Decidable Approximations on Generalized and Parameterized Discrete Timed Automata

Zhe Dang[1], Oscar H. Ibarra[2], and Richard A. Kemmerer[2]

[1] School of Electrical Engineering and Computer Science
Washington State University, Pullman, WA 99164
zdang@eecs.wsu.edu
[2] Department of Computer Science
University of California, Santa Barbara, CA 93106
{ibarra,kemm}@cs.ucsb.edu

Abstract. We consider generalized discrete timed automata with general linear relations over clocks and parameterized constants as clock constraints and with parameterized durations. We look at three approximation techniques (i.e., the r-reset-bounded approximation, the B-bounded approximation, and the $\langle B, r\rangle$-crossing-bounded approximation), and derive automata-theoretic characterizations of the binary reachability under these approximations. The characterizations allow us to show that the safety analysis problem is decidable for generalized discrete timed automata with unit durations and for deterministic generalized discrete timed automata with parameterized durations. An example specification written in ASTRAL is used to run a number of experiments using one of the approximation techniques.

1 Introduction

As a standard model for analyzing real-time systems, timed automata [3] have received enormous attention during the past decade. A fundamental result in the theory of timed automata shows that region reachability for timed automata is decidable [3]. This result has been very useful in defining various real-time logics, appropriate model checking algorithms and tools [2, 4, 16, 17, 24, 25, 27–29] for verifying real-time systems (see [1] for a survey).

However, not every real-time system can be modeled as a timed automaton since a complex (not necessarily large) real-time system might contain non-region (e.g., Presburger) clock constraints. However, the "Turing computing" power of such augmented automata prevents automatic verification of properties such as reachability [3, 5]. Therefore, decidable approximation techniques are of great interest, since these techniques would provide a user some degree of confidence in analyzing and debugging complex real-time specifications. In contrast to the most direct approximation techniques [5, 10–12] that bound the number of transitions to a fixed number, the approximation techniques presented in this paper restrict the clock behaviors but do not necessarily bound the number of transition iterations to be finite.

In this paper, we focus on timed automata with integer-valued clocks, i.e., discrete timed automata, and extend them to generalized discrete timed automata by allowing

general linear relations over clocks and parameterized constants as clock constraints. Furthermore, the duration of a transition can be a parameterized constant. These generalizations have practical motivation. For example, many complex real-world specifications [6, 8, 10, 22] written in the real-time specification language ASTRAL [6] use generalized clock constraints and parameterized durations in almost every specification. Therefore, the results presented in this paper may be useful in implementing an automatic specification debugger for complex real-time specification languages like ASTRAL.

We investigate three approximation techniques in this paper. The r-reset-bounded approximation bounds the number of clock resets by a given positive integer r for each clock. The B-bounded approximation requires that each clock reset happens before its value exceeds a given (integral) time B. Combining these two, the $\langle B, r\rangle$-crossing-bounded approximation requires that for each clock, there are at most r times that the clock resets after its value exceeds B. Given an approximation technique, we will focus on the binary reachability characterization of a generalized discrete timed automaton. Binary reachability has recently been proposed and investigated for a number of timed systems [7, 9, 14, 15], which makes it possible to reason about "non-region" properties for these systems. We first show that, when a generalized discrete timed automaton has unit duration, the binary reachability under any one of the three approximations is Presburger. Then by considering a generalized discrete timed automaton with parameterized durations, we show that the binary reachability under any one of the three approximations has a 2DCM-padding when the machine is deterministic. Specifically, we show that the "padded language" for binary reachability can be accepted by a deterministic two-way counter machine with one reversal-bounded counter [20]. The case for nondeterministic generalized discrete timed automata is open. These are good characterizations in the sense that the validity of Presburger formulas as well as the emptiness problem for these counter machines are decidable. This allows us to establish, in principle, decidable results for the (Presburger) safety analysis problem for generalized discrete timed automata under the approximations. The 2DCM-padding characterization is particularly interesting, since binary reachability is not necessarily semilinear.

Due to space limitation, all the proofs are omitted. For a complete exposition see [13].

