# Temporal Proof Methodologies for Real-time Systems*

Thomas A. Henzinger
Department of Computer Science
Stanford University

Zohar Manna
Department of Computer Science
Stanford University
and
Department of Applied Mathematics
The Weizmann Institute of Science

Amir Pnueli
Department of Applied Mathematics
The Weizmann Institute of Science

**Abstract.** We extend the specification language of temporal logic, the corresponding verification framework, and the underlying computational model to deal with real-time properties of concurrent and reactive systems. A global, discrete, and asynchronous clock is incorporated into the model by defining the abstract notion of a real-time transition system as a conservative extension of traditional transition systems: qualitative fairness requirements are replaced (and superseded) by quantitative lower-bound and upper-bound real-time requirements for transitions.

We show how to model real-time systems that communicate either through shared variables or by message passing, and how to represent the important real-time constructs of priorities (interrupts), scheduling, and timeouts in this framework.

Two styles for the specification of real-time properties are presented. The first style uses bounded versions of the temporal operators; the real-time requirements expressed in this style are classified into *bounded-invariance* and *bounded-response* properties. The second specification style allows explicit references to the current time through a special clock variable.

Corresponding to the two styles of specification, we present and compare two very different proof methodologies for the verification of real-time properties that are expressed in these styles. For the *bounded-operator* style, we provide a complete set of proof rules to establish lower and upper real-time bounds for response properties of real-time transition systems. For the *explicit-clock* style, we exploit the observation that when given access to the clock, every real-time property can be reformulated as a safety property, and use the standard temporal rules for establishing safety properties.

## 1 Introduction

It is self-evident that the most sensitive and critical among reactive systems, and therefore the ones for which formal approaches are needed most direly, are *real-time* systems. The qualitative requirement of *responsiveness*, that every environment stimulus $p$ must be followed by a system response $q$, is no longer adequate for real-time systems; it has to be replaced by the stronger quantitative requirement of *timed responsiveness*, which imposes a bound on the time interval that is allowed between the stimulus $p$ and the response $q$.

Over the past few years, there have been several suggestions for extending the expressive power of temporal logic, which has been used successfully to specify qual-

*This research was supported in part by an IBM graduate fellowship, by the National Science Foundation grants CCR-89-11512 and CCR-89-13641, by the Defense Advanced Research Projects Agency under contract N00039-84-C-0211, by the United States Air Force Office of Scientific Research under contract AFOSR-90-0057, and by the European Community ESPRIT Basic Research Action project 3096 (SPEC).

itative properties of reactive systems, to handle timing constraints. These attempts can be roughly classified into two approaches.

The first approach, to which we refer as the *bounded-operator* approach, introduces for each temporal operator, such as $\Diamond$, one or more time-bounded versions. For example, while the formula $\Diamond q$ states that the event $q$ will happen "eventually" but puts no time bound on when it will happen, the formula $\Diamond_{\leq 3} q$ predicts an occurrence of $q$ within 3 time units from now. This approach to the specification of timing properties has been advocated first in [KVdR83] and [Ko89], where the language is called MTL; it is analyzed for its complexity and expressiveness in [EMSS89] and [AH90].

An alternative approach to the specification of timing constraints of reactive systems introduces no new temporal operators but interprets one of the nonrigid state variables (we use the variable $\mathbf{t}$) as the current time at each state. We refer to this approach as the *explicit-clock* approach, because the only new element is the ability to refer explicitly to the clock. Scattered examples of this method to express timing properties are presented in [PdR82], [Ro84], and in [Ha88], [PH88], where it is referred to as GCTL. A more systematic exposition of this logic and its applications can be found in [Os90], where it is called RTTL.

To compare the two approaches, consider the requirement of a timed response of $q$ to $p$ within at most 3 time units. In the bounded-operator approach, this requirement is specified by the formula

$$p \;\Rightarrow\; \Diamond_{\leq 3}\, q,$$

while in the explicit-clock approach it is expressed by the formula

$$(p \wedge \mathbf{t} = T) \;\Rightarrow\; \Diamond(q \wedge \mathbf{t} \leq T+3),$$

where the rigid variable $T$ is used to record the time at the $p$-state.

The main contribution of this paper is the elaboration of two proof systems that correspond, respectively, to the two styles for the specification of timing requirements. We hope to convince the reader that proving timing properties of reactive systems is an activity that does not require significantly more complex tools than the verification of qualitative (untimed) properties.

It is a well-known observation that with the introduction of an explicit clock, all properties become *safety* properties. For example, the timed response property that, in either style, is expressed above by a *liveness*-like formula (employing the operator $\Diamond$) can alternatively be specified by a formula that uses the clock variable $\mathbf{t}$ and the safety operator $\mathsf{U}$ (*unless*):

$$(p \wedge \mathbf{t} = T) \;\Rightarrow\; (\mathbf{t} \leq T+3)\,\mathsf{U}\, q.$$

This formula states that if $p$ happens at time $T$, then from this point on, the time will not exceed $T+3$ either forever (which is ruled out by an axiom that requires the time to progress eventually) or until $q$ happens. It follows that $q$ must occur within 3 time units from $p$.

Consequently, no new proof rules are necessary for the explicit-clock style of timed specification; all properties can, in principle, be verified using a standard, uniform set of timeless rules.

On the other hand, when pursuing the bounded-operator style of specification, one discerns a clear dichotomy between *upper-bound* properties such as the bounded-response formula $p \Rightarrow \Diamond_{\leq 3}\, q$ considered above, and *lower-bound* properties, such as the bounded-invariance formula

$$p \;\Rightarrow\; \Box_{<3}\, \neg q,$$

which states that $q$ cannot happen sooner than 3 time units after any occurrence of $p$. Upper-bound properties assert that something good will happen within a specified amount of time, while lower-bound properties assert that nothing bad will happen for a certain amount of time.

Clearly, the class of upper-bound properties bear a close resemblance to *liveness* properties, while the class of lower-bound properties closely resemble *safety* properties. Consequently, the proof system we present cultivates this similarity by including separate proof principles for the classes of lower- and upper-bound properties. These proof principles can easily be seen to be natural extensions of the proof rules for the untimed safety and liveness classes, respectively. The proof system, which consists of a small number of basic rules for either class, is shown to be (relative) complete.

In our model, we assume a global, discrete, and asynchronous clock, whose actions (clock ticks) are interleaved with other system actions ([HMP90]). In some other work aimed at the formal analysis of real-time systems, it has been claimed that while this *interleaving* model of computation may be adequate for the qualitative analysis of reactive systems, it is inappropriate for the real-time analysis of programs, and a more realistic model, such as *maximal parallelism* or even *continuous time*, is needed. One of the points that we demonstrate in this paper is a refutation of this claim. We show that by a careful incorporation of time into the interleaving model, we can still model adequately most of the phenomena that occur in the timed execution of programs, and yet retain the important economic advantage of interleaving models, namely, that any point only one transition can occur and has to be analyzed.

However, when time is of the essence, we can no longer ignore the difference between multiprocessing, where each parallel task is executed on a separate machine, and multiprogramming, where several tasks reside on the same machine. This is because questions of interrupts, scheduling of tasks, and priorities may strongly influence the ability of the system to meet its timing constraints. Consequently, any suggestion for a formal analysis of real-time systems should demonstrate its ability to conveniently and naturally represent these constructs and reason about them. We demonstrate this ability for our approach and include, in the full paper, the verification of a real-time property that depends on a particular scheduling policy.

