# Checking Timed Automata for Some Discretisable Duration Properties

Zhao Jianhua and Dang Van Hung

The United Nations University
International Institute for Software Technology
UNU/IIST, P.O.Box 3058, Macau
e-mail: dvh@iist.unu.edu

**Abstract.** In this paper we propose a technique for checking a timed automaton w.r.t a class of discretisable duration properties. It is shown in the paper that if only integral constants are allowed in the time constraints of the timed automata, a linear duration property or an inter-state duration property is satisfied by a timed automaton if and only if it is satisfied by all of the integral behaviours of the automaton. Therefore, checking linear duration properties or inter-state duration properties can be done by using the integer time verification techniques. Based on this, some algorithms are proposed. We also develop a technique for reducing the search-space for the proposed algorithms. This technique is specially suitable to our case, and can be applied to checking other duration properties as well.

## 1 Introduction

In the last a few years, some verification algorithms for checking timed automata w.r.t. some duration properties have been developed. These duration properties constrain the accumulated time during which certain predicates hold, and are represented formally by Duration Calculus formulas. A solution to the problem for the general case has been given in [8] using mixed integer and linear programming techniques. Since the mixed integer and linear programming techniques are complicated, many authors have searched for a simpler solution by restricting themselves to some special cases of the problem.

It has been shown that by using the linear programming techniques which are much simpler than the mixed integer and linear programming ones, the problem can be solved when the timed automata are real-time automata [1], or in more general, are represented by a so-called timed regular expressions [2]. The technique has been generalised for verifying linear duration invariants of the parallel composition of real-time automata in [6] and of a class of hybrid automata in [3].

In [4], we proposed a verification algorithm for checking a parallel composition of real-time automata w.r.t. a linear duration property by using the state-exploration technique. In that paper, we introduced the compatibility relation between configurations to avoid exhaustive state-exploration. The basic idea of

our algorithm is that if only integral constants are allowed in the time constraints of the automata, a linear duration property is satisfied by the automata if and only if it is satisfied by just the integral computations. Therefore, checking linear duration properties can be done by using the integer time verification techniques.

In this paper, we develop this idea further by introducing a class of so-called discretisable Duration Calculus formulas in which linear duration properties are included. We also give algorithms for checking if a timed automaton satisfies a linear duration property or an inter-state duration property which are discretisable DC formulas.

Since we use the integer time verification techniques to check a system w.r.t. a discretisable Duration Calculus formula, we need to develop a technique for reducing the search-space which is better suitable to our case than the techniques in the literature. In order to do so, we define the set of 'floating indices' for an integral behaviour and show that, if the property to be checked is violated then it must be violated by an integral behaviour having no floating indices. Therefore, we do not need to check those behaviours for the property once we discover that they have floating indices. We propose some techniques to identify the behaviours of which all the right extensions have floating index set. By using these techniques, we can improve the algorithms by reducing the space and time complexity considerably.

## 2 Discretisable Duration Properties of Timed Automata

In this section, we first recall some basic notations of timed automata that will be used in the paper. For some class of real-time properties, it happens that they are satisfied by a timed automaton in the real time domain iff they are satisfied by the automaton in the discrete time domain. We call such properties discretisable properties. We define also in this section the discretisability of a real-time property for a given timed automaton.

For a set of clock variables $X$, we use $\Phi(X)$ to denote the set of time constraints which are conjunctions of the formulas of the form $x \leq c$ or $x \geq c$, where $x \in X$ and $c \in \mathsf{N} \cup \{0\}$ ($\mathsf{N}$ is the set of natural numbers).

**Definition 1.** *A timed automaton $M$ is a tuple $\langle S, s^0, C, E, I\rangle$ , where*

1. *$S$ is a finite set of locations;*
2. *$s^0 \in S$ is an initial location;*
3. *$C$ is a finite set of clocks;*
4. *$E \subseteq S \times S \times 2^C \times \Phi(C)$ is a finite set of transitions. For a transition $e = \langle s_1, s_2, \lambda, \gamma\rangle$, the locations $s_1$ and $s_2$ are respectively called source location and target locations of transition $e$ and denoted by $\overleftarrow{e}$ and $\overrightarrow{e}$ respectively, the set $\lambda$ is called set of clocks reset by $e$ and is denoted by $\lambda_e$, and the time constraint $\gamma$ is called clock constraint of $e$ and is denoted by $\gamma_e$;*
5. *$I$ is a mapping from $S$ to $\Phi(C)$. The mapping $I$ assigns to each state $s \in S$ a time constraint $I(s)$ which is called invariant of location $s$. $I(s)$ is a conjunction of formulas of the form $x \leq c$ , where $x \in X$ and $c \in \mathsf{N}$ or*

$I(s) = True$. *Intuitively, $M$ can stay in $s$ only if the clock valuation satisfies the invariant $I(s)$.*

Because the formula $(x \leq c_1) \wedge (x \leq c_2)$ is equivalent to $x \leq \min(c_1, c_2)$, and the formula $(x \geq c_1) \wedge (x \geq c_2)$ is equivalent to $x \geq \max(c_1, c_2)$, we can assume that for any transition $e$, for any location $s$ there is at most one formula of the form $x \leq c$ and at most a formula of the form $x \geq c$ in $\gamma_e$ or $I(s)$ for each clock $x$,

**Example.** A timed automaton is shown in Figure 1. It has two clocks: $x$ and $y$. There are four locations $A, B, C$ and $D$ in this automaton. The invariants associated with the locations are $I(A) = x \leq 10$, $I(C) = x \leq 10$ and $I(B) = I(P) = True$.

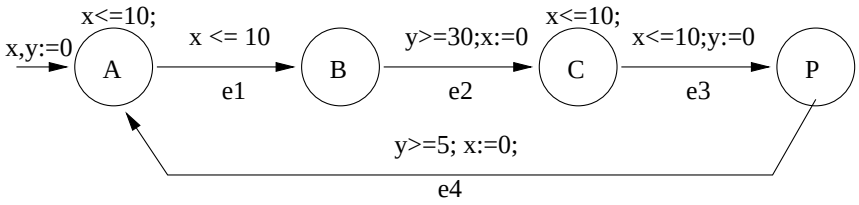


**Fig. 1.** A Timed Automaton

A clock valuation $u \in \mathsf{R}^C$ is an assignment of a non-negative real to each clock in $C$. A clock valuation $u$ satisfies a time constraint $\gamma \in \Phi(C)$, denoted by $u \in \gamma$, iff $\gamma$ evaluates to true when replacing each clock variable $x$ by its value $u(x)$. A configuration of the timed automaton $M$ is a pair $\langle s, u \rangle$, where $s$ is a location and $u$ is a clock valuation satisfying $u \in I(s)$. When the timed automaton $M$ is in a configuration $\langle s, u \rangle$, it can evolve either by moving through a transition to another location (*discrete step*), or by letting time pass without changing to another location (*time lapse step*). Formally, we define the consecutive relation on the set of configurations of $M$ as follows.

1. *Discrete step:* Let $e$ be a transition of $M$. Let $u$ and $u'$ be clock valuations such that $\langle \overleftarrow{e}, u \rangle$ and $\langle \overrightarrow{e}, u' \rangle$ are configurations. Then $\langle \overleftarrow{e}, u \rangle \xrightarrow{e} \langle \overrightarrow{e}, u' \rangle$ iff $u \in \gamma_e$ and $u' = u[\lambda_e \mapsto 0]$.
2. *Time lapse step:* Let $d$ be a non-negative real. Let $v + d$ denote the clock valuation $v'$ defined by $v'(x) = v(x) + d$ for all $x \in C$. Then $\langle s, u \rangle \xrightarrow{d} \langle s, u + d \rangle$ iff $u + d \in I(s)$. (Because $I(s)$ is a conjunction of the formula of the form $x \leq c$, $u + d \in I(s)$ implies that the clock valuation $u + \varepsilon \in I(s)$ for all $\varepsilon$, $0 \leq \varepsilon \leq d$.)

