Verifying Hybrid Systems *

Zohar Manna[1] and Amir Pnueli[2]

[1] Department of Computer Science, Stanford University
Stanford, CA 94305
e-mail: `manna@cs.stanford.edu`

[2] Department of Applied Mathematics and Computer Science
Weizmann Institute, Rehovot, Israel
e-mail: `amir@wisdom.weizmann.ac.il`

Abstract. Hybrid systems are modeled as *phase transition systems* with sampling semantics. By identifying a set of *important events* it is ensured that all significant state changes are observed, thus correcting previous drawbacks of the sampling computations semantics. A proof rule for verifying properties of hybrid systems is presented and illustrated on several examples.

Keywords: Temporal logic, real-time, specification, verification, hybrid systems, statecharts, proof rules, phase transition system, sampling semantics, important events.

1 Introduction

Hybrid systems are reactive systems that intermix discrete and continuous components. Typical examples are digital controllers that interact with continuously changing physical environments.

A formal model for hybrid systems was proposed in [MMP92], based on the notion of *phase transition systems* (PTS). Two types of semantics were considered in [MMP92]. The first semantics, to which we refer here as the *super dense* semantics, is based on *hybrid traces* which view computations as functions from pairs of a real number (representing time) and an index (representing discrete steps) to states. The second semantics is that of *sampling computations*, which sample the continuous behavior of a hybrid system at countably many observation points.

It is argued in [MMP92] that the super dense semantics provides a more accurate and faithful description of hybrid behaviors. In particular, it never

* This research was supported in part by the National Science Foundation under grant CCR-92-23226, by the Defense Advanced Research Projects Agency under contract NAG2-703, by the United States Air Force Office of Scientific Research under contract F49620-93-1-0139, by the European Community ESPRIT Basic Research Action Project 6021 (REACT) and by the France-Israel project for cooperation in Computer Science.

misses events of a variable crossing a threshold or a state predicate changing truth value. Sampling computations, on the other hand, may miss such an event if it happens to occur between two observation points.

However, when comparing the two approaches with respect to ease of verification, it is obvious that this is a wide open question under the super dense view, while verification under the sampling view can be obtained by generalizing the sequence-based methods used for reactive and real-time systems. That such a generalization is possible has been indicated in [SBM92] and [AL92].

In this paper, we improve the accuracy with which sampling computations describe hybrid behaviors. This is done by augmenting the model of phase transition systems by a finite set of *important events* that should never be missed. Subsequently, we present a rule for proving invariance properties of hybrid systems under the sampling computations semantics.

The power of this rule is illustrated by proving several invariance properties of hybrid systems.

2 Computational Model: Phase Transition System

Hybrid systems are modeled as phase transition systems [MMP92]. The definition presented here introduces an additional component: a set of important events, whose purpose is to ensure that all significant state changes are observed in every sampling computation.

As the time domain we take the nonnegative reals R^+. In some cases, we also need its extension $\mathsf{R}^\infty = \mathsf{R}^+ \cup \{\infty\}$.

A *phase transition system* (PTS) $\Phi = \langle V, \Theta, \mathcal{T}, \mathcal{A}, \mathcal{I} \rangle$ consists of:

- $V = \{u_1, ..., u_n\}$: A finite set of *state variables*. The set $V = V_d \cup V_c$ is partitioned into V_d the set of *discrete variables* and V_c the set of *continuous variables*. Continuous variables have always the type *real*. The discrete variables can be of any type.
  We define a *state* s to be a type-consistent interpretation of V, assigning to each variable $u \in V$ a value $s[u]$ over its domain. We denote by Σ the set of all states.
- Θ : The *initial condition*. A satisfiable assertion characterizing the initial states.
- $\mathcal{T}$: A finite set of *transitions*. Each transition $\tau \in \mathcal{T}$ is a function

$$\tau : \Sigma \mapsto 2^{\Sigma},$$

  mapping each state $s \in \Sigma$ into a (possibly empty) set of τ-*successor* states $\tau(s) \subseteq \Sigma$.
  The function associated with a transition τ is represented by an assertion $\rho_\tau(V, V')$, called the *transition relation*, which relates a state $s \in \Sigma$ to its τ-successor $s' \in \tau(s)$ by referring to both unprimed and primed versions of the state variables. An unprimed version of a state variable refers to its value in s, while a primed version of the same variable refers to its value in s'. For

example, the assertion $x' = x + 1$ states that the value of x in s' is greater by 1 than its value in s.
The enabledness of a transition τ can be expressed by the formula

$$En(\tau): \ (\exists V')\rho_\tau(V, V'),$$

which is true in s iff s has some τ-successor. The enabling condition of a transition τ can always be written as $\delta \wedge \kappa$, where δ is the largest subformula that does not depend on continuous variables. We call κ the *continuous part* of the enabling condition, and denote it by $En_c(\tau)$.
A transition can also change the value of a continuous variable.

- $\mathcal{A}$: A finite set of *activities*. Each activity $\alpha \in \mathcal{A}$ is a conditional differential equation of the form:

  $$p \ \rightarrow \ \dot{x} = e,$$

  where p is a predicate over V_d called the *activation condition* of α, $x \in V_c$ is a continuous state variable, and e is an expression over V. We say that the activity α *governs* variable x. Activity α is said to be *active* in state s if its activation condition p holds on s. If p is *true*, it may be omitted.
  It is required that the activation conditions of the activities that govern any variable x are exhaustive and exclusive, i.e., exactly one of them holds on any state.
- A set of *important events* $\mathcal{I}$. This is a finite set of assertions that includes at least the assertions $En_c(\tau)$, for each $\tau \in \mathcal{T}$. These are assertions such that changes in their truth values must be observable. Usually, $\mathcal{I}$ includes, in addition to the assertions $\{En_c(\tau)\}$, all the assertions that appear in specifications of the system.

We introduce a special variable T, called the *clock variable*. At any point in an execution of a system, T has a value over R^+ representing the current time. The set of variables $V_T = V \cup \{T\}$ is called the set of *situation variables*. A type consistent interpretation of V_T is called a *situation*, and the set of all situations is denoted by Σ_T. Often, we represent a situation as a pair $\langle s, t\rangle$ where s is a state and $t \in \mathsf{R}^+$ is the interpretation of the clock T.

In [MP93], we present a similar model for hybrid systems which associates with each transition a pair of real numbers $l_\tau \leq u_\tau$ representing a lower and upper bound on the length of time a transition can be continuously enabled without being taken. The approach taken here makes no such association as part of the model. Instead, it is recommended that each transition that needs such bounds employs a special continuous *timer* variable that grows linearly with time and constrains the length of intervals in which the transition was continuously enabled.

Activity Successors

Consider a phase transition system Φ, and let $\langle s_1, t_1\rangle$ and $\langle s_2, t_2\rangle$ be two situations of Φ with $t_1 < t_2$. An *evolution* from $\langle s_1, t_1\rangle$ to $\langle s_2, t_2\rangle$ consists of a set of functions $F: \{f_x(t) \mid x \in V_c\}$ that are differentiable in the interval $[t_1, t_2]$ and satisfy the following requirements:

- $f_x(t_1) = s_1[x]$ and $f_x(t_2) = s_2[x]$. Thus, the values of $f_x(t)$ at the boundaries of the interval $[t_1, t_2]$ agree with the interpretation of x by s_1 and s_2, respectively.
- $s_1[y] = s_2[y]$ for every $y \in V_d$. That is, states s_1 and s_2 agree on the values of all discrete variables.
- For every activity $p \rightarrow \dot{x} = e$ that governs x, if p holds at s_1, then f_x satisfies the differential equation

  $$\dot{f}_x(t) = e(F)$$

  in the interval $[t_1, t_2]$, where the expression $e(F)$ is obtained from e by replacing each occurrence of a variable $y \in V_c$ by the function $f_y(t)$.
- For every assertion $\varphi \in \mathcal{I}$, $\varphi(t)$ has a uniform truth value for all $t \in (t_1, t_2)$, which equals either $\varphi(t_1)$ or $\varphi(t_2)$.

The last requirement ensures that the truth value of every important assertion $\varphi \in \mathcal{I}$ is uniform throughout the interior of the evolution interval, and matches its value at one of the endpoints of the interval. In particular, it disallows a change in the truth value of φ in an internal point. It also guarantees that any value assumed by φ at internal points is also represented at one of the endpoints. This implies that φ cannot be true at both endpoints but false in the middle, nor false at both endpoints but true in the middle.

If such an evolution exists, we say that the situation $\langle s_2, t_2\rangle$ is an *activity successor* of the situation $\langle s_1, t_1\rangle$. Assuming that the differential equations satisfy some reasonable healthiness conditions, such as the Lipschitz condition, there exists at most one evolution from $\langle s_1, t_1\rangle$ to $\langle s_2, t_2\rangle$. In fact, the functions F are uniquely determined by the situation $\langle s_1, t_1\rangle$.

We denote by $\mathcal{A}(\langle s_1, t_1\rangle)$ the set of all activity successors of $\langle s_1, t_1\rangle$.

Consider, for example, a trivial phase transition system with a single (continuous) state variable x, no transitions, a single activity α given by $\alpha: \ \dot{x} = -1$, and an empty $\mathcal{I}$. Then, the following are examples of a situation and its activity successor:

$$\begin{aligned} s_1 &: \langle x: \ \ 0\,, T: 1\rangle \in \mathcal{A}(s_0: \langle x: 1\,, T: 0\rangle) \\ s_2 &: \langle x: -1\,, T: 2\rangle \in \mathcal{A}(s_1: \langle x: 0\,, T: 1\rangle) \\ s_2 &: \langle x: -1\,, T: 2\rangle \in \mathcal{A}(s_0: \langle x: 1\,, T: 0\rangle) \end{aligned}$$

Sampling Computations

A *sampling computation* of a phase transition system $\Phi: \langle V, \Theta, \mathcal{T}, \mathcal{A}, \mathcal{I}\rangle$ is an infinite sequence of situations

$$\sigma: \langle s_0, t_0\rangle,\ \langle s_1, t_1\rangle,\ \langle s_2, t_2\rangle,\ \ldots$$

satisfying:

- *Initiation:* $s_0 \models \Theta$ and $t_0 = 0$.
- *Consecution:* For each $j = 0, 1, \ldots,$
  - Either $t_j = t_{j+1}$ and $s_{j+1} \in \tau(s_j)$ for some transition $\tau \in \mathcal{T}$ — transition τ is taken at j, or
  - $\langle s_{j+1}, t_{j+1} \rangle$ is an activity successor of $\langle s_j, t_j \rangle$ (implying $t_j < t_{j+1}$) — a continuous phase takes place at step j.
- *Urgency:* No continuous step may disable an enabled transition. Thus, when some transition τ is enabled but every continuous step of positive length will cause it to become disabled, τ must be taken or disabled before any continuous step.
- *Time Divergence:* As i increases, t_i grows beyond any bound.

