, C_{201}, C_{3000}\}$ and $\alpha = \alpha \cup \{C_{000}\}$. Now the partition is

$$\begin{aligned} \pi_7 = \; & \{C_{000}, C_{001}, C_{01}, C_{1000}, C_{1001}, C_{101}, C_{11}, \\ & C_{200}, C_{201}, C_{21}, C_{3000}, C_{3001}, C_{301}, C_{31}\}. \end{aligned}$$

Since C_{000} is stable in π_7, we have $\alpha = \beta = \{C_{000}, C_{001}, C_{1000}, C_{200}, C_{201}, C_{3000}\}$. Notice that no region in α contains any bad states from R_f. Therefore, the invariant property has been verified.

Fixpoint computation: mutual-exclusion protocol

Let A be the product of the two hybrid automata defined in Figure 4, for $a = 2$ and $b = 3$. We use the fixpoint computation procedure to prove that the formula $pc \neq (4, D)$ is an invariant of A. It follows that the protocol ensures mutual exclusion.

Let $R_i = \langle pc = (1, A)\rangle$ be the region of initial states and let $R_f = \langle pc = (4, D)\rangle$ be the region of "bad" states. Let R^i denote the value of $R = pre(R_{cur})$ after the i-th iteration of the algorithm. Then $R^0 = R_f$ and

$$R^1 \;=\; pre(R_{cur}) \;=\; \langle (pc_1 = 3 \wedge pc_2 = D \wedge 1 = k) \vee (pc_1 = 4 \wedge pc_2 = C \wedge 2 = k)\rangle,$$

where pc_i, for $i = 1, 2$, denotes the i-th component of the control variable pc. We keep computing $R^{i+1} = pre(R^i)$ as long as R^i and R_i are disjoint:

$$\begin{aligned}
R^2 \;=\; & \langle (pc_1 = 2 \wedge pc_2 = D \wedge x \leq 2) \vee (pc_1 = 4 \wedge pc_2 = B \wedge y \leq 2) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge y \leq 2 \wedge 13 \leq 11x - 10\,y \wedge 1 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge x \leq 2 \wedge 8 \leq -11x + 10\,y \wedge 2 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge x \leq 2 \wedge 30 \leq 10\,y \wedge 2 = k) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge y \leq 2 \wedge 3 \leq x \wedge 1 = k)\rangle \\
R^3 \;=\; & \langle (pc_1 = 1 \wedge pc_2 = D \wedge 0 = k) \vee (pc_1 = 4 \wedge pc_2 = A \wedge 0 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge 2 = k \wedge x \leq 2 \wedge 8 \leq -11x + 10y) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge 1 = k \wedge y \leq 2 \wedge -20 \leq -10y \wedge 13 \leq 11x - 10y) \\
& \vee (pc_1 = 2 \wedge pc_2 = B \wedge x \leq 2 \wedge y \leq 2 \wedge 13 \leq -10\,y)\rangle \\
R^4 \;=\; & \langle (pc_1 = 4 \wedge pc_2 = D) \vee (pc_1 = 3 \wedge pc_2 = D \wedge 0 = k) \\
& \vee (pc_1 = 4 \wedge pc_2 = C \wedge 0 = k) \vee (pc_1 = 3 \wedge pc_2 = D \wedge 1 = k) \\
& \vee (pc_1 = 4 \wedge pc_2 = C \wedge 2 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = A \wedge 8 \leq -11x \wedge 0 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = B \wedge y \leq 2 \wedge 8 \leq -11x) \\
& \vee (pc_1 = 2 \wedge pc_2 = B \wedge x \leq 2 \wedge y \leq 2 \wedge 13 \leq -10\,y) \\
& \vee (pc_1 = 1 \wedge pc_2 = B \wedge 0 = k \wedge y \leq 2 \wedge 13 \leq -10\,y)\rangle \\
R^5 \;=\; & \langle (pc_1 = 3 \wedge pc_2 = D \wedge 1 = k) \vee (pc_1 = 4 \wedge pc_2 = C \wedge 2 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = D \wedge x \leq 2) \vee (pc_1 = 4 \wedge pc_2 = B \wedge y \leq 2) \\
& \vee (pc_1 = 2 \wedge pc_2 = A \wedge 8 \leq -11x \wedge 0 = k) \\
& \vee (pc_1 = 1 \wedge pc_2 = B \wedge y \leq 2 \wedge 13 \leq -10\,y \wedge 0 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge 8 \leq -11x \wedge 38 \leq -11x + 10\,y \wedge 0 = k) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge y \leq 2 \wedge 13 \leq 11x - 10\,y \wedge 1 = k) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge y \leq 2 \wedge 3 \leq x \wedge 1 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge x \leq 2 \wedge 8 \leq -11x + 10\,y \wedge 2 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge x \leq 2 \wedge 3 \leq y \wedge 2 = k) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge y \leq 2 \wedge 13 \leq -10\,y \wedge 46 \leq 11x - 10\,y \wedge 0 = k)\rangle \\
R^6 \;=\; & \langle (pc_1 = 1 \wedge pc_2 = D \wedge 0 = k) \vee (\wedge pc_1 = 4 \wedge pc_2 = A \wedge 0 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = D \wedge x \leq 2) \vee (pc_1 = 4 \wedge pc_2 = B \wedge y \leq 2) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge 8 \leq -11x \wedge 38 \leq -11x + 10\,y \wedge 0 = k) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge y \leq 2 \wedge 13 \leq 11x - 10\,y \wedge 1 = k) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge y \leq 2 \wedge 3 \leq x \wedge 1 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge x \leq 2 \wedge 8 \leq -11x + 10\,y \wedge 2 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = C \wedge x \leq 2 \wedge 3 \leq y \wedge 2 = k) \\
& \vee (pc_1 = 2 \wedge pc_2 = B \wedge x \leq 2 \wedge y \leq 2 \wedge 13 \leq -10\,y) \\
& \vee (pc_1 = 3 \wedge pc_2 = B \wedge y \leq 2 \wedge 13 \leq -10\,y \wedge 46 \leq 11x - 10\,y \wedge 0 = k)\rangle
\end{aligned}$$

