Computing Accumulated Delays in Real-time Systems

Rajeev Alur[1], Costas Courcoubetis[2] *, Thomas A. Henzinger[3] **

[1] AT&T Bell Laboratories, Murray Hill
[2] Department of Computer Science, University of Crete, Greece
[3] Department of Computer Science, Cornell University, Ithaca

Abstract. We present a verification algorithm for duration properties of finite-state real-time systems. While simple real-time properties constrain the total elapsed time between events, duration properties constrain the accumulated time during which certain state predicates hold. We formalize the concept of durations by introducing duration measures for (dense-time) timed automata. Given a timed automaton with a duration measure, a start and a target state, and a duration constraint, the *duration-bounded reachability problem* asks if there is a run of the automaton from the start state to the target state such that the accumulated duration along the run satisfies the constraint. Our main result is a novel decision procedure for solving the duration-bounded reachability problem. We also prove that the problem is PSPACE-complete and demonstrate how the solution can be used to verify interesting duration properties of real-time systems.

1 Introduction

Over the past decade, model checking [CES86, QS82] has emerged as a powerful tool for the automatic verification of finite-state concurrent systems. Recently the model checking paradigm has been extended to real-time systems ([ACD90, EMSS90, AFH91, HNSY92]). Thus, given the description of a finite-state system together with its timing assumptions, there are algorithms to test whether such a system satisfies a specification written in a real-time temporal logic. A typical property specifiable in real-time temporal logics is the following *time-bounded causality* property:

> A response is obtained whenever a ringer has been pressed *continuously* for 2 seconds. — (*)

The standard real-time temporal logics, however, have limited expressiveness and cannot specify some properties we may want to verify of a given system. In

* Partially supported by the BRA ESPRIT project REACT.
** Supported in part by the National Science Foundation under grant CCR-9200794 and by the United States Air Force Office of Scientific Research under contract F49620-93-1-0056.

particular, they do not allow us to reason about the accumulated durations of state predicates. As an example, consider the following *duration-bounded causality* property:

> A response is obtained whenever a ringer has been pressed, *possibly intermittently*, for a total duration of 2 seconds. — (**)

To specify this duration property, we need to measure the accumulated time spent in the state that models "the ringer is pressed" over a given interval of time. For this purpose, the concept of duration operators on state predicates was introduced in the *Calculus of Durations* [CHR91]. In that paper, an axiom system is given to prove duration properties of real-time systems.

Here we address the *algorithmic* verification problem for duration properties of finite-state real-time systems. We use the formalism of timed automata [Dil89, AD90] to represent finite-state real-time systems. A timed automaton operates with a finite-state control and a finite number of fictitious time gauges called *clocks*, which allow the annotation of the state-transition graph of the system with timing constraints. The state of a timed automaton includes, apart from the location of the control, also the real-numbered values of all its clocks. Consequently, the state space of a timed automaton is infinite, and this complicates its analysis. The basic question about a timed automaton is the following *time-bounded reachability* problem:

> Given a start state σ, a target state σ', and an interval I, is there a run of the automaton starting in state σ and ending in state σ' such that the total elapsed time is in the interval I? — (†)

The solution to this problem relies on a partition of the state space into finitely many regions and the construction of a quotient called the *region graph* of the timed automaton [AD90]. The states within the same region are equivalent with respect to many standard questions. In particular, the region graph can be used for testing emptiness of a timed automaton [AD90], for checking time-bounded branching formulas [ACD90], for testing bisimulation equivalence of states [Č92], and for computing bounds on delays [CY91]. Unfortunately, the region graph is *not* adequate for testing the duration properties such as the duration-bounded causality property (**); that is, of two runs that start in different states within the same region, one may satisfy the duration-bounded causality property, whereas the other one does not. Thus new techniques are needed to analyze duration properties.

To introduce the concept of durations in a timed automaton, we associate a *duration measure*, a nonnegative integer, with each of the control locations. The duration measure of a location gives the rate at which the accumulated duration increases while the automaton control resides in the location. For example, a duration measure of 0 means that the time spent in the location is not accumulated, a duration measure of 1 means that the time spent in the location is accumulated, and a duration measure of 2 means that the integral increases at twice the rate of time. The time-bounded reachability problem (†) can now be generalized to the *duration-bounded reachability* problem as follows:

Given a start state σ, a target state σ', a duration measure, and an interval I, is there a run of the automaton starting in state σ and ending in state σ' such that the accumulated duration along the run is in the interval I? — (††)

We provide a solution to the duration-bounded reachability problem, and we show the problem to be PSPACE-complete. Our algorithm can be used to verify many interesting duration properties of finite-state real-time systems, such as the duration-bounded causality property (**).