Let $\langle s_1, u_1 \rangle$ and $\langle s_2, u_2 \rangle$ be two configurations. We write $\langle s_1, u_1 \rangle \xrightarrow{e,d} \langle s_2, u_2 \rangle$ iff $\langle s_1, u_1 \rangle \xrightarrow{d} \langle s_1, u_1 + d \rangle$ and $\langle s_1, u_1 + d \rangle \xrightarrow{e} \langle s_2, u_2 \rangle$, or equivalently, $\overleftarrow{e} = s_1$,

$\overrightarrow{e} = s_2$, $u_1 + d \in I(s_1) \wedge \gamma_e$ and $u_2 = (u_1 + d)[\lambda_e \mapsto 0] \in I(s_2)$. The pair $(e, d)$ is called a combined step.

A *run* $\alpha$ of $M$ is an evolution history of $M$ written as

$$\alpha = \langle s^0, u_0 \rangle \xrightarrow{e_1,d_1} \langle s_1, u_1 \rangle \xrightarrow{e_2,d_2} \langle s_2, u_2 \rangle \xrightarrow{e_3,d_3} \ldots \xrightarrow{e_n,d_n} \langle s_n, u_n \rangle ,$$

where $u_0$ assigns 0 to all clocks in $C$. The timed-stamped transition sequence

$$\tau = (e_0, 0), (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n) ,$$

where $t_i = \sum_{j=1}^{i} d_j$ , and $e_0$ is the virtual initial transition, is called the behaviour corresponding to the run $\alpha$.

A run $\alpha$ is said to be integral iff all the $d_i$s occurred in its combined steps are integers. The corresponding behaviour of an integral run is also called integral behaviour.

In this paper, we consider only those timed automata which satisfy that given a possitive real $\eta$ any behaviour $(e_0, 0), \ldots, (e_n, t_n)$ is a proper prefix of another behaviour $(e_0, 0), \ldots, (e_m, t_m)$ for which $t_m > \eta$. It is not difficult to see that any non-zeno timed automaton can be modified without any change of the semantics to satisfy this restriction.


**Fig. 2.** DC model defined by a behaviour $\tau$.

The problem we are concerning is to verify the timed automata $M$ w.r.t. a real-time property. A real-time property of the automaton $M$ is defined as a predicate over its runs. A property $D$ is said to be satisfied by $M$ iff $D$ is satisfied by all of its runs. When $D$ is a property for the integral runs, i.e. $D$ is a property of $M$ in discrete time domain, verifying for the property $D$ of the automaton $M$ is reduced to an integer time verification. Such a property $D$ is said to be discrete. In general, we define the discretisability of a property for $M$ as follows.

**Definition 2.** *A real-time property $D$ of a timed automaton $M$ is said to be discretisable iff the property $D$ is satisfied by the automaton $M$ exactly when $D$ is satisfied by all the integral behaviours of $M$.*

Therefore, if $D$ is discretisable w.r.t. $M$ then verifying $M \models D$ can be done by using the integral-time verification methods, which are well established.

Although the discretisability is rather restrict, it is satisfied by many interesting verification problems.

Each run $\alpha$ of the timed automaton $M$ can be viewed as a model of the Duration Calculus formulas over state variables which are locations of $M$ in the natural way (see, e.g. [1] for more details). Namely, a behaviour $\tau = (e_0, 0), (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)$ defines a function $f_\tau$ from Reals (time) to the set $S$ of locations by $f_\tau(t) = s$ iff $t_i \leq t < t_{i+1}$ and $\overleftarrow{e}_{i+1} = s$. Thus, given a location $s$, $s$ defines the boolean function $s(t)$ ($s$ is overloaded to be a boolean function) by $s(t) = 1$ iff $f_\tau(t) = s$, and the duration $\int s$ of $s$ over an interval $[b, e]$ along the behaviour $\tau$ is defined as $\int_b^e s(t)dt$. Let $\Psi = \sum_{s \in S} c_s \int s$ be a linear duration expression. We use $\Psi_i^j(\tau)$ to denote the value of $\Psi$ over the interval $[t_i, t_j]$ along $\tau$, i.e. the value of $\Psi$ over the behaviour fragment $(e_i, t_i), (e_{i+1}, t_{i+1}), \ldots, (e_j, t_j)$. So, $\Psi_i^j(\tau) = \sum_{s \in S} c_s \int_{t_i}^{t_j} s(t)d(t) = \sum_{k=i+1}^{j} c_{\overleftarrow{e}_k}(t_k - t_{k-1})$. For the simplicity, we write $\Psi(\tau)$ for $\Psi_0^n(\tau)$.

**Example.** Let $\tau = (e_0, 0), (e_1, 5), (e_2, 35), (e_3, 45), (e_4, 50)$ be a behaviour of the timed automaton shown in Figure 1. Figure 2 shows how the functions $A(t)$, $B(t)$ and $C(t)$ are defined. The duration $\int A$ over the interval $[0, 50]$ along $\tau$ is $\int_0^{50} A(t)dt = 5$.

**Proposition 1.** *Let $M$ be a timed automaton. Let $D_1$, $D_2$ be DC formulas of the following forms*

$$\begin{array}{ll} D_1 \mathrel{\widehat{=}} \sum_{s' \in S} c_{s'} \int s' \leq a & \textit{(Linear Duration Properties)} \\ D_2 \mathrel{\widehat{=}} \Box(\lceil\!\lceil s \rceil\!\rceil^0 \frown \lceil\!\lceil \neg s \rceil\!\rceil \frown \lceil\!\lceil s \rceil\!\rceil^0 \Rightarrow & \\ \qquad \sum_{s' \in S} c_{s'} \int s' \leq a) & \textit{(Inter-State Duration Properties)}, \end{array}$$

*where $S$ is the set of locations of $M$, $s \in S$, and all $c_{s'}$ and $a$ are reals. Then, $D_1$ and $D_2$ are discretisable w.r.t. $M$.*

Given a behaviour $\tau$ of $M$, $\lceil\!\lceil s \rceil\!\rceil^0$ is a DC formula which is true at an interval $[t, t']$ iff $t = t' \wedge s(t) = 1$. $\lceil\!\lceil \neg s \rceil\!\rceil$ is true at an interval $[t_1, t_2]$ iff $\forall t \bullet t_1 < t < t_2 \Rightarrow s(t) = 0$. Thus, the premise of formula $D_2$ is satisfied at the interval of time $[a, b]$ iff $s(a) = s(b) = 1$ and $s(t) = 0$ for all $a < t < b$. Consequently, the timed automaton $M$ satisfies $D_1$ iff for each behaviour $\tau$ of $M$ it holds $\Psi(\tau) \leq a$ . The timed automaton $M$ satisfies $D_2$ iff there is no behaviour $\tau = (e_0, 0), (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)$ satisfying that:

1. $\overrightarrow{e_n} = s$ and there is another transition $e_i$ in $\tau$ satisfying $\overleftarrow{e}_i = s$.
2. there is no transition between $e_i$ (excluded) and $e_n$ of which the source location is $s$.
3. $\Psi_i^n(\tau) > a$.

Because only the fragment between $e_i$ and $e_n$ makes contribution to the value of $\Psi$ when checking $D_2$, we call this fragment *effective fragment* for $D_2$. For the sake of convenience, we call any behaviour the effective fragment for $D_1$.

**Example.** For the timed automaton shown in Figure 1, it can be seen easily that the integrated duration the automaton stays in state $C$ is at most one third of the duration it stays in other states. This property can be expressed as the following linear duration property

$$3\int C - \int A - \int B - \int P \leq 0\,.$$

The property that for each single period between leaving $A$ and entering $A$, the time the automaton stays in $C$ at most half of the time it stays in others can be expressed as the following inter-state duration property

$$\Box(\llbracket A \rrbracket^0 \frown \llbracket \neg A \rrbracket \frown \llbracket A \rrbracket^0 \Rightarrow 2\int C - \int P - \int B \leq 0)\,.$$

## 3 Exhaustive Exploration Algorithms for Checking Discretisable Duration Properties

Let $M$ be a timed automaton. Because we are only interested in the discretisable properties, from now on we will consider only the integral runs and behaviours of the timed automaton $M$. Note that in all the configurations of an integeral run the value of clocks are integers.