Example

Consider a simple phase transition system Φ_1 given by:

State Variables $V = V_c$	: $\{x\}$
Initial Condition Θ	: $x = 1$
Transitions $\mathcal{T}$	: $\{\tau\}$, where $\rho_\tau : (x = -1) \wedge (x' = 1)$
Activities $\mathcal{A}$	: $\{\alpha\}$, where $\alpha : \dot{x} = -1$
Important events $\mathcal{I}$	: $\{x = -1\}$

Fig. 1 depicts the full behavior of system Φ_1 as a function from $T \in \mathsf{R}^+$ to the value of x. Note that this is not really a function because at $T = 2$ (and all other

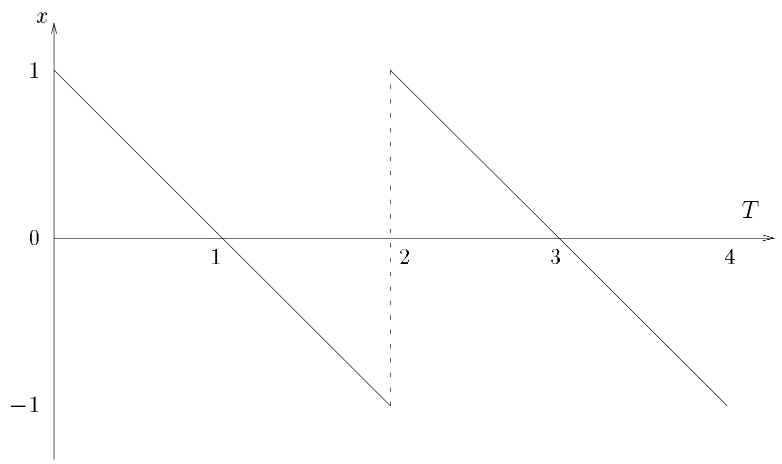

Fig. 1. Full behavior of hybrid system Φ_1.

even positive integers) x has two values: -1 which is the value it attains at the end of the continuous phase, but also $+1$ which is the result of taking transition τ at these points.

There are (uncountably) many sampling computations that correspond to this full behavior.

For example, the sampling computation

$$\begin{array}{ll}\sigma_1 : & s_0 : \langle x:1\,,\, T:0\rangle \xrightarrow{\alpha} s_1 : \langle x:-1\,,\, T:2\rangle \xrightarrow{\tau} \\ & s_2 : \langle x:1\,,\, T:2\rangle \xrightarrow{\alpha} s_3 : \langle x:-1\,,\, T:4\rangle \xrightarrow{\tau} \\ & \ldots \end{array}$$

corresponds to sampling the full behavior as shown in Fig. 2.

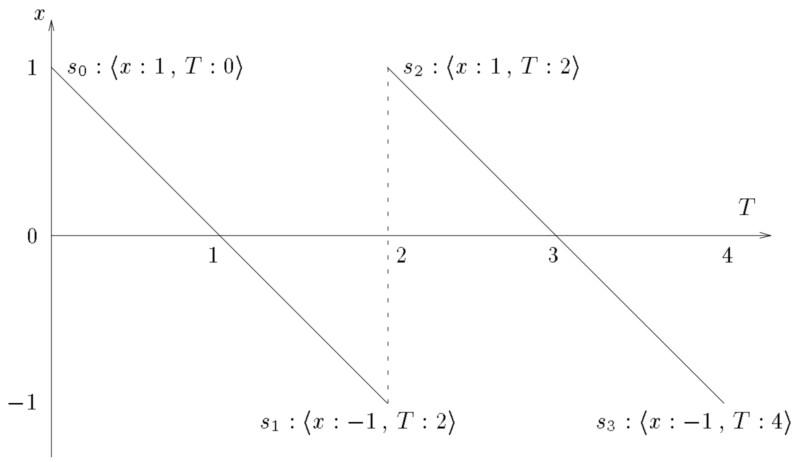

Fig. 2. Sampling computation σ_1.

A more frequent sampling leads to the computation

$$\begin{array}{ll}\sigma_2 : & s_0 : \langle x:\ \ 1\,,\, T:0\rangle \xrightarrow{\alpha} s_1 : \langle x:\ \ 0\,,\, T:1\rangle \xrightarrow{\alpha} \\ & s_2 : \langle x:-1\,,\, T:2\rangle \xrightarrow{\tau} s_3 : \langle x:\ \ 1\,,\, T:2\rangle \xrightarrow{\alpha} \\ & s_4 : \langle x:\ \ 0\,,\, T:3\rangle \xrightarrow{\alpha} s_5 : \langle x:-1\,,\, T:4\rangle \xrightarrow{\tau} \\ & \ldots \end{array}$$

whose sampling points are shown in Fig. 3.

In comparison, consider phase transition system Φ_2 which is identical to Φ_1 in all components, except for $\mathcal{I}$, which is given by

Important events $\mathcal{I}_2 : \{x = -1,\ x = 0\}$.

Thus, system Φ_2 considers the assertion $x = 0$ to be an important event in addition to $x = -1$, which is the enabling condition of transition τ. The computation

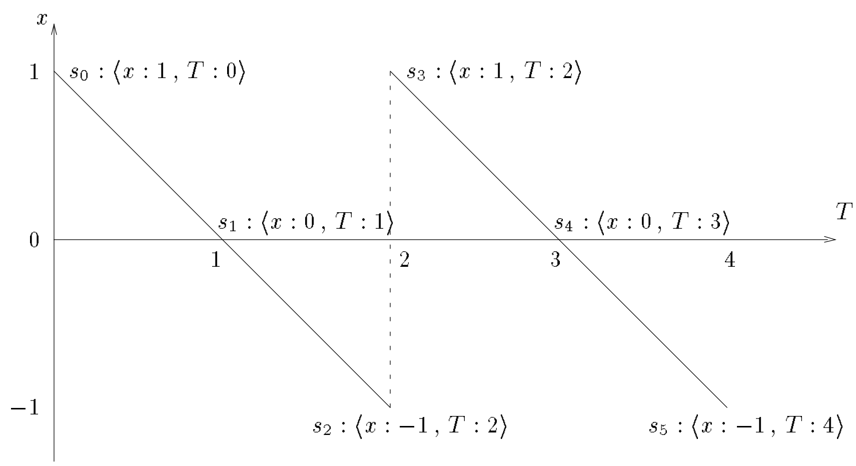

Fig. 3. Sampling computation σ_2.

σ_2 is a sampling computation of Φ_2 as well. However, the sequence σ_1 is not. Informally, this is because σ_1 fails to observe the (infinitely many) points at which x becomes 0. Formally, $s_1 : \langle x : -1\ ,\ T : 2\rangle$ is no longer an activity successor of $s_0 : \langle x : 1\ ,\ T : 0\rangle$ because the evolution from s_0 to s_1 did not respect the condition that important assertions do not change their truth value in the middle of a continuous step.

Zeno Computations and Progressive Systems

The requirement of time divergence excludes *Zeno computations* in which there are infinitely many state-changes within a finite time interval [AL92]. Unfortunately, not every phase transition system is guaranteed to have computations that satisfy all the requirements given above.

Consider, for example, the PTS Φ_3 given by:

State Variables V	: $\{x, y\}$
Initial Condition Θ	: $x = y = 0$
Transitions $\mathcal{T}$	: $\{\tau\}$, where $\rho_\tau :\ (x = 0) \wedge (y' = y + 1)$
Activities $\mathcal{A}$	: $\{\alpha\}$, where $\alpha :\ \dot{x} = 1$
Important events $\mathcal{I}$	: $\{x = 0\}$

This PTS does not have a computation. This is because, as long as $x = 0$, transition τ is enabled. Any continuous step will disable τ. Therefore, τ must be taken infinitely many times, not allowing time to progress.

A PTS is called *progressive* if it has at least one computation. From now on, we restrict our attention to progressive transition systems.

Super-Dense Computations

In addition to sampling computations, [MMP92] presents another class of computations, to which we refer here as *super-dense computations*. Sampling computations have the signature $\mathsf{N} \mapsto \Sigma \times \mathsf{R}^+$, that is, each natural number $j = 0, 1, \ldots$ is mapped to a pair consisting of a state and a real-time stamp, i.e., a situation.

In contrast, super-dense computations have the signature $\mathsf{R}^+ \times \mathsf{N} \mapsto \Sigma$, that is, each pair $\langle t, i \rangle$, where $t \in \mathsf{R}^+$ and $i \in \mathsf{N}$, is mapped to a state $s \in \Sigma$. The pair $\langle t, i \rangle$ identifies a time stamp t and a step number i. The step numbers correspond to the transitions that are taken at time instant t.

For example, the (single) super-dense computation produced by phase transition system Φ_1 is given by a function $x(r, i)$ from $\mathsf{R}^+ \times \mathsf{N}$ to R defined as

$$x(r,i) = \begin{cases} 1 & \text{for } \quad r = 0 \quad \text{and } i \geq 0 \\ 1 - r & \text{for } 0 < r < 2 \text{ and } i \geq 0 \\ -1 & \text{for } \quad r = 2 \quad \text{and } i = 0 \\ 1 & \text{for } \quad r = 2 \quad \text{and } i \geq 1 \\ 3 - r & \text{for } 2 < r < 4 \text{ and } i \geq 0 \\ -1 & \text{for } \quad r = 4 \quad \text{and } i = 0 \\ 1 & \text{for } \quad r = 4 \quad \text{and } i \geq 1 \\ & \qquad \vdots \end{cases}$$

An argument, made in [MMP92], claims that the super-dense semantics provides a more precise representation of the behavior of hybrid systems than the semantics of sampling computations. The main criticism of sampling computations complains that some important events may fail to be observed, such as the event of x becoming 0, to which computation σ_1 is oblivious.

As previously seen, this difficulty has been overcome by the introduction of the important event component. Consequently, in this paper we continue to use the sampling-computation semantics. The advantages of the sampling semantics are that it is simpler than the super-dense semantics and conforms better with sequence based verification methods.

System Description by Hybrid Statecharts

Hybrid systems can be conveniently described by an extension of statecharts [Har87] called *hybrid statecharts*. The main extension is

- States may be labeled by (unconditional) differential equations. The implication is that the activity associated with the differential equation is active precisely when the state it labels is active.

We illustrate this form of description by the example of *Cat and Mouse* taken from [MMP92]. At time $T = 0$, a mouse starts running from a certain position on the floor in a straight line towards a hole in the wall, which is at a distance X_0 from the initial position. The mouse runs at a constant velocity v_m. After a delay of Δ time units, a cat is released at the same initial position and chases the mouse at velocity v_c along the same path. Will the cat catch the mouse, or will the mouse find sanctuary while the cat crashes against the wall?

The statechart in Fig. 4 describes the possible scenarios.