2 Preliminaries

Let V be a finite set of variables over the integers. An *atomic linear relation* on V is defined as $\Sigma_{v\in V} a_v v < b$, where a_v and b are integers. A *linear relation* on V is constructed from a finite number of atomic linear relations using $\neg$ and $\wedge$. $\mathcal{L}_V$ denotes the set of all linear relations on V. An *atomic Presburger relation* on V is either an atomic linear relation on V or a linear congruence $\Sigma_{v\in V} a_v v = b \bmod d$, where a_v, b and d are integers. A Presburger formula can always be constructed from atomic Presburger relations using $\neg$ and $\wedge$. Let N be the set of integers with N^+ denoting the set of nonnegative integers. A *clock* is a variable over N^+. A *generalized discrete timed automaton* $\mathcal{A}$ is a tuple $\langle S, \boldsymbol{C}, \boldsymbol{X}, E\rangle$ where S is a finite set of *(control) states*, $\boldsymbol{X}$ is a finite set of clocks, $\boldsymbol{C}$ is a finite set of parameterized constants,

and $E \subseteq S \times (\boldsymbol{C} \cup \{0,1\}) \times 2^{\boldsymbol{X}} \times \mathcal{L}_{\boldsymbol{X} \cup \boldsymbol{C}} \times S$ is a finite set of edges. Each edge $\langle s, d, \lambda, l, s' \rangle$ denotes a *transition (or an edge)* from s to s' with *enabling condition* $l \in \mathcal{L}_{\boldsymbol{X} \cup \boldsymbol{C}}$. $d \in \boldsymbol{C} \cup \{0,1\}$ is a parameterized constant indicating the duration of this transition. $\lambda \subseteq \boldsymbol{X}$ is the set clocks that are reset as a result of this transition. When λ is empty, the duration, which is a parameterized constant in $\boldsymbol{C}$ or integer constant 1, must be positive. Clock resets take no time. Thus, when λ is not empty, the duration d on this edge is simply 0.

The semantics is defined as follows. $\alpha \in S \times (N^+)^{|\boldsymbol{C}|} \times (N^+)^{|\boldsymbol{X}|}$ is called a *configuration* with α_q being the state under this configuration. α_{x_i} and α_{c_j} denote the value of the clock x_i and the value of the parameterized constant c_j, respectively. Note that each clock and parameterized constant are nonnegative, i.e., in N^+. $\alpha \rightarrow^{\langle s,d,\lambda,l,s' \rangle} \alpha'$ denotes a one-step transition along an edge in $\mathcal{A}$ satisfying

- Constant values do not change, i.e., for each $c \in \boldsymbol{C}$, $\alpha_c = \alpha'_c$,
- The state s is set to a new location s', i.e., $\alpha_q = s, \alpha'_q = s'$,
- Each clock changes according to the edge given. When there is no clock reset on this edge, i.e., $\lambda = \emptyset$, all the clocks synchronously progress by the amount of the duration, i.e., for each $x \in \boldsymbol{X}$, $\alpha'_x = \alpha_x + \alpha_d$. In this case, the duration is positive, i.e., $\alpha_d > 0$. When $\lambda \neq \emptyset$, clocks in λ reset to 0 and all the other clocks are unchanged. Thus, clock resets take no time. That is, for each $x \in \lambda$, $\alpha'_x = 0$ and for each $x \notin \lambda$, $\alpha'_x = \alpha_x$.
- The enabling condition is satisfied. That is, $l(\alpha)$ is true.

We simply write $\alpha \rightarrow \alpha'$ if α can reach α' by a one-step transition. $\mathcal{A}$ is *deterministic* if for any configuration α there is at most one β such that $\alpha \rightarrow \beta$. A *path* $\alpha_0 \cdots \alpha_k$ satisfies $\alpha_i \rightarrow \alpha_{i+1}$ for each i. Write $\alpha \rightsquigarrow_{\mathcal{A}} \beta$ if α reaches β through a path. In the definition of $\mathcal{A}$, there is no input. This is because the input is always one-way for timed automata and, therefore, input symbols can be built into the control states. $\mathcal{A}$ has parameterized durations. If we restrict each duration on an edge without clock resets to be 1, then $\mathcal{A}$ is called a generalized discrete timed automaton *with unit durations*.