It is well known that the region equivalence relation over configurations of a timed automaton is of finite index, see e.g. [5], based on which one can construct the reachability graph for the timed automaton, and many interesting properties of the automaton can be verified by investigating the reachability graph. In this section, we will use this technique to check a timed automaton w.r.t. a linear duration property or an inter-state duration property.

Let $K_x$ be the largest constant compared with the clock $x$ in the time constraints and state invariants of $M$, the region-equivalence relation restricted into integral configurations is defined as follows.

**Definition 3.** *Integral configurations $\langle s, u\rangle$ and $\langle s, v\rangle$ are region-equivalent, denoted $\langle s, u\rangle \equiv \langle s, v\rangle$, iff for each clock $x$, either $u(x) = v(x)$ or $u(x) > K_x \wedge v(x) > K_x$.*

From the definition, it is obvious that the number of the equivalence classes of the region-equivalence relation is bounded by $|S|\Pi_{x\in C}(K_x + 1)$ (see, e.g. [5]).

**Proposition 2.** *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be integral configurations satisfying $\mathcal{C}_1 \equiv \mathcal{C}_2$. Then, for any integer $d$ and transition $e$, $\mathcal{C}_1 \xrightarrow{e,d} \mathcal{C}_1'$ iff there is a configuration $\mathcal{C}_2'$ satisfying $\mathcal{C}_2 \xrightarrow{e,d} \mathcal{C}_2'$ and $\mathcal{C}_1' \equiv \mathcal{C}_2'$. (This means that $\equiv$ is a strong bisimulation.)*

Let $K = \max\{K_x \mid x \in C\}$. Based on the equivalence relation $\equiv$, the reachability graph $G$ of the timed automaton $M$ is built as follows. Let $G = (V, A)$, where $V$ is the set of nodes and $A$ is the set of edges. The set $V$ and $A$ are defined as follows. Each element of $V$ is an equivalence class.

1. The equivalence class $[\langle s^0, \overline{0}\rangle]$ is in the set $V$ of nodes.
2. Let $[\mathcal{C}]$ be an equivalence class in $V$. For a transition $e$, for an integer $d \leq K + 1$, if $\mathcal{C} \xrightarrow{e,d} \mathcal{C}'$ for some configuration $\mathcal{C}'$ then $[\mathcal{C}']$ is also in $V$, and the edge $([\mathcal{C}] \rightarrow [\mathcal{C}'])$ is in the set $A$.

From the Proposition 2, given an equivalence class $[\mathcal{C}]$ in $V$, we can use an arbitrary configuration in $[\mathcal{C}]$ to calculate the successive nodes of $[\mathcal{C}]$. So the reachability graph can be generated algorithmically.

From the definition of $\equiv$, the configurations in the same node have the same location. So we say the location of a node the location of any configuration in it.

Let $n$ and $n'$ be two nodes satisfying that there is an edge from $n$ to $n'$. Given a linear duration expression $\Psi = \sum_{s \in S} c_s \int s$, the $\Psi$-distance between $n$ and $n'$ is defined as $\sup\{c_s d \mid \exists e \bullet \mathcal{C} \xrightarrow{e,d} \mathcal{C}' \wedge \mathcal{C} \in n \wedge \mathcal{C}' \in n'\}$, where $s$ is the location of $n$. The $\Psi$-distance between $n$ and $n'$ says how much the value of $\Psi$ along a behaviour ending at a configuration in $n$ can maximally increase if we extend the behaviour by making one combined step to a configuration in $n'$. Given a path $p = n_1, n_2, \ldots, n_k$ of $G$, we say the run fragment $\mathcal{C}_1 \xrightarrow{e_1,d_1} \mathcal{C}_2 \xrightarrow{e_2,d_2} \ldots \xrightarrow{e_{k-1},d_{k-1}} \mathcal{C}_k$ is a run fragment corresponding to $p$ iff for all $i$, $(1 \leq i \leq k)$, $\mathcal{C}_i \in n_i$. We also say the sequence $e_1, e_2, \ldots, e_k$ is a corresponding transition sequence of $p$. From the definition of $G$, we have the following proposition.

**Proposition 3.** *Let $\Psi$ be a linear duration expression. Any run fragment $\alpha$ is a corresponding run fragment of a path $p$ in the reachability graph with the $\Psi$-length non less than the value of $\Psi$ along $\alpha$. Reversely, for a path $p$ of the reachability graph, if the $\Psi$-length of $p$ is finite, there exists a run fragment $\alpha'$ corresponding to $p$ such that $\Psi(\alpha')$ equals to the $\Psi$-length of $p$.*

*Proof.* The first half of the proposition follows immediately from the definition of the $\Psi$-distance between two nodes in the reachability graph. The second half of the proposition is proved as follows. Let $n$ and $n'$ be two nodes in the reachability graph such that there is an edge from $n$ to $n'$. From Proposition 2, if $\Psi$-distance from $n$ to $n'$ is finite, for any configuration $\mathcal{C}$ in $n$, we can find a transition $e$, an integer time $d$ and a configuration $\mathcal{C}'$ in $n'$ such that $c_s d$ equals to the $\Psi$-distance and $\mathcal{C} \xrightarrow{e,d} \mathcal{C}'$, where $s$ is the location of $n$. So given a path $p = n_1, n_2, \ldots, n_m$, we can construct a run fragment $\alpha$ starting from an arbitrary configuration $\mathcal{C}_1$ in $n_1$, $\alpha = \mathcal{C}_1 \xrightarrow{e_1,d_1} \mathcal{C}_2 \xrightarrow{e_2,d_2} \ldots \xrightarrow{e_{m-1},d_{m-1}} \mathcal{C}_m$ such that $\mathcal{C}_i \in n_i$ and $c_{s_i} d_i$ equals to the $\Psi$-distance from $n_i$ to $n_{i+1}$, where $s_i$ is the location of $\mathcal{C}_i$. That is, the value of $\Psi$ along $\alpha$ equals to the $\Psi$-length of $p$.

In the following we present algorithms for deciding whether a linear duration property or an inter-state duration property is satisfied by the timed automaton

$M$ based on investigating the reachability graph of $M$. These algorithms can be generalised for other discretisable properties as well.

**Checking linear duration properties.** Let $\Psi$ be a linear duration expression and $\Psi \leq a$ be a linear duration property. From the above observation, if there is a cycle in the reachability graph of $M$ along which the value of $\Psi$ is positive then because each node in the graph is reachable from the initial one, we can build a run which makes the value of $\Psi$ as large as possible by repeating the cycle. This means that the linear duration property is violated, and we are done. Otherwise, by investigating all the paths in the reachability graph from the initial node having no cycle (the number of such paths is finite), we can decide if the property is satisfied. Therefore, we have the following algorithm for checking for the linear duration property $\Psi \leq a$.

Associating with each node $n$ in the reachability graph, there is a real variable $\Psi_n$. For each node $n$, $\Psi_n$ stores the $\Psi$-length of the longest path among those paths which have been checked and of which the last node is $n$. Let $\overline{p}$, $\overline{\Psi}$ be auxilary variables. The auxiliary variable $\overline{p}$ , whose type is the set of lists of nodes, is used to record the current path. The auxiliary variable $\overline{\Psi}$ is of the type real and is used to record the $\Psi$-length of $\overline{p}$. The initial value of $\overline{\Psi}$ is 0. Initially, each node in the reachability graph is marked 'unexplored'. The value of $\Psi_n$ is not defined if $n$ is marked unexplored.

