Temporal Proof Methodologies for Real-time Systems*†

Thomas A. Henzinger‡ Zohar Manna‡§ Amir Pnueli§

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 reactive systems. The abstract notion of a real-time transition system is defined as a conservative extension of traditional transition systems: qualitative fairness requirements are replaced (and superseded) by quantitative lower-bound and upper-bound real-time requirements for transitions.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 exhibit two styles for the specification of real-time properties. The first approach uses bounded versions of the temporal operators, while the second approach 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.

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 qualitative 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 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

$$(p \wedge \mathrm{t} = T) \rightarrow \Diamond(q \wedge \mathrm{t} \leq T + 3),$$

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,

*The full version of this paper is available as a technical report ([HMP91]).

†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).

‡Department of Computer Science, Stanford University, Stanford, CA 94305.

§Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel 76100.

to the two styles for the specification of timing requirements.

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 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 bears a close resemblance to *liveness* properties, while the class of lower-bound properties closely resembles *safety* properties. 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.

In our model, we assume a global, discrete, and asynchronous clock, whose actions (clock ticks) are interleaved with the 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. Yet we retain the important economic advantage of interleaving models, namely, that at any point only one transition can occur and has to be analyzed.

In Section 2, we define the abstract computational model of a real-time transition system and illustrate how 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 a real-time system as a set 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* model of real-time multiprocessing systems that communicate through shared variables, and show how to interpret the concrete constructs within the abstract model. In the full paper, we also show how message-passing and multiprogramming systems can be modeled in this framework ([HMP91]).

2.1 Abstract model: Real-time transition system

The basic computational model we use is that of a *transition system* ([MP89a]), 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.[1]

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 Σ 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.

[1] To simplify the verification methods, the definition of a real-time transition system has been significantly changed since introduced in [HMP90]. That preliminary work is, in the current paper, also extended by the treatment of nondeterminism and by completeness results.

- 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 τ_I. Every transition $\tau \in \mathcal{T}$ is a binary accessibility relation on Σ; that is, it defines for every state $\sigma \in \Sigma$ a (possibly empty) set of τ-successors $\tau(\sigma) \subseteq \Sigma$. We require that $\sigma'(\mathit{first}) = \mathit{false}$ for all τ-successors $\sigma' \in \tau(\sigma)$ of every state σ.

  The transition τ is *enabled* on σ iff $\tau(\sigma) \neq \emptyset$. Let σ^+ denote the state that differs from σ at most in the interpretation of the variable *first*, to which σ^+ assigns the value *false*. The *idle* (stutter) transition
  $$\tau_I = \{(\sigma, \sigma^+) : \sigma \in \Sigma\}$$
  is enabled on every state.
- a *minimal delay* $l_\tau \in \mathsf{N}$ for every transition $\tau \in \mathcal{T}$. In particular, $l_{\tau_I} = 0$.
- a *maximal delay* $u_\tau \in \mathsf{N} \cup \{\infty\}$ for every transition $\tau \in \mathcal{T}$. We require that $u_\tau \geq l_\tau$ for all $\tau \in \mathcal{T}$, and $u_{\tau_I} = \infty$ (for notational convenience, we assume that $\infty \geq n$ for all $n \in \mathsf{N}$).

  Let $\mathcal{T}_0 \subseteq \mathcal{T}$ be the set of transitions with the maximal delay 0. To allow time to progress, we put a restriction on these transitions. We require that there is no sequence of states $\sigma_0 \sigma_1 \ldots \sigma_n$ such that $n > |\mathcal{T}_0|$ and, for all $0 \leq i < n$, $\sigma_{i+1} \in \tau(\sigma_i)$ for some transition $\tau \in \mathcal{T}_0$.

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

- [*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.

By $\rho^i = (\sigma^i, \mathsf{T}^i)$ we denote the i-th *suffix* of ρ; it consists of the infinite sequence $\sigma^i = \sigma_i \sigma_{i+1} \ldots$ of states and the infinite sequence $\mathsf{T}^i = \mathsf{T}_i \mathsf{T}_{i+1} \ldots$ of times. Note that ρ^i is, for all $i \geq 0$, again a timed state sequence; that is, the set of timed state sequences is closed under suffices.

The timed state sequence $\rho = (\sigma, \mathsf{T})$ is an *initialized 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*] $\sigma_0 \in \Theta$ and $\mathsf{T}_0 = 0$; that is, the time of the initial state is 0.
- [*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 τ is *taken* at position i and *completed* at position $i+1$. At both stuttering steps and clock ticks, the idle transition τ_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 τ is taken at position j,
  then $\mathsf{T}_j \geq l_\tau$ and τ is enabled on σ_i.

  In other words, once enabled, τ is delayed for at least l_τ clock ticks; it can be taken only after being continuously enabled for l_τ 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 τ is not enabled on σ_j,
  or τ is taken at position j.

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

The computations of a real-time transition system are obtained by closing the set of initialized computations under suffices: the timed state sequence ρ is an *computation* of S iff ρ is a suffix of an initialized computation of S.

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 τ_I. The computations of any real-time transition system are, furthermore, closed under *stuttering*: the addition of finitely many stuttering steps to a timed state sequence does not alter the property of being a computation of S.

Also observe that the state component of any computation of S is, in the sense of [MP89a], a computation of the (untimed) transition system S^- that results from S by disregarding all delays. It follows that ordinary timeless reasoning is sound for real-time transition systems.

The timing constraints of S can be viewed as filters that prohibit certain possible behaviors of the underlying untimed system S^-. Special cases are a minimal delay 0 and a maximal delay ∞ for a transition τ. While the former does not rule out any computations of S^-, the latter adds to S^- a *weak-fairness* assumption: τ cannot be continuously enabled without being taken.

2.2 Concrete model: Shared variables

The 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.

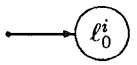
A *shared-variables multiprocessing system* P has the form

$$\{\theta\}[P_1 \| \dots \| P_m].$$

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 θ, called the *data precondition* of P, restricts the initial values of the variables in $U = U_s \cup \bigcup_{1 \leq i \leq m} U_i$.