In Section 2, we define the abstract computational model of a real-time transition system and illustrate how a wide variety of concrete real-time systems and real-time phenomena can be mapped into this model. Section 3 introduces the bounded-operator specification language, and Section 4 presents a proof system for this language. In Section 5, we discuss the alternative, explicit-clock, approach. Section 6 concludes by giving completeness results for both methods.

# 2 Computational Model

We define the semantics of real-time systems as sets of timed behaviors. This is done in two steps. First, we introduce the *abstract* notion of a real-time transition system and identify the possible timed behaviors (computations) of any such system. Then, we consider the *concrete* models of shared-variables, multiprogramming, and message-passing real-time systems and show how to interpret the concrete constructs within the abstract model.

## 2.1 Abstract model: Real-time transition system

The basic computational model we use is that of a *transition system* ([MP89]), which we generalize by adding real-time constraints. We classify the real-time constraints into two categories: *lower-* and *upper-bound* requirements. They assure that transitions are taken neither too early nor too late, respectively.

A *real-time transition system* $S = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$ consists of the following components:

- a finite set $V$ of *variables*, including the boolean variable *first*.
- a set $\Sigma$ of *states*. Every state $\sigma \in \Sigma$ is an interpretation of $V$; that is, it assigns to every variable $x \in V$ a value $\sigma(x)$ in its domain.
- a set $\Theta \subseteq \Sigma$ of *initial states*. We require that $\sigma(\mathit{first}) = \mathit{true}$ for all initial states $\sigma \in \Theta$.
- a finite set $\mathcal{T}$ of *transitions*, including the idle transition $\tau_I$. Every transition $\tau \in \mathcal{T}$ is a binary accessibility relation on $\Sigma$; that is, it defines for every state $\sigma \in \Sigma$ a (possibly empty) set of $\tau$-successors $\tau(\sigma) \subseteq \Sigma$. We require that $\sigma'(\mathit{first}) = \mathit{false}$ for all $\tau$-successors $\sigma' \in \tau(\sigma)$ of every state $\sigma$.

  The transition $\tau$ is *enabled* on $\sigma$ iff $\tau(\sigma) \neq \emptyset$; a set $T$ of transitions is enabled on $\sigma$ iff some transition in $T$ is enabled on $\sigma$. The *idle* (stutter) transition

  $$\tau_I = \{(\sigma, \sigma_{[\mathit{first}:=\mathit{false}]}) : \sigma \in \Sigma\}$$

  is enabled on every state.[1]
- a *minimal delay* $l_\tau \in \mathsf{N}$ for every transition $\tau \in \mathcal{T}$. In particular, $l_{\tau_I} = 0$.
- a *maximal delay* $u_\tau \in \mathsf{N}^\infty$ for every transition $\tau \in \mathcal{T}$.[2] We require that $u_\tau \geq l_\tau$ for all $\tau \in \mathcal{T}$, and $u_{\tau_I} = \infty$.

A *timed state sequence* $\rho = (\sigma, \mathsf{T})$ consists of an infinite sequence $\sigma$ of states $\sigma_i \in \Sigma$, $i \geq 0$, and an infinite sequence $\mathsf{T}$ of corresponding times $\mathsf{T}_i \in \mathsf{N}$, $i \geq 0$, that satisfy the following conditions:

- [*Initiality*] $\mathsf{T}_0 = 0$; that is, the time of the initial state is 0.
- [*Bounded monotonicity*] For all $i \geq 0$,

  either $\mathsf{T}_{i+1} = \mathsf{T}_i$,
  or $\mathsf{T}_{i+1} = \mathsf{T}_i + 1$ and $\sigma_{i+1} = \sigma_i$;

  that is, the time never decreases. It may increase, by at most 1, only between two consecutive states that are identical. The case that the time stays the same between two identical states is referred to as a *stuttering* step; the case that the time increases is called a *clock tick*.
- [*Progress*] For all $i \geq 0$ there is some $j > i$ such that $\mathsf{T}_i < \mathsf{T}_j$; that is, the time never stagnates. Thus there are infinitely many clock ticks.

The timed state sequence $\rho = (\sigma, \mathsf{T})$ is a *computation* (run) of the real-time transition system $S = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$ iff it satisfies the following requirements:

- [*Initiality*] $\sigma_0 \in \Theta$.

[1] By $\sigma_{[x:=d]}$ we denote the state $\sigma'$ that differs from $\sigma$ at most in the interpretation of the variable $x$, to which $\sigma'$ assigns the value $d$.

[2] Let $\mathsf{N}^\infty = \mathsf{N} \cup \{\infty\}$. For notational convenience, we assume that $\infty \geq n$ for all $n \in \mathsf{N}^\infty$.

- [*Consecution*] For all $i \geq 0$ there is a transition $\tau \in \mathcal{T}$ such that $\sigma_{i+1} \in \tau(\sigma_i)$. We say that $\tau$ is *taken* at position $i$; a set $T$ of transitions is taken at position $i$ iff some transition in $T$ is taken at $i$. At both stuttering steps and clock ticks, the idle transition $\tau_I$ is taken.

- [*Lower bound*] For every transition $\tau \in \mathcal{T}$ and all positions $i \geq 0$ and $j \geq i$ such that $\mathsf{T}_j < \mathsf{T}_i + l_\tau$,

  if $\tau$ is taken at position $j$,
  then $\tau$ is enabled on $\sigma_i$.

  In other words, once enabled, $\tau$ is delayed for at least $l_\tau$ clock ticks; it can be taken only after being continuously enabled for $l_\tau$ time units.

- [*Upper bound*] For every transition $\tau \in \mathcal{T}$ and position $i \geq 0$, there is some position $j \geq i$ with $\mathsf{T}_j \leq \mathsf{T}_i + u_\tau$ such that

  either $\tau$ is not enabled on $\sigma_j$,
  or $\tau$ is taken at $j$.

  In other words, once enabled, $\tau$ is delayed for at most $u_\tau$ clock ticks; it cannot be continuously enabled for more than $u_\tau$ time units without being taken.

Note that we consider all computations of the system $S$ to be infinite; finite (terminating as well as deadlocking) computations can be represented by infinite extensions that add only idle transitions $\tau_I$. Also observe that while any minimal delay 0 can be discarded without changing the computations of $S$, every maximal delay $\infty$ adds to $S$ a *weak-fairness* assumption (in the sense of [MP89]).

## 2.2 Concrete model: Shared variables

The first kind of concrete real-time systems we consider consist of a fixed number of sequential programs that are executed in parallel, on separate processors, and communicate through a shared memory.

A *shared-variables multiprocessing system* $P$ has the form $\{\theta\}[P_1 \| \ldots \| P_m]$. Each *process* $P_i$, $1 \leq i \leq m$, is a sequential nondeterministic real-time program over the finite set $U_i$ of *private* (local) *data variables*, and the finite set $U_s$ of *shared data variables*. The formula $\theta$, called the *data precondition* of $P$, restricts the initial values of the variables in $U = U_s \cup \bigcup_{1 \leq i \leq m} U_i$.

The real-time programs $P_i$ can be alternatively presented in a textual programming language, or as transition diagrams. We shall use the latter, graphical, representation.

A *transition diagram* for the process $P_i$ is a finite directed graph whose vertices $L_i = \{\ell^i_0, \ldots \ell^i_{n_i}\}$ are called *locations*; $\ell^i_0$ is considered to be the *entry* location:

$$\longrightarrow \textcircled{\ell^i_0}$$

The intended meaning of the entry location is that the control of the process $P_i$ starts at the location $\ell^i_0$ at time 0.