$\overline{\Psi} := 0$; $\overline{p} =\ll [\langle s^0, \overline{0}\rangle] \gg$; $\Psi_{[\langle s^0,\overline{0}\rangle]} := 0$; {*initialisation*}
`while True do`
`begin`
  `if` $\overline{p} =\ll\gg$ `return true`;
  `while` the last node of $\overline{p}$ has no new successive node `do`
  `begin` {*back-tracking and re-calculating* $\overline{\Psi}$}
    delete the last node of $\overline{p}$;
    `if` $\overline{p} =\ll\gg$ `return true`;
    $\overline{\Psi} := \Psi_n$ , where $n$ is the last node of $\overline{p}$;
  `end`
  $n'$ := the last node of $\overline{p}$;
  $n$ := a new successive node of $n'$;
  Append $n$ to $\overline{p}$;
  {*calculating* $\Psi$*-length of* $p$}
  $\overline{\Psi} := \overline{\Psi} + l$, where $l$ is the $\Psi$-distance from $n'$ to $n$;
  `if` $\overline{\Psi} > a$ `return False`;
  `if` $n$ appears in $\overline{p}$ at least twice `and` $\overline{\Psi} > \Psi_n$ `then`
    `return False`;{*a positive circle has been found*}
  `if` $n$ is explored `and` $\Psi_n \geq \overline{\Psi}$ `then`
  `begin` {*back-tracking and re-calculating the* $\Psi$*-length*}
    delete the last node of $\overline{p}$;
    $\overline{\Psi} := \Psi_{n'}$;
  `end`
  `else begin`
    $\Psi_n := \overline{\Psi}$;

marked $n$ as explored;
end
end.

**Checking inter-state duration properties.** Let $\Psi \widehat{=} \sum_{s' \in S} c'_s \int s'$ be a linear duration expression. For checking whether $M$ satisfies an inter-state duration property $\Box(\llbracket s \rrbracket^0 \frown \llbracket \neg s \rrbracket \frown \llbracket s \rrbracket^0 \Rightarrow \Psi \leq a)$, we investigate in the reachability graph only those paths that start and end at nodes with the location $s$ and pass no node with the location $s$. For convenience, we say a node with location $s$ is an $s$-node. If for such a path there is a cycle passing no $s$-node that make the value of $\Psi$ positive then similarly to the above, we can conclude that the property to be checked is violated by $M$. Otherwise, by investigating the paths having no cycle like that, we can have the answer to the problem. Before giving the formal algorithm, let us consider an example.

**Example.** Fig. 3 shows a small part of the reachability graph of the timed automaton in Fig. 1. To check the inter-state duration property

$$\Box(\llbracket A \rrbracket^0 \frown \llbracket \neg A \rrbracket \frown \llbracket A \rrbracket^0 \Rightarrow 2\int C - \int P - \int B \leq 0)\,,$$

we have to check only the paths $n2 \rightarrow n4 \rightarrow n5 \rightarrow n6$ and $n3 \rightarrow n4 \rightarrow n5 \rightarrow n6$.

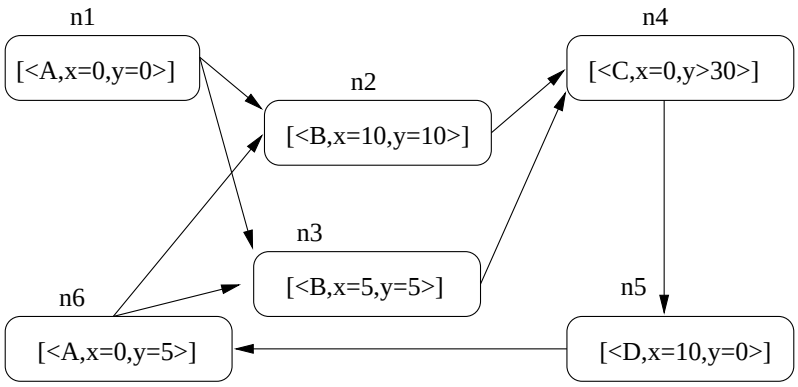


**Fig. 3.** A Part of Reachability Graph of the Timed Automaton

The algorithm is given as follows. Similarly to the previous algorithm, there is a real variable $\Psi_n$ associated with each node $n$ in the reachability graph. For each $n$, $\Psi_n$ records the biggest $\Psi$-length of the paths satisfying $\llbracket s \rrbracket^0 \hat{} \llbracket \neg s \rrbracket$ which have been checked and of which the last node is $n$. An auxiliary variable $\overline{p}$, whose type is the set of lists of nodes, is used to record the current path. An auxiliary variable $\overline{\Psi}$ is a real variable used to record the $\Psi$-length of the path $\overline{p}$. Let $m, n, n'$ be auxiliary variables of the type integral configurations. Like in the previous algorithm, initially, each node in the reachability graph is marked as unexplored. The value of $\Psi_n$ is undefined if $n$ is marked as unexplored.

**Step 1. Removing irrelevant nodes**
Remove all the nodes $n$ satisfying that $n$ is not an $s$-node, and cannot be the starting point of a path leading to an $s$-node.

**Step 2. Checking the behaviours**
To check whether $M$ satisfies $\Box(\llbracket s \rrbracket^0 \frown \llbracket \neg s \rrbracket \frown \llbracket s \rrbracket^0 \Rightarrow \Psi \leq a)$, all we have to do is to check that for each node $n$ which is a successive node of an $s$-node, there is no path starting from $n$, ending at an $s$-node and of which the $\Psi$-length is greater than $a$.

Given a node $m$ as a parameter, the following procedure returns `False` if and only if there is a path starting from $m$, ending at an $s$-node and of which the $\Psi$-length is greater than $a$. For each node $m$ which is a successive node of an $s$-node, we use $m$ as a parameter to call the procedure. This algorithm returns `True` iff the procedure returns `True` for each of such nodes.

$\overline{\Psi} := 0$; $\overline{p} :=\ll m \gg$; $\Psi_m := 0$;
`while True do`
`begin`
  `if` $\overline{p} =\ll\gg$ `return True`;
  `while` the last node of $\overline{p}$ has no new successive node `do`
  `begin`{*back-tracking and re-calculating the* $\Psi$*-length*}
    delete the last node of $\overline{p}$;
    `if` $\overline{p} =\ll\gg$ `return True`;
    $\overline{\Psi} = \Psi_n$, where $n$ is the last node of $\overline{p}$;
  `end`
  $n'$ := the last node of $\overline{p}$;
  $n$ := a new successive node of $n'$;
  append $n$ to $p$;
  $\overline{\Psi} := \Psi + l$, where $l$ is the $\Psi$-distance from $n'$ to $n$;
  `if` $\overline{\Psi} > a$ `and` the location of $n$ is $s$
    `then return False`;{*a counter-example found*}
  `if` $n$ appears in $\overline{p}$ at lease twice `and` $\overline{\Psi} > \Psi_n$
    `then return False`;{*positive circle found*}
  `if` ($n$ is explored `and` $\overline{\Psi} \leq \Psi_n$)
    `or` (the location of $n$ is $s$ `and` $\overline{\Psi} \leq a$) `then`
  `begin`{*back-tracking and re-calculating the* $\Psi$*-length*}
    delete the last node of $\overline{p}$;
    $\overline{\Psi} := \Psi_{n'}$, where $n'$ is the last node of $\overline{p}$;
  `end`
  `else begin`
    $\Psi_n := \overline{\Psi}$;
    mark $n$ as explored;
  `end`
`end`.

## 4 Floating Index Sets and Reduction of Reachability Graphs

In the previous section, we have presented two algorithms for deciding the satisfaction of some discretizable properties by the timed automata. These algorithms are based on the investigation in the reachability graphs. In general, the reachability graph for a timed automaton is so big that the exhaustive investigation in it becomes impossible. In this section we propose a technique for reducing the size of the graph to make the algorithms more efficient.

Let, in this section, $M$ be a timed automaton. Let $\tau = (e_0, 0), (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)$ be an integral behaviour (of $M$). We say that an index $i$ is floating iff by modifying $t_i$ a little bit (increasing and decreasing) we still get a behaviour. Note that for the untimed sequence $e_0 e_1 \ldots e_n$ of $\tau$ there exist a matrix $A$ and a vector $\overline{C}$ such that for a tuple of reals $\overline{X} = (x_0, x_1, \ldots, x_n)$ the sequence $(e_0, x_0)(e_1, x_1)\ldots(e_n, x_n)$ is a behaviour of $M$ iff the tuple $\overline{X}$ is a solution of the inequality system $A\overline{X} \leq \overline{C}$. Therefore, if there exists a floating index for the behaviour, the tuple $\overline{T} = (t_0, t_1, \ldots, t_n)$ is not a vertex of the polyhedron defined by $A\overline{X} \leq \overline{C}$. In the theory of linear programming it is well-known that a bounded linear function on a polyhedron always reaches its maximal value at one of the vertices of the polyhedron. This motivates our idea for this section.

