Generalized Discrete Timed Automata: Decidable Approximations for Safety Verification ⋆

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 timed automaton can be considered as a finite automaton augmented with a finite number of clocks. The clocks can be reset or progress at the same rate, and can be tested against clock constraints in the form of clock regions (i.e., comparisons of a clock or the difference of two clocks against an integer constant, e.g., $x - y > 8$, where x and y are clocks.). 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, 15, 16, 23, 24, 28] for verifying real-time systems (see [1] for a survey).

However, not every real-time system can be modeled as a timed automaton. A complex (not necessarily large) real-time system might contain non-region clock constraints, such as $x_1 - x_2 > x_3 - x_4 + c$, where x_1, x_2, x_3, x_4 are clocks, and c is a

⋆ The work by Zhe Dang and Richard A. Kemmerer has been supported in part by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Material Command, USAF, under agreement number F30602-97-1-0207. The work by Oscar H. Ibarra has been supported in part by NSF grant IRI-9700370.

⋆⋆ Corresponding author.

parameterized (or unspecified) constant. Though there are practical needs to augment timed automata with more complex clock constraints, 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, 21] 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, for each clock, the clock value is less than a given positive integer B every time when the clock resets (but after the last reset, the clock can go unbounded.). 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 is greater or equal to 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, 13, 14], 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 [19]. The case for nondeterministic generalized discrete timed automata is open. These are good characterizations in the sense that the validity of Presburger formulas can be verified by these machines, and the emptiness problem for such machines is 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.

The paper is organized as follows. Section 2 gives the basic definitions and preliminary results that are used in the paper. Section 3 presents the decidable characterization

of binary reachability for generalized discrete timed automata with unit durations and with parameterized durations, under the approximations. Section 4 formulates the safety analysis problem and its decidability and proposes an open problem concerning nondeterministic generalized discrete timed automata with parameterized durations. Section 5 shows an example from an ASTRAL specification of the railroad crossing benchmark. Finally, in Section 6, conclusions and future work are summarized.

2 Generalized Discrete Timed Automata

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*.

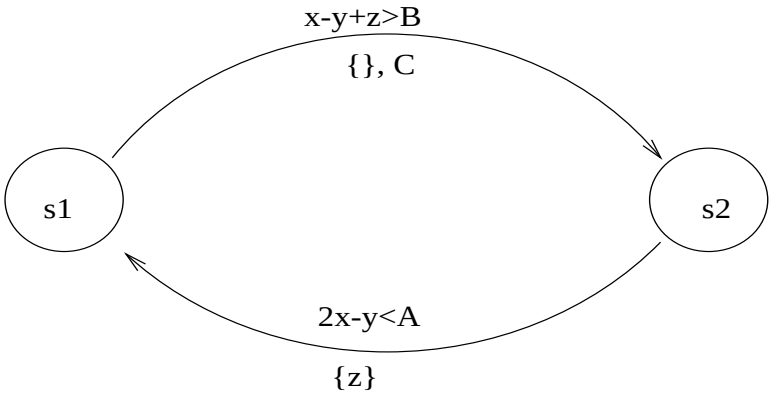

Fig. 1. An example generalized discrete timed automaton

Example 1. Figure 1 shows a generalized discrete timed automaton with parameterized constants A, B, C and clocks x, y, z. The automaton has two transitions T_1 (that is from state s_1 to state s_2) and T_2 (that is from state s_2 to state s_1). As shown in the figure, T_1 has an enabling condition of $x - y + z > B$, an empty clock reset set, and a parameterized duration C. T_2 has an enabling condition of $2x - y < A$, a clock reset set $\{z\}$, and a zero duration. According to the semantics, the following is an instance of an execution of the automaton when $A = 4, B = 1, C = 2$: $\langle s_1, x = 1, y = 1, z = 3\rangle \rightarrow^{T_1} \langle s_2, x = 3, y = 3, z = 5\rangle \rightarrow^{T_2} \langle s_1, x = 3, y = 3, z = 0\rangle$. ∎

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 problem* of $\mathcal{A}$ is whether a state $s \in S$ is state reachable. It is noticed that the region reachability (whether clock values in a clock region can reach clock values in another clock region) can be formulated as state reachability by applying a simple modification to the automaton

being considered. 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, 14].