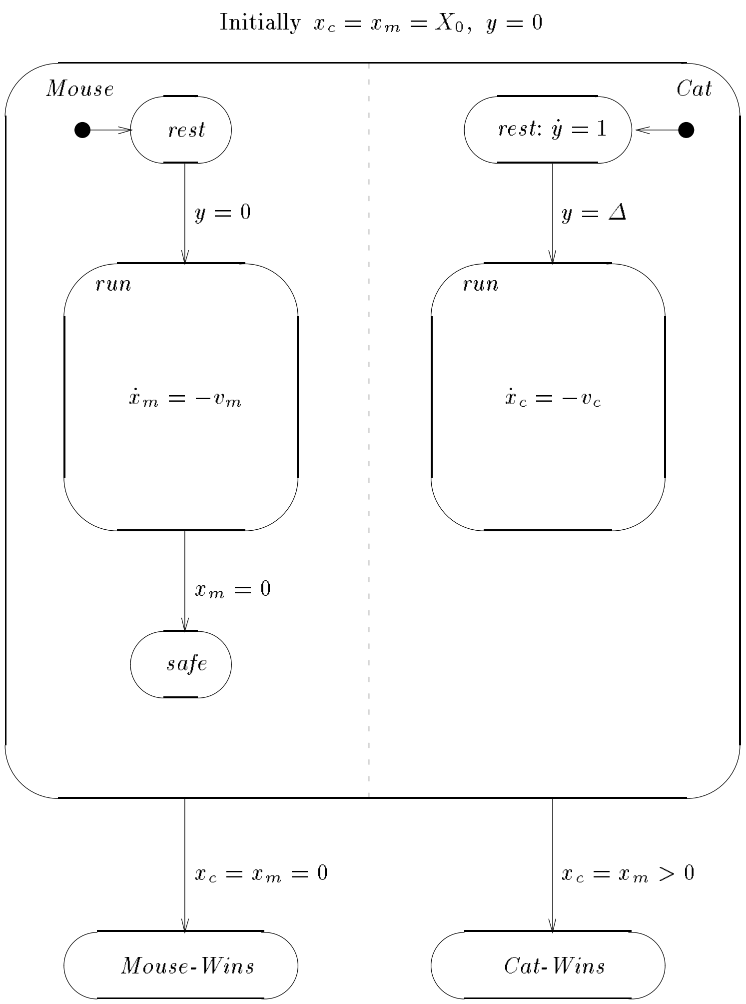

Fig. 4. Specification of Cat and Mouse.

The specification (and underlying phase transition system) uses the continuous state variables x_m and x_c, measuring the distance of the mouse and the cat, respectively, from the wall. The continuous variable y serves as a timer, measuring the waiting time of the cat before it starts running. The specification refers to the constants X_0, v_m, v_c, and Δ.

A behavior of the system starts with states $M.rest$ and $C.rest$ active, variables x_m and x_c set to the initial value X_0, and variable y set to 0. The mouse proceeds immediately to the state of running, in which its variable x_m changes continuously according to the equation $\dot{x}_m = -v_m$. The cat waits for a delay of Δ before entering its running state, using the timer variable y that increases linearly with time. There are several possible termination scenarios. If the event $x_m = 0$ happens first, the mouse reaches sanctuary and moves to state *safe*, where it waits for the cat to reach the wall. As soon as this happens, detectable by the condition $x_c = x_m = 0$ becoming true, the system moves to state *Mouse-Wins*. The other possibility is that the event $x_c = x_m > 0$ occurs first, which means that the cat overtook the mouse before the mouse reached sanctuary. In this case they both stop running and the system moves to state *Cat-Wins*. The compound conditions $x_c = x_m = 0$ and $x_c = x_m > 0$ stand for the conjunctions $x_c = x_m \wedge x_m = 0$ and $x_c = x_m \wedge x_m > 0$, respectively.

This diagram illustrates the typical interleaving between continuous activities and discrete state changes which, in this example, only involve changes of control.

The idea of using statecharts with continuous activities associated with certain states (usually basic ones) was already suggested in [Har84]. According to this suggestion, these states are associated with activities that represent physical (and therefore possibly continuous) operations and interactions with the environment.

The Underlying Phase Transition System

Following the graphical representation, we identify the phase transition system underlying the picture of Fig. 4. We refer to states in the diagram that do not enclose other states as *basic states*.

- *State Variables:* Given by $V_c = \{x_c, x_m, y\}$ and $V_d = \{\pi\}$. Variable π is a control variable whose value is a set of basic states of the statechart.
- *Initial Condition:* Given by

$$\Theta : \pi = \{M.rest, C.rest\} \ \wedge \ x_c = x_m = X_0 \ \wedge \ y = 0.$$

- *Transitions:* Listed together with the transition relations associated with them.

$$\begin{aligned}
M.rest\text{-}run &: M.rest \in \pi \ \wedge \ y = 0 \ \wedge \ \pi' = \pi - \{M.rest\} \cup \{M.run\} \\
C.rest\text{-}run &: C.rest \in \pi \ \wedge \ y = \Delta \ \wedge \ \pi' = \pi - \{C.rest\} \cup \{C.run\} \\
M.run\text{-}safe &: M.run \in \pi \ \wedge \ x_m = 0 \ \wedge \ \pi' = \pi - \{M.run\} \cup \{M.safe\} \\
M.win &: (Active \cap \pi) \neq \phi \ \wedge \ x_c = x_m = 0 \ \wedge \ \pi' = \{Mouse\text{-}Wins\} \\
C.win &: (Active \cap \pi) \neq \phi \ \wedge \ x_c = x_m > 0 \ \wedge \ \pi' = \{Cat\text{-}Wins\}
\end{aligned}$$

The set *Active* stands for the set of basic states

$$\{M.rest,\ M.run,\ M.safe,\ C.rest,\ C.run\}.$$

- *Activities:* Six activities represent the running activities of the two participants and the progress of the timer. Their equations are given by

$$
\begin{aligned}
\alpha_m^{on} &: M.run \in \pi \rightarrow \dot{x}_m = -v_m \\
\alpha_m^{off} &: M.run \notin \pi \rightarrow \dot{x}_m = 0 \\
\alpha_c^{on} &: C.run \in \pi \rightarrow \dot{x}_c = -v_c \\
\alpha_c^{off} &: C.run \notin \pi \rightarrow \dot{x}_c = 0 \\
\alpha_y^{on} &: C.rest \in \pi \rightarrow \dot{y} = 1 \\
\alpha_y^{off} &: C.rest \notin \pi \rightarrow \dot{y} = 0
\end{aligned}
$$

- *Important Events:* Given by

$$\mathcal{I} : \{x_m = 0,\ x_c = x_m,\ x_m > 0,\ y = 0,\ y = \Delta\}$$

System Description by Textual Programs

In [MP91b], we introduced a simple concurrent programming language. It is possible to extend this simple programming language to represent timed and hybrid behaviors as well. The resulting language is a subset of the language *Statext* introduced in [KP92b], and shown there to have expressive power equal to that of hybrid statecharts.

The main extensions to the language are:

- Each statement is assigned lower and upper bounds for the length of time the statement may be continuously enabled before it must be executed. Typically, these bounds may apply uniformly to all statements. For example, we may assign time bounds $[1, 5]$ to the transitions associated with all statements. These time bounds are referenced in the transition associated with the statement.
- We introduce the *preemption* statement $S_1 \ \mathsf{U}\ S_2$. The intended meaning of the *preemption* statement is
  - Execute S_1 forever or until it terminates, or
  - Execute S_1 until a first step of S_2 is taken, and then continue to execute S_2.
- Differential equations are also acceptable as statements of the extended language. A differential equation statement never terminates, and the only way to get out of it is by preemption, using the U construct. Statements consisting of differential equations are associated with activities, in contrast to all other statements which are associated with transitions. Thus, the statement

  $$\ell :\ \dot{x} = e$$

  gives rise to the activity

  $$\alpha_\ell :\quad \ell \in \pi \ \rightarrow\ \dot{x} = e.$$

  We refer to this activity as an *explicit activity* since it corresponds to an explicit statement in the program.

Besides the explicit activities, each continuous variable $x \in V_c$ also has an implicit *default activity* α_x^{off} which controls its continuous change when none of the explicit activities governing x is active. If $\ell_1, \ldots, \ell_m$ are all the statements giving rise to explicit activities for x, then its default activity is given by

$$\alpha_x^{off}: \quad \{\ell_1, \ldots, \ell_m\} \cap \pi = \phi \;\rightarrow\; \dot{x} = 0.$$

Instead of giving a full definition of the extended language, we will illustrate it on an example.

Example: Fig. 5 presents a simple program consisting of two processes communicating by the shared variable x, initially set to 0. Process P_1 keeps incrementing variable y as long as $x < 1$. Process P_2 represents a continuous component that lets x grow linearly from 0 until it reaches the value 1. At that point, statement m_1 intervenes and shuts off the continuous growth of x, terminating process P_2. Once process P_1 detects that $x = 1$, it terminates too. Note that, once $x = 1$, statement m_1 must be taken because, otherwise, x will continue to grow beyond 1, disabling the transition associated with m_1.

$$\begin{array}{c}
y\text{: } \mathbf{integer\ where}\ y = 0 \\
x\text{: } \mathbf{real\ \ where}\ x = 0 \\[1ex]
\left[\begin{array}{l}
\ell_0 : \mathbf{while}\ x < 1\ \mathbf{do} \\
\qquad \left[\ell_1 :\ y := y + 1\right] \\
\ell_2 :
\end{array}\right]
\;\Big\|\;
\left[\begin{array}{l}
[m_0 : \dot{x} = 0.2] \sqcup [m_1 : \mathbf{await}\ x = 1] \\
m_2 :
\end{array}\right] \\[1ex]
-\ P_1\ - \qquad\qquad\qquad\qquad -\ P_2\ -
\end{array}$$

Fig. 5. Program HYB-T: A Hybrid textual Program

With each of the statements of P_1, we associate the uniform time bounds $[1, 5]$.

We show how to define a phase transition system $\Phi_{\text{HYB-T}}$ corresponding to program HYB-T. To enforce the timing requirements for the statements of process P_1, we introduce a (continuous) timer variable z.

- *State Variables:* $V = V_c : \{x, z\} \cup V_d : \{\pi, y\}$.
  Variable π is a control variable ranging over subsets of the locations of program HYB-T: $\{\ell_0, \ell_1, \ell_2, m_0, m_1, m_2\}$. The value of π in a state denotes all the locations of the program in which control currently resides.
- *Initial Condition*

$$\Theta : \pi = \{\ell_0, m_0, m_1\} \;\wedge\; x = y = z = 0$$

- *Transitions:* $\mathcal{T}$: $\{\ell_0, \ell_1, m_1\}$ with transition relations:

$$\rho_{\ell_0} : \ell_0 \in \pi \;\wedge\; 1 \leq z \leq 5 \;\wedge\; \begin{pmatrix} x < 1 \;\wedge\; \pi' = \pi - \{\ell_0\} \cup \{\ell_1\} \\ \vee \\ x \geq 1 \;\wedge\; \pi' = \pi - \{\ell_0\} \cup \{\ell_2\} \end{pmatrix} \;\wedge\; z' = 0$$

$$\rho_{\ell_1} : \ell_1 \in \pi \;\wedge\; 1 \leq z \leq 5 \;\wedge\; \pi' = \pi - \{\ell_1\} \cup \{\ell_0\} \;\wedge\; y' = y + 1 \;\wedge\; z' = 0$$

$$\rho_{m_1} : m_1 \in \pi \;\wedge\; x = 1 \;\wedge\; \pi' = \pi - \{m_0, m_1\} \cup \{m_2\}$$

  As usual, a transition τ preserves the value of every variable whose primed version does not appear in the transition relation for τ. For example, transition ℓ_0 preserves the values of variables x and y.
- *Activities:* $\mathcal{A}$: $\{\alpha_x^{on}, \alpha_x^{off}, \alpha_z\}$ given by:

$$\begin{aligned} \alpha_x^{on} &: m_0 \in \pi \rightarrow \dot{x} = 0.2 \\ \alpha_x^{off} &: m_0 \notin \pi \rightarrow \dot{x} = 0 \\ \alpha_z &: \qquad\qquad\;\; \dot{z} = 1 \end{aligned}$$

- *Important Events:* $\mathcal{I}$: $\{x = 1,\; 1 \leq z,\; z \leq 5\}$.