Formally, a non-empty set $T = \{i_1, \ldots, i_k\} \subseteq \{0, 1, \ldots, n\}$ is said to be a *floating index set* (FIS) of $\tau$ iff the sequences $\tau^l(T)$ and $\tau^r(T)$ defined below are also behaviours. The sequences $\tau^l(T)$ and $\tau^r(T)$ are defined as

$$\tau^l(T) = (e_0, 0)(e_1, t'_1), (e_2, t'_2), \ldots, (e_n, t'_n)$$
$$\tau^r(T) = (e_0, 0)(e_1, t''_1), (e_2, t''_2), \ldots, (e_n, t''_n)$$

where

$$t'_i = \begin{cases} t_i - 1, & \text{if } i \in T \\ t_i\,, & \text{if } i \notin T \end{cases}, \quad t''_i = \begin{cases} t_i + 1, & \text{if } i \in T \\ t_i\,, & \text{if } i \notin T \end{cases}.$$

Let $D$ be either a linear duration property or an inter-state duration property. Let $\Psi = \sum_{s \in S} c_s \int s$ be the linear duration expression associated with $D$. As mentioned earlier, the value of $\Psi$ for the fragment from $t_k$ to $t_n$ along $\tau$ evaluates to $\sum_{j=k}^{n-1} c_{\overrightarrow{e}_j}(t_{j+1} - t_j)$. From the definition of $\tau^l(T)$ and $\tau^r(T)$, either $\Psi_k^n(\tau^l(T)) \geq \Psi_k^n(\tau)$ or $\Psi_k^n(\tau^r(T)) \geq \Psi_k^n(\tau)$. That is, from a behaviour having an FIS, we can construct another behaviour which is more likely to violate $D$. We will show that if a timed automaton does not satisfy $D$, either there is a behaviour having no FIS and violating $D$, or there is a location which can make unlimited contribution to the $\Psi$ value.

Let $\tau = (e_0, 0), (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)$ be an integral behaviour and $i, j$ be two indices ($1 \leq j < i \leq n$). We say that the index $j$ *left-binds* the index $i$, or $i$ *right-binds* the index $j$, (denoted $j \overset{\tau}{\hookleftarrow}_l i$ or $i \overset{\tau}{\hookleftarrow}_r j$) iff either $t_j = t_i$ or there is a clock $x \in C$ such that

1. $e_i$ resets $x$, and

2. $x \backsim t_i - t_j$ ($\backsim$ can be $\leq$ or $\geq$) is in the clock constraint of $e_i$ or the state invariant of $\overleftarrow{e}_i$, and
3. there is no transition between $e_j$ and $e_i$ which resets $x$.

Let $j \overset{\tau}{\hookleftarrow}_l i$. If $t_j = t_i$, because $e_j$ must take place before $e_i$ we can not increase $t_j$ without increasing $t_i$ or decrease $t_i$ without changing $t_j$. If $j$ and $i$ satisfy the conditions 1, 2 and 3, when $\backsim$ is $\leq$, we can not increase $t_i$ without increasing $t_j$ or decrease $t_j$ with decreasing $t_i$, when $\backsim$ is $\geq$, we can not increase $t_i$ without increasing $t_j$ or decrease $t_i$ with decreasing $t_j$.

We define relation $\overset{\tau}{\hookleftarrow}$ as the symmetrical and transitive closure of $\overset{\tau}{\hookleftarrow}_l$. Given two indices $i, j$ of $\tau$ satisfying $i \overset{\tau}{\hookleftarrow} j$, it is obvious that for any FIS $T$ of $\tau$, $i \in T$ iff $j \in T$. In fact, we have the following proposition.

**Proposition 4.** *Let $M$ be a timed automaton. Let $\tau$ be an integral behaviour of $M$. The behaviour $\tau$ has no FIS iff for each non-zero index $i$ of $\tau$, $0 \overset{\tau}{\hookleftarrow} i$.*

**Example.** Let $\tau = (e_0, 0), (e_1, 5), (e_2, 35), (e_3, 45), (e_4, 50)$ be a behaviour of the timed automaton shown in Figure 1. From the definition of $\overset{\tau}{\hookleftarrow}_l$, $2 \overset{\tau}{\hookleftarrow}_l 3$ and $3 \overset{\tau}{\hookleftarrow}_l 4$. The set $\{2, 3, 4\}$ is an FIS and $\{2, 3\}$ is not an FIS. Let $\Psi = 3\int C - \int A - \int B - \int P$ be a linear duration property. The value of $\Psi$ along $\tau$ is $\Psi(\tau) = 3 \times 10 - 5 - 30 - 5 = -10$. By decreasing the time stamps associated with $e_2$, $e_3$ and $e_4$, we get a new behaviour $\tau' = (e_0, 0), (e_1, 5), (e_2, 34), (e_3, 44), (e_4, 49)$. The value of $\Psi$ along $\tau'$ is $3 \times 10 - 5 - 29 - 5 = -9$. That is, based on $\tau$ and an FIS $\{2, 3, 4\}$ of $\tau$, we construct another behaviour $\tau'$ of which the value of $\Psi$ is larger.

The following proposition says that either the 'maximum' of $\Psi$ is achieved at a behaviour having no floating index set, or there is a location in which the automaton can stay for unlimited time to make the $\Psi$ value unlimited large. The latter implies that the value of $\Psi$ is unbounded on the set of behaviour.

**Proposition 5.** *Let $D$ be a linear duration property or an inter-state duration property. If the automaton $M$ does not satisfy $D$, there should be a behaviour $\tau$ violating $D$ and satisfying that*

1. *either $\tau$ has no FIS, or*
2. *there is an index $i$ in the effective segment satisfying $c_{\overleftarrow{e}_i} > 0 \wedge t_i - t_{i-1} > K$ ($K$ is the largest constant appeared in the time constraints and state invariants), and furthermore, for each index $j$ it holds that $j > i \Rightarrow i \overset{\tau}{\hookleftarrow} j$ and $j < i \Rightarrow 0 \overset{\tau}{\hookleftarrow} j$*

Together with Proposition 4, this proposition demonstrates that in order to verify for a linear duration property or an inter-state duration property, we have to check only those behaviours $\tau$ which satisfy that for each index $i$ ($i \neq 0$)

1. either there is another index $j$, $j < i$ and $j \overset{\tau}{\hookleftarrow} i$,
2. or $c_{\overleftarrow{e}_i} > 0 \wedge t_i - t_{i-1} > K$ ($K$ is the largest constant appeared in the time constraints and state invariants).

From this, when investigating the reachability graph $G$, we need only to check the paths satisfying that at least one of its corresponding behaviours satisfies the above conditions. So, if there is a node $n$ satisfying that there is no such a path which passes through $n$, $n$ can be removed without effecting the result of the investigation. Therefore, let $G_r$ be the graph (called 'reduced reachability graph') derived from the reachability graph $G$ by removing all of such nodes, then we can obtain the answer to the problem by investigating $G_r$ instead of $G$. In general, $G_r$ is much smaller than $G$. Hence, the efficiency of the algorithm is better. However, it is difficult to generate $G_r$. In the following, we will give a procedure to generate a reachability graph $G'$ satisfying $G_r \subseteq G' \subseteq G$.

More specifically, we will show that given a timed automaton $M$, we can construct a set $\mathcal{D}$ of tuples $(e_1, e_2, d)$ with the following property. If a behaviour $\tau$ is a prefix of a behaviour satisfying the conditions described in Proposition 5, then for each index $i > 0$ there is another index $j < i$ such that the tuple $(e_j, e_i, t_i - t_j)$ is in $\mathcal{D}$ and there is a clock reset by $e_j$, tested by $e_i$ and not reset by $e_k$ $(j < k < i)$. Moreover, if $\alpha = \mathcal{C}_0 \stackrel{e_1,d_1}{\longrightarrow} \mathcal{C}_1 \stackrel{e_2,d_2}{\longrightarrow} \mathcal{C}_2 \stackrel{e_3,d_3}{\longrightarrow} \ldots \stackrel{e_n,d_n}{\longrightarrow} \mathcal{C}_n$ is the run corresponding to such a behaviour $\tau$, for each $i$, there should be a node in $G_r$ containing $\mathcal{C}_i$.