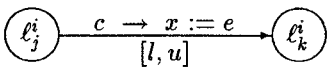
The real-time programs P_i can be alternatively presented in a textual programming language, or as transition diagrams. We shall use the latter, graphical, representation.

A *transition diagram* for the process P_i is a finite directed graph whose vertices $L_i = \{\ell_0^i, \dots \ell_{n_i}^i\}$ are called *locations*; ℓ_0^i is considered to be the *entry* location:



The intended meaning of the entry location is that the control of the process P_i starts at the location ℓ_0^i 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* $u \in \mathsf{N} \cup \{\infty\}$ such that $u \geq l$:

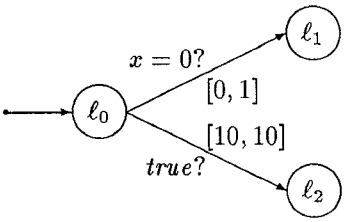


where the guard c is a boolean expression, x is a variable, and e an expression (the guard *true* and the delay interval $[0, \infty]$ are usually suppressed; the instruction $c \rightarrow x := x$ is often abbreviated to c?). We require that every cycle in the graph consists of no fewer than two edges, 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 ℓ_j^i for at least l time units during which the guard c has been continuously true, then P_i *may* proceed to the location ℓ_k^i. The maximal delay u ensures that whenever the control of the process P_i has resided at ℓ_j^i for u time units during which the guard c has been continuously true, then P_i *must* proceed to ℓ_k^i. In doing so, the control of P_i moves to the location ℓ_k^i "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 ℓ_0, the process P attempts to proceed to the location ℓ_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 ℓ_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 $S_P = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$:

- $V = U \cup \{\pi_1, \dots \pi_m, \mathit{first}\}$. Each *control variable* π_i, $1 \leq i \leq m$, ranges over the locations L_i of the corresponding process P_i.
- Σ contains all interpretations of V.
- Θ is the set of all states $\sigma \in \Sigma$ such that θ is true in σ, $\sigma(\pi_i) = \ell_0^i$ for all $1 \leq i \leq m$, and $\sigma(\mathit{first}) = \mathit{true}$.
- $\mathcal{T}$ contains, in addition to τ_I, a transition τ_E for every edge E in the transition diagrams for $P_1, \dots P_m$. If E connects the source location ℓ_j^i to the target location ℓ_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 $\sigma'(\pi_i) = \ell_k^i$,
  c is true in σ, $\sigma'(x) = \sigma(e)$, $\sigma'(\mathit{first}) = \mathit{false}$,
  $\sigma'(y) = \sigma(y)$ for all $y \in V - \{\pi_i, x, \mathit{first}\}$.

  If τ_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 ∞). Then the state components of the initialized 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 ([MP89a]); weak fairness (progress) for every individual transition and, thus, for every process is guaranteed by the maximal delays ∞.

2.3 Other concrete models

The interleaving model for concurrency identifies true parallelism with (sequential) nondeterminism. 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 priorities, interrupts, and scheduling of tasks may strongly influence the ability of a system to meet its timing constraints.

Consider the following example of a real-time system in a multiprogramming environment. 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 or even individual transitions. 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.

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.

A formal treatment of priorities and scheduling, as well as real-time systems that communicate by message passing, is given in the full paper ([HMP91]).

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

Let $S = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$ be a real-time transition system. 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 Σ, and called *state formulas*. If the state formula p is true in state σ, we say that σ is a p-state.

We use the following abbreviations for state formulas:

- The *starting* condition *start* holds precisely in the initial states Θ.
- 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 τ-successors of σ. 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*(τ) asserts that τ is enabled. In particular, *enabled*(τ_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 the m processes P_i, $1 \leq i \leq m$. Let $at\_\ell_j^i$ stand for $\pi_i = \ell_j^i$; that is, the control of the process P_i is at the location ℓ_j^i. We abbreviate any disjunction $at\_\ell_j^i \vee at\_\ell_k^i$ further, to $at\_\ell_{j,k}^i$.

If θ is the data precondition of P, then the starting condition *start* is equivalent to

$$\theta \wedge \Big(\bigwedge_{1 \leq i \leq m} at\_\ell_0^i \Big) \wedge \mathit{first}.$$

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 ℓ_j^i to the location ℓ_k^i and is labeled by the instruction $c \rightarrow x := e$. Then, the enabling condition $enabled(\tau)$ is equivalent to

$$at\_\ell_j^i \wedge c,$$

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

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

where q' is obtained from q by replacing every variable with its primed version (for example, $(at\_\ell_k^i)'$ stands for $\pi_i' = \ell_k^i$).

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-invariance and bounded-response properties. Thus we restrict ourselves to the following temporal formulas:

- Every state formula p is a temporal formula; it is true over the timed state sequence $\rho = (\sigma, \mathsf{T})$ iff the initial state σ_0 is a p-state.
- Every boolean combination of temporal formulas is a temporal formula, whose truth over a timed state sequence is determined from the truth of its components in the obvious way.
- If p is a state formula, ϕ a temporal formula, and $l \in \mathsf{N}$, then $p\,\mathsf{U}_{\geq l}\,\phi$ is a temporal formula; it is true over the timed state sequence $\rho = (\sigma, \mathsf{T})$ iff either all σ_i, $i \geq 0$, are p-states, or there is some position $i \geq 0$ such that $\mathsf{T}_i \geq \mathsf{T}_0 + l$, ϕ is true over the i-th suffix ρ^i of ρ, and all σ_j, $0 \leq j < i$, are p-states.

  Note that the formula $p\,\mathsf{U}_{\geq 0}\,q$ is true over ρ iff the untimed *unless* formula $p\,\mathsf{U}\,q$ is true over the state component σ of ρ. We use the abbreviations $p\,\mathsf{U}\,\phi$, $\Box_{<l}\,p$, and $p\,\mathsf{U}^+_{\geq l}\,\phi$ for the formulas $p\,\mathsf{U}_{\geq 0}\,\phi$, $p\,\mathsf{U}_{\geq l}\,true$, and $p \wedge (p\,\mathsf{U}_{\geq l}\,\phi)$.
- If ϕ is a temporal formula and $u \in \mathsf{N}$, then $\Diamond_{\leq u}\,\phi$ is a temporal formula; it is true over the timed state sequence $\rho = (\sigma, \mathsf{T})$ iff there is some position $i \geq 0$ such that $\mathsf{T}_i \leq \mathsf{T}_0 + u$ and ϕ is true over the i-th suffix ρ^i of ρ.