Let us briefly outline our construction. Given a region R, a target state σ', and a path in the region graph from R to σ', we show that the lower and upper bounds on the accumulated duration over all the runs that start at some state in R and follow the chosen path, can be written as linear expressions over the variables that represent the clock values of the start state. In a first step, we provide an algorithm for computing these so-called *bound expressions*. In the next step, we define an infinite graph, the *bounds graph*, whose vertices are regions tagged with bound expressions that specify the set of possible accumulated duration values along any path to the target state. In the final step, we show how the infinite bounds graph can be collapsed into a finite graph to solve the duration-bounded reachability problem.

2 Timed Automata

Timed automata are a model for finite-state real-time systems [Dil89, AD90, ACD90]. Each automaton has a finite set of control locations and a finite set of real-valued clocks. All clocks proceed at the same rate, and thus each clock measures the amount of time that has elapsed since it was last reset. A transition of a timed automaton can be taken only if the current clock values satisfy the constraint that is associated with the transition. When taken, the transition changes the control location of the automaton and resets one of the clocks.

Formally, a *timed automaton* A is a triple (S, C, E), where

1. S is a finite set of *locations*;
2. C is a finite set of *clocks*;
3. E is a finite set of *transitions* of the form (s, s', μ, x), for a source location $s \in S$, a target location $s' \in S$, a clock constraint μ, and an associated clock $x \in C$. Each clock constraint is a boolean combination of atomic formulas of the form $y \leq k$ and $k \leq y$, for a clock $y \in C$ and a nonnegative integer constant $k \in \mathsf{N}$.

A configuration of the timed automaton A can be fully described by specifying the location of the control and the values of all clocks. A *clock valuation* $\nu \in \mathsf{R}^{|C|}$ is an assignment of nonnegative reals to the clocks in C. A *state* σ of A is a pair (s, ν) consisting of a location $s \in S$ and a clock valuation ν. We write Σ for the (infinite) set of states of A. As time elapses, the values of all clocks increase uniformly with time, thereby changing the state of A. Thus if the state

of A is (s, ν) at time t, then at time $t + \delta$, assuming that no transition occurs, the state of A will be $(s, \nu + \delta)$ where $\nu + \delta$ is the clock valuation that assigns $\nu(x) + \delta$ to each clock $x \in C$. The state of A may also change because of a transition $(s, s', \mu, x) \in E$. Such a transition can be taken only in a state whose location is s and whose clock valuation satisfies the constraint μ. The transition is instantaneous: after the transition, the automaton is in a state with location s' and the new clock valuation is $\nu[x := 0]$; that is, the clock $x \in C$ associated with the transition is reset to the value 0, and all other clocks remain unchanged.

The possible behaviors of the timed automaton A are then defined through a consecution relation on the states of A:

Transition successor For all states $(s, \nu) \in \Sigma$ and transitions $(s, s', \mu, x) \in E$, if $\nu \models \mu$ then $(s, \nu) \stackrel{0}{\Rightarrow} (s', \nu[x := 0])$.

Time successor For all states $(s, \nu) \in \Sigma$ and all time increments $\delta > 0$, $(s, \nu) \stackrel{\delta}{\Rightarrow} (s, \nu + \delta)$.

A state (s', ν') is a *successor* of the state (s, ν), written $(s, \nu) \Rightarrow (s', \nu')$, iff there exists a time value $\delta \geq 0$ such that $(s, \nu) \stackrel{\delta}{\Rightarrow} (s', \nu')$. The successor relation defines an infinite graph $K(A)$ on the state space Σ of A.

Depending on the application, a timed automaton may be augmented with additional components such as initial locations, accepting locations, input symbols as transition labels, or atomic propositions as location labels. We have chosen a very simple definition to illustrate the essential computational aspects of solving the reachability problems. Also, the original definition of a timed automaton allows a (possibly empty) set of clocks to be reset with each transition. Our requirement that precisely one clock is reset with each transition, does not affect the expressiveness.

Region Graphs

Let us review the known method for analyzing timed automata. The key to solving verification problems for a timed automaton is the construction of the so-called region graph of a timed automaton [AD90, ACD90]. The region graph of a timed automaton is a finite quotient of the infinite state graph that retains enough information to answer reachability questions.

Suppose we are given a timed automaton A and an equivalence relation $\cong$ on the states Σ of A. We write $[\sigma] \subseteq \Sigma$ for the equivalence class of states that contains the state $\sigma \in \Sigma$. The successor relation $\Rightarrow$ is extended as follows: we write $\sigma \Rightarrow [\sigma']$ if for some time value $\delta \geq 0$, $\sigma \stackrel{\delta}{\Rightarrow} \sigma'$ and $(\sigma + \delta') \in ([\sigma] \cup [\sigma'])$ for all $\delta' < \delta$. The *quotient graph* of A with respect to the given equivalence relation $\cong$, written $[K(A)]_{\cong}$, has an edge from $[\sigma]$ to $[\sigma']$ iff $\sigma \Rightarrow [\sigma']$. The equivalence relation $\cong$ is called *forward-stable* if whenever $\sigma \Rightarrow [\sigma']$ holds so does $\sigma'' \Rightarrow [\sigma']$ for all $\sigma'' \in [\sigma]$. The quotient graph with respect to a forward-stable relation can be used for solving reachability problems. If $\cong$ is forward-stable, and $[\sigma']$ is singleton, then $\sigma \Rightarrow^* \sigma'$ iff there is a path from $[\sigma]$ to $[\sigma']$ in the quotient graph $[K(A)]_{\cong}$.