Let $\tau = (e_0, 0), (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)$ be a behaviour. Let $i$ be an index of $\tau$. If there is an index $j$ satisfying $j < i$ and $j \stackrel{\tau}{\leftrightarrow} i$, then there is a sequence of indices starting from $i$, $j_1 = i, j_2, j_3, \ldots, j_k$ satisfying

1. $j_k < i$, $j_2, j_3, \ldots, j_{k-1} > i$, and
2. for each $p$ $(1 \leq p \leq k-1)$, $j_p \stackrel{\tau}{\leftrightarrow}_l j_{p+1}$ or $j_p \stackrel{\tau}{\leftrightarrow}_r j_{p+1}$ and
3. $(p \neq q) \Rightarrow (j_p \neq j_q)$.

The sequence can be viewed as a chain of indices, each index in the sequence left- or right- binds the next index. The distance $t_i - t_{j_k}$ between $e_{j_k}$ and $e_{j_1}$ $(e_i)$ equals to $\sum_{q=2}^{n} t_{j_q} - t_{j_{q-1}}$, and for each $q$, $t_{j_q} - t_{j_{q-1}}$ or $t_{j_{q-1}} - t_{j_q}$ equals to a constant appeared in the time constrains or the state invariants of $M$. Therefore, the tuple $(e_{j_k}, e_i, t_i - t_{j_k})$ must be in the set $\mathcal{D}$ which is defined as follows.

1. Let $e$ and $e'$ be two transitions. If a clock $x$ is reset by $e$, $x \backsim c$ is an enabling condition of $e'$ or a formula in $I(\overleftarrow{e})$, and there is a sequence of transitions $e_1(= e'), e_2, \ldots, e_{k-1}, e_k(= e)$ satisfying $\forall i(1 \leq i \leq k-1) \bullet \overrightarrow{e}_i = \overleftarrow{e}_{i+1} \wedge \mathrm{x} \notin \lambda_{e_i}$, then the tuple $(e, e', c)$ is in $\mathcal{D}$.
2. Let $e$ and $e'$ be two transitions. If there is a transition sequence $e_1(= e'), e_2, \ldots, e_{k-1}, e_k(= e)$ satisfying $\forall i(2 \leq i \leq k-1) \bullet \overrightarrow{e}_i = \overleftarrow{e}_{i+1}$ and these transitions can take place instantaneously, the tuple $(e, e', 0)$ is in $\mathcal{D}$.
3. Let $(e, e', d)$ and $(e'', e', d')$ be two tuples in $\mathcal{D}$. If $d' > d$ and if there is a transition sequence $e_1(= e''), e_2, \ldots, e_{k-1}, e_k(= e')$ satisfying $\forall i(1 \leq i \leq k-1) \bullet \overrightarrow{e}_i = \overleftarrow{e}_{i+1}$, $\exists j(1 < j < k) \bullet e_j = e$ and there is a clock $x$ reset by $e_1$ and does not reset by $e_i$ for all $1 < i < k$, then the tuple $(e'', e, d' - d)$ is also in $\mathcal{D}$.
4. $\mathcal{D}$ contains only these tuples.

**Proposition 6.** *Let $\tau = (e_0, 0), (e_1, t_1), \ldots, (e_n, t_n)$ be a behaviour. Let $i$ be an index of $\tau$. If there is an index $j$ satisfying $j < i$ and $j \overset{\tau}{\leftrightarrow} i$, there must be an index $j'$ satisfying $(e_{j'}, e_i, t_i - t_{j'}) \in \mathcal{D}$ and $j' < i$, and furthermore, if $t_i - t_{j'} > 0$, there must be a clock $x \in C$ which is reset by $e_{j'}$ and not reset by $e_k$ for all $j' < k < i$.*

Based on this proposition and Proposition 4, Proposition 5, we can give a necessary condition for a node $[\langle s, u\rangle]$ to be in $G_r$. Let $\alpha = \mathcal{C}_0 \overset{e_1,d_1}{\longrightarrow} \mathcal{C}_1 \overset{e_2,d_2}{\longrightarrow} \mathcal{C}_2 \overset{e_3,d_3}{\longrightarrow} \ldots \overset{e_n,d_n}{\longrightarrow} \mathcal{C}_n$ be a run satisfying the conditions described in Proposition 5. From the definition of $G_r$, each equivalence classes containing $\mathcal{C}_i$ $(0 \leq i \leq n)$ is in $G_r$. Let $\langle s, u\rangle$ be the configuration just before the $i$th transition takes place. There must exist a clock $x$, an index $j < i$ such that $e_j$ resets $x$, $x$ is not reset by the transitions between $e_{j+1}$ and $e_i$, and $(e_j, e_i, u(x)) \in \mathcal{D}$. From this observation, we can use the following algorithm to generate a reduced reachability graph.

This algorithm uses a depth-first method to generate the graph. The auxiliary variable $\overline{p}$ is used to record the current path. The type of two auxiliary variables $n$ and $n'$ is the set of nodes.

$\overline{p} :=\ll [\langle s^0, \overline{0}\rangle] \gg$;
`while` $\overline{p} \neq \ll\gg$ `do`
`begin`
  $n' :=$ the last node of $\overline{p}$;
  let $\langle s, u\rangle$ be a configuration in $n'$;
  `if` there is a new pair $(e, d)$ satisfying $\exists \mathcal{C} \bullet \langle s, u\rangle \overset{e,d}{\longrightarrow} \mathcal{C}$ and
    either $c_s > 0 \wedge d > K$
    or there is a sequence of transitions $e_1, e_2, \ldots, e_m$
      such that a clock $x$ is reset by $e_1$ and
      not reset by $e_k$ $(1 < k \leq m)$ and
      $e_m = e \wedge \overrightarrow{e}_i = \overleftarrow{e}_{i-1} \wedge (e_1, e, u(x) + d) \in \mathcal{D}$
    {This property is decidable}
  `then begin`
    $n :=$ $[\mathcal{C}]$ where $\langle s, u\rangle \overset{e,d}{\longrightarrow} \mathcal{C}$;
    `if` $n$ is not in the node set of graph `then`
    `begin`
      append $n$ to $\overline{p}$;
      add $n$ to the node set of the graph;
    `end`
    add $n' \rightarrow n$ to the edge set of the graph;
  `end`
  `else`
    delete the last node of $\overline{p}$;
`end`

**Proposition 7.** *Let $M$ be a timed automaton. Let $\Psi$ be a linear duration expression. Let $G'$ be the graph generated by the above algorithm. Let $D$ be a linear duration property $\Psi \leq a$ or an inter-state duration property $\Box(\llbracket s \rrbracket^0 \frown \llbracket \neg s \rrbracket \frown \llbracket s \rrbracket^0 \Rightarrow$*

$\Psi \leq a$). *The algorithms proposed in Section 3 working on $G'$ (instead of the reachability graph) can decide whether the timed automaton $M$ satisfies $D$.*

Notice that from the definition of the set $\mathcal{D}$, there is an algorithm to construct $\mathcal{D}$ for any timed automaton. The size of $\mathcal{D}$ is $\mathcal{O}(|E|^2K)$, where $E$ is the set of transitions and $K$ is the largest constant occurring in the enabling conditions and state invariants.

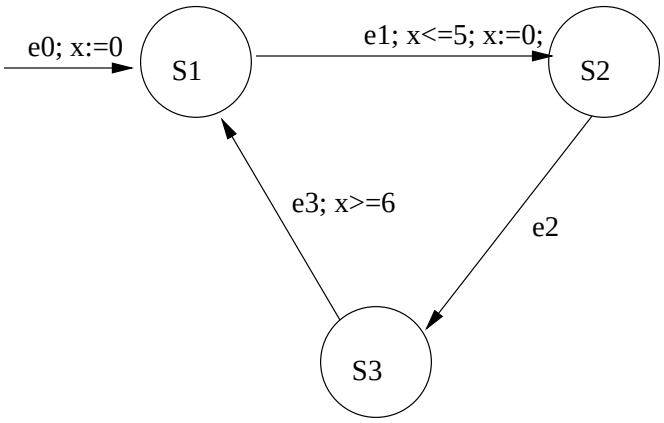


**Fig. 4.** A timed automaton

To show how to construct the set $\mathcal{D}$ and the 'reduced reachability graph', let us take the timed automaton shown in Fig.4 as an example. The set $\mathcal{D}$ for the automaton is constructed as follows.