Since $R^6 \subseteq R^3 \cup R^5$, a fixpoint is found in 6 iterations. Notice that the fixpoint $\bigcup_{0 \leq i \leq 5} R^i$ contains no initial states from R_i. Therefore, the invariant property has been verified.

Fixpoint computation: leaking gas burner

Let A be the integrator system defined in Figure 5. We use the fixpoint computation procedure to prove that the formula $y \geq 60 \rightarrow 20t \leq y$ is an invariant of A. It follows that the gas burner leaks at most one twentieth of the time in any interval of at least 60 seconds. Let $R_i = \langle pc = 1 \wedge t = 0 \wedge x = 0\rangle$ be the region of initial states and let $R_f = \langle y \geq 60 \wedge 20t > y\rangle$ be the region of "bad" states. Let R^i again denote the value of $R = pre(R_{cur})$ after the i-th iteration of the algorithm. Then:

$$
\begin{array}{rcl}
R^0 & = & R_f \\
R^1 & = & \langle (pc = 2 \wedge -19 < 20t - y \wedge 11 < 20t + x - y \wedge 2 < t \wedge 0 \leq t \wedge 0 \leq x) \\
 & & \vee (pc = 1 \wedge -19 < 20t - 19x - y \wedge 2 < t - x \wedge -1 \leq -x \wedge 0 \leq x)\rangle \\
R^2 & = & \langle (pc = 1 \wedge -8 < 20t - 19x - y \wedge 1 < t - x \wedge -1 \leq -x \wedge 0 \leq x) \\
 & & \vee (pc = 2 \wedge -19 < 20t - y \wedge 2 < t \wedge 11 < 20t + x - y \wedge 0 \leq x)\rangle \\
R^3 & = & \langle (pc = 2 \wedge -8 < 20t - y \wedge 1 < t \wedge 22 < 20t + x - y \wedge 0 \leq x) \\
 & & \vee (pc = 1 \wedge -8 < 20t - 19x - y \wedge 1 < t - x \wedge -1 \leq -x \wedge 0 \leq x)\rangle \\
R^4 & = & \langle (pc = 1 \wedge 0 < t - x \wedge 3 < 20t - 19x - y \wedge -1 \leq -x \wedge 0 \leq x) \\
 & & \vee (pc = 2 \wedge -8 < 20t - y \wedge 1 < t \wedge 22 < 20t + x - y \wedge 0 \leq x)\rangle \\
R^5 & = & \langle (pc = 2 \wedge 0 < t \wedge 3 < 20t - y \wedge 33 < 20t + x - y \wedge 0 \leq x) \\
 & & \vee (pc = 1 \wedge 0 < t - x \wedge 3 < 20t - 19x - y \wedge -1 \leq -x \wedge 0 \leq x)\rangle \\
R^6 & = & \langle (pc = 1 \wedge -1 < t - x \wedge 14 < 20t - 19x - y \wedge -1 \leq -x \wedge 0 \leq x) \\
 & & \vee (pc = 2 \wedge 0 < t \wedge 3 < 20t - y \wedge 33 < 20t + x - y \wedge 0 \leq x)\rangle \\
R^7 & = & \langle (pc = 2 \wedge 14 < 20t - y \wedge 44 < 20t + x - y \wedge 0 \leq t \wedge 0 \leq x) \\
 & & \vee (pc = 1 \wedge -1 < t - x \wedge 14 < 20t - 19x - y \wedge -1 \leq -x \wedge 0 \leq x)\rangle \\
R^8 & = & \langle (pc = 1 \wedge 25 < 20t - 19x - y \wedge -1 \leq t - x \wedge -1 \leq -x \wedge 0 \leq x) \\
 & & \vee (pc = 2 \wedge 14 < 20t - y \wedge 44 < 20t + x - y \wedge 0 \leq t \wedge 0 \leq x)\rangle \\
R^9 & = & \langle (pc = 2 \wedge 25 < 20t - y \wedge 55 < 20t + x - y \wedge 0 \leq t \wedge 0 \leq x) \\