To fully comprehend the behavior of this program, we present two possible computations of program HYB-T. In the presentation of these computations, we use the notation $\pi : \{\ell_i, m_{0,1}\}$ as abbreviation for $\pi : \{\ell_i, m_0, m_1\}$. The first computation attempts to maximize the value of y on termination.

$$\begin{aligned}
&\langle \pi : \{\ell_0, m_{0,1}\},\; x : 0.0,\; y : 0,\; z : 0,\; T : 0\rangle \xrightarrow{cont} \\
&\langle \pi : \{\ell_0, m_{0,1}\},\; x : 0.2,\; y : 0,\; z : 1,\; T : 1\rangle \xrightarrow{\ell_0} \\
&\langle \pi : \{\ell_1, m_{0,1}\},\; x : 0.2,\; y : 0,\; z : 0,\; T : 1\rangle \xrightarrow{cont} \\
&\langle \pi : \{\ell_1, m_{0,1}\},\; x : 0.4,\; y : 0,\; z : 1,\; T : 2\rangle \xrightarrow{\ell_1} \\
&\langle \pi : \{\ell_0, m_{0,1}\},\; x : 0.4,\; y : 1,\; z : 0,\; T : 2\rangle \xrightarrow{cont} \\
&\langle \pi : \{\ell_0, m_{0,1}\},\; x : 0.6,\; y : 1,\; z : 1,\; T : 3\rangle \xrightarrow{\ell_0} \\
&\langle \pi : \{\ell_1, m_{0,1}\},\; x : 0.6,\; y : 1,\; z : 0,\; T : 3\rangle \xrightarrow{cont} \\
&\langle \pi : \{\ell_1, m_{0,1}\},\; x : 0.8,\; y : 1,\; z : 1,\; T : 4\rangle \xrightarrow{\ell_1} \\
&\langle \pi : \{\ell_0, m_{0,1}\},\; x : 0.8,\; y : 2,\; z : 0,\; T : 4\rangle \xrightarrow{cont} \\
&\langle \pi : \{\ell_0, m_{0,1}\},\; x : 1.0,\; y : 2,\; z : 1,\; T : 5\rangle \xrightarrow{m_1} \\
&\langle \pi : \{\ell_0, m_2\}\;\;,\; x : 1.0,\; y : 2,\; z : 1,\; T : 5\rangle \xrightarrow{\ell_0} \\
&\langle \pi : \{\ell_2, m_2\}\;\;,\; x : 1.0,\; y : 2,\; z : 0,\; T : 5\rangle \xrightarrow{cont} \\
&\langle \pi : \{\ell_2, m_2\}\;\;,\; x : 1.0,\; y : 2,\; z : 1,\; T : 6\rangle \xrightarrow{cont} \\
&\ldots
\end{aligned}$$

The second computation attempts to minimize the value of y on termination.

$$\begin{aligned}
&\langle \pi : \{\ell_0, m_{0,1}\},\; x : 0.0,\; y : 0,\; z : 0,\; T : 0\rangle \xrightarrow{cont} \\
&\langle \pi : \{\ell_0, m_{0,1}\},\; x : 1.0,\; y : 0,\; z : 5,\; T : 5\rangle \xrightarrow{m_1} \\
&\langle \pi : \{\ell_0, m_2\}\;\;,\; x : 1.0,\; y : 0,\; z : 5,\; T : 5\rangle \xrightarrow{\ell_0} \\
&\langle \pi : \{\ell_2, m_2\}\;\;,\; x : 1.0,\; y : 0,\; z : 0,\; T : 5\rangle \xrightarrow{cont} \\
&\langle \pi : \{\ell_2, m_2\}\;\;,\; x : 1.0,\; y : 0,\; z : 1,\; T : 6\rangle \xrightarrow{cont} \\
&\ldots
\end{aligned}$$

3 A Requirement Specification Language

To specify properties of hybrid systems, we use the language of temporal logic, as presented in [MP91b], with the extension that state formulas (assertions) may refer to the variable T representing the real-time clock.

This logic has been considered under the name *Explicit Clock Temporal Logic* in [PH88], [HLP90], and [Ost90]. However, since it is always possible to introduce a continuous variable z with the activity $\dot{z} = 1$, leading to the fact that always $z = T$, one may argue that, under the hybrid context, this is no extension at all.

There have been several proposals for more radical extensions of the temporal language to express properties of real-time and hybrid systems. Some of the real-time extensions are surveyed in [AH92]. For our purposes here, classical temporal logic suffices.

We illustrate the use of temporal logic with explicit reference to the clock T to express two important timing properties of systems.

- *Bounded Response:* Every p should be followed by an occurrence of a q, not later than d time units. This property can be specified by either of the formulas:

$$p \,\wedge\, T = t_0 \;\Rightarrow\; \Diamond(q \,\wedge\, T \leq t_0 + d)$$
$$p \,\wedge\, T = t_0 \;\Rightarrow\; (T \leq t_0 + d)\, \mathcal{W}\, q$$

- *Minimal separation*: No q can occur earlier than d time units after an occurrence of p. This property can be specified by the formula:

$$p \,\wedge\, T = t_0 \;\Rightarrow\; \Box(T < t_0 + d \;\rightarrow\; \neg q)$$

The second formula for bounded response uses the operator $\mathcal{W}$ to state that time cannot progress beyond $t_0 + d$ before q occurs. While the first formula is a response formula (liveness), the second formula is a safety formula. The advantage of formulating properties by safety formulas is that their verification requires simple proof rules.

Temporal logic can also specify properties that refer to continuous variables. For example, to specify that the mouse will always escape the cat, for the system of Fig. 4, we can write the invariance formula

$$\Box(\mathit{Cat.run} \wedge (x_c = x_m) \rightarrow x_m = 0),$$

where we use names of states in a statechart as control predicates. Of course, such a property will not be valid over all computations of the Cat and Mouse system, unless some relation is established among the problem parameters: X_0, Δ, v_c, and v_m. Indeed, a sufficient condition for this property to be valid is:

$$\frac{X_0}{v_m} < \Delta + \frac{X_0}{v_c}.$$

The following axiom constrains the value of the clock T to be nonnegative.

$$\text{TIME-RANGE} : 0 \leq T$$

We may use this axiom freely in any reasoning step.

4 Verification

In this paper, we only consider verification of *invariance formulas* of the form

$$\Box\, p,$$

where p is an assertion that may refer to continuous variables as well as to the clock T.

As shown, for example, in [MP91a], such invariance formulas are proven using a very simple computational induction principle, that can be summarized as:

> To establish that φ is invariant over all computations of a PTS, show that
>
> I1. φ is implied by the initial condition Θ, and
>
> I2. φ is preserved by any step taken in the computation.

To express the preservation of assertions over steps in a computation, we introduce the notion of a *verification condition* that claims that a computation step always leads from a state satisfying assertion p to a state satisfying assertion q.

4.1 Verification Conditions for Hybrid Systems

As indicated in the definition of a hybrid computation, there are two kinds of steps that can lead from a situation $\langle s_i, t_i\rangle$ to its successor $\langle s_{i+1}, t_{i+1}\rangle$: a discrete step, caused by taking a transition, and a continuous step in which $\langle s_{i+1}, t_{i+1}\rangle$ is an activity successor of $\langle s_i, t_i\rangle$. Consequently, we have two types of verification conditions.

- Condition $\{p\}\,\tau\,\{q\}$ states that transition τ always leads from a p-state to a q-state.
- Condition $\{p\}\,\mathit{cont}\,\{q\}$ states that every continuous step always leads from a p-state to a q-state.

Transformation under Transitions

The condition $\{p\}\,\tau\,\{q\}$ is given by

$$\Big(\rho_\tau \;\wedge\; T' = T \;\wedge\; p\Big) \rightarrow q' \tag{1}$$

where ρ_τ is the transition relation corresponding to τ. The *primed version* q' of the assertion q is obtained by replacing each variable in q by its primed version. In addition to ρ_τ, which ensures that the state variables V' are related to V according to the transition relation, the verification condition requires also that time does not progress in this step.

Consider, for example, the verification condition

$$\{at\_\ell_1 \;\wedge\; T \leq 5 + z\}\;\ell_1\;\{at\_\ell_0 \;\wedge\; T \leq 10 + z\} \tag{2}$$

for program HYB-T. Here, the expressions at_ℓ_1 and at_ℓ_0 abbreviate the formulas $\ell_1 \in \pi$ and $\ell_0 \in \pi$. Expanding the definition of the verification condition, we get

$$\underbrace{\cdots \pi' = \pi - \{\ell_1\} \cup \{\ell_0\} \cdots \wedge z \leq 5 \cdots \wedge z' = 0}_{\rho_{\ell_1}} \wedge T' = T \ \wedge \ \underbrace{\cdots \wedge T \leq 5 + z}_{p}$$
$$\rightarrow \qquad \underbrace{(at\_\ell_0)' \ \wedge \ T' \leq 10 + z'}_{q'}.$$

Clearly $\pi' = \pi - \{\ell_1\} \cup \{\ell_0\}$ implies $(at\_\ell_0)'$. Since $T' = T$, we obtain from $T \leq 5 + z$ and $z \leq 5$ the inequality $T' \leq 10$. From $z' = 0$, it follows that $T' \leq 10 + z'$.

Transformation under Continuous Steps

To formulate the verification condition over continuous steps, we consider an evolution from a situation that can be described as $\langle V, T\rangle$ to the situation $\langle V', T'\rangle$, assuming that $T' > T$.

An *activity selection* is a mapping $g : V_c \mapsto \mathcal{A}$, assigning to each continuous variable $x \in V_c$ an activity $g(x)$ in its governing set. Assume that the activity selected by g for each continuous state variable $x \in V_c$ is

$$p_x^g \rightarrow \dot{x} = e^g.$$

Let $F^g = \{f_x^g(t)\}$ be a set of functions, one for each continuous variable $x \in V_c$, such that

- $f_x^g(T) = x$,
- The equation

  $$\dot{x}_x^g(t) = e^g(F^g)$$

  is satisfied in the range $t \in [T, T']$, where $e^g(F^g)$ is obtained from e^g by replacing each occurrence of $y \in V_c$ by $f_y^g(t)$.