Temporal-logic aficionados will readily recognize the operators $\mathsf{U}_{\geq l}$, $\Box_{<l}$, and $\Diamond_{\leq u}$ as time-bounded versions of the conventional (untimed) *unless*, *always*, and *eventually* operators ([MP89]). With these bounded temporal operators we can express two important classes of real-time properties (for a general addition of time-bounded operators to linear temporal logic, see [AH90]).

- 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 $l \in \mathsf{N}$. Recall that 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 σ_i is a p-state and $\mathsf{T}_j < \mathsf{T}_i + l$,
  > then σ_j is a q-state;

  that is, no p-state is followed by a $\neg q$-state within time less than l.
- 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 $u \in \mathsf{N}$. Recall that 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 σ_i is a p-state,
  > then there is some q-state σ_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.

From now on, we use the convention that the letters p, q, r as well as φ (and primed versions) denote state formulas, while the letters ϕ, ψ, and χ stand for arbitrary temporal formulas.

We say that a temporal formula is *S-valid* iff it is true over all computations of the real-time transition

system S; (general) validity (i.e., truth under every interpretation) implies of course S-validity for every system S. 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 *monotonicity* rule U-**MON**, which allows us to weaken any of the three arguments of the bounded-unless operator:

$$\textbf{U-MON}\quad \frac{p \rightarrow p' \qquad \phi \rightarrow \phi' \qquad l' \leq l}{(p \,\mathsf{U}_{\geq l}\, \phi) \rightarrow (p' \,\mathsf{U}_{\geq l'}\, \phi')}$$

It is not hard to see that this rule is S-sound for every real-time transition system S. A similar, also universally S-sound, monotonicity rule holds for the bounded-eventuality operator:

$$\Diamond\textbf{-MON}\quad \frac{\phi \rightarrow \phi' \qquad u' \geq u}{(\Diamond_{\leq u}\, \phi) \rightarrow (\Diamond_{\leq u'}\, \phi')}$$

We will refer to applications of these two weakening rules in derivations through the simple annotation "by monotonicity."

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* and *induction* rules combine real-time properties into more complicated ones.

First we present the methodology to verify deterministic systems (S is deterministic if any two guards that are associated with outgoing edges of the same vertex in the transition diagram of S are disjoint). Nondeterministic systems require more complex (conditional) single-step reasoning and are treated at the end of this section.

4.1 Single-step rules

The *single-step lower-bound* rule uses the minimal delay $l_\tau \in \mathsf{N}$ of a transition $\tau \in \mathcal{T}$ to infer a bounded-unless formula (by $\mathcal{T} - \tau$ we denote the set difference $\mathcal{T} - \{\tau\}$):

$$\textbf{U-SS}\quad \frac{\begin{array}{l} p \rightarrow (\mathit{first} \vee \neg \mathit{enabled}(\tau)) \\ p \rightarrow \varphi \\ \{\varphi\}\, \mathcal{T} - \tau\, \{\varphi\} \\ \varphi \rightarrow q \\ (\varphi \wedge \mathit{enabled}(\tau)) \rightarrow r \end{array}}{p \rightarrow q \,\mathsf{U}_{\geq l_\tau}\, r}$$

The rule U-**SS** derives a *temporal* (bounded-unless) formula from premises all of which are *state* formulas. The state formula φ is called the *invariant* of the rule. Choosing r to be *true*, the rule infers a bounded-invariance property,

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

(note that the last premise holds trivially in this case).

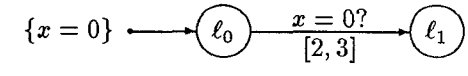
To see why the rule U-**SS** is S-sound, observe that whenever the transition τ is not enabled, it cannot be taken for at least l_τ time units. Also, τ cannot be taken within the first l_τ time units of any initialized computation.

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

$$\Diamond\textbf{-SS}\quad \frac{\begin{array}{l} p \rightarrow (\varphi \vee q) \\ \varphi \rightarrow \mathit{enabled}(\tau) \\ \{\varphi\}\, \mathcal{T} - \tau\, \{\varphi \vee q\} \\ \{\varphi\}\, \tau\, \{q\} \end{array}}{p \rightarrow \Diamond_{\leq u_\tau}\, q}$$

This rule derives a *temporal* bounded-response formula from premises all of which are *state* formulas. The state formula φ is again called the *invariant* of the rule. To see why the rule $\Diamond$-**SS** is S-sound, recall that the transition τ has to be taken before it would be continuously enabled for more than u_τ time units.

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 ℓ_1. Because of the delay interval $[2, 3]$ of the transition $\tau_{0\rightarrow 1}$, the final location ℓ_1 cannot be reached before time 2 and must be reached by time 3.

Using single-step reasoning, we can carry out a formal proof of this analysis. The bounded-invariance property that P does not terminate before time 2,

$$\mathit{start} \rightarrow \Box_{<2}\, \neg \mathit{at}\_\ell_1,$$

is established by an application of the single-step lower-bound rule U-**SS** with respect to the transition $\tau_{0\rightarrow 1}$

(let the invariant φ be at_ℓ_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 $\Diamond$-**SS** 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 transitivity rules.

The *transitive lower-bound* rule combines two bounded-unless formulas:

$$\textbf{U-TRANS}\quad \frac{\phi \;\rightarrow\; p\, \mathsf{U}_{\geq l_1}\, \chi \qquad \chi \;\rightarrow\; q\, \mathsf{U}_{\geq l_2}\, \psi}{\phi \;\rightarrow\; (p \vee q)\, \mathsf{U}_{\geq l_1 + l_2}\, \psi}$$

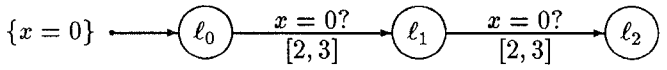
We refer to the formula χ as the *link* of the rule.