Each edge in the graph is labeled by a guarded instruction, a *minimal delay* $l \in \mathsf{N}$ and a *maximal delay* $u \in \mathsf{N}^\infty$ such that $u \geq l$:

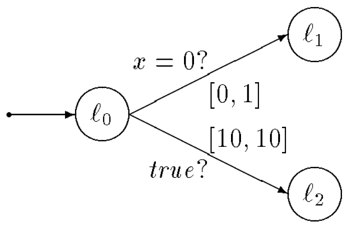
$$\ell^i_j \xrightarrow[{[l,u]}]{c \;\rightarrow\; x := e} \ell^i_k$$

where $c$ is a boolean expression, $x$ a variable, and $e$ an expression (the guard *true* and the delay interval $[0, \infty]$ are usually suppressed; the instruction $c \;\rightarrow\; x := x$ is often abbreviated to $c$?). We require that every cycle in the graph consists of no fewer than two edges, at least one of which is labeled by a positive (nonzero) maximal delay.

The intended operational meaning of the given edge is as follows. The minimal delay $l$ guarantees that whenever the control of the process $P_i$ has resided at the location $\ell^i_j$ for at least $l$ time units during which the guard $c$ has been continuously true, then $P_i$ *may* proceed to the location $\ell^i_k$. The maximal delay $u$ ensures that whenever the control of the process $P_i$ has resided at $\ell^i_j$ for $u$ time units during which the guard $c$ has been continuously true, then $P_i$ *must* proceed to $\ell^i_k$. In doing so, the control of $P_i$ moves to the location $\ell^i_k$ "instantaneously," and the current value of $e$ is assigned to $x$.

In general, a process may have to proceed via several edges all of whose guards have been continuously true for their corresponding maximal delays. In this case, any such edge is chosen nondeterministically.

To demonstrate the scope of this model, we show how the typical real-time application of a *timeout* situation can be represented. Consider the process $P$ with the following transition diagram:




When at the location $\ell_0$, the process $P$ attempts to proceed to the location $\ell_1$ for 10 time units, by checking the value of $x$ at least once every time unit. If the value of $x$ is different from 0 at least once every time unit, then $P$ may not succeed and has to proceed to the alternative location $\ell_2$ at time 10.

This operational view of the concrete model can be captured by a simple translation. With the given shared-variables multiprocessing system $P$, we associate the following real-time transition system $S_P = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$:

- $V = U \cup \{\pi_1, \ldots \pi_m, \mathit{first}\}$. Each *control variable* $\pi_i$, $1 \leq i \leq m$, ranges over the locations $L_i$ of the corresponding process $P_i$.

- $\Sigma$ contains all interpretations of $V$.

- $\Theta = \{\sigma \in \Sigma : \sigma \models \theta,\ \sigma(\pi_i) = \ell_0^i$ for all $1 \leq i \leq m$, and $\sigma(\mathit{first}) = \mathit{true}\}$.

- $\mathcal{T}$ contains, in addition to $\tau_I$, a transition $\tau_E$ for every edge $E$ in the transition diagrams for $P_1, \ldots P_m$. If $E$ connects the source location $\ell_j^i$ to the target location $\ell_k^i$ and is labeled by the instruction $c \rightarrow x := e$, then $\sigma' \in \tau_E(\sigma)$ iff

  - $\sigma(\pi_i) = \ell_j^i$ and $\sigma'(\pi_i) = \ell_k^i$,
  - $\sigma \models c$, $\sigma'(x) = \sigma(e)$, $\sigma'(\mathit{first}) = \mathit{false}$, and
  - $\sigma'(y) = \sigma(y)$ for all $y \in V - \{\pi_i, x, \mathit{first}\}$.

  If $\tau_E$ is uniquely determined by its source and target locations, we often write $\tau_{j \rightarrow k}^i$.

- For every edge $E$ labeled by the minimal delay $l$, let $l_{\tau_E} = l$.

- For every edge $E$ labeled by the maximal delay $u$, let $u_{\tau_E} = u$.

This translation defines the set of possible computations of the concrete system $P$ as a set of timed state sequences.

We remark that the translation is conservative over the untimed case. Suppose that the system $P$ contains no delay labels (recall that, in this case, all minimal delays are 0 and all maximal delays are $\infty$). Then the state components of the computations of $S_P$ are precisely the legal execution sequences of $P$, as defined in the interleaving model of concurrency, that are weakly fair with respect to every transition ([MP89]). Weak fairness (progress) for every individual transition and, thus, for every process is guaranteed by the maximal delays $\infty$: no transition can be continuously enabled without being taken.

## 2.3 Concrete model: Priorities and scheduling

Next, we model real-time systems that consist of a fixed number of sequential programs that are executed on a *single* processor.

Let us motivate the need for real-time requirements in multiprogramming environments by describing a typical application. Suppose that the process $P_1$ responds to an external event $e_1$ and the process $P_2$ responds to the event $e_2$, and that the response to $e_1$ is more urgent.

In the simplest case, every process is assigned a static *priority*; that is, we assign a higher priority to $P_1$ than to $P_2$. Then, whenever $P_2$ is executed and $e_1$ happens, $P_2$ is interrupted and $P_1$ responds to $e_1$, after which $P_2$ resumes. Given the frequencies with which $e_1$ and $e_2$ occur as well as upper bounds on the times that the processes take to respond if they are not interrupted, a typical real-time analysis of the system infers upper bounds on the response times.

In a more general setting, priorities are assigned to regions of processes. This is to disallow certain critical regions to be interrupted by other processes.

Priorities can be incoropated into the shared-variables model by restricting the enableness of transitions: a transition is only enabled if no transition with a higher priority is enabled. A formal treatment is given in the full paper.

If a more flexible scheduling policy than static priorities is required, we may add an additional process, the *scheduler*, which has the highest priority and determines which process is executed at any given time. We model scheduling transitions such as *interrupt* and *resume* by a shared variable, whose value determines the process whose transitions are currently enabled. Details are again deferred to the full paper.

## 2.4 Concrete model: Message passing

In the full paper, we show how synchronous (CSP-style) message passing can be modeled by real-time transition systems. For this purpose, we need to extend the abstract model by fair-selection requirements, which ensure that a communication transition is taken within a finite number of opportunities.

A *fair-selection* requirement $(T_1, T_2, n)$ consists of two sets $T_1 \subseteq T_2 \subseteq \mathcal{T}$ of transitions and a counter $n \in \mathbb{N}^\infty$. A timed state sequence $\rho = (\sigma, \mathsf{T})$ satisfies the requirement $(T_1, T_2, n)$ iff

> if $T_2$ is taken at $n$ positions $i_1 < \ldots < i_n$
> and $T_1$ is enabled on all $\sigma_j$, $i_1 \leq j \leq i_n$, at which $T_2$ is taken,
> then $T_1$ is taken at some $i_1 \leq j \leq i_n$.

Fair-selection requirements are finitary versions of *strong-fairness* assumptions (in the sense of [MP89]): the requirement $(T_1, T_2, \infty)$ states that $T_1$ cannot be enabled infinitely often without being taken (given an appropriate choice of $T_2$).

# 3 Specification Language

Having settled on our computational model, we need a sufficiently expressive language that is interpreted over timed state sequences in order to specify real-time systems. We distinguish between *state* formulas, which assert properties of individual states of a computation, and *temporal* formulas, which assert properties of entire computations.