We assume that we know how to express the functions f_x^g in a closed form, referring to x, T, and t. For example, if g selects the activity $\dot{x} = 2$ for $x \in V_c$, then $f_x^g(t)$ is given by

$$f_x^g(t) = x + 2 \cdot (t - T).$$

The Condition

With each possible activity selection function g, we associate a verification condition $\{p\}\, g\, \{q\}$, which is given by

$$\rho_{cont}^g \ \wedge \ p \rightarrow q',$$

where the relation ρ_{cont}^g stands for

$$\bigwedge_{x \in V_d} x' = x \;\wedge\; \bigwedge_{x \in V_c} x' = f_x^g(T') \;\wedge\; \bigwedge_{x \in V_c} p_x^g \;\wedge\; T' > T \;\wedge\; \bigwedge_{\tau \in \mathcal{T}} \Big(En(\tau) \rightarrow En'(\tau)\Big)$$

$$\wedge \bigwedge_{\varphi \in \mathcal{I}} \left(\begin{array}{c} (\;\;\varphi \;\vee\;\;\; \varphi') \;\wedge\; \Big(\forall t : (T < t < T') : \;\;\varphi^g(t)\Big) \\ \vee \\ (\neg\varphi \;\vee\; \neg\varphi') \;\wedge\; \Big(\forall t : (T < t < T') : \neg\varphi^g(t)\Big) \end{array} \right)$$

The first conjunct of the formula states that all discrete variables are not changed in a continuous step. The next conjunct states that the value of $f_x^g(T')$ agrees with x'. The third conjunct requires that the activation condition p_x^g holds at the start of the continuous step. Since p_x^g depends only on discrete values, it must hold throughout. The fourth conjunct requires that time progresses by a positive amount. The last conjunct in the first line ensures that the progress of time cannot disable any enabled transition, corresponding to the urgency requirement.

The next line requires that every important assertion φ has the same truth value throughout the interior of the evolution interval, and that this value agrees with the value of φ at one of the interval's boundaries, i.e., with either φ or φ'. The formula $\varphi^g(t)$ is obtained from φ by replacing each occurrence of $y \in V_c$ by $f_y^g(t)$.

Consider, for example, the condition $\{-1 \leq x \leq 1\}\, g\, \{-1 \leq x \leq 1\}$ for system Φ_1. Since there is only one activity, there is only one activity selection g, i.e., the one that selects this activity. The evolution function f_x^g is given by $f_x^g(t) = x - (t - T)$. Consequently, the verification condition is given (after some simplifications) by:

$$\underbrace{\begin{bmatrix} x' = x - (T' - T) \;\wedge\; T' > T \;\wedge \\ (x = -1) \;\rightarrow\; (x' = -1) \qquad \wedge \\ \begin{bmatrix} \cdots \wedge (\forall t : (T < t < T') : x - (t - T) = -1) \\ \vee \\ \cdots \wedge (\forall t : (T < t < T') : x - (t - T) \neq -1) \end{bmatrix} \end{bmatrix}}_{\rho^g_{cont}} \wedge \underbrace{-1 \leq x \leq 1}_{p} \quad \rightarrow \quad \underbrace{-1 \leq x' \leq 1}_{p'}$$

Clearly, since $T' > T$, it is impossible to have $x - (t - T) = -1$ for all $t \in (T, T')$. Consequently, ρ^g_{cont} implies that $x - (t - T) \neq -1$ for all $t \in (T, T')$. The case $x = -1$ is also impossible since, by the second line of ρ^g_{cont}, it would imply $x' = -1$ while, on the other hand, $x' = x - (T' - T) < x = -1$. We conclude that $x \neq -1$ which, by p, implies $x > -1$. Since $x - (t - T) \neq -1$ for all $t \in (T, T')$, $x > -1$ implies that $x - (t - T) > -1$ for all $t \in (T, T')$. Therefore, in the limit of t approaching T', $x' = x - (T' - T) \geq -1$.

For the other inequality, from $T' > T$ it follows that $x' = x-(T'-T) < x \leq 1$ and therefore $x' \leq 1$.

Finally, we define the verification condition over a continuous step to be

$$\{p\}\ cont\ \{q\}: \quad \bigwedge_{g} \{p\}\ g\ \{q\},$$

where the conjunction of the individual conditions $\{p\}\ g\ \{q\}$ is taken over all possible activity selection functions g.

4.2 The Invariance Rule

The following rule establishes the validity of an invariance formula $\square\, p$ for an assertion p over a given PTS Φ.

$$\begin{array}{lll}
\text{Rule INV} & & \\
& \text{I1. } \Theta\ \wedge\ T=0\ \rightarrow\ \varphi & \\
& \text{I2. } \{\varphi\}\ \tau\ \{\varphi\} & \text{for every } \tau \in \mathcal{T} \\
& \text{I3. } \{\varphi\}\ g\ \{\varphi\} & \text{for every activity selection } g \\
& \text{I4. } \varphi\ \rightarrow\ p & \\
\hline
& \quad \square\, p &
\end{array}$$

Premise I1 establishes that the auxiliary assertion φ is initially true. Premise I2 requires that φ is preserved by every transition. Premise I3 requires that φ is preserved by any continuous step. It follows that all situations in a computation of Φ satisfy φ. By premise I4, they also satisfy p.

To illustrate the use of this rule, consider system Φ_1. We will prove that the invariant $\square(-1 \leq x \leq 1)$ is valid over all sampling computations of this system. We use rule INV with $p = \varphi : -1 \leq x \leq 1$.

Premise I1 assumes the form

$$\underbrace{x=1}_{\Theta} \wedge T=0 \rightarrow \underbrace{-1 \leq x \leq 1}_{\varphi},$$

which is obviously valid.

Premise I2 assumes the form

$$\underbrace{\cdots \wedge (x'=1) \wedge \cdots}_{\rho_\tau} \ \wedge \ \underbrace{\cdots}_{\varphi} \rightarrow \underbrace{-1 \leq x' \leq 1}_{\varphi'}$$

which is also obviously valid.

Premise I3 requires showing that $-1 \leq x \leq 1$ is preserved under a continuous step. This has been verified above.

Premise I4 is trivial since $p = \varphi$.

Case Splitting

It often happens that the assertion φ appearing in rule INV naturally splits into a disjunction:

$$\varphi : \varphi_0 \ \lor \ \varphi_1 \ \lor \ \cdots \ \lor \ \varphi_m$$

In this case, we may use the following requirements to establish premises I1–I4 of rule INV.

D1. $\Theta \ \land \ T = 0 \ \rightarrow \ \varphi_0$.

D2. For every $i = 0, \ldots, m$ and transition τ, there exist indices $j_1, \ldots, j_k$ (depending on i and τ), such that

$$\{\varphi_i\}\, \tau \, \{\varphi_{j_1} \ \lor \ \cdots \ \lor \ \varphi_{j_k}\}.$$

We call $\varphi_{j_1}, \ldots, \varphi_{j_k}$ the τ-*successors* of φ_i.

D3. For every $i = 0, \ldots, m$ and activity selection g,

$$\{\varphi_i\}\, g \, \{\varphi_i\}.$$

D4. For every $i = 0, \ldots, m$,

$$\varphi_i \ \rightarrow \ p.$$

The *scheme* of an invariance proof under case splitting consists of the list of assertions $\varphi_0, \ldots, \varphi_m$ and the identification, for each $i = 0, \ldots, m$ and $\tau \in \mathcal{T}$, of the τ-successors of φ_i.

A scheme is defined to be *valid* with respect to an assertion p if requirements D1–D4 are satisfied.

It is obvious that a valid scheme identifies a set of verification conditions that can serve as the premises to rule INV. Indeed, we have the following claim.

Claim 1 *A scheme that is valid with respect to an assertion p establishes the validity of the formula*

$$\Box \, p$$

over all sampling computations of the given PTS Φ.

In presenting an invariance proof, we rarely list the full set of verification conditions and prove them in detail. Instead, it is often sufficient to present the scheme of the proof as convincing evidence of the proof's validity.

Consider a proof of the property

$$\Box(x < 1 \ \rightarrow \ at\_\ell_{0,1})$$

for program HYB-T, where $at\_\ell_{0,1}$ is an abbreviation for $at\_\ell_0 \lor at\_\ell_1$.

A scheme of the proof of this property can be given by the assertion list

$$\varphi_0 : at\_\ell_0 \ \land \ x \leq 1$$
$$\varphi_1 : at\_\ell_1 \ \land \ x \leq 1$$
$$\varphi_2 : x \geq 1$$

and the successor identification

	ℓ_0	ℓ_1	m_1
φ_0	φ_1, φ_2	φ_0	φ_2
φ_1	φ_1	φ_0	φ_2
φ_2	φ_2	φ_2	φ_2

It is not difficult to see that this scheme is valid with respect to $p : x < 1 \rightarrow at\_\ell_{0,1}$.

In some cases, we encounter an assertion φ_i and a transition τ such that φ_i implies that τ is disabled. An example of this is assertion φ_0 and transition ℓ_1, which is disabled on all states satisfying φ_0. In such a case, we can fill the τ-successors entry by an arbitrary set of assertions. This is because the conjunction $\rho_\tau \wedge \varphi_i$ appearing on the left-hand side of the implication in the verification condition is false and implies anything. In all of these cases, we consistently put φ_i as its own τ-successor.

Proof Diagrams

A convenient way of presenting a proof scheme is by proof diagrams.

A *proof diagram* representation of a proof scheme consists of

- A node corresponding to each assertion $\varphi_0, \ldots, \varphi_m$.
- The initial node is annotated by an entry arrow.
- For each node φ_i, transition τ, and the set of φ_i's τ-successors $\varphi_{j_1} \ldots, \varphi_{j_k}$, there is an edge connecting φ_i to each φ_{j_r}, $r = 1, \ldots, k$, labeled by τ. If φ_i is the only τ-successor of itself, the edge is not drawn.

In Fig. 6 we present a proof diagram corresponding to the scheme for the proof of $\Box(x < 1 \rightarrow at\_\ell_{0,1})$.

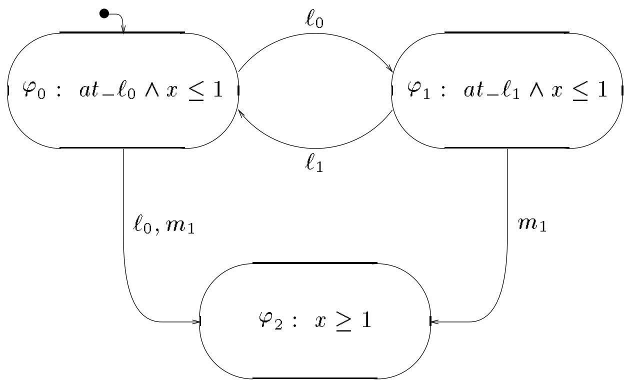

Fig. 6. A proof diagram.