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, 14].

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 given 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 j'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.

Figure 2 shows the behaviors of a clock under the three approximations with some bound B and $r = 4$. Figure 2 (a) is the case for r-reset-bounded approximation, where the clock is reset for at most four times. Figure 2 (b) is the case for B-bounded approximation, where the clock value always stays below the bound B before the last reset (but there could be many resets). Figure 2 (c) is the case for $\langle B, r\rangle$-crossing-bounded approximation, where the clock crosses the bound B but for at most four times before the last reset.

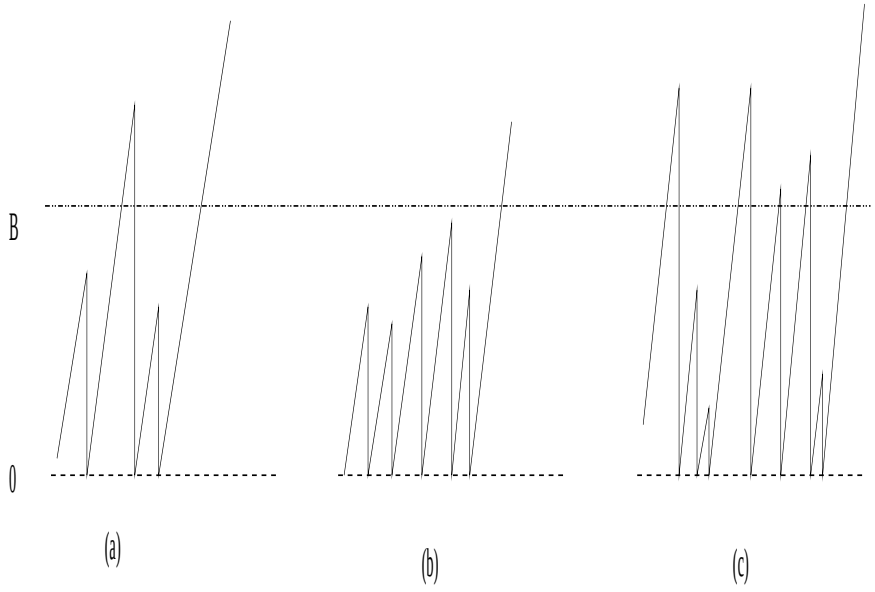

Fig. 2. Behaviors of a clock under the three approximations

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 set of *(control) states* $Q = \{1, 2, \cdots, |Q|\}$, and a finite number of counters $x_1, \cdots, x_k$ with integer counter values. Each counter can add 1, subtract 1, or stay unchanged. These counter assignments are called *standard assignments*. M can also test whether a counter is equal to, greater than, or less than an integer constant. These

tests are called *standard tests*. When an NCM M is used as a language recognizer, we attach a separate one-way read-only input tape to the machine and assign a state in Q as the final state. M *accepts* an input iff it can reach the final state. It is well-known that counter machines with two counters have undecidable halting problem. Thus, we have to restrict the behaviors of the counters. One such restriction is to limit the number of reversals a counter can move. A counter is *r-reversal-bounded* if it changes mode between nondecreasing and nonincreasing at most r times. A counter is *r-strong-reversal-bounded* if it changes mode between strictly increasing, strictly decreasing and unchanged at most r times. For instance, the following sequence of counter values: $0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 3, 2, 1, 1, 1, 1, \cdots$ exhibits only one counter reversal. However, it has 11 strong reversals. M is *(strong-)reversal-bounded* if each counter in M is r-(strong-)reversal-bounded for some r. We note that a (strong-)reversal-bounded M does not necessarily limit the number of moves or the number of reachable configurations to be finite.

Let $(j, v_1, \cdots, v_k)$ denote the *configuration* of an NCM M (without input tape) when it is in state $j \in Q$, counter x_i has value $v_i \in N$ for $i = 1, 2, \cdots, k$. Similar to $\leadsto_{\mathcal{A}}$, the binary reachability $\leadsto_M$ of M is defined as all the pairs (α, β) of configurations of M such that α can reach β.