1. From the first rule, the tuples $(e_0, e_1, 5)$, $(e_0, e_1, 7)$, $(e_1, e_1, 5)$, $(e_1, e_1, 7)$, $(e_1, e_2, 6)$ are in $\mathcal{D}$.
2. From the second rule, the tuples $(e_1, e_2, 0)$, $(e_1, e_3, 0)$ and $(e_2, e_3, 0)$ are in $\mathcal{D}$.
3. According to the third rule, the tuple $(e_1, e_2, 6)$ is added to $\mathcal{D}$ because $(e_1, e_3, 6)$ and $(e_2, e_3, 0)$ is in $\mathcal{D}$.

The 'reduced' reachability graph is generated based on the set $\mathcal{D}$ as follows. $[\langle S1, x = 0\rangle]$ is the initial node. Its successive node is $[\langle S2, x = 0\rangle]$. Because $(e_1, e_2, 6)$ and $(e_1, e_2, 6)$ in $D$, two successive nodes, $[\langle S3, x = 0\rangle]$ and $[\langle S3, x = 6\rangle]$, of $[\langle S2, x = 0\rangle]$ are also added to the graph. Because the tuples $(e_1, e_3, 6)$, $(e_1, e_3, 0)$ and $(e_2, e_3, 0)$ are in $\mathcal{D}$, the node $[\langle S1, x = 6\rangle]$ is added to the graph as the successive node of $[\langle S3, x = 0\rangle]$ and $[\langle S3, x = 6\rangle]$. No other successive node of these two nodes can be added to the graph. So, the 'reduced' reachability graph is shown in Fig 5. It should be noticed that if we don't use this technique, and we build the reachability graph according to its definition, then the nodes

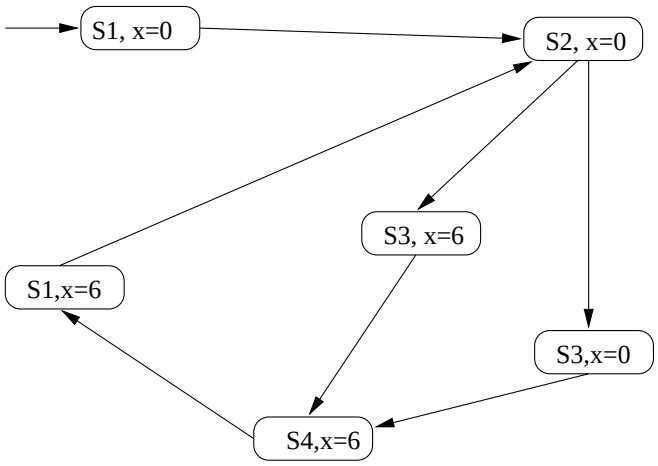


**Fig. 5.** A reduced reachability graph

$[\langle S3, x = 1\rangle]$, $[\langle S3, x = 2\rangle]$, $[\langle S3, x = 3\rangle]$, $[\langle S3, x = 4\rangle]$, $[\langle S3, x = 5\rangle]$, $[\langle S3, x > 6\rangle]$, $[\langle S4, x > 6\rangle]$, $[\langle S1, x > 6\rangle]$ would be included in the graph, too.

## 5 Example

As an example, let us take the railroad crossing system [7] and see how to generate a reduced reachability graph. We have trains, a railroad crossing monitor, and a gate controller which are subject to the following constraints (see Figure 6).

1. The monitor has four states to express the positions of train: state $A$ for train approaching beyond $1/2$ mile, state $B$ for train approaching within $1/2$ mile, state $C$ for train crossing, and state $P$ for train just passed.
2. The controller has four states to express the positions of the gate: state $U$ for the gate being up, state $Md$ for the gate moving down, state $Dn$ for the gate being down and $Mu$ for the gate moving up.

When the system starts, the monitor is in state $A$ and the controller is in state $U$. In state $A$, when the monitor detects that a train is approaching within $1/2$ mile, it enters state $B$, and at the same time if the controller is in state $U$ or state $Mu$, it must enters state $Md$. Namely, when the gate is up or is moving up, and detects that the monitor enters state $B$, it must start moving down. When the train enters the crossing, the monitor enters state $C$, and when the train has passed, it enters state $P$. When the monitor changes its state from $C$ to $P$ then at the same time the monitor changes its state from $Dn$ to $Mu$. This means, when the gate is down, and detects that the monitor enters state $P$, it begins to move up. In addition, due to the speed of trains and the safety

distance between trains, it takes at least 10 time units and at most 30 time units for a train to go to the crossing after entering state $B$. The train must pass the crossing with in 20 time units. When a train has passed, a new train could come after at least 100 time units. That means that the monitor stays in $B$ at least 10 time units and at most 30 time units each time and in $P$ at least 100 time units each time. Furthermore, assume that it takes the gate at most 10 time units to move down or move up, and hence, the controller can stay at $Md$ and $Mu$ at most 10 time units each time. Automata modelling the railroad crossing system are depicted in Figure 6. The timed automaton modelling the whole system is shown in Figure 7. The state $T$ is an auxiliary state used to identify the time point the monitor enters the state $B$. The system will stay in $T$ for 0 time unit.

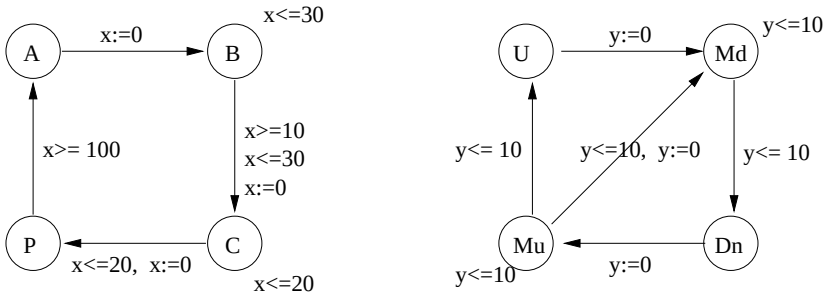


**Fig. 6.** Railroad Crossing Monitor and Gate

From the timed automaton in Fig. 7, the following properties can be derived for the Railroad Crossing System:

1. The trains cross the crossing safely. That is, each time the monitor is in state $C$, the gate must in state $Dn$. From Fig. 7, it can be seen that the only dangerous state is $(C, Md)$. However, the composed system can not stay in this state for any non-zero time. This property can be expressed as a linear duration property.
$$\int(C, Md) \leq 0\,.$$
2. In each period between two arrivals of trains, there should be at least 30 time units when the gate is in state U. That is, people have enough time to cross the railroad crossing. This property can be expressed as the inter-state duration property:
$$\Box(\llbracket T \rrbracket^0 \frown \llbracket \neg T \rrbracket \frown \llbracket T \rrbracket^0 \Rightarrow -\int(A, U) - \int(P, U) \leq -30).$$

From the definition, the set $\mathcal{D}$ of constraints about distance between two transitions built according to its definition is as follows.