The *transitive upper-bound* rule combines two bounded-response formulas:

$$\Diamond\textbf{-TRANS}\quad \frac{\phi \;\rightarrow\; \Diamond_{\leq u_1}\, \chi \qquad \chi \;\rightarrow\; \Diamond_{\leq u_2}\, \psi}{\phi \;\rightarrow\; \Diamond_{\leq u_1 + u_2}\, \psi}$$

The formula χ is again called 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 **U-TRANS**, it suffices to show the two premises

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

Both premises can be established by single-step lower-bound reasoning. To show the premise (1), we apply the rule **U-SS** with respect to the transition $\tau_{0\rightarrow 1}$, using the invariant at_ℓ_0; the premise (2) follows from the rule **U-SS** with respect to the transition $\tau_{1\rightarrow 2}$ and the invariant $at\_\ell_{0,1}$.

The upper bound on the termination of P,

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

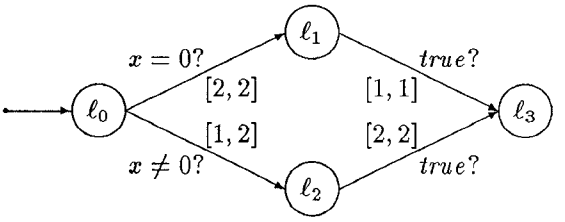
is concluded by the transitive upper-bound rule $\Diamond$-**TRANS**. It suffices to show the premises

$$\begin{array}{l} start \;\rightarrow\; \Diamond_{\leq 3}(at\_\ell_1 \wedge x = 0), \\ (at\_\ell_1 \wedge x = 0) \;\rightarrow\; \Diamond_{\leq 3}\, at\_\ell_2, \end{array}$$

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_ℓ_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* $\tau_{0\rightarrow 1}$ is taken.

For an example with a (deterministic) branching structure, consider the process P' with the following transition diagram:



We show that P' terminates either at time 3 or at time 4.

The proof requires a case analysis on the initial value of x, which determines which path of the transition diagram is taken. The lower bound

$$start \;\rightarrow\; \Box_{<3}\, \neg at\_\ell_3$$

is implied by the two bounded-invariance formulas

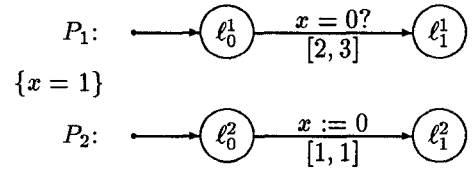
$$\begin{array}{l} (start \wedge x = 0) \;\rightarrow\; \Box_{<3}\, \neg at\_\ell_3, \\ (start \wedge x \neq 0) \;\rightarrow\; \Box_{<3}\, \neg at\_\ell_3, \end{array}$$

each of which can be derived by transitive lower-bound reasoning (use the links $at\_\ell_0 \wedge x = 0$ and $at\_\ell_0 \wedge x \neq 0$, respectively). The upper bound

$$start \;\rightarrow\; \Diamond_{\leq 4}\, at\_\ell_3$$

follows by a similar case analysis and transitive upper-bound reasoning.

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 ℓ_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 ℓ_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 **U-TRANS**. It suffices to show the premises

$$start \;\rightarrow\; (\neg at\_\ell_1^1)\, \mathsf{U}_{\geq 1}\, (at\_\ell_0^1 \wedge x = 1),$$
$$(at\_\ell_0^1 \wedge x = 1) \;\rightarrow\; (\neg at\_\ell_1^1)\, \mathsf{U}_{\geq 2}\, true,$$

both of which follow from single-step lower-bound reasoning. Similarly, the transitive upper-bound rule **◇-TRANS** 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 Induction rules

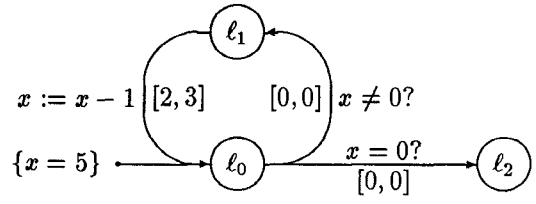
To prove lower and upper bounds on the execution time of program loops, we need to combine a state-dependent number of bounded-invariance or bounded-response properties. For this purpose it is economical to have induction schemes.

The *inductive lower-bound* rule **U-IND** generalizes the transitive lower-bound rule **U-TRANS**; it combines a potentially large number of similar bounded-unless formulas in a single proof step. Assume that the new, rigid (static) variable $i \in V$ (i.e., $i \notin V$ and $i = i'$) ranges over the natural numbers N; for any $n \in \mathsf{N}$:

$$\textbf{U-IND}\quad \frac{(\varphi(i) \wedge i > 0) \;\rightarrow\; p\, \mathsf{U}_{\geq l}\, \varphi(i-1)}{\varphi(n) \;\rightarrow\; p\, \mathsf{U}_{\geq n \cdot l}\, \varphi(0)}$$

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. Note that every instance of the rule **U-IND**, for any constant $n \in \mathsf{N}$, is derivable from the transitive lower-bound rule **U-TRANS**.

For a demonstration of inductive lower-bound reasoning, consider 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 ℓ_2. Since x starts out with the value 5, and each decrement operation takes at least 2 time units, while the tests are instantaneous, the final location ℓ_2 cannot be reached before time 10.

This lower bound,

$$start \;\rightarrow\; \Box_{<10} \neg at\_\ell_2,$$

follows by transitivity and monotonicity from the two bounded-unless properties

(1) $start \;\rightarrow\; (\neg at\_\ell_2)\, \mathsf{U}_{\geq 2}\, (at\_\ell_1 \wedge x = 5)$,
(2) $(at\_\ell_1 \wedge x = 5) \;\rightarrow\; (\neg at\_\ell_2)\, \mathsf{U}_{\geq 8}\, (at\_\ell_1 \wedge x = 1)$.

The first property, (1), is enforced by two single-step lower bounds; the second property, (2), can be derived by the inductive lower-bound rule **U-IND** from the premise

$$(at\_\ell_1 \wedge x = i{+}1 \wedge i > 0) \;\rightarrow\; (\neg at\_\ell_2)\, \mathsf{U}_{\geq 2}\, (at\_\ell_1 \wedge x = i),$$

which is concluded by transitive reasoning.