 & & \vee (pc = 1 \wedge 25 < 20t - 19x - y \wedge -1 \leq t - x \wedge -1 \leq -x \wedge 0 \leq x)\rangle
\end{array}
$$

Since $R^9 \subseteq R^8$, a fixpoint is found in 9 iterations. As the fixpoint $\bigcup_{0 \leq i \leq 8} R^i$ contains no initial states from R_i, the invariant property has been verified.

Acknowledgements. Amir Pnueli and Joseph Sifakis have influenced the ideas contained in this paper through numerous discussions.

References

[ACH+92] R. Alur, C. Courcoubetis, N. Halbwachs, D.L. Dill, and H. Wong-Toi. Minimization of timed transition systems. In W.R. Cleaveland, editor, *CONCUR 92: Theories of Concurrency*, Lecture Notes in Computer Science 630, pages 340–354. Springer-Verlag, 1992.

[AD90] R. Alur and D.L. Dill. Automata for modeling real-time systems. In M.S. Paterson, editor, *ICALP 90: Automata, Languages, and Programming*, Lecture Notes in Computer Science 443, pages 322–335. Springer-Verlag, 1990.

[BFH90] A. Bouajjani, J.C. Fernandez, and N. Halbwachs. Minimal model generation. In R.P. Kurshan and E.M. Clarke, editors, *CAV 90: Automatic Verification Methods for Finite-state Systems*, Lecture Notes in Computer Science 531, pages 197–203. Springer-Verlag, 1990.

[Čerāns] K. Čerāns. Decidability of bisimulation equivalence for parallel timer processes. In *CAV 92: Automatic Verification Methods for Finite-state Systems*, Lecture Notes in Computer Science. Springer-Verlag. To appear.

[CHR91] Z. Chaochen, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.

[HNSY92] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science*, pages 394–406. IEEE Computer Society Press, 1992.

[HPS83] D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 26(2):222–243, 1983.

[Lam87] L. Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems*, 5(1):1–11, 1987.

[LY92] D. Lee and M. Yannakakis. Online minimization of transition systems. In *Proceedings of the 24th Annual Symposium on Theory of Computing*. ACM Press, 1992.

[MMP92] O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 447–484. Springer-Verlag, 1992.

[NSOY] X. Nicollin, J. Sifakis, A. Olivero, and S. Yovine. An approach to the description and analysis of hybrid systems. To appear.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine. From ATP to timed graphs and hybrid systems. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, Lecture Notes in Computer Science 600, pages 549–572. Springer-Verlag, 1992.

[WME92] F. Wang, A.K. Mok, and E.A. Emerson. Real-time distributed system specification and verification in asynchronous propositional temporal logic. In *Proceedings of the 12th International Conference on Software Engineering*, 1992.