It is known that the emptiness problem for reversal-bounded NCMs (when used as language recognizers) is decidable [18]. When a reversal-bounded NCM uses linear relations as tests instead of using standard tests, we have the following result.

Theorem 5. *Suppose M is a nondeterministic strong-reversal-bounded multicounter machine without input tape that uses linear relations (on the counters and parameterized constants) as tests instead of using standard tests. Then its binary reachability $\leadsto_M$ is Presburger [20].*

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 [18]. 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 [19].*

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 Decidability 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., $x := 0$.

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

Proof. Let $\mathcal{A}$ be a generalized discrete timed automaton with unit durations. Note that when a path of $\mathcal{A}$ contains no clock resets, i.e., all the clocks increase at the same rate 1, clocks (understood as counters) are 0-strong-reversal-bounded. The theorem follows from Theorem 5. ∎

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.*

Proof. Let $\mathcal{A}$ be a generalized discrete timed automaton with unit durations.

(1). r-reset-bounded reachability $\rightsquigarrow^r_{\mathcal{A}}$: An r-reset-bounded path of $\mathcal{A}$ can be divided into at most $r \cdot |\boldsymbol{X}|$ many phases, where within each phase no clock resets. Two consecutive phases are connected by a one-step transition in $\mathcal{A}$ that contains clock resets. Thus, $\rightsquigarrow^r_{\mathcal{A}}$ is the "concatenation" of all these phases and the transitions by using existential elimination. The result follows because from Lemma 2 each phase is Presburger and Presburger formulas are closed under quantification.

(2). B-bounded reachability $\rightsquigarrow^B_{\mathcal{A}}$: Consider a B-bounded path of $\mathcal{A}$. According to the definition of B-bounded reachability, within the path each clock either never exceeds the value B or it exceeds B. But once a clock exceeds B, it never resets afterwards. Thus, with respect to each clock, a path can be divided into at most two phases. Values of the clock in the first phase do not exceed B while the clock in the second phase never resets. Thus, when all clocks are considered, a path can be divided into at most $|\boldsymbol{X}|+1$ phases such that within each phase a clock either never resets or resets (for an unbounded number of times) but with values not exceeding B. The result follows by concatenating these phases, where, in a phase that a clock does not exceed B, a finite variable (bounded by B) is used to represent the clock, and in a phase that a clock never resets, Lemma 2 is used.

(3). $\langle B, r\rangle$-crossing-bounded reachability $\rightsquigarrow_{\mathcal{A}}^{\langle B,r\rangle}$: Consider a $\langle B, r\rangle$-crossing-bounded path of $\mathcal{A}$. From the definition, each clock in the path can only cross the boundary B at most r times. Thus, the path can be divided into at most $r \cdot |\boldsymbol{X}| + 1$ phases with the first $r \cdot |\boldsymbol{X}|$ phases being B-bounded paths. The last one is either a B-bounded path or a path with no clock resets. Phases are connected by a one-step transition with at least one clock reset. Thus, the result follows from (1) and (2) above. ∎

The case when $\mathcal{A}$ has 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 $\rightsquigarrow_{\mathcal{A}}^{0}$, the 0-reset-bounded approximation of $\mathcal{A}$. $\alpha \rightsquigarrow_{\mathcal{A}}^{0} \beta$ 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 $\alpha \rightsquigarrow_{\mathcal{A}}^{0} \beta$ could have an unbounded number of clock resets. Therefore, for $\mathcal{A}$ with parameterized durations, Theorem 7 is not applicable.

Before proceeding to prove some further results, we first give some properties of linear relations. Let $\mathcal{R}$ be the (finite) set of all the atomic linear relations that appear in all the enabling conditions of $\mathcal{A}$. Each atomic linear relation is obviously convex in clocks $\boldsymbol{X}$ for any fixed values for parameterized constants in $\boldsymbol{C}$. Define the set of *labels*: $\Sigma = \{\bigwedge_{R_i \in \mathcal{R}} \sigma_i : \text{ each } \sigma_i = R_i \text{ or } \sigma_i = \neg R_i\}$. Each label is a linear relation over $\boldsymbol{X}$ and $\boldsymbol{C}$, and it is convex in clocks $\boldsymbol{X}$. A configuration α has label $\sigma \in \Sigma$ if the values of the clocks and the parameterized constants in α satisfy the linear relation σ. By the definition of Σ, each configuration has a unique label.