The inductive lower-bound rule has a twin that combines several similar bounded-response formulas by adding up there upper bounds u. In fact, both induction rules can be generalized, by letting the bounds l and u vary as functions of i. In its more general form, we state only the *inductive upper-bound* rule. It uses again a new, rigid variable $i \in V$ that ranges over the natural numbers N; for any $n \in \mathsf{N}$:

$$\textbf{◇-IND}\quad \frac{(\varphi(i) \wedge i > 0) \;\rightarrow\; \Diamond_{\leq u_i}\, \varphi(i-1)}{\varphi(n) \;\rightarrow\; \Diamond_{\leq \Sigma_{0 < i \leq n}\, u_i}\, \varphi(0)}$$

Every instance of this rule is derivable from the transitive upper-bound rule **◇-TRANS**.

The general form of the inductive upper-bound rule is useful to prove upper bounds for programs with loops whose execution time is not uniform. An example for such a system is the *odd-even* variant of the process P in Figure 1.

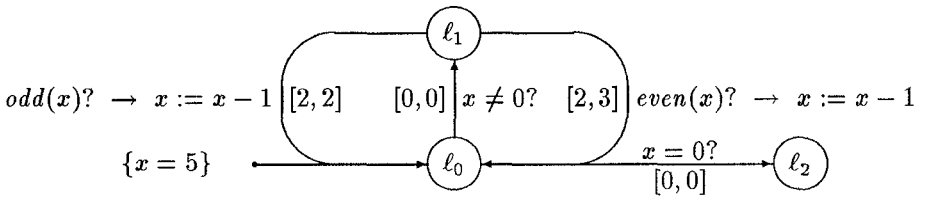
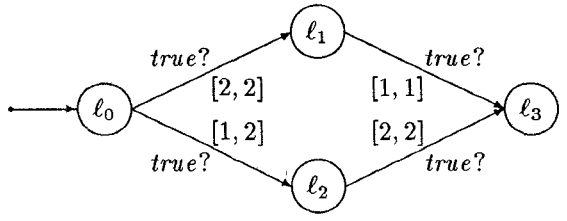


Figure 1: Odd-even

4.4 Conditional rules

Unfortunately, our proof system is not strong enough to show tight bounds on *nondeterministic* systems. To see this, consider the following nondeterministic variant P of a process encountered previously:



As before, P terminates either at time 3 or at time 4. However, during an execution of P, one of the two transitions $\tau_{0\to1}$ and $\tau_{0\to2}$ is chosen nondeterministically. Thus we cannot carry out a case analysis with respect to a state formula that selects a unique guard.

Instead, we proceed in two steps. First we establish an untimed formula that enumerates all possible nondeterministic choices. Then we decorate the unbounded temporal formula with time bounds.

Step 1. To establish the S-validity of a temporal formula ϕ that is built only from connectives and U (i.e., $\mathsf{U}_{\geq 0}$) operators, it suffices to show that ϕ is true over all (untimed) computations of the transition system S^- underlying S. This can be achieved with the help of any conventional (timeless) proof system (for instance, the proof system of [MP89b]).

To derive the lower bound 3 on the termination of our example P, we show the untimed formula

$$start \rightarrow \begin{pmatrix} start\,\mathsf{U}^+\, at\_\ell_0\,\mathsf{U}^+\, at\_\ell_1 \vee \\ start\,\mathsf{U}^+\, at\_\ell_0\,\mathsf{U}^+\, at\_\ell_2 \end{pmatrix} \qquad (\dagger)$$

(nested unless operators associate to the right).

Step 2. To add time bounds to this disjunction of nested unless formulas, we need *conditional* single-step rules. They establish single-step real-time bounds under the assumption that a particular disjunct has been chosen.

The *conditional single-step lower-bound* rule uses the minimal delay $l_\tau \in \mathsf{N}$ of a transition $\tau \in \mathcal{T}$:

U-CSS

$$\frac{p \rightarrow ((first \wedge l = 0) \vee \neg enabled(\tau)) \quad \{q\}\,\mathcal{T} - \tau\,\{q \vee \neg r\}}{(p\,\mathsf{U}^+_{\geq l}\,q\,\mathsf{U}^+\,(r \wedge \phi)) \rightarrow (p\,\mathsf{U}^+_{\geq l}\,q\,\mathsf{U}^+_{\geq l_\tau}\,(r \wedge \phi))}$$

This presentation represents really *two* rules, depending on whether $l = 0$, which is, strictly speaking, not a state formula. The rule is **U-CSS** S-sound for any temporal formula ϕ.

In our example, we use the conditional single-step lower-bound rule **U-CSS** with respect to the transitions $\tau_{0\to1}$ and $\tau_{0\to2}$ to derive the conditional single-step bounds

$$(start\,\mathsf{U}^+\, at\_\ell_0\,\mathsf{U}^+\, at\_\ell_1) \rightarrow (start\,\mathsf{U}^+\, at\_\ell_0\,\mathsf{U}^+_{\geq 2}\, at\_\ell_1),$$
$$(start\,\mathsf{U}^+\, at\_\ell_0\,\mathsf{U}^+\, at\_\ell_2) \rightarrow (start\,\mathsf{U}^+\, at\_\ell_0\,\mathsf{U}^+_{\geq 1}\, at\_\ell_2).$$

They allow us to conclude, from (†),

$$start \rightarrow \begin{pmatrix} start\,\mathsf{U}^+\, at\_\ell_0\,\mathsf{U}^+_{\geq 2}\, at\_\ell_1 \vee \\ start\,\mathsf{U}^+\, at\_\ell_0\,\mathsf{U}^+_{\geq 1}\, at\_\ell_2 \end{pmatrix} \qquad (\ddagger)$$

To collapse nested unless operators, we use the valid temporal formula **U-COLL**:

U-COLL

$$(p\,\mathsf{U}_{\geq l_1}\,q\,\mathsf{U}_{\geq l_2}\,\phi) \rightarrow ((p \vee q)\,\mathsf{U}_{\geq l_1 + l_2}\,\phi)$$

This formula can be derived from the transitive lower-bound rule **U-TRANS**.