A diagram is said to be *valid* with respect to assertion p if the scheme it represents is valid with respect to p.

Statechart Conventions

There are several conventions inspired by the visual language of statecharts [Har87] that improve the presentation and readability of proof diagrams. We extend the notion of a directed graph into a structured directed graph by allowing *compound nodes* that may encapsulate other nodes, and edges that may depart or arrive at compound nodes. A node that does not encapsulate other nodes is called a *basic node*.

We use the following conventions:

- Labels of compound nodes: a diagram containing a compound node n, labeled by an assertion φ and encapsulating nodes $n_1, \ldots, n_k$ with assertions $\varphi_1, \ldots, \varphi_k$, is equivalent to a diagram in which n is unlabeled and nodes $n_1, \ldots, n_k$ are labeled by $\varphi_1 \wedge \varphi, \ldots, \varphi_k \wedge \varphi$.
- Edges entering and exiting compound nodes: a diagram containing an edge e connecting node A to a compound node n encapsulating nodes $n_1, \ldots, n_k$ is equivalent to a diagram in which there is an edge connecting A to each n_i, $i = 1, \ldots, k$, with the same label as e. Similarly, an edge e connecting the compound node n to node B is the same as having a separate edge connecting each n_i, $i = 1, \ldots, k$, to B with the same label as e.

With these conventions we can redraw the proof diagram of Fig. 6 as shown in Fig. 7. Note that the common conjunct $x \leq 1$ has been factored out of nodes n_0

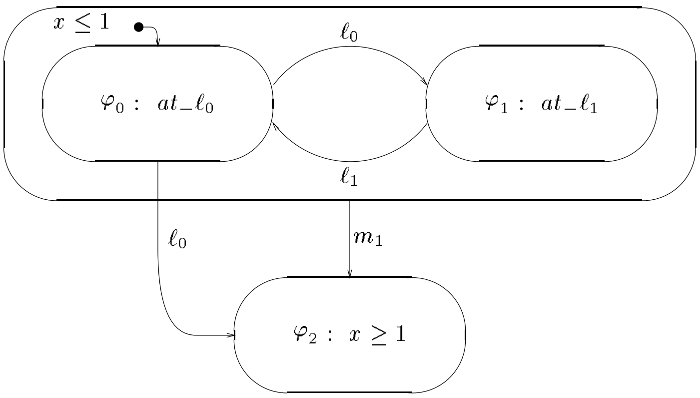

Fig. 7. A structured proof diagram.

and n_1 and now appears as the label of the compound node encapsulating them.

Termination of Program HYB-T within 15 Time Units

We will prove that program HYB-T terminates within 15 time units. This property can be expressed by the invariance

$$\square(T \leq 15 \;\vee\; (at\_\ell_2 \;\wedge\; at\_m_2)).$$

This formula states that if more than 15 time units have elapsed, then the program must have terminated.

The proof diagram presented in Fig. 8 establishes the validity of this formula over program HYB-T. The notation $at\_\ell_{0,1}$ is an abbreviation for $at\_\ell_0 \vee at\_\ell_1$.

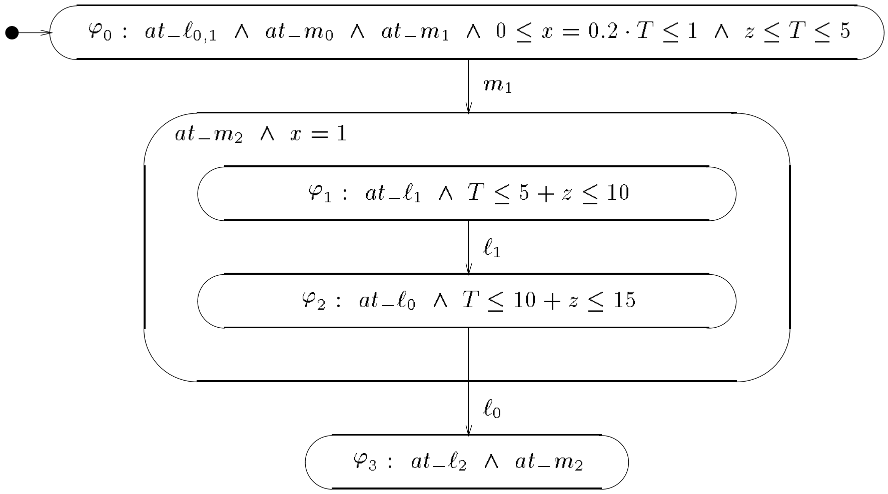

Fig. 8. A proof diagram for a timed property.

It is not difficult to show that the diagram is valid with respect to the assertion

$$T \leq 15 \;\vee\; (at\_\ell_2 \wedge at\_m_2).$$

For example, we show that assertion φ_0 is preserved under a continuous step. The only relevant activity selection function g is the one that picks α_x^{on} : $at\_m_0 \rightarrow \dot{x} = 0.2$ for x and $\alpha_z : \dot{z} = 1$ for z. For this choice of g, the evolution of x and z is given by $f_x^g(t) = x + 0.2 \cdot (t - T)$ and $f_z^g(t) = z + (t - T)$. Consequently, the appropriate verification condition (after some simplifications) is

$$\left.\begin{array}{l} \pi' = \pi \;\wedge\; x' = x + 0.2\cdot(T'-T) \;\wedge\; z' = z + (T'-T) \;\wedge\; T' > T \;\wedge\; \cdots \\ \wedge\,(x=1) \;\rightarrow (x'=1) \;\wedge\; \cdots \\ \wedge\,(\forall t : T < t < T' : x + 0.2\cdot(t-T) \neq 1) \;\wedge\; \cdots \end{array}\right\} \rho^{g}_{cont}$$

$$\wedge \quad \underbrace{at\_\ell_{0,1} \;\wedge\; at\_m_0 \;\wedge\; at\_m_1 \;\wedge\; 0 \leq x = 0.2\cdot T \leq 1 \;\wedge\; z \leq T \leq 5}_{\varphi_0}$$

$$\rightarrow$$

$$\underbrace{(at\_\ell_{0,1})' \;\wedge\; (at\_m_0)' \;\wedge\; (at\_m_1)' \;\wedge\; 0 \leq x' = 0.2\cdot T' \leq 1 \;\wedge\; z' \leq T' \leq 5}_{\varphi'_0}$$

Obviously, ρ^g_{cont} implies that $x' > x$. Therefore, $(x=1) \rightarrow (x'=1)$ implies that $x \neq 1$. It follows from φ_0 that $x < 1$. Since $f^g_x(t) = x + 0.2\cdot(t-T) \neq 1$ for all $t \in (T, T')$, it follows that $f^g_x(t) < 1$ for all $t \in (T, T')$ and, in the limit of t approaching T', $f^g_x(T') = x' \leq 1$.

We use these facts to show that ρ^g_{cont} and φ_0 imply φ'_0.

- $(at\_\ell_{0,1})' \wedge (at\_m_0)' \wedge (at\_m_1)'$ follows from $\pi' = \pi$.
- $0 \leq x' = 0.2\cdot T'$ follows from $x' = x + 0.2\cdot(T'-T)$, $T' > T$, and $0 \leq x = 0.2\cdot T$.
- Taking the limit over $(\forall t : T < t < T' : x + 0.2\cdot(t-T) < 1)$ as t tends to T', yields $x' = x + 0.2\cdot(T'-T) \leq 1$.
- $z' \leq T'$ follows from $z' = z + (T'-T)$ and $z \leq T$.
- $T' \leq 5$ follows from $x' = 0.2\cdot T' \leq 1$.

This establishes that program HYB-T terminates within 15 time units.

Proving Untimed Properties of Timed Systems

The previous example concentrated on proving timed properties, i.e., properties in which time is explicitly mentioned. Another interesting class of properties consists of properties that do not refer to time directly but whose validity over a program P is a consequence of the timing constraints satisfied by the computations of P.

For example, the property $\Box(y \leq 3)$ is valid over all computations of program HYB-T.

In Fig. 9 we present a proof diagram that is valid with respect to the assertion $y \leq 3$.

Let us present some arguments for the validity of this diagram. First, let us check that all assertions appearing in the diagram are preserved by a continuous step. The interesting cases are assertions that refer to T, x, and z. As shown above, the assertion $0 \leq x = 0.2\cdot T \leq 1$ is preserved by any continuous step taken while at_m_1 holds. Additional assertions referring to T are

$2\cdot y + z \leq T \leq 5$ while $at\_\ell_0 \wedge at\_m_0 \wedge at\_m_1$ holds,

$2\cdot y + 1 + z \leq T \leq 5$ while $at\_\ell_1 \wedge at\_m_0 \wedge at\_m_1$ holds.

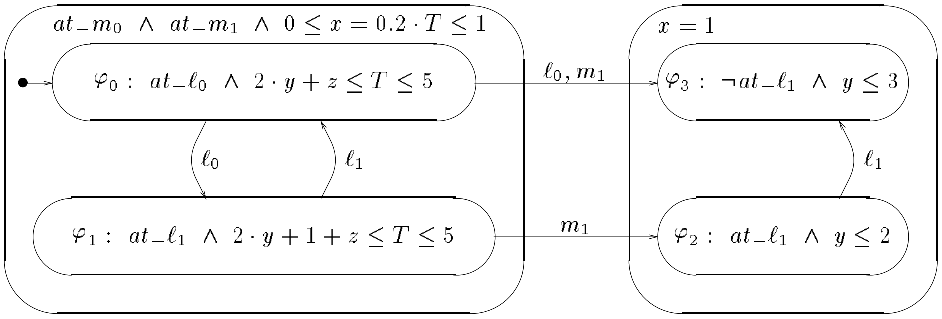

Fig. 9. A proof diagram for untimed properties.

In both cases, when time progresses from T to $T' > T$, both sides of the inequality increase by $T' - T$. The value of the clock T remains bounded by 5 as long as control is at m_1.

Next, we should check the verification conditions corresponding to the transitions labeling edges in the diagram. The verification condition corresponding to ℓ_0 is

$$\rho_{\ell_0} \;\wedge\; T' = T \;\wedge\; \underbrace{\cdots \;\wedge\; x = 0.2 \cdot T \le 1 \;\wedge\; at\_\ell_0 \;\wedge\; 2 \cdot y + z \le T \le 5}_{\varphi_0} \quad \rightarrow$$
$$\underbrace{(at\_\ell_1)' \;\wedge\; 2 \cdot y' + 1 + z' \le T' \le 5}_{\varphi_1'} \vee \underbrace{x' = 1 \;\wedge\; \neg(at\_\ell_1)' \;\wedge\; y' \le 3}_{\varphi_3'}$$

The transition relation ρ_{ℓ_0} implies $y' = y$, $z \ge 1$ and $z' = 0$. According to φ_0, $x \le 1$. We distinguish two cases: $x < 1$ and $x = 1$.