When $\mathcal{A}$ with unit durations contains no parameterized constants and enabling conditions are *clock constraints* or *clock regions* in the form of Boolean combinations of $x - y \# c$ and $x \# c$, where c is an integer, x and y are clocks and $\# \in \{<, =, >\}$, $\mathcal{A}$ is equivalent to the standard timed automata (with integer-valued clocks) [3]. On the other hand, when $\mathcal{A}$ with unit durations contains enabling conditions only in the form of Boolean combinations of $x \# c$, where c is a parameterized constant or an integer, x is a clock (when c is a parameterized constant, x is called a *parameterized clock*), $\mathcal{A}$ is a *parameterized timed automaton* (with integer-valued clocks) [5].

There are two kinds of reachability for $\mathcal{A}$. They are state reachability and binary reachability. Assume a state s_0 is designated as the *initial state* of $\mathcal{A}$. A state $s \in S$ is *state reachable* in $\mathcal{A}$ if there is an initial configuration α_0 (whose state is s_0 and whose clock values are all 0) and a configuration α with $\alpha_q = s$ such that α is reachable from the initial configuration, i.e., $\alpha_0 \rightsquigarrow_{\mathcal{A}} \alpha$. The state reachability set is the set of all states s such that s is state reachable in $\mathcal{A}$. The *state reachability problem* of $\mathcal{A}$ is whether a state $s \in S$ is state reachable. The following are some known results about the state reachability problem.

Theorem 1. *(1). The state reachability problem is decidable for standard timed automata [3]. (2). The state reachability problem is decidable for parameterized timed automata with only one parameterized clock (but can have many unparameterized clocks) [5]. (3). The state reachability problem is undecidable for generalized discrete timed automata.*

Actually, Theorem 1 (3) follows from the following special cases.

Theorem 2. *(1). The state reachability problem is undecidable for standard timed automata when we allow "+" operations in clock constraints, e.g., $x+y-z<5$ [3]. (2). The state reachability problem is undecidable for parameterized timed automata (with more than 2 parameterized clocks) [5].*

On the other hand, *binary reachability* is the set of all configuration pairs $\langle\alpha,\beta\rangle$ such that $\alpha \rightsquigarrow_{\mathcal{A}} \beta$ (we use $\rightsquigarrow_{\mathcal{A}}$ in the paper). Characterizations of the binary reachability of timed automata have recently been established.

Theorem 3. *(1). The binary reachability of timed automata with real valued clocks is definable in the additive theory over reals and integers [7, 9]. (2). The binary reachability of timed automata with integer valued clocks is definable by a Presburger formula [7, 15].*

Characterizations of binary reachability help us to reason about non-region properties of timed systems. For instance, consider the following property: "for every configuration α there exists a configuration β such that $\alpha \rightsquigarrow_{\mathcal{A}} \beta$ and the clock x_1 in β is the sum of clocks x_1 and x_2 in α." Though constraint $\beta_{x_1} = \alpha_{x_1} + \alpha_{x_2}$ is not in the form of a clock region, this property can be automatically verified for timed automata [7, 9, 15].

However, for generalized discrete timed automata, the binary reachability $\rightsquigarrow_{\mathcal{A}}$ is too strong to have an interesting characterization. In particular, even the membership problem for binary reachability, i.e., deciding whether two given configurations α and β satisfy $\alpha \rightsquigarrow_{\mathcal{A}} \beta$ for a generalized discrete timed automaton $\mathcal{A}$, is undecidable. This follows from the fact that a two-counter machine can be simulated by a generalized discrete timed automaton $\mathcal{A}$, as shown in the the proofs of Theorem 2 (1) (2) [3, 5]. Thus, the membership problem can be reduced to the halting problem for two-counter machines, which is undecidable.