The *region graph* $R(A)$ of a timed automaton A is a finite quotient graph with respect to the particular equivalence relation defined below. For $x \in C$, let c_x be the largest constant that the clock x is compared to in any constraint of A. For $t \in \mathsf{R}$, let $\langle t \rangle$ denote the fractional part of t, and let $\lfloor t \rfloor$ denote its integral part. Two states (s, ν) and (s, ν') are *region-equivalent*, written $(s, \nu) \cong (s, \nu')$, iff

1. for each clock $x \in C$, either $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ or both $\nu(x)$ and $\nu'(x)$ are greater than c_x; and
2. for all $x, y \in C$, $\langle \nu(x) \rangle \leq \langle \nu(y) \rangle$ iff $\langle \nu'(x) \rangle \leq \langle \nu'(y) \rangle$, and $\langle \nu(x) \rangle = 0$ iff $\langle \nu'(x) \rangle = 0$.

A *region* $R \subseteq \Sigma$ is a $\cong$-equivalence class of states. Note that a region is fully specified by a location, the integral parts of all clock values, and the ordering of the fractional parts of the clock values. For instance, if C contains three clocks x, y, and z, then the region $[s, x = 1, y = 0.2, z = 1.3]$ contains all states (s, ν) with $\nu(x) = 1$, $\lfloor \nu(y) \rfloor = 0$, $\lfloor \nu(z) \rfloor = 1$, and $0 < \langle \nu(y) \rangle < \langle \nu(z) \rangle$. Notice that there are only finitely many regions. The finiteness follows from the fact that the exact value of the integral part of a clock x is recorded only if it is smaller than c_x. The number of regions is bounded by $|S| \cdot 2^n \cdot n! \cdot \Pi_{x \in C}(c_x + 1)$, where n is the number of clocks.

The region graph $R(A)$ of the timed automaton A is the quotient graph with respect to the equivalence $\cong$. It is easy to check that the relation $\cong$ is forward-stable, and hence the region graph can be used to solve reachability problems. Let us define its edges explicitly. A region R is a *boundary region* iff there is some clock x such that R satisfies $\langle x \rangle = 0$. A region that is not a boundary region is called an *open region*. For a boundary region R, we define its predecessor region $pred(R)$ to be the open region R' such that for all states $(s, \nu) \in R'$, there is a time increment $\delta > 0$ such that $(s, \nu + \delta) \in R$ and $(s, \nu + \delta') \in R'$ for all $\delta' < \delta$. Similarly, let the successor region $succ(R)$ of R be the open region R' such that for all states $(s, \nu) \in R'$, there is a time value $\delta > 0$ such that $(s, \nu - \delta) \in R$ and $(s, \nu - \delta') \in R'$ for all $\delta' < \delta$. The state of an automaton belongs to a boundary region R only instantaneously. Just before that instant the state belongs to $pred(R)$, and just after that instant the state belongs to $succ(R)$. For example, for $R = [s, x = 1, y = 0.2, z = 1.3]$, $pred(R)$ is the equivalence class $[s, x = 0.9, y = 0.1, z = 1.2]$ and $succ(R)$ is the equivalence class $[s, x = 1.01, y = 0.21, z = 1.31]$.

The edges of the region graph $R(A)$ are defined as follows:

Transition edges If $(s, \nu) \stackrel{0}{\Rightarrow} (s', \nu')$, then there is an edge from the region $[s, \nu]$ to $[s', \nu']$.

Time edges For each boundary region R, there is an edge from $pred(R)$ to R, and an edge from R to $succ(R)$.

The region graph is useful in solving the time-bounded reachability problem for a timed automaton, and also for checking specifications written in the real-time logic TCTL [ACD90].

3 Reachability analysis for duration constraints

Duration constraints

Let A be a timed automaton. A *duration measure* is a function *dur* from the locations of A to the nonnegative integers. A *duration constraint* is of the form $\int dur \in I$, where *dur* is a duration measure, and I is an interval of the real line with integer end-points.