From (‡) we obtain by collapsing and monotonicity

$$start \rightarrow (at\_\ell_0\,\mathsf{U}^+_{\geq 2}\, at\_\ell_1 \vee at\_\ell_0\,\mathsf{U}^+_{\geq 1}\, at\_\ell_2);$$

that is, using the (untimed) validity $p \rightarrow p\,\mathsf{U}\,p$ and monotonicity,

$$start \rightarrow \begin{pmatrix} at\_\ell_0\,\mathsf{U}^+_{\geq 2}\, at\_\ell_1\,\mathsf{U}^+\, at\_\ell_1 \vee \\ at\_\ell_0\,\mathsf{U}^+_{\geq 1}\, at\_\ell_2\,\mathsf{U}^+\, at\_\ell_2 \end{pmatrix}.$$

Adding conditional single-step lower bounds for the transitions $\tau_{1\to3}$ and $\tau_{2\to3}$ gives

$$start \rightarrow \begin{pmatrix} at\_\ell_0\,\mathsf{U}^+_{\geq 2}\, at\_\ell_1\,\mathsf{U}^+_{\geq 1}\, at\_\ell_1 \vee \\ at\_\ell_0\,\mathsf{U}^+_{\geq 1}\, at\_\ell_2\,\mathsf{U}^+_{\geq 2}\, at\_\ell_2 \end{pmatrix},$$

and by collapsing and monotonicity we finally arrive at the desired bounded-invariance property

$$start \rightarrow \Box_{<3} \neg at\_\ell_3.$$

Conditional upper-bound reasoning does not require the nesting of unless operators. The *conditional single-step upper-bound* rule uses the maximal delay $u_\tau \in \mathsf{N}$ of a transition $\tau \in \mathcal{T}$:

$$\diamond\text{-CSS} \quad \frac{p \rightarrow enabled(\tau) \quad \{p\}\,\tau\,\{\neg p\}}{(p \,\mathsf{U}\, q) \rightarrow \diamond_{\leq u_\tau} q}$$

Note that the second premise of this rule is trivially valid if τ becomes disabled by being taken, as is the case for all transitions of a real-time transition system that is associated with a shared-variables multiprocessing system (recall that we have ruled out selfloops in transition diagrams).

It is also worth pointing out that both conditional lower-bound and conditional upper-bound reasoning rely only on assumptions that are safety (unless) formulas.

To derive the upper bound 4 on the termination of our example P, we show first the untimed formula

$$start \rightarrow (at\_\ell_0 \,\mathsf{U}\, at\_\ell_1 \vee at\_\ell_0 \,\mathsf{U}\, at\_\ell_2).$$

By the conditional single-step upper-bound rule $\diamond$-**CSS** with respect to the transitions $\tau_{0\rightarrow 1}$ and $\tau_{0\rightarrow 2}$, we derive the conditional single-step bounds

$$(at\_\ell_0 \,\mathsf{U}\, at\_\ell_1) \rightarrow \diamond_{\leq 2}\, at\_\ell_1,$$
$$(at\_\ell_0 \,\mathsf{U}\, at\_\ell_2) \rightarrow \diamond_{\leq 2}\, at\_\ell_2.$$

They allow us to conclude

$$start \rightarrow (\diamond_{\leq 2}\, at\_\ell_1 \vee \diamond_{\leq 2}\, at\_\ell_2).$$

Now we can proceed by *unconditional* upper-bound reasoning to arrive at the desired bounded-response property

$$start \rightarrow \diamond_{\leq 4}\, at\_\ell_3.$$

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 Explicit-clock transition systems

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

- The *clock variable* $\mathbf{t}$ ranges over the natural numbers N; it records, in every state σ_i of a computation $\rho = (\sigma, \mathsf{T})$, the corresponding time T_i.
- The *delay counters* δ_τ, $\tau \in \mathcal{T}$, range over N; they record, in every state of a computation, for how many clock ticks the transition τ has been continuously enabled without being taken.

  We often write $\delta^i_{j\rightarrow k}$ short for $\delta_{\tau^i_{j\rightarrow k}}$.

The *explicit-clock transition system*

$$S^* = \langle V^*, \Sigma^*, \Theta^*, \mathcal{T}^*, \mathcal{W}, \mathcal{F} \rangle$$

associated with S is defined to be the following (untimed) fair transition system (in the sense of [MP89a]):

- $V^* = V \cup \{\mathbf{t}\} \cup \{\delta_\tau : \tau \in \mathcal{T}\}$.
- Σ^* contains all interpretations of V^*.
- Θ^* contains all extensions σ^* 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 τ^* such that $(\sigma^*_1, \sigma^*_2) \in \tau^*$ iff, for all $\tau' \in \mathcal{T}$,

  $(\sigma_1, \sigma_2) \in \tau$,
  $\sigma^*_1(\delta_\tau) \geq l_\tau$,
  $\sigma^*_2(\mathbf{t}) = \sigma^*_1(\mathbf{t})$,
  $$\sigma^*_2(\delta_{\tau'}) = \begin{cases} \sigma^*_1(\delta_{\tau'}) & \text{if } \tau' \neq \tau \text{ is enabled on } \sigma_2 \\ 0 & \text{otherwise.} \end{cases}$$

  Note that the second clause, $\sigma^*_1(\delta_\tau) \geq l_\tau$, enforces all lower bounds.

  In addition, $\mathcal{T}^*$ contains the *tick* transition τ_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.

- $\mathcal{W}$ contains, for every transition τ^* with $\tau \in \mathcal{T}$, a *weak-fairness* requirement that assures that no transition τ^* can be continuously enabled without being taken.
- $\mathcal{F}$ contains a *strong-fairness* requirement for the tick transition τ_T that assures that τ_T cannot be enabled infinitely often without being taken.

The real-time transition system S and the explicit-clock transition system S^* are related in the following way: for every initialized computation $\rho = (\sigma, \mathsf{T})$ of S, there is a unique (untimed) initialized computation σ^* of S^* such that, for all $i \geq 0$, the state σ_i^* is an extension of σ_i to V^* and $\sigma_i^*(\mathbf{t}) = \mathsf{T}_i$; and for every computation σ^* of S^*, the timed state sequence $(\sigma, \sigma^*(\mathbf{t}))$ is a computation of S if every state σ_i, $i \geq 0$, is the restriction of σ_i^* to V.