In the following, we consider a generalized discrete timed automaton $\mathcal{A}$ (with parameterized durations). Currently, we cannot show a decidable characterization of $\rightsquigarrow_{\mathcal{A}}^{0}$, 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.

Lemma 3. *Let $\mathcal{A}$ be a deterministic generalized discrete timed automaton. Then $\rightsquigarrow_{\mathcal{A}}^{0}$ has a 2DCM-padding.*

Proof. Let $\mathcal{A} = \langle S_0, S, \boldsymbol{C}, \boldsymbol{X}, E\rangle$. Since $\rightsquigarrow_{\mathcal{A}}^{0}$ contains no clock resets, we may assume that all edges in $\mathcal{A}$ with clock resets are not present in $\mathcal{A}$. Let β and β' be two configurations. Denote Σ to be the set of labels. Since clocks in $\mathcal{A}$ synchronously increase at the same rate, due to the convexity of each label in Σ, without loss of generality, we can further partition a path for $\beta \rightsquigarrow_{\mathcal{A}}^{0} \beta'$ into the following sequence, $\alpha_0 \nearrow^1 \alpha'_0 \to \alpha_1 \nearrow^1 \alpha'_1 \cdots \to \alpha_m \nearrow^1 \alpha'_m$, where m is bounded by the number of labels in Σ, and $\alpha_0 = \beta$, $\alpha'_m = \beta'$. Each "$\nearrow^1$" represents a sequence of one-step transitions through configurations with the same label. Each "$\to$" represents a one-step transition. Notice that the number of labels is independent of the choice of the sequence. Thus, from Lemma 1, it suffices to show that both $\to$ and $\nearrow^1$ have 2DCM-paddings.

We first show the case for $\nearrow^1$. The one for $\to$, which is trivial, is left for the reader. Assume $\nearrow^1$ is through configurations with the same label σ. By definition, a label gives the truth value for each enabling condition. We first drop all edges e from E with $l_e \wedge \sigma$ false. Then the enabling conditions on all the remaining edges are replaced by true. The

resulting automaton is written as $\mathcal{A}^-$. Notice that $\mathcal{A}^-$ is deterministic. That is, each node in $\mathcal{A}^-$ cannot have more than one outgoing edge, from the definition of σ and the fact that $\mathcal{A}$ is deterministic. Given two configurations γ^s and $\gamma^{s'}$ at control states s and s', respectively, a key observation is:

$\gamma^s \nearrow^1 \gamma^{s'}$ iff $\gamma^s \rightsquigarrow \gamma^{s'}$ in $\mathcal{A}^-$ and both γ^s and $\gamma^{s'}$ have label σ.

That is, $\gamma^s \nearrow^1 \gamma^{s'}$ if and only if γ^s reaches $\gamma^{s'}$ in $\mathcal{A}^-$ with the first configuration γ^s and the last configuration $\gamma^{s'}$ having the same label σ. The convexity of each label ensures that all the intermediate configurations also have the same label. Since $\mathcal{A}^-$ is deterministic, any path from s to s' can be written as $\mathbf{u}\mathbf{v}^i\mathbf{w}$ for some i, with the length of $\mathbf{u}$,$\mathbf{v}$ and $\mathbf{w}$ bounded by the number of control states $|S|$. That is, the path can be concatenated by a short path and a short cycle followed by another short path. In order to show $\nearrow^1$ has a 2DCM-padding, we construct the required 2DCM(1, r) M. M operates on an input tape with the following format: $B_s\#B_{s'}\#\mathbf{u}\#\mathbf{v}\#\mathbf{w}$, where B_s and $B_{s'}$ are string encodings of the two configurations γ^s and $\gamma^{s'}$. $\mathbf{u}$,$\mathbf{v}$ and $\mathbf{w}$ are short sequences of control states, which could be empty, with $\mathbf{u}$ started by s and $\mathbf{w}$ ended by s'. From the above key observation, M only needs to check:

(1). the input tape is in the correct format. That is, $\mathcal{A}^-$ actually has a path $\mathbf{u}\mathbf{v}\mathbf{w}$. This can be done by M using a finite table look-up. Configurations γ^s and $\gamma^{s'}$ agree on each (positive) parameterized constant. This can be done with exactly $|\boldsymbol{C}|$ counter reversals.