Let *dur* be a duration measure. We extend the state space of A to evaluate the integral $\int dur$ along the runs of A. An *extended state* is a pair (σ, ϵ) consisting of a state σ of A and a nonnegative real ϵ. The successor relation on states is extended as follows:

Transition successor For all extended states (s, ν, ϵ) and transitions (s, s', μ, x) such that $\nu \models \mu$, let $(s, \nu, \epsilon) \stackrel{0}{\Rightarrow} (s', \nu[x := 0], \epsilon)$.

Time successor For all extended states (s, ν, ϵ) and all time increments $\delta > 0$, let $(s, \nu, \epsilon) \stackrel{\delta}{\Rightarrow} (s, \nu + \delta, \epsilon + \delta \cdot dur(s))$.

We solve the following duration-bounded reachability problem between regions:

> Given an initial region R_0, a final region R_f, and a duration constraint $\int dur \in I$, whether there exists a state in R_0 from which a state in R_f can be reached such that the path satisfies the constraint $\int dur \in I$; that is, whether for some $\sigma \in R_0$, $\sigma' \in R_f$, and $\delta \in I$, $(\sigma, 0) \Rightarrow^* (\sigma', \delta)$ holds.

Bound-labeled regions

Let R_0 and R_f be two regions and let *dur* be a duration measure. We will determine the set I of possible values of the integral $\int dur$ such that $(\sigma, 0) \Rightarrow^* (\sigma', \delta)$ for some $\sigma \in R_0$, $\sigma' \in R_f$, and $\delta \in I$. To compute the lower and upper bounds on the integral $\int dur$ along a path of the region graph, we refine the graph by labeling all regions with expressions that describe the extremal values of the integral.

We build an infinite graph with vertices of the form (R, ℓ, u), where R is a region, and ℓ and u are linear expressions over the clock variables. The intended meaning of the bound expressions ℓ and u is that from a state $(s, \nu) \in R$, a state in the target region R_f can be reached, and in moving to R_f, the set of possible values of the integral $\int dur$ has infimum ℓ and supremum u, both of which are functions of the current clock values ν.

The bound expressions ℓ and u labeling a region R will be of a special form. Suppose that $C = \{x_1, \ldots, x_n\}$ is the set of clock variables and that for all states $(s, \nu) \in R$, ν satisfies

$$0 \leq \langle x_1 \rangle \leq \cdots \leq \langle x_n \rangle < 1;$$

that is, x_1 is the clock with the smallest fractional part and x_n is the clock with the largest fractional part. The fractional parts of all n clocks partition the unit interval into $n + 1$ subintervals of length $e_0, \ldots, e_n$:

$$
\begin{aligned}
e_0 &= \langle x_1 \rangle, \\
e_1 &= \langle x_2 \rangle - \langle x_1 \rangle, \\
&\vdots \\
e_{n-1} &= \langle x_n \rangle - \langle x_{n-1} \rangle, \\
e_n &= 1 - \langle x_n \rangle.
\end{aligned}
$$

A *bound expression* for R is a positive linear combination of the expressions $e_0, \ldots, e_n$. We denote bound expressions by $(n+1)$-tuples of nonnegative integer coefficients and write $(a_0, \ldots, a_n)$ for the bound expression $(a_0 \cdot e_0 + \cdots a_n \cdot e_n)$. For any expression e and a clock valuation ν, we write $[e]_\nu$ to denote the result of evaluating e using the clock values given by ν. Note that when time advances, the value of a bound expression changes at $a_0 - a_n$ times the rate of time.

A *bound-labeled region* (R, ℓ, u) consists of a region $R \in R(A)$, and two bound expressions ℓ and u for R such that for every state $(s, \nu) \in R$, $[u]_\nu \geq [\ell]_\nu$. We construct $B_{dur,R_f}(A)$, the *bounds graph* of A for duration measure *dur* and target region R_f. The vertices of $B_{dur,R_f}(A)$ are the bound-labeled regions of A and the special vertex R_f.

We begin with finding bounds for the paths that reach R_f without going through any other regions. Suppose that R_f is an open region with duration measure d. The target region R_f is reachable from a state $(s, \nu) \in R_f$ by remaining in R_f for at least by 0 and at most $[1 - \langle x_n \rangle]_\nu$ time units. Hence we add an edge in the bounds graph to R_f from (R_f, ℓ, u) for $\ell = (0, \ldots, 0, 0)$ and $u = (0, \ldots, 0, d)$. If R_f is a boundary region, we add an edge to R_f from (R_f, ℓ, u) for $\ell = u = (0, \ldots, 0, 0)$.

Now let us look at longer paths to reach the target region R_f. For each edge from R to R' in the region graph, the bounds graph has an edge from (R, ℓ, u) to (R', ℓ', u') if the bound expressions ℓ and u are related to ℓ' and u' according to certain rules. We discuss here only the lower bounds; the rules for updating the upper bounds are similar and will be given in the full paper.