5.2 Explicit-clock formulas

Now we translate every bounded-invariance and bounded-response formula ϕ over V into an untimed unless formula ϕ^* that contains the clock variable $\mathbf{t}$. The *explicit-clock formula* ϕ^* is constructed such that it it S^*-valid iff ϕ is S-valid:

- The explicit-clock translation of the bounded-invariance formula $p \rightarrow \Box_{<l} q$ is

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

  for a *new*, *rigid* variable T (i.e., $T \notin V^*$ and $T = T'$).
- The explicit-clock translation of the bounded-response formula $p \rightarrow \Diamond_{\leq u} q$ is

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

  for a new, rigid variable T.

Both unless formulas use the rigid variable T to record the time of the p-state. In the case of bounded-response properties the explicit-clock translation exploits the fact that the time is guaranteed to reach and surpass $T + u$, for any value of T.

The proof that the explicit-clock formula ϕ^* is indeed S^*-valid iff ϕ is S-valid is given in the full paper ([HMP91]).

This result leads to an alternative and quite different approach to the verification of real-time properties: to prove the S-validity of a real-time property ϕ (over V), we establish instead the S^*-validity of the untimed safety formula ϕ^* (in fact, the S-validity of any temporal formula ϕ can be concluded in this way).

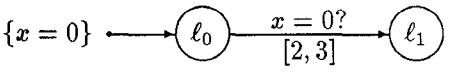
To show the unbounded unless formulas that result from translating bounded-invariance and bounded-response properties, a single timeless *unless* rule suffices:

UNLESS

$$\frac{p \rightarrow (\varphi \vee r) \qquad \{\varphi\}\, \mathcal{T}^*\, \{\varphi \vee r\} \qquad \varphi \rightarrow q}{p \rightarrow q \,\mathsf{U}\, r}$$

We emphasize that all premises are state formulas; the state formula φ is called the *invariant* 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,

$$\begin{aligned} \mathit{start} &\rightarrow (\neg \mathit{at}\_\ell_1) \,\mathsf{U}\, (\mathbf{t} \geq 2), \text{ and} \\ \mathit{start} &\rightarrow (\mathbf{t} \leq 3) \,\mathsf{U}\, \mathit{at}\_\ell_1, \end{aligned}$$

respectively, can be derived by the unless rule from the invariant

$$\mathit{at}\_\ell_0 \wedge x = 0 \wedge 0 \leq \mathbf{t} \leq 3 \wedge \mathbf{t} = \delta_{0 \rightarrow 1}.$$

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 (untimed) *liveness* properties ([OL82]), and uses relatively simple *local* invariants; the *explicit-clock* method employs only the simple unless rule — a (timeless) *safety* rule —, but requires a powerful *global* invariant.

This trade-off is illustrated vividly by a more elaborate example of real-time verification that is given at the end of the following section.

6 Completeness

The unless rule is known to be complete, relative to state reasoning, for establishing unless properties ([MP89b]). From the results of the previous section 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 show completeness in the case that all real-time constraints are either lower or upper bounds.

Theorem. *Let $S = \langle V, \Sigma, \Theta, \mathcal{T}, l, u \rangle$ be a real-time transition system such that either $l_\tau = 0$ for all $\tau \in \mathcal{T}$ or $u_\tau = \infty$ for all $\tau \in \mathcal{T}$. Let ϕ be a bounded-invariance or a bounded-response formula. If ϕ is S-valid, then it can be derived by the monotonicity, transitivity, and conditional single-step rules relative to untimed reasoning.*

Proof. We consider here only the case in which all maximal delays of S are ∞; the argument for the other case proceeds similarly and is given in the full paper ([HMP91]).

First we observe that, under the given restrictions, untimed reasoning is complete for timeless properties of S. This is because in the absence of finite maximal delays there is, modulo stuttering, a time sequence T for every (untimed) computation σ of the *weakly-fair* transition system S_W^- underlying S such that (σ, T) is a computation of S (add sufficiently many clock ticks between any two states). It follows that any untimed temporal formula that is S-valid is also S_W^--valid and, thus, can be established by untimed reasoning.

Any bounded-response property is either trivially not S-valid or can be established by untimed reasoning. Now suppose that the bounded-invariance property

$$p \rightarrow \Box_{<l} \neg q \qquad (\dagger)$$

is S-valid; we show that it can be derived within our proof system.

The main idea is to see that in order for (†) to be valid, for any p-state in a computation of S there has to be a sequence of nonoverlapping single-step lower bounds that add up to at least l before a q-state can be reached. We show that there are only finitely many such ways in which a q-state can be delayed for l time units; hence they can be enumerated by a single untimed formula.

Consider an arbitrary computation $\rho = (\sigma, \mathsf{T})$ of S such that σ_i, $i \geq 0$, is a p-state. Let σ_j be the first q-state with $j \geq i$; if no such state exists, let $j = \infty$. We write τ_k for the transition that is completed at position $k \geq 0$ of ρ.

An l-constraint pattern for $\sigma_{i..j}$ is a finite sequence of nonoverlapping single-step lower bounds between i and j that add up to at least l. Formally, a constraint pattern C is a sequence of transitions $\tau_{i_1}, \ldots \tau_{i_n}$. The pattern C is an l-constraint pattern iff

$$\sum_{1 \leq k \leq n} l_{\tau_{i_k}} \geq l;$$

it is a constraint pattern for $\sigma_{i..j}$ iff $i = i_0 < i_1 < \ldots < i_n \leq j$ and, for all $1 \leq k \leq n$, τ_{i_k} is not enabled on some $i_{k-1} \leq j_k < i_k$ (or $k = 1$ and $i = 0$).

Two constraint patterns are equivalent iff one is a subpattern of the other (i.e., can be obtained by omitting transitions). The proofs of the following two claims can be found in [HMP91].

Claim 1: There is an l-constraint pattern for $\sigma_{i..j}$.

Claim 2: There are only finitely many different equivalence classes of l-constraint patterns.