Theorem 4. *The membership problem for binary reachability is undecidable for generalized discrete timed automata.*

This undecidability result for generalized discrete timed automata leads us to consider the following three approximations of $\rightsquigarrow_{\mathcal{A}}$. Let r and B be fixed positive integers. The first approximation is *r-reset-bounded* reachability. A path $\alpha^0\alpha^1\cdots\alpha^k$ is called *r-reset-bounded* if each clock resets at most r times. Write $\alpha \rightsquigarrow^r_{\mathcal{A}} \beta$ if α reaches β through an r-reset-bounded path. The second approximation is *B-bounded* reachability. A path $\alpha^0\alpha^1\cdots\alpha^k$ is called *B-bounded* if for each $j<k$, each $x_i \in \boldsymbol{X}$, $|\alpha^j_{x_i} - \alpha^{j+1}_{x_i}| < B$. Write $\alpha \rightsquigarrow^B_{\mathcal{A}} \beta$ if α reaches β through a B-bounded path. The third approximation is *$\langle B, r\rangle$-crossing-bounded* reachability. A path $\alpha^0\alpha^1\cdots\alpha^k$ is called

$\langle B, r\rangle$-*crossing-bounded* if there are at most r many i's such that $|\alpha^j_{x_i} - \alpha^{j+1}_{x_i}| \geq B$. Write $\alpha \rightsquigarrow^{\langle B,r\rangle}_{\mathcal{A}} \beta$ if α reaches β through a $\langle B, r\rangle$-crossing-bounded path.

The main results in this paper show that the three approximations of binary reachability $\rightsquigarrow_{\mathcal{A}}$ have decidable characterizations, i.e., they can be accepted by a class of machines with a decidable emptiness problem. Before we proceed to show the results, some further definitions are needed.

A nondeterministic multicounter machine (NCM) is a nondeterministic machine with a finite number of states, a one-way input tape, and a finite number of integer counters. Each counter can be incremented by 1, decremented by 1, or stay unchanged. These counter assignments are called *standard assignments*. In addition, a counter can be tested against an integer constant. These tests are called *standard tests*. An NCM is *reversal-bounded* if each counter is reversal-bounded (i.e., it changes mode between nondecreasing and nonincreasing for some bounded number of times).

A tuple of integers can be encoded as a string by concatenating the unary representations of each integer with a separator. In this way, the binary reachability $\rightsquigarrow_{\mathcal{A}}$ can be treated as a language $\{[\alpha]\#[\beta] : \alpha \rightsquigarrow_{\mathcal{A}} \beta\}$, where $[\alpha]$ and $[\beta]$ are string encodings of configurations α and β separated by a delimiter "#". Obviously, same encoding applies to $\rightsquigarrow_M$, the binary reachability of an NCM M.

When an NCM is reversal-bounded, the emptiness problem is decidable. In fact, we have a stronger characterization.

Theorem 5. *A set of n-tuples of integers is definable by a Presburger formula iff it can be accepted by a reversal-bounded deterministic multicounter machine [19].*

The machines defined above, when used as language recognizers, have a one-way input tape. Suppose a two-way input is used instead. Let 2DCM(c, r) denote the class of deterministic machines with a two-way input tape and c r-reversal-bounded counters. Then the emptiness problem for 2DCM(c, r) when $c \geq 2$ and $r \geq 1$ is undecidable [19]. An interesting special case is when $c = 1$, i.e., there is only one counter. A language is *2DCM-recognizable* if it can be accepted by a 2DCM$(1, r)$.

Theorem 6. *The emptiness problem for 2DCM-recognizable languages is decidable [20].*

It is still open whether Theorem 6 holds for nondeterministic machines. That is, whether the emptiness problem for 2NCM(1,r), which is a nondeterministic r-reversal-bounded one counter machine with a two-way input tape, is decidable.

Given a generalized discrete timed automaton $\mathcal{A}$, consider a subset of configuration pairs, $R_{\mathcal{A}} \subseteq S \times (N^+)^{|\boldsymbol{C}|} \times (N^+)^{|\boldsymbol{X}|} \times S \times (N^+)^{|\boldsymbol{C}|} \times (N^+)^{|\boldsymbol{X}|}$. One can look at $R_{\mathcal{A}}$ as some sort of reachability relation. For a given $\mathcal{A}$, if a 2DCM$(1, r)$ $M_{\mathcal{A}}$ can be effectively constructed such that for every w, w is in $R_{\mathcal{A}}$ iff there exists a w' such that $M_{\mathcal{A}}$ accepts $w\#w'$, then we say that $R_{\mathcal{A}}$ has a *2DCM-padding*. From Theorem 6, it is routine to show the following lemma.