First let us consider an example. Suppose that $C = \{x, y, z\}$ and the boundary region R_1, which satisfies $0 = \langle x \rangle < \langle y \rangle < \langle z \rangle$, is labeled with the lower bound $\ell_1 = (0, a, b, c)$. This means that starting from a state $(s, \nu) \in R_1$, the lower bound on the integral $\int dur$ for reaching some state in R_f is

$$[\, a \cdot \langle y \rangle + b \cdot (\langle z \rangle - \langle y \rangle) + c \cdot (1 - \langle z \rangle) \,]_\nu.$$

Now consider the open predecessor region R_2 satisfying $0 < \langle y \rangle < \langle z \rangle < \langle x \rangle$. There is a time edge from R_2 to R_1 in the region graph. Let d_2 be the duration measure of R_2. We want to compute the lower bound label ℓ_2 for R_2 from ℓ_1. Staring in a state $(s, \nu) \in R_2$, the state (s, ν'), $\nu' = \nu + \delta$, of the region R_1 is reached after a time increment of δ that equals $[1 - \langle x \rangle]_\nu$, and

$$
\begin{aligned}
[\langle y \rangle]_{\nu+\delta} &= [\langle y \rangle + (1 - \langle x \rangle)]_\nu, \\
[\langle z \rangle - \langle y \rangle]_{\nu+\delta} &= [\langle z \rangle - \langle y \rangle]_\nu, \\
[1 - \langle z \rangle]_{\nu+\delta} &= [\langle x \rangle - \langle z \rangle]_\nu.
\end{aligned}
$$

Furthermore, from any state $(s, \nu) \in R_2$ the integral increases by $[d_2 \cdot (1 - \langle x \rangle)]_\nu$ before entering the region R_1. Hence, the new lower bound is

$$[\, a \cdot \langle y \rangle + a \cdot (1 - \langle x \rangle) + b \cdot (\langle z \rangle - \langle y \rangle) + c \cdot (\langle x \rangle - \langle z \rangle) + d_2 \cdot (1 - \langle x \rangle)\,]_\nu$$

and the label ℓ_2 is $(a, b, c, a + d_2)$. Next consider the predecessor region R_3 satisfying $0 = \langle y \rangle < \langle z \rangle < \langle x \rangle$ and the time edge from region R_3 to region R_2. The reader can verify that the updated lower bound label ℓ_3 of R_3 is $(a, b, c, a + d_2)$, same as ℓ_2.

The process repeats if we consider further time edges, so let us consider a transition edge from region R_4 to region R_3. We assume that the region R_4 is open with duration measure d_4, satisfies $0 < \langle z \rangle < \langle y \rangle < \langle x \rangle$, and the corresponding transition resets the clock y. Given a state $(s, \nu) \in R_4$, let ν' be the clock values immediately after the transition. The choice of ν' depends on when the transition occurs, and we want to minimize $[\ell_3]_{\nu'}$ over all such possible choices. First observe that independently of when the transition occurs

$$[\langle x \rangle - \langle z \rangle]_{\nu'} \;=\; [\langle x \rangle - \langle z \rangle]_\nu \;=\; [(\langle y \rangle - \langle z \rangle) + (\langle x \rangle - \langle y \rangle)]_\nu .$$

If the transition occurs immediately, then

$$\begin{aligned} [\langle z \rangle - \langle y \rangle]_{\nu'} &= [\langle z \rangle]_\nu , \\ [1 - \langle x \rangle]_{\nu'} &= [1 - \langle x \rangle]_\nu , \end{aligned}$$

and in this case the updated lower bound ℓ_4^1 is $(b, c, c, a + d_2)$. On the other hand, if the transition happens as late as possible the lower bound is computed as follows. Let ν_1 be the clock values just prior to the transition, where $[\langle x \rangle]_{\nu_1}$ is arbitrarily close to 1. Then the lower bound ℓ_4^2 starting from (s, ν) is the sum of $[d_4 \cdot (1 - \langle x \rangle)]_\nu$ ($1 - \langle x \rangle$ is the time to reach (s, ν_1)) and of the lower bound starting from (s', ν'). The later is equal to $[b \cdot (\langle z \rangle - \langle y \rangle) + c \cdot (\langle x \rangle - \langle z \rangle)]_{\nu'}$, and since $[\langle z \rangle - \langle y \rangle]_{\nu'} = [\langle z \rangle + (1 - \langle x \rangle)]_\nu$, this is equal to

$$[b \cdot (\langle z \rangle + (1 - \langle x \rangle)) + c \cdot (\langle x \rangle - \langle y \rangle) + c \cdot (\langle y \rangle - \langle z \rangle)]_\nu .$$

Hence, ℓ_4^2 is $(b, c, c, b + d_4)$. The new lower bound label ℓ_4 is ℓ_4^1 if $a + d_2 \leq b + d_4$, and ℓ_4^2 otherwise.