For the case $x < 1$ we show that $\rho_{\ell_0} \wedge T' = T \wedge \varphi_0$ implies φ_1'. From $T' = T \ge 2 \cdot y + z \ge 2 \cdot y + 1 = 2 \cdot y' + 1 + z'$, we obtain $2 \cdot y' + 1 + z' \le T'$. Clauses $T' = T$ and $T \le 5$ imply $T' \le 5$.

For the case $x = 1$ we show that $\rho_{\ell_0} \wedge T' = T \wedge \varphi_0$ implies φ_3'. The relation ρ_{ℓ_0} implies $x' = x$ and, since we assume $x = 1$, it follows that $x' = 1$. The relation ρ_{ℓ_0}, under $x = 1$, also implies $\neg(at\_\ell_1)'$. As ρ_{ℓ_0} implies $z \ge 1$, $2 \cdot y + z \le 5$ implies $y' = y \le 3$. Thus, all conjuncts of φ_3' are implied by the left-hand side of the implication. This establishes the verification condition for ℓ_0.

On taking transition m_1 from either φ_0 or φ_1, we have that either $2 \cdot y \le 5$ or $2 \cdot y + 1 \le 5$. In both cases, this implies $y \le 2.5 \le 3$.

It is also straightforward to show the condition

$$\Theta \wedge T = 0 \rightarrow at\_\ell_0 \wedge at\_m_0 \wedge at\_m_1 \;\wedge\; 0 \le x = 0.2 \cdot T \le 1 \;\wedge\; 2 \cdot y + z \le T \le 5,$$

since $\Theta \wedge T = 0$ implies $at\_\ell_0 \wedge at\_m_0 \wedge at\_m_1 \wedge x = y = z = T = 0$.

This shows that the diagram is valid with respect to $y \le 3$ and establishes that the formula

$$\Box(y \leq 3)$$

is valid over program HYB-T.

Verifying a Property of the Cat and Mouse System

Consider the property that, under the assumption

$$\frac{X_0}{v_m} < \Delta + \frac{X_0}{v_c} \tag{3}$$

all computations of the Cat and Mouse system satisfy

$$\Box(Cat.run \wedge (x_c = x_m) \rightarrow x_m = 0).$$

In Fig. 10, we present a proof diagram of this invariance property. In this diagram

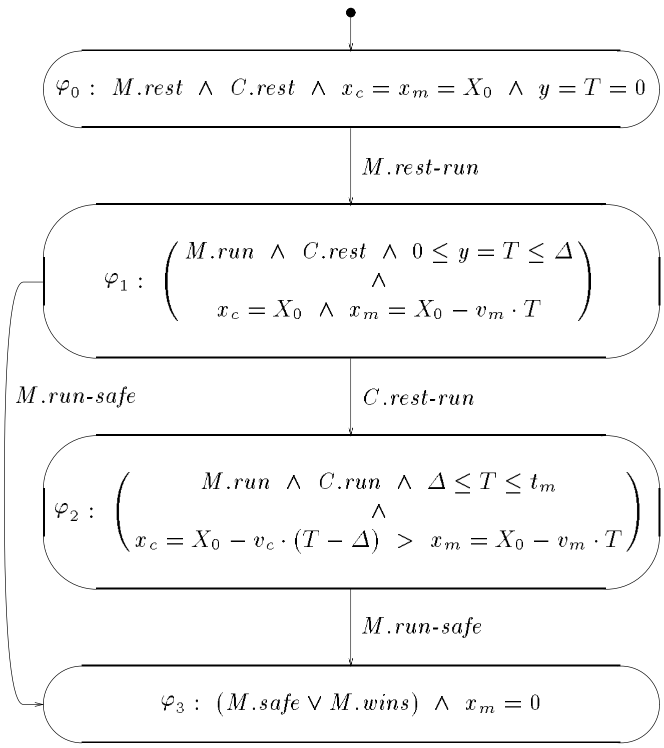

Fig. 10. A hybrid invariance proof diagram.

we use control assertions denoting that certain basic states are contained in π.

For example, $C.run$ stands for $Cat.run \in \pi$. We also use t_m for $\frac{X_0}{v_m}$, the time it takes the mouse to run the distance X_0.

It is not difficult to verify that the diagram is valid, including the preservation of all assertions under a continuous step. The only part that requires more attention is showing that the φ_2 conjunct

$$X_0 - v_c \cdot (T - \Delta) \ > \ x_m = X_0 - v_m \cdot T$$

is maintained until transition $M.run\text{-}safe$ becomes enabled, that is as long as x_m is nonnegative, which implies $T \leq t_m$. To show this, it is sufficient to show $v_c \cdot (T - \Delta) \ < \ v_m \cdot T$ which is equivalent to

$$\frac{v_m}{v_c} > 1 - \frac{\Delta}{T} \tag{4}$$

From inequality (3), we can obtain

$$\frac{v_m}{v_c} > 1 - \Delta \cdot \frac{v_m}{X_0}$$

which, using the definition of $t_m = \dfrac{X_0}{v_m}$, gives

$$\frac{v_m}{v_c} > 1 - \frac{\Delta}{t_m}. \tag{5}$$

Since $T \leq t_m$, we have $1 - \dfrac{\Delta}{t_m} \geq 1 - \dfrac{\Delta}{T}$ establishing (4).

It remains to show that

$$\underbrace{M.rest \ \wedge \ C.rest \ \wedge \ x_c = x_m = X_0 \ \wedge \ y = 0}_{\Theta} \wedge T = 0 \quad \rightarrow$$

$$\underbrace{M.rest \ \wedge \ C.rest \ \wedge \ x_c = x_m = X_0 \ \wedge \ y = T = 0}_{\varphi_0} \tag{6}$$

$$\varphi_0 \ \vee \ \cdots \ \vee \ \varphi_3 \rightarrow \Big(C.run \ \wedge \ x_c = x_m \ \rightarrow \ x_m = 0 \Big). \tag{7}$$

Implication (6) is obviously valid. To check implication (7), we observe that both φ_0 and φ_1 imply $\neg C.run$, φ_2 implies $x_c > x_m$ (under the assumption that $\Delta > 0$), and φ_3 implies $x_m = 0$.

This shows that under assumption (3), property

$$\Box(Cat.run \wedge (x_c = x_m) \rightarrow x_m = 0)$$

is valid for the Cat and Mouse system.

5 The Gas Burner Example

We conclude with an example of a Gas Burner system, presented in [CHR92]. Consider the hybrid statechart presented in Fig. 11. This statechart represents a

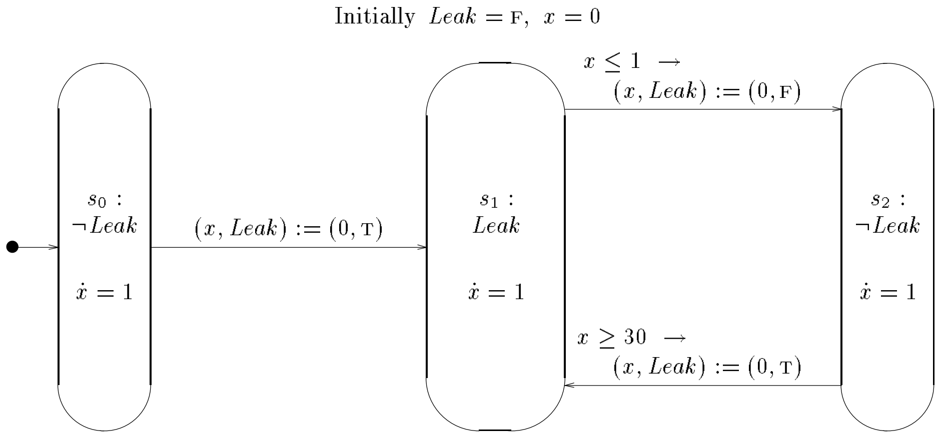

Fig. 11. GAS-BURNER: A gas burner system.

Gas Burner system that has three states: s_0, s_1, and s_2. There is a boolean state variable *Leak* whose value represents whether the system is currently leaking. For clarity, we have labeled each state with the value of *Leak* at the states. However, this labeling has no semantic meaning. The system employs a timer x to constrain the length of time that can be spent in each visit to states s_1 and s_2.

The verification problem posed in [CHR92] can be formulated as follows. Assuming

1. A continuous leaking period cannot extend beyond 1 time unit.
2. Two disjoint leaking periods are separated by a non-leaking period extending for at least 30 time units.

Prove:

- *Safety-Critical Requirement:* In any interval longer than 60, the *accumulated* leaking time is at most 5% of the interval length.

Obviously, the hybrid statechart of Fig. 11 satisfies assumptions 1 and 2. The only leaking state is s_1 and it is clear that the system cannot stay continuously in s_1 for more than 1 time unit and that, between two consecutive (but disjoint) visits to s_1, the system stays at the non-leaking state s_2 for at least 30 time units.

However, the property to be proved uses the notion of *accumulated time* in which some assertion, such as *Leak*, holds. This cannot be expressed directly in the temporal logic (TL) we use. The calculus of durations, introduced in [CHR92], has a special *duration* operator $\int p$ that measures the accumulated time p holds. Later, we will briefly consider an extension of TL which adopts the duration operator.

To handle this problem without extending the logic, we introduce additional auxiliary variables to the hybrid statechart representing the Gas Burner system. These variable measure the total time of an interval and the accumulated time in which variable *Leak* has been true. For simplicity, we consider the safety-critical requirement only for *initial intervals*, i.e., intervals starting at $T = 0$. The extension of the method to arbitrary intervals is straightforward.

Consider the hybrid statechart of Fig. 12. This hybrid statechart employs

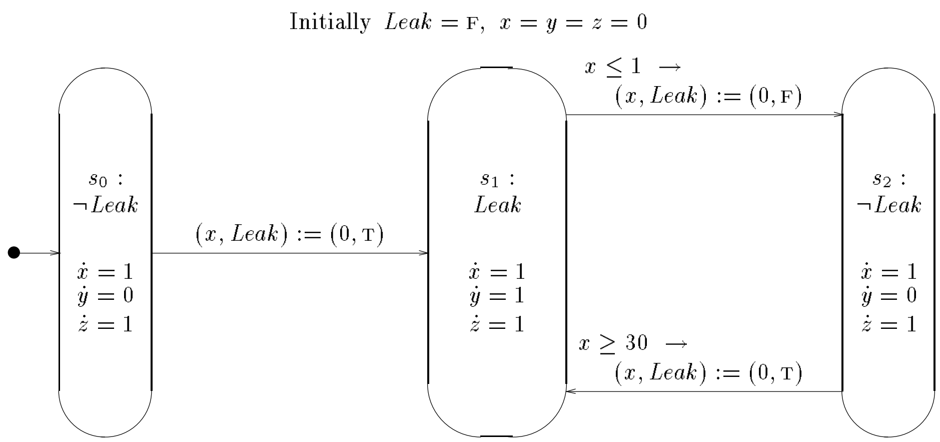

Fig. 12. H-GAS: The gas burner as a hybrid system.

three continuous variables as follows:

- As before, variable x measures the duration of time in each of the states s_0, s_1, and s_2. It is reset to 0 on entry to each of these states.
- Variable y measures the accumulated leaking time. It grows linearly in state s_2, and stays constant in any of the other states.
- Variable z measures the total elapsed time.