We add, for every transition $\tau \in \mathcal{T}$, the boolean variable $\mathit{completed}(\tau)$ to our language; it is intended to be true in a state σ_i, $i \geq 0$, of a computation $\rho = (\sigma, \tau)$ iff the transition τ is completed at position i of ρ. For our purpose, it turns out to be sufficient that $\mathit{completed}(\tau)$ satisfies the axiom

$$\{\mathit{true}\}\ \mathcal{T} - \tau\ \{\neg \mathit{completed}(\tau)\}. \qquad (\ddagger)$$

By Claim 1, there is an untimed formula of the form

$$(\neg q)\mathsf{U}(\neg q \wedge (\mathit{first} \vee \neg \mathit{enabled}(\tau_{i_0})))\mathsf{U}^+(\neg q)\mathsf{U}^+ \\ (\neg q \wedge \mathit{completed}(\tau_{i_0}))\mathsf{U}^+ \ldots \mathsf{U}^+(\neg q \wedge \mathit{completed}(\tau_{i_n}))$$

that is true over the i-th suffix of ρ. Since there are, by Claim 2, only finitely many formulas of this form, $p \rightarrow \psi$ for some finite disjunction ψ of nested unless formulas is S-valid and, thus, given by untimed reasoning.

From (‡) we infer by the conditional single-step lower-bound rule **U-CSS** with respect to any transition $\tau \in \mathcal{T}$ that

$$(\mathit{first} \vee \neg \mathit{enabled}(\tau))\ \mathsf{U}^+_{\geq l}\, \varphi\, \mathsf{U}^+\, (\mathit{completed}(\tau) \wedge \phi) \rightarrow \\ (\mathit{first} \vee \neg \mathit{enabled}(\tau))\ \mathsf{U}^+_{\geq l}\, \varphi\, \mathsf{U}^+_{\geq l_\tau}\, (\mathit{completed}(\tau) \wedge \phi)$$

for any state formula φ and temporal formula ϕ. Hence we can decorate the untimed nested unless formula with time bounds. By repeated collapsing and monotonicity similar to the sample lower-bound derivation of Section 4.4, we arrive at the desired bounded-invariance property (†). ■

In general, the situation is more complicated: *both* the lower- and upper-bound rules may be necessary to derive a bounded-invariance (or bounded-response) property.

This is demonstrated by the *increment-decrement* example in Figure 2. In this two-process system, 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.

Assuming that assignments cost at least 2 time units (instead of 1), tests still being free, the maximal value of y would be only 6, implying termination by time 80. It follows that the modification of individual *lower* bounds may affect a composite *upper* bound!

The proof of timely termination relies, accordingly, on an interplay of lower- and upper-bound rules. We

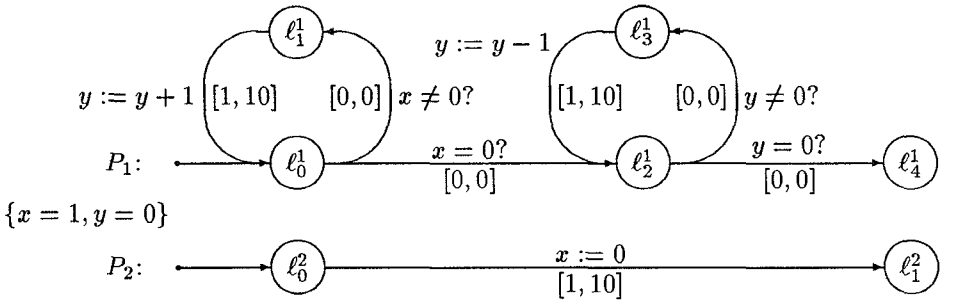


Figure 2: Increment-decrement

give only a brief sketch of the proof of the bounded-response property

$$start \rightarrow \Diamond_{\leq 130}\, at\_\ell_4^1;$$

more details, as well as a proof of the corresponding lower bound on termination, can be found in the full paper ([HMP91]).

The formal derivation follows the informal argument given above. First we show that within 10 time units P_1 can increase the value of y at most to 10:

$$start \rightarrow \Box_{<11}\,(0 \leq y \leq 10);$$

this is done by inductive lower-bound reasoning. Then we apply the new rule

$$\textbf{MIX}\quad \frac{u < l}{(\Diamond_{\leq u}\,\phi \wedge \Box_{<l}\,\psi) \rightarrow \Diamond_{\leq u}\,(\phi \wedge \psi)}$$

to the single-step upper bound

$$start \rightarrow \Diamond_{\leq 10}\,(at\_\ell_1^2 \wedge x = 0),$$

thus obtaining the bounded-response property

$$start \rightarrow \Diamond_{\leq 10}\,(0 \leq y \leq 10 \wedge at\_\ell_1^2 \wedge x = 0).$$

From here we proceed by pure upper-bound reasoning, performing a case analysis on the locations of P_1.

Alternatively, we can establish the upper bound 130 on the termination of P_1 by explicit-clock reasoning from the following global invariant (add that all variables are nonnegative):

$$\begin{array}{l}
(at_\ell_0^1 \wedge at_\ell_0^2 \wedge (y = \mathbf{t} = \delta^2_{0\to 1} = 0 \vee 1 \leq y \leq \mathbf{t} = \delta^2_{0\to 1})) \vee \\
(at_\ell_1^1 \wedge at_\ell_0^2 \wedge y + \delta^1_{1\to 0} \leq \mathbf{t} = \delta^2_{0\to 1}) \vee \\
(at_\ell_0^1 \wedge at_\ell_1^2 \wedge 1 \leq y \leq 11 \wedge \mathbf{t} \leq 20) \vee \\
(at_\ell_1^1 \wedge at_\ell_1^2 \wedge y \leq 10 \wedge \mathbf{t} \leq 10 + \delta^1_{1\to 0}) \vee \\
(at_\ell_2^1 \wedge at_\ell_1^2 \wedge y \leq 11 \wedge \mathbf{t} + 10y \leq 130) \vee \\
(at_\ell_3^1 \wedge at_\ell_1^2 \wedge 1 \leq y \leq 11 \wedge \mathbf{t} + 10y \leq 130 + \delta^1_{3\to 2}).
\end{array}$$

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.

[HMP91] T.A. Henzinger, Z. Manna, A. Pnueli, *Temporal Proof Methodologies for Real-time Systems*, Technical Report, Stanford University, 1991.

[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, W.-P. de Roever, "Rendez-vous 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.