We now formally define the edges of the bounds graph. Suppose that the region graph has an edge from R to R' and let d be the duration measure of R. Then the bounds graph has an edge from (R, ℓ, u) to (R', ℓ', u'), where the lower bound labels $\ell = (a_0, a_1, \ldots, a_n)$ and $\ell' = (a'_0, a'_1, \ldots, a'_n)$ are related as follows (and the upper bound labels are related according to similar rules). There are various cases to consider, depending on whether the edge from R to R' is a time edge or a transition edge:

Time edge 1 R' is a boundary region and $R = \mathit{pred}(R')$ is an open region:
for all $0 \leq i < n$, $a_i = a'_{i+1}$;
$a_n = a'_0 + d$.

Time edge 2 R is a boundary region and $R' = \mathit{succ}(R)$ is an open region:

for all $0 \leq i \leq n$, $a_i = a'_i$;

Transition edge R' is a boundary region, and suppose the clock with the k-th largest fractional part in R is reset:

for all $0 \leq i < k$, $a_i = a'_{i+1}$;
for all $k \leq i < n$, $a_i = a'_i$.

If $a'_n \leq a'_1 + d$, then the lower bound is achieved when the reset occurs immediately; in this case

$a_n = a'_n$.

If $a'_n > a'_1 + d$, then the lower bound is achieved when the reset is delayed as much as possible; in this case

$a_n = a'_1 + d$.

Reachability in the bounds graph

Given a state $\sigma = (s, \nu)$ and bound expressions ℓ and u, we define the interval $I(\ell, u, \sigma)$ of the real line as follows: if $[\ell]_\nu = [u]_\nu$, then $I(\ell, u, \sigma)$ is the singleton set containing $[\ell]_\nu$; otherwise $I(\ell, u, \sigma)$ denotes the open interval $([\ell]_\nu, [u]_\nu)$. The next lemma indicates how the bounds graph can be used to solve reachability problems with duration constraints.

Lemma 1. *For any region $R_f \in R(A)$, time value $\delta \in \mathsf{R}$, and state $\sigma \in \Sigma$, there exists a state $\sigma' \in R_f$ such that $(\sigma, 0) \Rightarrow^* (\sigma', \delta)$ iff there is a bound-labeled region $([\sigma], \ell, u)$ in the bounds graph $B_{dur,R_f}(A)$ from which there is a path to R_f and $\delta \in I(\ell, u, \sigma)$.*

We need some more definitions. For a bound-labeled region $B = (R, \ell, u)$, let $I(B)$ denote the union $\bigcup_{\sigma \in R} I(\ell, u, \sigma)$. It is easy to check that for any B, the set $I(B) \subseteq \mathsf{R}$ is an interval with integer endpoints. The left endpoint is the infimum of $[\ell]_\nu$ over all possible choices of ν subject to the ordering of the fractional parts of the clock values in R. This infimum is simply the smallest coefficient in the tuple representing ℓ. Similarly, the right endpoint of $I(B)$ is the supremum of $[u]_\nu$ over the allowed ordering of the fractional parts, and is the largest coefficient in u.

Now the desired reachability property between R_0 and R_f holds iff there is a bound-labeled region $B = (R_0, \ell, u)$ such that (i) there is a path from B to R_f in the bounds graph $B_{dur,R_f}(A)$, and (ii) $I(B) \cap I \neq \emptyset$.

To test the desired property, we start building the vertices of $B_{p,R_f}(A)$ from which R_f is reachable. This can be done in a breadth-first fashion starting from the target vertex R_f. On a particular path, the same region may appear with different bound expressions, but when the coefficients of the bound expressions become "sufficiently" large, some collapsing is possible.

Collapsing the bounds graph

Given a constant K, we define an equivalence relation over bound expressions as follows.

Let e_j, $j = 1, 2$, be bound expressions given by $(a_0^j, \ldots a_n^j)$. Define $e_1 \cong_K e_2$ iff for $0 \leq i \leq n$, either a_i^1 equals a_i^2 or both exceed K.

The equivalence relation $\cong_K$ is extended over bound-labeled regions as follows.

Two bound-labeled regions (R_1, ℓ_1, u_1) and (R_2, ℓ_2, u_2) are said to be equivalent with respect to $\cong_K$ iff (1) $R_1 = R_2$, (2) $\ell_1 \cong_K \ell_2$, and (3) $u_1 \cong_K u_2$.