With these variables, we can write the requirement that the accumulated leak time does not exceed 5% of the elapsed time as $y \leq 0.05 \cdot z$ or, equivalently, as $20 \cdot y \leq z$.

Consequently, to verify that the original hybrid system of Fig. 11 maintains the safety-critical requirement for initial intervals, it is sufficient to prove that all computations of the hybrid system of Fig. 12 satisfy the invariance property

$$\Box\Big(z \geq 60 \rightarrow 20 \cdot y \leq z\Big).$$

This is the first example in which the invention of the necessary auxiliary invariants is not immediately obvious. Therefore, we will spend some time on their derivation. We try to find a relation that continuously holds between y and z and that implies the requirement

$$z \geq 60 \rightarrow 20 \cdot y \leq z. \tag{8}$$

Consider a finite prefix of a computation. Let v_1 denote the number of times the leaking state s_1 is visited in this prefix. Since on each visit variable y grows by at most 1 time unit, we have

$$y \leq v_1$$

at the end of the prefix. On the same prefix, variable z can be bounded from below by the sum of the accumulated time spent at s_1 and the accumulated time spent at s_2, ignoring the time spent at s_0 which can be arbitrarily short. The accumulated time spent at s_1 equals y. Since between two consecutive visits to s_1 the computation visits s_2, the number of visits to s_2 is at least $v_1 - 1$, and each of these visits lasts at least 30 time units. We thus obtain

$$z \geq 30 \cdot (v_1 - 1) + y \geq 31 \cdot y - 30,$$

where the last inequality is obtained by replacing v_1 by the smaller or equal value y. This leads to:

$$z \geq 31 \cdot y - 30. \tag{9}$$

We will show that this relation implies requirement (8), that is

$$z \geq 31 \cdot y - 30 \quad \rightarrow \quad (z \geq 60 \ \rightarrow \ 20 \cdot y \leq z),$$

or, equivalently,

$$z \geq 31 \cdot y - 30 \ \wedge \ z \geq 60 \quad \rightarrow \quad 20 \cdot y \leq z.$$

By $z \geq 60$, which can be written as $30 \leq \dfrac{1}{2} \cdot z$, we can replace the value 30 in $z \geq 31 \cdot y - 30$ by the bigger or equal value $\frac{1}{2} \cdot z$ and obtain

$$z \geq 31 \cdot y - \frac{1}{2} \cdot z,$$

leading to

$$\begin{aligned} \frac{3}{2} \cdot z &\geq 31 \cdot y, \\ z &\geq \frac{62}{3} \cdot y > 20 \cdot y. \end{aligned}$$

We therefore start with the assumption that the inequality $z \geq 31 \cdot y - 30$ holds at all states in the computation. Working backwards, we can identify what versions of this invariant should hold on every visit to each of the states s_0, s_1, and s_2. This leads to the proof diagram presented in Fig. 13. Transitions in this diagram are identified by the names of the states in system H-GAS from which they exit. To facilitate the reading of the diagram, edges entering a node are

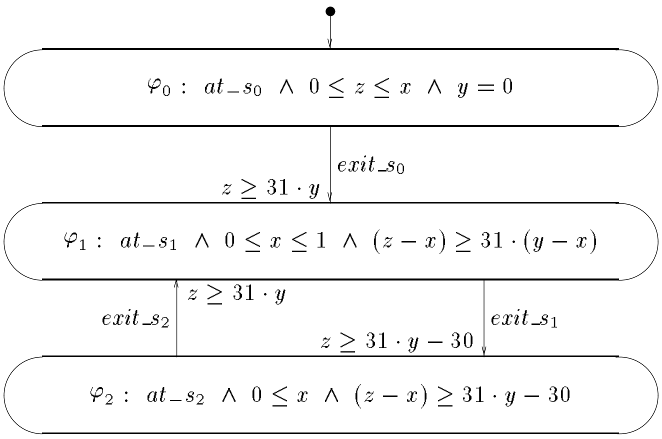

Fig. 13. An invariance proof diagram for the gas burner.

annotated by an assertion that holds whenever this node is entered. Thus, it can be shown that $z \ge 31 \cdot y$ (which is the same as $(z-x) \ge 31 \cdot (y-x)$ since $x = 0$ on entry) holds on entering the node (labeled by) φ_1 from either φ_0 or φ_2. Since x, y, and z all grow at the same linear rate within state s_1 (corresponding to node φ_1), the differences $z - x$ and $y - x$ maintain the values they had on entry. This explains why $(z-x) \ge 31 \cdot (y-x)$ is maintained within φ_1. On exit from φ_1 to φ_2, $x \le 1$, therefore, $(z-x) \ge 31 \cdot (y-x)$ implies $z \ge 31 \cdot y - 30 \cdot x \ge 31 \cdot y - 30$ on entry to φ_2. Since, within φ_2 both x and z grow at the same rate, while y remains the same, $(z-x) \ge 31 \cdot y - 30$ is maintained.

It is not difficult to see that the initial condition implies φ_0 and that each of φ_0, φ_1, or φ_2, implies $z \ge 31 \cdot y - 30$. Consequently, $z \ge 31 \cdot y - 30$ holds over all computations, establishing the validity of

$$\Box\Big(z \ge 60 \rightarrow 20 \cdot y \le z\Big).$$

To generalize this analysis to arbitrary (not necessarily initial) intervals, we can add a transition that nondeterministically resets the values of y and z to 0. This will start the measurements corresponding to an interval at an arbitrary point in time. In fact, the proof diagram of Fig. 13 is also valid for this system. It can be checked that assertions φ_0 and φ_1 in this diagram are preserved under simultaneous reset of y and z to 0.

Assertion φ_2 has to be weakened to

$$\varphi_2^* : \ at\_s_2 \ \wedge \ 0 \le x \ \wedge \ \begin{pmatrix} (z-x) \ge 31 \cdot y - 30 \\ \vee \\ z \ge 31 \cdot y = 0 \end{pmatrix}$$

Proof by an Extended Version of TL

The previous proof is based on a transformation that augments the original Gas Burner problem of Fig. 11 with the two additional variables y and z. We will now consider an alternative approach, which does not modify the given system but uses a stronger logic.

As is shown in [KP92a], it is possible to extend TL by adding to it the *duration function* $\int p$, which measures the accumulated time in which p has been true up to the present. We denote the extended logic by $\mathrm{TL}_{\int}$.

Several extensions are needed as a result of the new operator. The first extension is axiom DURATION-RANGE which bounds the range of the duration function and also relates it to the clock T.

$$\text{DURATION-RANGE} : 0 \leq \int \psi \leq T \qquad \text{for every formula } \psi$$

Since duration expressions can appear in assertions, it is also necessary to define the primed version of a duration expression $\int r$, denoted $(\int r)'$ for some assertion r. We will only consider the case that r depends on discrete variables. For this case, the value of $(\int r)'$ is given by

$$(\int r)' = \mathbf{if}\ r\ \mathbf{then} \int r + T' - T\ \mathbf{else} \int r.$$

This definition states that, if r holds at s, then the value of $\int r$ at $\langle s', t' \rangle$ is its value at $\langle s, t \rangle$ plus the time difference $t' - t$. Otherwise, it retains the same value it has at $\langle s, t \rangle$. Since r depends only on discrete variables, which are not changed by a continuous step, it is sufficient to check the value of r at $\langle s, t \rangle$.

If r depends also on continuous variables, then it should be included in the set $\mathcal{I}$, and $(\int r)'$ defined as part of the relation ρ^g_{cont}.

Using the logic $\mathrm{TL}_{\int}$, we can express the safety-critical requirement of system GAS-BURNER of Fig. 11 by the formula

$$\Box\Big(T \geq 60 \rightarrow 20 \cdot \int Leak \leq T\Big)$$

We can prove this property, using rule INV. The auxiliary invariant assertion used is inspired by the proof diagram of Fig. 13 and is given by

$$\begin{array}{l} at\_s_0 \ \wedge\ \int Leak = 0 \\ \quad \vee \\ at\_s_1 \ \wedge\ (T - x) \geq 31 \cdot (\int Leak - x) \\ \quad \vee \\ at\_s_2 \ \wedge\ (T - x) \geq 31 \cdot \int Leak - 30. \end{array}$$

This proof can also be presented as a proof diagram, resembling very much the diagram of Fig. 13. The main difference is that we replace y and z by $\int Leak$ and T, respectively.

Acknowledgement

We gratefully acknowledge the help of Eddie Chang, Yonit Kesten, and Oded Maler for their careful reading of the manuscript and thank them for many helpful suggestions.

References

[AH92] R. Alur and T. Henzinger. Logics and models of real time: A survey. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lect. Notes in Comp. Sci.*, pages 74–106. Springer-Verlag, 1992.

[AL92] M. Abadi and L. Lamport. An old-fashioned recipe for real time. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lect. Notes in Comp. Sci.*, pages 1–27. Springer-Verlag, 1992.

[CHR92] Z. Chaochen, C.A.R Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1992.

[Har84] D. Harel. Statecharts: A visual approach to complex systems. Technical report, Dept. of Applied Mathematics, Weizmann Institute of Science CS84-05, 1984.

[Har87] D. Harel. Statecharts: A visual formalism for complex systems. *Sci. Comp. Prog.*, 8:231–274, 1987.

[HLP90] E. Harel, O. Lichtenstein, and A. Pnueli. Explicit clock temporal logic. In *Proc. 5th IEEE Symp. Logic in Comp. Sci.*, pages 402–413, 1990.

[KP92a] Y. Kesten and A. Pnueli. Age before beauty. Technical report, Dept. of Applied Mathematics, Weizmann Institute of Science, 1992.

[KP92b] Y. Kesten and A. Pnueli. Timed and hybrid statecharts and their textual representation. In J. Vytopil, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 571 of *Lect. Notes in Comp. Sci.*, pages 591–619. Springer-Verlag, 1992.

[MMP92] O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lect. Notes in Comp. Sci.*, pages 447–484. Springer-Verlag, 1992.

[MP91a] Z. Manna and A. Pnueli. Completing the temporal picture. *Theor. Comp. Sci.*, 83(1):97–130, 1991.

[MP91b] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, New York, 1991.

[MP93] Z. Manna and A. Pnueli. Models for reactivity. *Acta Informatica*, 1993. To appear.

[Ost90] J.S. Ostroff. *Temporal Logic of Real-Time Systems.* Advanced Software Development Series. Research Studies Press (John Wiley & Sons), Taunton, England, 1990.

[PH88] A. Pnueli and E. Harel. Applications of temporal logic to the specification of real time systems. In M. Joseph, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 331 of *Lect. Notes in Comp. Sci.*, pages 84–98. Springer-Verlag, 1988.

[SBM92] F. B. Schneider, B. Bloom, and K. Marzullo. Putting time into proof outlines. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lect. Notes in Comp. Sci.*, pages 618–639. Springer-Verlag, 1992.