## 3.1 State formulas

Given a real-time transition system $S = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$, we assume a first-order language with equality that contains interpreted function and predicate symbols to express operations and relations on the domains of the variables in $V$. Formulas of this language are interpreted over the states in $\Sigma$, and called *state formulas.* If the state formula $p$ is true in state $\sigma$, we say that $\sigma$ is a $p$-state.

We use the following abbreviations for state formulas:

- The *starting* condition *start* holds precisely in the initial states $\Theta$.
- For any transition $\tau \in \mathcal{T}$ and state formulas $p$ and $q$, the *verification* condition $\{p\}\tau\{q\}$ asserts that if $p$ is true of a state $\sigma \in \Sigma$, then $q$ is true of all $\tau$-successors of $\sigma$. For any set $T \subseteq \mathcal{T}$ of transitions, we write $\{p\}T\{q\}$ for the conjunction $\bigwedge_{\tau \in T}\{p\}\tau\{q\}$ of all individual verification conditions.
- For any transition $\tau \in \mathcal{T}$, the *enabling* condition $enabled(\tau)$ asserts that $\tau$ is enabled. In particular, $enabled(\tau_I)$ is equivalent to *true.*

For the case that the real-time transition system $S$ is associated with a shared-variables multiprocessing system $P$, it is easy to see that the starting condition, verification conditions, and enabling conditions can indeed be expressed by state formulas. Suppose that $P$ consists of $m$ processes $P_i$, $1 \leq i \leq m$. Let $at\_\ell^i_j$ stand for $\pi_i = \ell^i_j$; that is, the control of the process $P_i$ is at the location $\ell^i_j$.

If $\theta$ is the data precondition of $P$, then the starting condition *start* is equivalent to

$$\theta \wedge \left( \bigwedge_{1 \leq i \leq m} at\_\ell^i_0 \right) \wedge \mathit{first}.$$

Let $\tau \in \mathcal{T}$ be a transition of $S$, and $E$ the corresponding edge in the transition diagram for $P$; assume that $E$ connects the location $\ell^i_j$ to the location $\ell^i_k$ and is labeled by the instruction $c \;\rightarrow\; x := e$. Then, the enabling condition $enabled(\tau)$ is equivalent to

$$at\_\ell^i_j \wedge c,$$

and the verification condition $\{p\}\tau\{q\}$ is equivalent to

$$\left( \begin{array}{l} p \wedge enabled(\tau) \wedge (at\_\ell^i_k)' \wedge \neg \mathit{first}' \wedge \\ (x' = e) \wedge \bigwedge_{y \in V - \{\pi_i, x, \mathit{first}\}} (y' = y) \end{array} \right) \rightarrow q',$$

where $q'$ is obtained from $q$ by replacing every variable with its primed version (for example, $(at\_\ell^i_k)'$ stands for $\pi'_i = \ell^i_k$).

## 3.2 Temporal formulas

*Temporal formulas* are constructed from state formulas by boolean connectives and bounded temporal operators; they are interpreted over timed state sequences. In this paper, we are mostly interested in proving two important classes of real-time properties — bounded-response and bounded-invariance properties. Thus we restrict ourselves to three kinds of temporal formulas.

- A *bounded-response* property asserts that something will happen within a certain amount of time. A typical application of bounded response is to state an upper bound $u$ on the termination of a system $S$: if started at time 0, then $S$ is guaranteed to reach a final state no later than at time $u$.

  Formally, we express bounded-response properties by temporal formulas of the form

  $$p \Rightarrow \Diamond_{\leq u}\, q,$$

  for state formulas $p$ and $q$ and $u \in \mathsf{N}$. The formula $p \Rightarrow \Diamond_{\leq u}\, q$ is true over the timed state sequence $\rho = (\sigma, \mathsf{T})$ iff, for all $i \geq 0$,

  > if $\sigma_i$ is a $p$-state,
  > then there is some $q$-state $\sigma_j$, $j \geq i$, such
  > that $\mathsf{T}_j \leq \mathsf{T}_i + u$;

  that is, every $p$-state is followed by a $q$-state within time $u$.

- A *bounded-invariance* property asserts that something will hold continuously for a certain amount of time; it is often used to specify that something will not happen for a certain amount of time. A typical application of bounded invariance is to state a lower bound $l$ on the termination of a system $S$: if started at time 0, then $S$ will not reach a final state before time $l$.

  Formally, we express bounded-invariance properties by temporal formulas of the form

  $$p \Rightarrow \Box_{<l}\, q,$$

  for state formulas $p$ and $q$ and $l \in \mathsf{N}$. The formula $p \Rightarrow \Box_{<l}\, q$ is true over the timed state sequence $\rho = (\sigma, \mathsf{T})$ iff, for all $i \geq 0$ and $j \geq i$,

  if $\sigma_i$ is a $p$-state and $\mathsf{T}_j < \mathsf{T}_i + l$,
  then $\sigma_j$ is a $q$-state;

  that is, no $p$-state is followed by a $\neg q$-state within time less than $l$.

- To prove bounded-invariance properties, we sometimes need to be able to express a stronger assertion than bounded invariance, that a $p$-state can be followed by a $\neg q$-state only if *two* conditions are met: the time difference is at least $l$, and there is an intermediate $r$-state. This *bounded-unless* property is expressed by the temporal formula

  $$p \Rightarrow q\, \mathsf{U}_{\geq l}\, r,$$

  for state formulas $p$, $q$, and $r$, and $l \in \mathsf{N}$; it is true over the timed state sequence $\rho = (\sigma, \mathsf{T})$ iff, for all $i \geq 0$,

  if $\sigma_i$ is a $p$-state,
  then either all subsequent $\sigma_j$, $j \geq i$, are $q$-states,
  or there is some $r$-state $\sigma_j$, $j \geq i$, such that $\mathsf{T}_j \geq \mathsf{T}_i + l$ and all intermediate $\sigma_k$, $i \leq k < j$, are $q$-states;

  that is, every $p$-state is followed by a (possibly infinite) sequence of $q$-states until there is an $r$-state, which cannot be closer than time $l$.

While temporal-logic aficionados will readily recognize the three classes of formulas we have introduced as time-bounded versions of conventional, composite, *invariance*, *response*, and *unless* formulas ([MP89]), for our purpose it suffices to consider them primitive (for a general addition of time-bounded operators to linear temporal logic, see [AH90]).

We say that a temporal formula is *$S$-valid* iff it is true over all computations of the real-time transition system $S$; for state formulas we do not distinguish between $S$-validity and (general) validity (i.e., truth under every interpretation). A proof rule is called *$S$-sound* iff the $S$-validity of all premises implies the $S$-validity of the conclusion.

Any $S$-sound rule can be used for verifying properties of the system $S$. Consider the following *bounded-invariance* rule **BD-INV**, which allows us to conclude the bounded-invariance formula $p \Rightarrow \Box_{<l}\, q$ from the bounded-unless formula $p \Rightarrow q\, \mathsf{U}_{\geq l}\, r$, for any state formulas $p$, $q$, and $r$:

$$\textbf{BD-INV} \qquad \frac{p \Rightarrow q\, \mathsf{U}_{\geq l}\, r}{p \Rightarrow \Box_{<l}\, q}$$

It is not hard to convince ourselves that this rule is $S$-sound for every real-time transition system $S$.