We say that a bound expression e is K-bounded if all the coefficients in e are at most K. Clearly, for each bound expression e, there exists a K-bounded expression e' such that $e \cong_K e'$. For a given region R, the number of K-bounded expressions is at most $(K+1)^{n+1}$, and hence

Lemma 2. *For any K, the number of different equivalence classes of bound-labeled regions under $\cong_K$ is bounded by $|S| \cdot n! \cdot 2^n \cdot (K+1)^{2(n+1)} \cdot \Pi_{x \in C} c_x$, where $n = |C|$.*

Thus the number of equivalence classes of bound-labeled regions is exponential in the length of the input to the problem. Also the equivalence relation $\cong_K$ is a bisimulation, or is stable with respect to reachability:

Lemma 3. *If the bounds graph $B_{dur,R_f}(A)$ contains an edge from a bound-labeled region B_1 to a bound-labeled region B_2, and $B_1 \cong_K B_3$, then there exists a bound-labeled region B_4 such that $B_2 \cong_K B_4$ and the bounds graph contains an edge from B_3 to B_4.*

Recall that we are searching for a bound-labeled region B from which R_f is reachable and $I(B)$ has nonempty intersection with I. Also recall that the end-points of $I(B)$ are obtained by choosing the smallest and the largest coefficients from the associated bound expressions. If two bound-labeled regions B_1 and B_2 are equivalent with respect to $\cong_K$, then the corresponding end-points of the two intervals $I(B_1)$ and $I(B_2)$ are either the same or greater than K. This leads to the following:

Lemma 4. *Consider two bound-labeled regions B_1 and B_2, and an interval I whose end-points do not exceed K. If $B_1 \cong_K B_2$, then $I \cap I(B_1) = \emptyset$ iff $I \cap I(B_2) = \emptyset$.*

This gives an algorithm for deciding the desired property. If the duration constraint we want to test is $\int dur \in I$, then we choose K to be the right end-point of I. We start building the bounds graph starting from the target vertex R_f in a breadth-first manner. The coefficients of the bound expressions are always bounded by K by identifying equivalent bound-labeled regions. The desired property holds upon visiting a vertex $B = (R_0, \ell, u)$ with $I(B) \cap I \neq \emptyset$. The property does not hold if the search terminates otherwise. The time complexity of the search is linear in the number of regions labeled with K-bounded expressions. Notice that each such bound-labeled region can be represented in space polynomial in the length of the input, and hence the search can be performed in PSPACE:

Theorem 5. *Given an initial region R_0 and a target region R_f, and a duration constraint $\int dur \in I$, the problem of testing whether there exist states $\sigma \in R_0$ and $\sigma' \in R_f$, and a time value $\delta \in I$, such that $(\sigma, 0) \Rightarrow^* (\sigma', \delta)$ is decidable in PSPACE.*

We point out that the reachability problem for a timed automaton is PSPACE-hard, and this implies that the duration-bounded reachability is also PSPACE-hard.

4 Discussion

In the previous section we solved the duration-bounded reachability problem between two specified regions. The construction can be used for many related problems as outlined below.

It should be clear that the initial or final region can be replaced either by a specified state (with rational clock values) or by a specified location (i.e., a set of regions). For instance, suppose we are given an initial state σ, a target state σ', a duration constraint $\int dur \in I$, and we want to decide whether $(\sigma, 0) \Rightarrow (\sigma', \delta)$ for some $\delta \in I$. Assuming σ and σ' assign *rational* values to all the clocks, we can choose an appropriate time unit so that the regions $[\sigma]$ and $[\sigma']$ are singletons. Then the solution for the reachability problem between the regions $[\sigma]$ and $[\sigma']$ is actually a solution to the reachability between the states. Thus the reachability problem between states (††) is also solvable in PSPACE.

Another example of a property we can test is the following. Given a finite-state system modeled as a timed automaton, and integers δ, a, and b, we want to verify whether in any time span of length δ, the system always spends at least a and at most b accumulated time units in a given set of locations. For instance, consider a railroad crossing, similar to the one that appears in various papers on real-time verification. If the minimum separation between two consecutive trains entering the gate is known, then we can use our algorithm to verify a property such as "in any interval of length 1 hour, the gate is closed for at most 5 minutes." Notice that testing this property requires computing the total accumulated delay for which the gate is closed.

Now let us outline how to test a duration property such as the duration-bounded causality (**) which says that "a response is obtained whenever a ringer has been pressed, possibly intermittently, for a total duration of 2 seconds." Assume that each location of the timed automaton is labeled with state predicates p (denoting that the ringer is pressed), and q (denoting the response). The duration measure is defined so that $dur(s) = 1$ if p is true in s, and $dur(s) = 0$ if p is false in s. The labeling of the locations is also extended to regions and bound-labeled regions. The desired duration-bounded causality property *does not* hold iff there are regions R_0, R_f, q holding in R_f, and a bound labeled region $B = (R_0, \ell, u)$ such that $I(B) \cap (2, \infty) = \emptyset$, and in the bounds graph B_{dur,R_f}, there is a path from B to R_f that visits only $(\neg q)$-labeled vertices. In general, the algorithm for testing reachability can be extended to incorporate simple (untimed) logical properties of the paths.