Lemma 1. *(1). The emptiness problem for $\{R_{\mathcal{A}}\}_{\mathcal{A}}$ having 2DCM-paddings is decidable. (2). If $R^1_{\mathcal{A}}$ and $R^2_{\mathcal{A}}$ have 2DCM-paddings, then so do the join $R^1_{\mathcal{A}} \cup R^2_{\mathcal{A}}$ and the composition $R^1_{\mathcal{A}} \circ R^2_{\mathcal{A}}$.*

3 Main Results

Denote $\rightsquigarrow^0_{\mathcal{A}}$ to be the binary reachability of a generalized discrete timed automaton $\mathcal{A}$ through a path without clock resets. Note that $\mathcal{A}$ itself can be considered as a nondeterministic multicounter machine. However, tests in $\mathcal{A}$, which are linear relations on clocks and parameterized constants, are not standard. Assignments in $\mathcal{A}$ in the form of $x := x + d$ with d a parameterized constant are also not standard. But, if $\mathcal{A}$ has only unit durations, the assignments are standard except when a clock reset occurs, i.e., $x := 0$. The following result follows our recent results on strong-reversal-bounded NCMs [21].

Lemma 2. *Suppose $\mathcal{A}$ is a generalized discrete timed automaton with unit durations. Then $\rightsquigarrow^0_{\mathcal{A}}$ is Presburger.*

The following theorem gives a characterization of the three approximations of $\rightsquigarrow_{\mathcal{A}}$ when $\mathcal{A}$ has unit durations. The proof cuts a reachability path of $\mathcal{A}$ into a finite number of phases and each phase can be further characterized by $\rightsquigarrow^0_{\mathcal{A}}$ using Lemma 2.

Theorem 7. *Suppose $\mathcal{A}$ is a generalized discrete timed automaton with unit durations. Then $\rightsquigarrow^r_{\mathcal{A}}$, $\rightsquigarrow^B_{\mathcal{A}}$ and $\rightsquigarrow^{\langle B,r\rangle}_{\mathcal{A}}$ are Presburger.*

The case for $\mathcal{A}$ with parameterized durations is more complicated. In principle, $\mathcal{A}$ with parameterized durations can be simulated by an $\mathcal{A}'$ with unit durations. This is done by simply introducing a new clock z to test whether the parameterized duration of a transition is reached, and after the transition is fired z is reset to 0. However, the three approximations on $\mathcal{A}$ are not equivalent to those on $\mathcal{A}'$. The reason is as follows. Consider $\rightsquigarrow^0_{\mathcal{A}}$, the 0-reset-bounded approximation of $\mathcal{A}$. $\alpha \rightsquigarrow^0_{\mathcal{A}} \beta$ if α can reach β through a path without clock resets. But for $\mathcal{A}'$ each transition with a parameterized duration causes clock z to reset. Thus, a path in $\mathcal{A}'$ witnessing $\alpha \rightsquigarrow^0_{\mathcal{A}} \beta$ could have an unbounded number of clock resets. Therefore, for $\mathcal{A}$ with parameterized durations, Theorem 7 is not applicable.

In the following, we consider a generalized discrete timed automaton $\mathcal{A}$ (with parameterized durations). Currently, we cannot show a decidable characterization of $\rightsquigarrow^0_{\mathcal{A}}$, since we find it is related to the emptiness problem of 2NCM(1,r), which is still open. However, by restricting $\mathcal{A}$ to be deterministic, the following results can be established. The proof uses the fact that the disjunction of the atomic linear relations that appear in all the enabling conditions of $\mathcal{A}$ is equivalent to a union of convex linear relations.

Lemma 3. *$\rightsquigarrow^0_{\mathcal{A}}$ has a 2DCM-padding for a deterministic generalized discrete timed automaton $\mathcal{A}$.*

Again, by using Lemma 3 and the idea of cutting a reachability path of $\mathcal{A}$ into phases, the following theorem gives a characterization of the three approximations of $\rightsquigarrow_{\mathcal{A}}$.