# 4 Verification by Bounded-operator Reasoning

We show how to prove that a given a real-time transition system $S = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$ satisfies its specification. In particular, we present a deductive system to establish the $S$-validity of bounded-invariance and bounded-response properties. The proof rules fall into three categories: the *single-step* rules derive real-time properties that follow from the lower- or upper-bound requirement for a single transition, while the *transitivity*, *disjunction*, and *induction* rules combine real-time properties into more complicated ones.

## 4.1 Single-step rules

The *single-step lower-bound* rule uses the minimal delay $l_\tau$ of a transition $\tau \in \mathcal{T}$ to infer a bounded-unless formula:

$$\textbf{SS-LB} \qquad \frac{\begin{array}{ll} (1) & p \rightarrow (\mathit{first} \vee \neg \mathit{enabled}(\tau)) \\ (2) & p \rightarrow \varphi \\ (3) & \{\varphi\}\, \mathcal{T} - \tau\, \{\varphi\} \\ (4) & \varphi \rightarrow (q \wedge r) \end{array}}{p \Rightarrow q\, \mathsf{U}_{\geq l_\tau}\, r}$$

By $\mathcal{T} - \tau$ we denote the set difference $\mathcal{T} - \{\tau\}$. The state formula $\varphi$ is called the *invariant* of the rule. We point out that the rule **SS-LB** derives a *temporal* (bounded-unless) formula from premises all of which are *state* formulas.

¿From the conclusion of the rule **SS-LB** we can infer the bounded-invariance property

$$p \Rightarrow \Box_{<l_\tau}\, q$$

by an application of the bounded-invariance rule **BD-INV**, independently of the state formula $r$. Thus we may choose $r$ to be *true*, in which case the premise (4) reduces to $\varphi \rightarrow q$. We shall often neglect to mention this simple proof step explicitly.

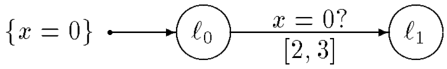
The *single-step upper-bound* rule uses the maximal delay $u_\tau$ of a transition $\tau \in \mathcal{T}$ to infer a bounded-response formula:

$$\boxed{\begin{array}{lll} \textbf{SS-UB} & (1) & p \rightarrow (\varphi \vee q) \\ & (2) & \varphi \rightarrow enabled(\tau) \\ & (3) & \{\varphi\}\, \mathcal{T} - \tau \,\{\varphi \vee q\} \\ & (4) & \{\varphi\}\, \tau \,\{q\} \\ \hline & & p \Rightarrow \Diamond_{\leq u_\tau}\, q \end{array}}$$

This rule derives a *temporal* bounded-response formula from premises all of which are *state* formulas. The state formula $\varphi$ is called the *invariant* of the rule.

The proofs that the rules **SS-LB** and **SS-UB** are $S$-sound are deferred to the full paper.

To demonstrate a typical application of the single-step rules, consider the single-process system $P$ with the data precondition $x = 0$ and the following transition diagram:




The process $P$ confirms that $x = 0$ and proceeds to the location $\ell_1$. Because of the delay interval $[2, 3]$ of the transition $\tau_{0\rightarrow 1}$, the final location $\ell_1$ cannot be reached before time 2 and must be reached by time 3.

In the full paper we carry out a formal proof of this analysis. The bounded-invariance property that $P$ does not terminate before time 2,

$$start \;\Rightarrow\; \Box_{<2}\, \neg at\_\ell_1,$$

is established by an application of the single-step lower-bound rule **SS-LB** with respect to the transition $\tau_{0\rightarrow 1}$ (let the invariant $\varphi$ be $at\_\ell_0$). The bounded-response property that $P$ terminates by time 3,

$$start \;\Rightarrow\; \Diamond_{\leq 3}\, at\_\ell_1,$$

follows from the single-step upper-bound rule **SS-UB** with respect to the transition $\tau_{0\rightarrow 1}$ (use the invariant $at\_\ell_0 \wedge x = 0$).

## 4.2 Transitivity rules

To join a finite number of successive real-time constraints into a more complicated real-time property, we introduce the transitivity rules.

The *transitive lower-bound* rule **TRANS-LB** combines two bounded-unless properties:

$$\boxed{\begin{array}{lll} \textbf{TRANS-LB} & (1) & p \Rightarrow q\, \mathsf{U}_{\geq l_1}\, r \\ & (2) & r \Rightarrow q\, \mathsf{U}_{\geq l_2}\, s \\ \hline & & p \Rightarrow q\, \mathsf{U}_{\geq l_1 + l_2}\, s \end{array}}$$

We refer to the state formula $r$ as the *link* of the rule.

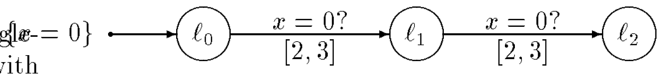
The *transitive upper-bound* rule **TRANS-UB** combines two bounded-response properties:

$$\boxed{\begin{array}{lll} \textbf{TRANS-UB} & (1) & p \Rightarrow \Diamond_{\leq u_1}\, r \\ & (2) & r \Rightarrow \Diamond_{\leq u_2}\, q \\ \hline & & p \Rightarrow \Diamond_{\leq u_1 + u_2}\, q \end{array}}$$

The state formula $r$ again the *link* of the rule.

Both transitivity rules are easily seen to be $S$-sound for every real-time transition system $S$.

We demonstrate the application of the transitivity rules by examining the single-process system $P$ with the following transition diagram:




We want to show that $P$ terminates not before time 4 and not after time 6.

First, we prove the lower bound on the termination of $P$:

$$start \;\Rightarrow\; \Box_{<4}\, \neg at\_\ell_2.$$

By the transitive lower-bound rule **TRANS-LB**, it suffices to show the premises

$$\begin{array}{lrcl} (1) & start & \Rightarrow & (\neg at\_\ell_2)\, \mathsf{U}_{\geq 2}\, at\_\ell_0, \\ (2) & at\_\ell_0 & \Rightarrow & (\neg at\_\ell_2)\, \mathsf{U}_{\geq 2}\, true. \end{array}$$

Both premises can be established by single-step lower-bound reasoning. To show the premise (1), we apply the rule **SS-LB** with respect to the transition $\tau_{0\rightarrow 1}$, using the invariant $at\_\ell_0$; the premise (2) follows from the rule **SS-LB** with respect to the transition $\tau_{1\rightarrow 2}$ and the invariant $at\_\ell_{0,1}$.

The upper bound on the termination of the system $P$ from above,

$$start \;\Rightarrow\; \Diamond_{\leq 6}\, at\_\ell_2,$$

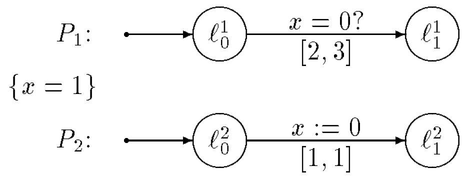
is derived by the transitive upper-bound rule **TRANS-UB**. It suffices to show the premises

$$\begin{array}{lrcl} (1) & start & \Rightarrow & \Diamond_{\leq 3}\, (at\_\ell_1 \wedge x = 0), \\ (2) & (at\_\ell_1 \wedge x = 0) & \Rightarrow & \Diamond_{\leq 3}\, at\_\ell_2, \end{array}$$

both of which can be established by single-step upper-bound reasoning (use the invariants $at\_\ell_0 \wedge x = 0$ and $at\_\ell_1 \wedge x = 0$, respectively).