(2). γ^s and $\gamma^{s'}$ have the same label σ. Note that σ is a linear relation. Checking satisfiability of a linear relation over configurations γ^s and $\gamma^{s'}$ can be done by accumulating the counter while reading the value of each clock and constant stored in $B_s\#B_{s'}$ on the two-way input tape. Since σ is fixed, this needs only $\delta \cdot (|\boldsymbol{C}| + |\boldsymbol{X}| + 1)$ counter reversals, where δ is the number of atomic linear relations in σ.

(3). M also needs to check that for each $x \in \boldsymbol{X}$, the net clock change $\gamma_x^{s'} - \gamma_x^s$ is the same. This needs $4|\boldsymbol{X}|$ counter reversals. Also, M makes sure that the net clock change agrees with the net clock change on a path $\mathbf{u}\mathbf{v}^i\mathbf{w}$ for some i. Denote $\Delta_{\mathbf{u}}$, $\Delta_{\mathbf{v}}$, $\Delta_{\mathbf{w}}$ to be the sum of the durations of the edges in sequences $\mathbf{u}, \mathbf{v}, \mathbf{w}$, respectively. Let $x \in \boldsymbol{X}$. The counter in M first calculates $\gamma_x^{s'} - \gamma_x^s - \Delta_{\mathbf{u}} - \Delta_{\mathbf{w}}$. This can be done by at most 1 counter reversal. Then M tests whether the counter ≤ 0, and doing the same test again by subtracting $\Delta_{\mathbf{v}}$ from the counter until the test succeeds. If the counter is equal to 0, then M accepts, else M rejects. Note that M has only one counter, thus $\Delta_{\mathbf{v}}$ cannot be stored. Since $\mathbf{v}$ is a short path, M instead subtracts each duration (that is a constant, stored in B_s) one by one in $\mathbf{v}$.

Thus, M accepts some input as above iff $\gamma^s \nearrow^1 \gamma^{s'}$. This completes the construction of the required 2DCM(1,r) machine. ∎

The following theorem gives a characterization of the three approximations of $\rightsquigarrow_{\mathcal{A}}$.

Theorem 8. *Let $\mathcal{A}$ be a deterministic discrete timed automaton. Then, $\rightsquigarrow_{\mathcal{A}}^r$, $\rightsquigarrow_{\mathcal{A}}^B$ and $\rightsquigarrow_{\mathcal{A}}^{\langle B,r\rangle}$ have 2DCM-paddings.*

Proof. Similar to the proof of Theorem 7 by "concatenating" the 2DCM-padding for each phase using Lemma 1 and Lemma 3. ∎

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.

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

Proof. Let $\mathcal{A}$ be a generalized discrete timed automata with unit durations. Without loss of generality, consider r-reset-bounded approximation. The safety analysis problem is equivalent to the existence of α and β such that $\alpha \in I, \alpha \rightsquigarrow^r_{\mathcal{A}} \beta$, and $\beta \in \neg P$. From Theorem 7, $\rightsquigarrow^r_{\mathcal{A}}$ is Presburger. Using the fact that P and I are Presburger and Presburger formulas are closed under quantification, the theorem follows. ∎

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

Proof. Let $\mathcal{A}$ be a deterministic generalized discrete timed automaton. We only prove the result for the case of the r-reset-bounded approximation, the others being similar. From Theorem 8, $\rightsquigarrow^r_{\mathcal{A}}$ has a 2DCM-padding. That is, we can construct a 2DCM(1,t) M such that, for all configurations α and β, $\alpha \rightsquigarrow^r_{\mathcal{A}} \beta$ iff $\exists w'(M$ accepts $\langle \alpha, \beta\rangle \# w')$. Let P and I be two sets of configurations definable by Presburger formulas. It is observed that P is a safety property with respect to the initial condition I under the approximation iff there exist α and β such that, $\alpha \in I, \alpha \rightsquigarrow^r_{\mathcal{A}} \beta$ and $\beta \in \neg P$. Using the known fact that a Presburger relation can also be accepted by a deterministic two-way counter machine with one reversal-bounded counter, the above M can be further modified to check $\alpha \in I$ and $\beta \in \neg P$. Thus, the safety analysis problem is equivalent to testing the emptiness of M, which is decidable from Lemma 1. ∎