The duration-bounded reachability problem has been studied, independently, in [KPSY92] also. Their approach is quite different from ours as indicated below. They first show that the problem is solvable in the discrete-time case; i.e., in the case where all the transitions of the timed automaton are assumed to occur at integer time values. Next they prove that the discrete-time solution is actually a solution to the original problem (i.e., the dense-time case) under the following two assumptions: (i) the constraints of the timed automaton use only positive Boolean combinations of non-strict inequalities, and the automaton can take many transition steps in zero time, and (ii) the duration constraint is of the form $\int dur \sim k$ for the comparison relation $\sim \in \{\leq, <, \geq, >\}$ and $k \in \mathsf{N}$. The first requirement ensures that the runs of the timed automaton are closed under *digitization* (i.e. truncating real time values with respect to arbitrary $\epsilon \in [0, 1)$). The second requirement rules out the duration constraints of the form $\int dur = 2$ or $2 \leq \int dur \leq 3$. This approach of proving that the discrete and dense solutions coincide gives a simpler solution than ours, and it also admits duration measures that associate negative rates with some locations. However, both the requirements (i) and (ii) are crucial for this approach, and for computing duration properties in general, the analysis is likely to require techniques designed for dense-time case. We also note that, when the timed automaton contains a single clock, [KPSY92] gives an algorithm to check more complex duration constraints such as $\int dur \in I \wedge \int dur' \in I'$ for different duration measures dur and dur'.

An alternative approach to analyzing duration constraints of a timed automaton is to add variables that measure accumulated durations directly to the timed-automaton model. An *integration graph* [KPSY92] is essentially a timed automaton where the accumulated duration $\int dur$, called the *integrator*, is a component of the automaton state. The integrator, like an ordinary clock, can be reset with a transition of the automaton, and the constraints associated with the transitions can test integrator values also. The reachability problem for such integration graphs is undecidable even in a very restricted setting [KPSY92, ACHH92]. In this paper, the duration constraints are not a part of the system, but constitute a part of the property we want to test. Thus, this distinction between whether we strengthen the model or the specification language with the duration constraints, is crucial to the decidability of the resulting verification problem.

The expressiveness of the specification language can be increased further; it is possible to define a temporal logic with duration constraints. While our construction allows immediately the test of validity for some formulas of such a logic over a timed automaton, the symbolic fixpoint computation procedure of [ACHH92] gives a semi-decision procedure for model-checking problem. The decidability of the model checking problem remains an open problem. The model checking problem is closely related to the following problem:

> Given an initial region R_0, a target state σ', a duration constraint $\int dur \in I$, compute the set $R_0' \subseteq R_0$ consisting of states $\sigma \in R_0$ for which there exists $\delta \in I$ such that $(\sigma, 0) \Rightarrow^* (\sigma', \delta)$.

Each bound-labeled region (R_0, ℓ, u) from which R_f is reachable in the bounds

graph $B_{dur,R_f}(A)$ contributes the subregion $\{\sigma \in R_0 \mid I \cap I(\ell, u, \sigma) \neq \emptyset\}$ to finding R_0'. In general, this is an infinite union, possibly of singleton sets, and hence seems hard to manipulate.

References

[ACD90] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking for real-time systems. In *Proceedings of the Fifth IEEE Symposium on Logic in Computer Science*, pages 414–425, 1990.

[ACHH92] R. Alur, C. Courcoubetis, T.A. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Proceedings of the Workshop on Theory of Hybrid Systems*, 1992. To appear.

[AD90] R. Alur and D.L. Dill. Automata for modeling real-time systems. In *Automata, Languages and Programming: Proceedings of the 17th ICALP*, Lecture Notes in Computer Science 443, pages 322–335. Springer-Verlag, 1990.

[AFH91] R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. In *Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing*, pages 139–152, 1991.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal-logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

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

[CY91] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. In *Proceedings of the Third Workshop on Computer-Aided Verification*, Lecture Notes in Computer Science 575, pages 399–409, 1991.

[Dil89] D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, Lecture Notes in Computer Science 407, pages 197–212. Springer-Verlag, 1989.

[EMSS90] E.A. Emerson, A.K. Mok, A.P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In E.M. Clarke and R.P. Kurshan, editors, *Computer-Aided Verification, 2nd International Conference, CAV'90*, Lecture Notes in Computer Science 531, pages 136–145, 1990.

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

[KPSY92] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In *Proceedings of the Workshop on Theory of Hybrid Systems*, 1992. To appear.

[QS82] J.P. Queille and J. Sifakis. Specification and verification of concurrent programs in CESAR. In *Proceedings of the 5th International Symposium on Programming*, Lecture Notes in Computer Science 137, pages 195–220, 1982.

[Čer92] K. Čerāns. Decidability of bisimulation equivalence for parallel timer processes. In *Proceedings of the Fourth Workshop on Computer-Aided Verification*, Lecture Notes in Computer Science, 1992. To appear.