Note that for lower-bound reasoning the link $at\_\ell_0$ identifies the last state *before* the transition $\tau_{0\rightarrow 1}$ is taken, while for upper-bound reasoning the link $at\_\ell_1 \wedge x = 0$ refers to the first state *after* $\tau_{0\to 1}$ is taken.

So far we have examined only single-process examples. In general, several processes that communicate through shared variables interfere with each other. Consider the two-process system with the data precondition $x = 1$ and the following transition diagrams:




The first process, $P_1$, is identical to a previous example; with a minimal delay of 2 time units and a maximal delay of 3 time units, it confirms that $x = 0$ and proceeds to the location $\ell_1^1$. However, this time the value of $x$ is not 0 from the very beginning, but set to 0 by the second process, $P_2$, only at time 1. Thus, $P_1$ can reach its final location $\ell_1^1$ no earlier than at time 3 and no later than at time 4.

For a formal proof we need the transitivity rules. The bounded-invariance property

$$start \;\Rightarrow\; \Box_{<3}\, \neg at\_\ell_1^1$$

is established by an application of the transitive lower-bound rule **TRANS-LB**. It suffices to show the premises

$$\begin{array}{ll} (1) & start \;\Rightarrow\; (\neg at\_\ell_1^1)\, \mathsf{U}_{\geq 1}\, (at\_\ell_0^1 \wedge x = 1), \\ (2) & (at\_\ell_0^1 \wedge x = 1) \;\Rightarrow\; (\neg at\_\ell_1^1)\, \mathsf{U}_{\geq 2}\, true, \end{array}$$

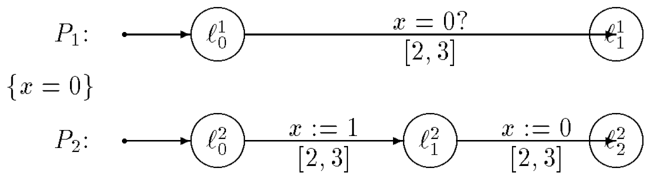
both of which follow from single-step lower-bound reasoning. Similarly, the transitive upper-bound rule **TRANS-UB** is used to show the bounded-response property

$$start \;\Rightarrow\; \Diamond_{\leq 4}\, at\_\ell_1^1$$

from the link $at\_\ell_0^1 \wedge x = 0$.

## 4.3 Disjunction rules

It turns out that the simple transitivity rules are not powerful enough to handle the nondeterminism resulting from multiple processes, which may require a case analysis. Consider the following two-process system:




In any analysis of this system, we have to distinguish two cases. The first process, $P1$, may reach its final location $\ell_1^1$ before the second process, $P2$, sets the value of $x$ to 1, or vice versa. In the latter case, $P1$ has to wait until $P2$ resets the value of $x$ to 0. Consequently, $P1$ may terminate as early as at time 2 and as late as at time 9.

In a formal proof, we have to put together overlapping real-time constraints. For this purpose we introduce the disjunction rules.

The *disjunctive lower-bound* rule **DIS-LB** combines two overlapping bounded-unless properties:

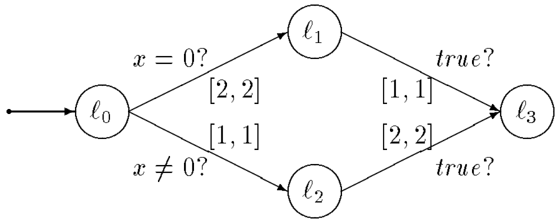
$$\boxed{\begin{array}{lll} \textbf{DIS-LB} & (1) & p_1 \Rightarrow q\, \mathsf{U}_{\geq l_1}\, r \\ & (2) & p_2 \Rightarrow q\, \mathsf{U}_{\geq l_2}\, r \\ \hline & & (p_1 \vee p_2) \;\Rightarrow\; q\, \mathsf{U}_{\geq min(l_1, l_2)}\, r \end{array}}$$

The *disjunctive upper-bound* rule **DIS-UB** combines two overlapping bounded-response properties:

$$\boxed{\begin{array}{lll} \textbf{DIS-UB} & (1) & p_1 \Rightarrow \Diamond_{\leq u_1}\, q \\ & (2) & p_2 \Rightarrow \Diamond_{\leq u_2}\, q \\ \hline & & (p_1 \vee p_2) \;\Rightarrow\; \Diamond_{\leq max(u_1, u_2)}\, q \end{array}}$$

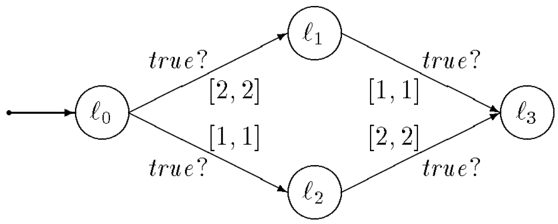
Both disjunction rules are $S$-sound for every real-time transition system $S$.

For an application of the disjunction rules, consider the process $P$ with the following transition diagram:




We show that $P$ terminates at time 3, independently of the initial value of $x$.

We remark at this point that our proof system is not strong enough to show tight bounds on *nondeterministic* systems. To see this, consider the following nondeterministic variant $P'$ of the process $P$:




During an execution of $P'$, one (if any) of the two transitions $\tau_{0\to 1}$ and $\tau_{0\to 2}$ is chosen nondeterministically (in general, a vertex of a transition diagram is *nondeterministic* iff any two guards that are associated with outgoing edges are not disjoint).

### 4.4 Induction rules

To prove lower and upper bounds on the execution time of program loops, we need to combine a state-dependent number of bounded-invariance or bounded-response properties. For this purpose we introduce two induction rules.

The *inductive lower-bound* rule **IND-LB** establishes a bounded-unless formula. Assume that the variable $i \in V$ ranges over the natural numbers $\mathsf{N}$; for any $n \in \mathsf{N}$:

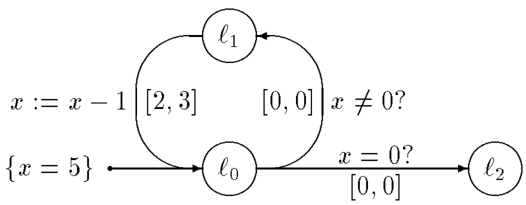
$$\begin{array}{lll} \textbf{IND-LB} & (1) & p \rightarrow \varphi(n) \\ & (2) & (\varphi(i) \wedge i > 0) \Rightarrow q \,\mathsf{U}_{\geq l}\, \varphi(i-1) \\ & (3) & \varphi(0) \rightarrow r \\ \hline & & p \Rightarrow q \,\mathsf{U}_{\geq n \cdot l}\, r \end{array}$$

By $\varphi(i-1)$ we denote the state formula that results from the inductive invariant $\varphi(i)$ by replacing all occurrences of the variable $i$ with the expression $i-1$; the formulas $\varphi(n)$ and $\varphi(0)$ are obtained analogously.