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 [19]. It follows that Theorem 11 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 of 2NCM(1,r). This is also an example showing that binary reachability is much stronger than state reachability. Another form that can be considered is generalized discrete timed automata with clock constraints $x\#c$ (c is an integer). Thus, they are standard timed automata with parameterized durations. The state reachability problem for these automata is decidable. The idea is to translate a transition with a parameterized duration d into a loop with unit duration and add a new clock z, which is tested against d, and use Theorem 1 (2) since z is the only parameterized clock. But again, it is open whether the safety property analysis problem is decidable for these automata. This problem is simpler than the previous open problem, but the answer is currently not clear.

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, there are nonsemilinear languages that are 2DCM(1,r)-recognizable [19].

5 A Verification Example

In practice, allowing generalized clock constraints and parameterized durations makes it possible to specify more complex real-time systems. In this section, we take an example specification [22] of the railroad crossing benchmark [17], 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 as follows:

```
TRANSITION enter_R
  ENTRY    [ TIME : enter_dur ]
            ~train_in_R
  EXIT
            train_in_R = TRUE.
```

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 following environment clause:

```
  Call ( enter_R, now )
& EXISTS t: time
   ( t >= 0 & t <= now & Call[2] (enter_R, t))
     -> Call(enter_R) - Call[2] (enter_R) > RIImin.
```

It says that two consecutive calls of enter_R must be separated by at least RIImin many time units. Assuming the environment holds, the safety property of the process is specified as a *schedule*:

```
  ( now >= response_time & Call (enter_R, now-response_time)
       -> train_in_R )
& ( now >= RIImin & Call (enter_R, now-RIImin)
       -> ~train_in_R ).
```

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 (note that, according to the assumptions on the parameterized constants, `response_time` is a parameterized constant not less than `enter_dur`, and satisfying `RIImin >= response_time + exit_dur`.). 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 [25], 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 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, *"Binary reachability analysis of timed pushdown automata with dense clocks," CAV'01*, LNCS **2102**, pp. 506-517
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, P. San Pietro and R. A. Kemmerer, *"On Presburger liveness of discrete timed automata," STACS'01*, LNCS 2010, pp. 132-143
14. 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
15. T. A. Henzinger and Pei-Hsin Ho, *"HyTech: the Cornell hybrid technology tool," Hybrid Systems II*, LNCS **999**, 1995, pp. 265-294
16. T. A. Henzinger, X. Nicollin, J. Sifakis and S. Yovine. *"Symbolic model checking for real-time systems," Information and Computation*, **111** (1994) 193-244
17. C. Heitmeyer and N. Lynch, *"The generalized railroad crossing: a case study in formal verification of real-time systems," RTSS'94*, pp. 120-131
18. O. H. Ibarra, *"Reversal-bounded multicounter machines and their decision problems," J. ACM*, **25** (1978) 116-133
19. 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
20. 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
21. P. Kolano, *"Tools and techniques for the design and systematic analysis of real-Time systems,"* Ph.D. Thesis, University of California, Santa Barbara, 1999
22. 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
23. K. G. Larsen, P. Pettersson and W. Yi, *"UPPAAL in a nutshell," International Journal on Software Tools for Technology Transfer*, **1** (1997) 134-152
24. F. Laroussinie, K. G. Larsen and C. Weise, *"From timed automata to logic - and back," MFCS'95*, LNCS **969**, pp. 529-539
25. W. Pugh, *"The Omega test: a fast and practical integer programming algorithm for dependence analysis," CACM*, **8** (1992) 102-104
26. J. Raskin and P. Schobben, *"State clock logic: a decidable real-time logic," HART'97*, LNCS 1201, pp. 33-47
27. T. Wilke, *"Specifying timed state sequences in powerful decidable logics and timed automata,"* LNCS 863, pp. 694-715, 1994
28. S. Yovine, *"A verification tool for real-time systems," International Journal on Software Tools for Technology Transfer*, **1** (1997): 123-133