Theorem 8. *Suppose $\mathcal{A}$ is a deterministic discrete timed automaton. Then, $\rightsquigarrow^r_{\mathcal{A}}$, $\rightsquigarrow^B_{\mathcal{A}}$ and $\rightsquigarrow^{\langle B,r\rangle}_{\mathcal{A}}$ have 2DCM-paddings.*

4 Verification of Safety Properties

Consider P and I, two sets of configurations of a generalized discrete timed automaton $\mathcal{A}$ definable by Presburger formulas. If, starting from a configuration in I, $\mathcal{A}$ can only reach configurations in P, then P is a *safety property* with respect to the initial condition I. The following is an example:

"starting from a configuration satisfying $x_1 - x_2 + x_3 - x_4 > c + d$, $\mathcal{A}$ cannot reach a configuration satisfying $2x_1 + x_2 < c \wedge x_3 - 3x_4 > 2d - c$."

The *safety analysis problem* is to determine whether P is a safety property with respect to the initial condition I. The following two theorems follow from Theorem 7, Theorem 8, and Lemma 1.

Theorem 9. *The safety analysis problem is decidable for generalized discrete timed automata with unit durations and with any one of the following approximations: r-reset-boundedness, B-boundedness and $\langle B, r\rangle$-crossing-boundedness.*

Theorem 10. *The safety analysis problem is decidable for deterministic generalized discrete timed automata with any one of the following approximations: r-reset- boundedness, B-boundedness and $\langle B, r\rangle$-crossing-boundedness.*

Remark: Theorem 10 can be strengthened. The class of languages accepted by deterministic two-way counter machines with one reversal-bounded counter is closed under Boolean operations [20]. It follows that Theorem 10 remains valid even if the sets of configurations P (property) and I (initial condition) are sets accepted by these machines. ∎

It is desirable to consider the decidability of the safety analysis problem for generalized discrete timed automata under some special form but without using the approximations. One such form is parameterized timed automata with only one parameterized clock. As stated in Theorem 1 (2), the state reachability problem is decidable. However, surprisingly, the safety analysis problem, i.e.,

"Deciding whether P is a safety property with respect to the initial condition I for a parameterized timed automaton with only one parameterized clock, where both P and I are definable by Presburger formulas"

is still open. This problem is closely related to the open problem of the decidability of the emptiness problem for 2NCM(1,r). It is also worthwhile to point out that the characterizations of the binary reachability of deterministic generalized discrete timed automata under the approximations, as shown in Theorem 8, are not necessarily Presburger. In fact, 2DCM(1,r) can accept a class of nonlinear languages [20].

5 A Verification Example

In practice, allowing parameterized clock constraints and durations makes it possible to specify more complex real-time systems. In this section, we take an example specification [23] of the railroad crossing benchmark [18], which is written in ASTRAL [6]. The specification specifies a system consisting of a set of railroad tracks that intersect a street where cars may cross the tracks. A gate is located at the crossing to prevent

cars from crossing the tracks when a train is near. A sensor on each track detects the arrival of trains on that track. The critical requirement of the system is that whenever a train is in the crossing the gate must be down, and when no train has been in between the sensors and the crossing for a reasonable amount of time, the gate must be up. The complete ASTRAL specification of the railroad crossing system, which was written by Paul Kolano, can be found at `http://www.cs.ucsb.edu/~dang`.

The ASTRAL specification includes a global specification and two process specifications: one is `Gate` and the other is `Sensor`. A transition system is specified inside each process specification along with local assumptions and local properties. ASTRAL adopts a modularized view of the specified system at the level of correctness proofs: the global properties in the global specification can be verified by using global assumptions and local properties (instead of the actual transition behaviors) of each process instance; local properties of a process instance can be verified by using the local assumptions, the local imported variable clause and the transition system of the instance, without looking at the transition behaviors of the other process instances (A reader need not worry about the possibility of circular proofs. The ASTRAL proof theory is sound, see [8] for details.).