The *inductive upper-bound* rule **IND-UB** uses again a variable $i \in V$ that ranges over the natural numbers $\mathsf{N}$. For any $n \in \mathsf{N}$:

$$\begin{array}{lll} \textbf{IND-UB} & (1) & p \rightarrow \varphi(n) \\ & (2) & (\varphi(i) \wedge i > 0) \Rightarrow \Diamond_{\leq u}\, \varphi(i-1) \\ & (3) & \varphi(0) \rightarrow q \\ \hline & & p \Rightarrow \Diamond_{\leq n \cdot u}\, q \end{array}$$

In the full paper, we show both induction rules to be $S$-sound for every real-time transition system $S$, and demonstrate their application on the following single-process system $P$:



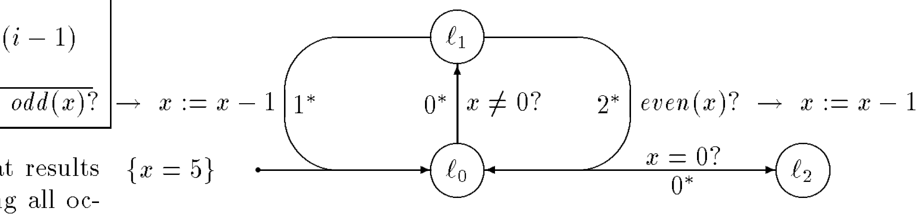

The process $P$ decrements the value of $x$ until it is 0, at which point $P$ proceeds to the location $\ell_2$. Since $x$ starts out with the value 5, and each decrement operation takes at least 2 and at most 3 time units, while the tests are instantaneous, the final location $\ell_2$ is reached not before time 10 and not after time 15. This is proved in the full paper.

The induction rules can be generalized, by letting the bounds $l$ and $u$ vary as functions of $i$. We state only the general inductive upper-bound rule, **IND-GUB**. For any $n \in \mathsf{N}$:

$$\begin{array}{lll} \textbf{IND-GUB} & (1) & p \rightarrow \varphi(n) \\ & (2) & (\varphi(i) \wedge i > 0) \Rightarrow \Diamond_{\leq u_i}\, \varphi(i-1) \\ & (3) & \varphi(0) \rightarrow q \\ \hline & & p \Rightarrow \Diamond_{\leq \Sigma_{0<i\leq n} u_i}\, q \end{array}$$

This general rule is still $S$-sound for every real-time transition system $S$.

The general inductive upper-bound rule is needed to prove upper bounds for programs with loops whose execution time is state-dependent. Consider the following process $P$:




# 5 Verification by Explicit-clock Reasoning

We point out that none of our state formulas is able to refer to the value of the time, and the only real-time references that are admitted in temporal formulas are bounded temporal operators. In this section, we investigate the consequences of extending the notion of state, by adding a variable $\mathbf{t}$ that represents, in every state, the current time.

This extension is interesting, because once we are given explicit access to the global clock through a clock variable $\mathbf{t}$, both bounded-invariance and bounded-response properties can alternatively be formulated as (unbounded) unless properties and, consequently, verified by conventional (timeless) techniques for establishing safety properties.

### 5.1 Extended real-time transition systems

Given a real-time transition system $S = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$, we introduce the following new variables:

- The *clock variable* $\mathbf{t}$ ranges over the natural numbers $\mathsf{N}$; it records, in every state $\sigma_i$ of a computation $\rho = (\sigma, \mathsf{T})$, the corresponding time $\mathsf{T}_i$.
- The *delay counters* $\delta_\tau$, $\tau \in \mathcal{T} - \{\tau_I\}$, range over $\mathsf{N}^\infty$; they record, in every state of a computation, for how many clock ticks the transition $\tau$ has been (continuously) enabled.

The *extension* $S^* = \langle V^*, \Sigma^*, \Theta^*, \mathcal{T}^*, l^*, u^* \rangle$ of $S$ is defined to be the following real-time transition system:

- $V^* = V \cup \{\mathbf{t}\} \cup \{\delta_\tau : \tau \in \mathcal{T}\}$.
- $\Sigma^*$ contains all interpretations of $V^*$.
- $\Theta^*$ contains all extensions $\sigma^*$ of interpretations $\sigma \in \Theta$ such that, for all $\tau \in \mathcal{T}$,

  $\sigma^*(\mathbf{t}) = 0$ and
  $\sigma^*(\delta_\tau) = 0$.

- $\mathcal{T}^*$ contains, for every $\tau \in \mathcal{T}$, a transition $\tau^*$ such that $(\sigma_1^*, \sigma_2^*) \in \tau^*$ iff, for all $\tau' \in \mathcal{T}$,

  $$(\sigma_1, \sigma_2) \in \tau,$$
  $$\sigma_1^*(\delta_\tau) \geq l_\tau,$$
  $$\sigma_2^*(\mathbf{t}) = \sigma_1^*(\mathbf{t}),$$
  $$\sigma_2^*(\delta_{\tau'}) = \begin{cases} \sigma_1^*(\delta_{\tau'}) & \text{if } \tau' \text{ is enabled on } \sigma_2 \\ 0 & \text{otherwise.} \end{cases}$$

  Note that the second clause, $\sigma_1^*(\delta_\tau) \geq l_\tau$, enforces all lower bounds.

  In addition, $\mathcal{T}^*$ contains the *tick* transition $\tau_T$ such that $(\sigma_1^*, \sigma_2^*) \in \tau_T$ iff, for all $\tau' \in \mathcal{T}$,

  $$\sigma_1 = \sigma_2,$$
  $$\sigma_2^*(\mathbf{t}) = \sigma_1^*(\mathbf{t}) + 1,$$
  $$\sigma_2^*(\delta_{\tau'}) = \begin{cases} \sigma_1^*(\delta_{\tau'}) + 1 & \text{if } \tau' \text{ is enabled on } \sigma_1 \\ 0 & \text{otherwise,} \end{cases}$$
  $$\sigma_2^*(\delta_{\tau'}) \leq u_{\tau'}.$$

  The last clause enforces all finite upper bounds.

- $l_\tau^* = 0$.
- $u_\tau^* = \infty$. These weak-fairness conditions enforce all infinite upper bounds.

The real-time transition system $S$ and its extension $S^*$ are, in the full paper, shown to be equivalent in the following sense: for every computation $\rho = (\sigma, \mathsf{T})$ of $S$, there is a computation $\rho^* = (\sigma^*, \mathsf{T})$ of $S^*$ such that every state $\sigma_i^*$, $i \geq 0$, is an extension of $\sigma_i$ to $V^*$; and for every computation $(\sigma^*, \mathsf{T})$ of $S^*$, the timed state sequence $(\sigma, \mathsf{T})$ is a computation of $S$ if every state $\sigma_i$, $i \geq 0$, is the restriction of $\sigma_i^*$ to $V$.

Thus, to show a temporal formula $\phi$ (over $V$) to be $S$-valid, it suffices to show the $S^*$-validity of $\phi$.

## 5.2 Proof rule

We show that any temporal formula over $V^*$ is $S^*$-equivalent to an (unbounded) unless formula.

*Unless* properties assert that something will hold continuously either forever, or until terminated by the occurrence of another event. They are expressed by temporal formulas of the form

$$p \Rightarrow q \,\mathsf{U}\, r,$$

for state formulas $p, q$, and $r$. The formula $p \Rightarrow q \,\mathsf{U}\, r$ is true over the timed state sequence $\rho = (\sigma, \mathsf{T})$ iff, for all $i \geq 0$,

> if $\sigma_i$ is a $p$-state,
> then either all subsequent $\sigma_j$, $j \geq i$, are $q$-states,
> or there is some $r$-state $\sigma_j$, $j \geq i$, such that all intermediate $\sigma_k$, $i \leq k < j$, are $q$-states;

that is, every $p$-state is followed by a (possibly infinite) sequence of $q$-states until there is an $r$-state.

In particular, the bounded-invariance property

$$p \Rightarrow \Box_{<l}\, q$$

is $S^*$-equivalent to the unbounded unless property

$$p \wedge (\mathbf{t} = T) \;\Rightarrow\; q \,\mathsf{U}\, (\mathbf{t} \geq T + l),$$

(i.e., both formulas are true over the same computations of $S^*$), and the bounded-response property

$$p \Rightarrow \Diamond_{\leq u}\, q$$

is $S^*$-equivalent to the unless property

$$p \wedge (\mathbf{t} = T) \;\Rightarrow\; (\mathbf{t} \leq T + u) \,\mathsf{U}\, q$$

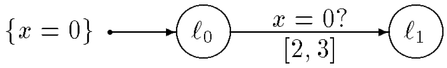
if $q$ is a state formula over $V$. Both unless formulas make use of the *rigid* (static) variable $T$ (i.e., $T = T'$) to record the time of the $p$-state. Note that the latter equivalence is based on the fact that the time is guaranteed to reach and surpass $T + u$, for any value of $T$.

This observation lead to an alternative and quite different approach to the verification of real-time properties: to prove the $S$-validity of a real-time property $\phi$ (over $V$), we establish instead the $S^*$-validity of an $S^*$-equivalent unless property $\phi^*$ (over $V^*$). This can be done by applying the timeless *unless* rule ([MP89]):

| **UNLESS** | (1) | $p \rightarrow (\varphi \vee r)$ |
|---|---|---|
| | (2) | $\{\varphi\}\, \mathcal{T}^*\, \{\varphi \vee r\}$ |
| | (3) | $\varphi \rightarrow q$ |
| | | $p \Rightarrow q \,\mathsf{U}\, r$ |

We emphasize that all premises are state formulas; the state formula $\varphi$ is called the *intermediate assertion* of the rule.

To demonstrate this kind of "explicit-clock" real-time reasoning, consider again the single-process system $P$ with the data precondition $x = 0$ and the following transition diagram:




Both the lower and the upper bound on the termination of $P$,

$$start \;\Rightarrow\; (\neg at\_\ell_1)\, \mathsf{U}\, (\mathbf{t} \geq 2)$$

and

$$start \;\Rightarrow\; (\mathbf{t} \leq 3)\, \mathsf{U}\, at\ell_1,$$

respectively, can be derived by the unless rule **UNLESS**. To establish the lower bound, we use the intermediate assertion

$$at\_\ell_0 \wedge (0 \leq \mathbf{t} \leq 2) \wedge (\mathbf{t} = \delta_{\tau_{0\to1}}),$$

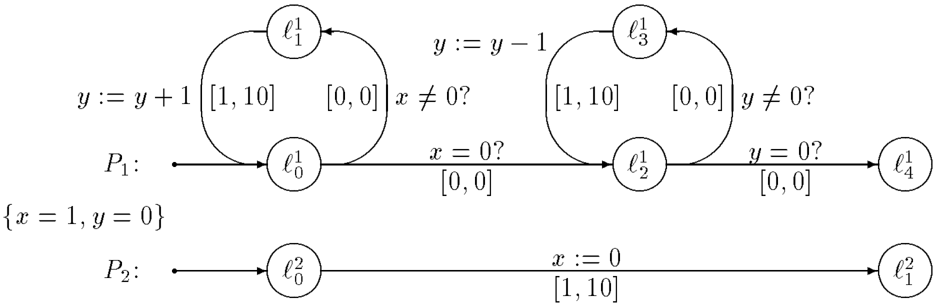
which has to be strengthened by $x = 0$ to imply the upper bound.

While the "hidden-clock" verification style presented in the previous section refers to time only through time-bounded temporal operators, explicit-clock reasoning uses ordinary, timeless, temporal operators and refers to the time in state formulas. Both styles trade off the complexity of the temporal proof structure against the complexity of the state invariants: the *hidden-clock* approach relies on complex proof structures similar to the proof lattices used to establish ordinary (timeless) *liveness* properties ([OL82], [MP89]), and uses relatively simple invariants; the *explicit-clock* method employs only the simple unless rule — a (timeless) *safety* rule —, but requires powerful intermediate assertions.

# 6 Completeness

The unless rule is known to be complete, relative to state reasoning, for establishing unless properties ([MP89b]). It follows immediately that explicit-clock reasoning is relative complete for showing bounded-invariance as well as bounded-response properties.

As for bounded-operator reasoning, we first observe that *both* the lower- and upper-bound rules may be necessary for proving a bounded-invariance (or bounded-response) property. This is demonstrated by the following example:




In other words, we assume that assignments cost at least 1 and at most 10 time units, while tests are free.

In this example, the first process, $P_1$, consumes the maximal amount of time if its first loop, in which the value of $y$ is incremented, is executed as often (fast) as possible — 11 times. In this worst (slowest) case, $P_1$ terminates by time 130. Consequently, the proof of timely termination relies on an interplay of lower- and upper-bound rules; it is given in the full paper.

It follows that, for completeness purposes, the lower-bound rules cannot be considered independently of the upper-bound rules. In the full paper, we show that the combined set of rules is sufficient to establish both bounded-invariance and bounded-response properties (relative to state reasoning). The proof proceeds similarly to completeness arguments in the untimed case ([MP89b]); it assumes a state language that is strong enough to assert that one state is (not) reachable from another state within a certain amount of time.

**Acknowledgements.** We thank Rajeev Alur for many helpful discussions.

## References


[AH90] R. Alur, T.A. Henzinger, "Real-time logics: complexity and expressiveness," 5th IEEE LICS, 1990.

[EMSS89] E.A. Emerson, A.K. Mok, A.P. Sistla, J. Srinivasan, "Quantitative temporal reasoning," *Automatic Verification of Finite-state Systems* (J. Sifakis, ed.), Springer *LNCS* **407**, 1989.

[Ha88] E. Harel, *Temporal Analysis of Real-time Systems*, M.S. Thesis, Weizmann Institute, 1988.

[HMP90] T.A. Henzinger, Z. Manna, A. Pnueli, "An interleaving model for real time," 5th Jerusalem Conf. on Information Technology, 1990.

[Ko89] R. Koymans, *Specifying Message Passing and Time-critical Systems with Temporal Logic*, Ph.D. Thesis, Eindhoven Univ. of Tech., 1989.

[KVdR83] R. Koymans, J. Vytopil, W.-P. de Roever, "Real-time programming and asynchronous message passing," 2nd ACM PODC, 1983.

[MP89a] Z. Manna, A. Pnueli, "The anchored version of the temporal framework," *Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency* (J.W. deBakker, W.-P. de Roever, and G. Rozenberg, eds.), Springer *LNCS* **354**, 1989.

[MP89b] Z. Manna, A. Pnueli, "Completing the temporal picture," 16th EATCS ICALP, 1989.

[OL82] S. Owicki, L. Lamport, "Proving liveness properties of concurrent programs," ACM *TOPLAS* **4**, 1982.

[Os90] J.S. Ostroff, *Temporal Logic for Real-time Systems*, Research Studies Press, 1989.

[PdR82] A. Pnueli and W.-P. de Roever, "Rendezvous with Ada — a proof-theoretical view," SigPlan AdaTEC, 1982.

[PH88] A. Pnueli, E. Harel, "Applications of temporal logic to the specification of real-time systems," *Formal Techniques in Real-time and Fault-tolerant Systems*, Springer *LNCS* **331**, 1988.

[Ro84] D. Ron, *Temporal Verification of Communication Protocols*, M.S. Thesis, Weizmann Institute, 1984.