We take the `Sensor` process specification to see how parameterized durations and parameterized clock constraints are used in ASTRAL. `Sensor` reports whether a train is beyond the crossing, which is indicated by a Boolean variable `train_in_R`. The process specification has only two transitions `enter_R` and `exit_R`, which have parameterized durations `enter_dur` and `exit_dur`, respectively. Transition `enter_R` changes `train_in_R` from `FALSE` to `TRUE` with duration `enter_dur`. Transition `exit_R` sets `train_in_R` back to `FALSE` after the slowest train moves out of the crossing. That is,

```
TRANSITION exit_I
  ENTRY           [ TIME : exit_dur ]
            train_in_R
          & now - Start ( enter_R ) >= RIImin - exit_dur
  EXIT
            train_in_R = FALSE,
```

where `now` indicates the current time, `Start(enter_R)` is the most recent start time of transition `enter_R`, and `RIImin` is a parameterized constant indicating the time for the slowest train to move out of the crossing. One of the two transitions is nondeterministically chosen to fire as long as the `ENTRY` condition of the chosen transition is satisfied. However, if neither of them is fireable, the process idles: `now` progresses by one time unit and `train_in_R` does not change.

But, according to this semantics, transition `enter_R` must fire immediately whenever transition `exit_R` completes. This is not the intended behavior of `Sensor`. In fact, it is desirable that `enter_R` fires only when a train actually comes. But `enter_R` specified as above only tells what happened (set `train_in_R` to `TRUE`) when a train comes, instead of the time when a train comes. The pattern of a train's arrival is controlled by the environment of the process. In this specification, transition `enter_R` is declared as exported. That is, it must be called by the environment in order to fire. In `Sensor`, the environment is specified by the environment clause, which states that two consecutive calls of `enter_R` must be separated by at least `RIImin` many time units.

The safety property of the process is specified as a *schedule*, which has two conjuncts. The first conjunct of the schedule says that a train will be sensed within `enter_dur` many time units after a call of transition `enter_R` is placed. The second conjunct of the schedule says that the sensor will be reset when the slowest train is beyond the crossing. It is assumed that initially `now` is 0 and `train_in_R` is `FALSE`.

We manually translated `Sensor` into a generalized discrete timed automaton and computed the transitive closure of the one-step transition of the automaton using the Omega Library [26], which is a tool to manipulate Presburger formulas. Experiments were run on a Sun workstation with 4 CPUs and 256M real memory and 512M swap memory. Unfortunately, the closure, even when the durations (`enter_dur` and `exit_dur`) were set to 1, could not be computed. Then, we used the B-bounded approximation on the automaton with $B = 3$. This time, the binary reachability $\rightsquigarrow_{\mathcal{A}}^{B}$ can be computed in about one minute of CPU time and using 170M memory. But the other two approximation approaches were still too expensive to calculate.

6 Conclusions and Future Work

We studied generalized discrete timed automata with general linear relations over clocks and parameterized constants as clock constraints and with parameterized durations. We focused on three approximation techniques and automata-theoretic characterizations of binary reachability under these approximations. The characterizations allow us to show that the safety analysis problem is decidable with respect to generalized discrete timed automata with unit durations, and deterministic generalized discrete timed automata with parameterized durations (modulo the approximations). We used an example specification written in ASTRAL to run a number of experiments using one of the approximation techniques. The results of the experiments show that further improvements to the approximations have to be developed, since currently they are not practical for large specifications.

For future work, we want to investigate how the approximation techniques proposed in this paper can be combined with existing image-approximation techniques [12] in debugging infinite state systems. Solutions to this problem would lead to an implementation of an effective specification debugger for large real-time specifications. Another research issue is how to extend the results in this paper to the case of generalized timed automata with dense clocks. Recent ideas used in [9] may provide some insights.

References

1. R. Alur, *"Timed automata", CAV'99*, LNCS **1633**, pp. 8-22
2. R. Alur, C. Courcoubetis and D. Dill, *"Model-checking in dense real time," Information and Computation*, **104** (1993) 2-34
3. R. Alur and D. Dill, *"A theory of timed automata," TCS*, **126** (1994) 183-236
4. R. Alur, T. A. Henzinger, *"A really temporal logic," J. ACM*, **41** (1994) 181-204
5. R. Alur, T. A. Henzinger and M. Y. Vardi, *"Parametric real-time reasoning," STOC'93*, pp. 592-601
6. A. Coen-Porisini, C. Ghezzi and R. A. Kemmerer, *"Specification of real-time systems using ASTRAL," TSE*, **23** (1997) 572-598

7. H. Comon and Y. Jurski, *"Timed automata and the theory of real numbers," CONCUR'99*, LNCS 1664, pp. 242-257
8. A. Coen-Porisini, R. A. Kemmerer and D. Mandrioli, *"A formal framework for ASTRAL intralevel proof obligations," TSE*, **20** (1994) 548-561
9. Z. Dang, *"A decidable binary reachability characterization of timed pushdown automata with dense clocks,"* To appear in *CAV'01*, LNCS
10. Z. Dang and R. A. Kemmerer, *"Using the ASTRAL model checker to analyze Mobile IP," ICSE'99*, pp. 132-141
11. Z. Dang and R. A. Kemmerer, *"A symbolic model checker for testing ASTRAL real-time specifications," RTCSA'99*, pp. 174-181
12. Z. Dang and R. A. Kemmerer, *"Using the ASTRAL symbolic model checker as a specification debugger: three approximation techniques," ICSE'00*, pp. 345-354
13. Z. Dang, O. H. Ibarra and R. A. Kemmerer, `http://www.eecs.wsu.edu/~zdang`, the full version of this paper
14. Z. Dang, P. San Pietro and R. A. Kemmerer, *"On Presburger liveness of discrete timed automata," STACS'01*, LNCS 2010, pp. 132-143
15. Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer and J. Su, *"Binary reachability analysis of discrete pushdown timed automata," CAV'00*, LNCS 1855, pp. 69-84
16. T. A. Henzinger and Pei-Hsin Ho, *"HyTech: the Cornell hybrid technology tool," Hybrid Systems II*, LNCS **999**, 1995, pp. 265-294
17. T. A. Henzinger, X. Nicollin, J. Sifakis and S. Yovine. *"Symbolic model checking for real-time systems," Information and Computation*, **111** (1994) 193-244
18. C. Heitmeyer and N. Lynch, *"The generalized railroad crossing: a case study in formal verification of real-time systems," RTSS'94*, pp. 120-131
19. O. H. Ibarra, *"Reversal-bounded multicounter machines and their decision problems," J. ACM*, **25** (1978) 116-133
20. O. H. Ibarra, T. Jiang, N. Tran and H. Wang, "New decidability results concerning two-way counter machines," *SIAM J. Comput.*, **24** (1995) 123-137
21. O. H. Ibarra, J. Su, Z. Dang, T. Bultan and R. A. Kemmerer, *"Counter machines: decidable properties and applications to verification problems,", MFCS'00*, LNCS **1893**, pp. 426-435
22. P. Kolano, *"Tools and techniques for the design and systematic analysis of real-Time systems,"* Ph.D. Thesis, University of California, Santa Barbara, 1999
23. P. Z. Kolano, Z. Dang and R. A. Kemmerer, *"The design and analysis of real-time systems using the ASTRAL software development environment," Annals of Software Engineering*, **7** (1999) 177-210
24. K. G. Larsen, P. Pettersson and W. Yi, *"UPPAAL in a nutshell," International Journal on Software Tools for Technology Transfer*, **1** (1997) 134-152
25. F. Laroussinie, K. G. Larsen and C. Weise, *"From timed automata to logic - and back," MFCS'95*, LNCS **969**, pp. 529-539
26. W. Pugh, *"The Omega test: a fast and practical integer programming algorithm for dependence analysis," CACM*, **8** (1992) 102-104
27. J. Raskin and P. Schobben, *"State clock logic: a decidable real-time logic," HART'97*, LNCS 1201, pp. 33-47
28. T. Wilke, *"Specifying timed state sequences in powerful decidable logics and timed automata,"* LNCS 863, pp. 694-715, 1994
29. S. Yovine, *"A verification tool for real-time systems," International Journal on Software Tools for Technology Transfer*, **1** (1997): 123-133