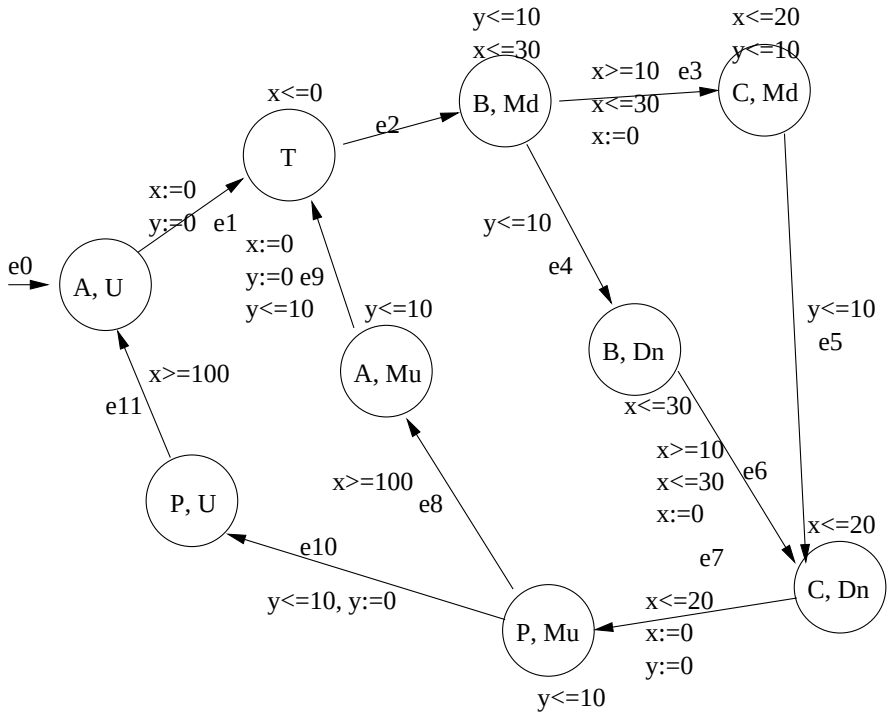


**Fig. 7.** Timed Automaton for Railroad Crossing System

$\{(e_1, e_2, 0), (e_9, e_2, 0), (e_1, e_3, 10), (e_9, e_3, 10)$
$(e_1, e_3, 30), (e_9, e_3, 30), (e_1, e_4, 10), (e_9, e_4, 10)$
$(e_1, e_4, 30), (e_9, e_4, 30), (e_3, e_5, 20), (e_1, e_5, 10)$
$(e_9, e_5, 10), (e_1, e_6, 10), (e_1, e_6, 30), (e_9, e_6, 10)$
$(e_3, e_7, 20), (e_6, e_7, 20), (e_7, e_8, 10), (e_7, e_8, 100)$
$(e_7, e_9, 10), (e_7, e_{10}, 10), (e_7, e_{11}, 100), (e_1, e_2, 0)$
$(e_9, e_2, 0), (e_2, e_8, 0), (e_3, e_5, 0), (e_4, e_6, 0)$
$(e_5, e_7, 0), (e_6, e_7, 0), (e_7, e_{10}, 0), (e_1, e_4, 0)$
$(e_9, e_4, 0), (e_5, e_{10}, 0), (e_6, e_{10}, 0), (e_{11}, e_1, 0)\}$

Based on this set, we can generate a reduced reachability graph. The graph is shown in Fig. 8 , which is much smaller than the original one.

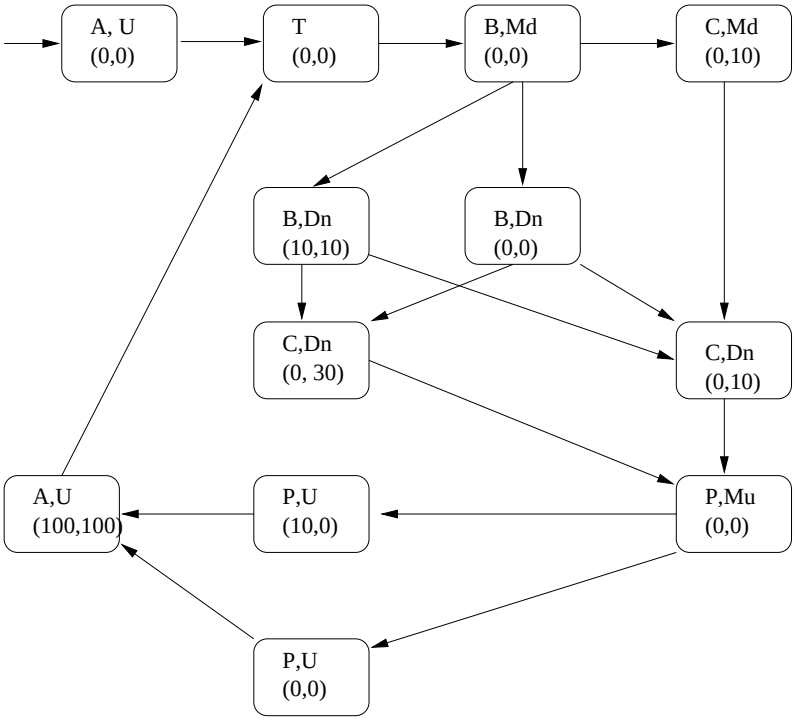


**Fig. 8.** The reduced reachability graph of Railroad Crossing System.

Applying the procedures in the previous sections on this reduced graph, we can decide that the Rail Road Crossing System satisfies the two properties mentioned above.

## 6 Conclusion

We have presented a technique for verifying for a linear duration property or an inter-state duration property of timed automata using integer verification methods. One characteristic of our technique is to reduce the problem into a search on a reachability graph with weights for edges, which has been well-studied. We have also given a procedure to generate a reduced weighted reachability graph which makes our algorithms more efficient. In comparison to the techniques in the literature, see e.g. [5], our technique is simpler, and for some cases, we can get smaller reduced reachability graph, which makes the computation complexity lower than theirs.

## References


1. Zhou Chaochen, Zhang Jingzhong, Yang Lu, and Li Xiaoshan. Linear Duration Invariants. Research Report 11, UNU/IIST, P.O.Box 3058, Macau, July 1993. Published in: *Formal Techniques in Real-Time and Fault-Tolerant systems*, LNCS 863, 1994.
2. Li Xuan Dong and Dang Van Hung. Checking Linear Duration Invariants by Linear Programming. Research Report 70, UNU/IIST, P.O.Box 3058, Macau, May 1996. Published in Joxan Jaffar and Roland H. C. Yap (Eds.), *Concurrency and Parallelism, Programming, Networking, and Security* LNCS 1179, Springer-Verlag, Dec 1996, pp. 321–332.
3. Li Xuan Dong, Dang Van Hung, and Zheng Tao. Checking Hybrid Automata for Linear Duration Invariants. Research Report 109, UNU/IIST, P.O.Box 3058, Macau, June 1997. Published in R.K.Shamasundar, K.Ueda (Eds.), *Advances in Computing Science*, Lecture Notes in Computer Science 1345, Springer-Verlag 1997, pp. 166-180.
4. Zhao Jianhua and Dang Van Hung. On Checking Real-Time Parallel Systems for Linear Duration Properties. Technical Report 130, UNU/IIST, P.O.Box 3058, Macau, January 1998. Presented at and published in the proceedings of *Formal Techniques in Real-Time and Fault-Tolerant Systems* 5th International Symposium, Lyngby, Denmark, September 1998 (FTRTFT'98), LNCS 1486, pp. 241–250, Springer-Verlag, 1998.
5. R. Alur and C. Courcoubetis and T.A. Henzinger. Computing accumulated delays in real-time systems. *Formal Methods in System Design*, pages 137–156, 1997.
6. Pham Hong Thai and Dang Van Hung. Checking a Regular Class of Duration Calculus Models for Linear Duration Invariants. Technical Report 118, UNU/IIST, P.O.Box 3058, Macau, July 1997. Presented at and published in the Proceedings of the *International Symposium on Software Engineering for Parallel and Distributed Systems* (PDSE'98), 20 – 21 April 1998, Kyoto, Japan, Bernd Kramer, Naoshi Uchihira, Peter Croll and Stefano Russo (Eds), IEEE Computer Society Press, 1998, pp. 61 – 71.
7. F. Wang, A.K.Mok, and E.A.Emerson. Distributed Real-Time System Specification and Verification in APTL. *ACM Transactions on Software Engineering and Methodology*, 2(4):346–378, October 1993.
8. Y.Kesten, A.Pnueli, J.Sifakis, and S.Yovine. Integration Graphs: A Class of Decidable Hybrid Systems. In *Hybrid System*, number 736 in LNCS, pages 179–208, 1993.

## A Proof of Some Propositions

**Proposition** 1 Let $M$ be a timed automaton. Let $D_1$, $D_2$ be DC formulas of the following forms

$$\begin{array}{l} D_1 \mathrel{\widehat{=}} \sum_{s\in S} c_s \int s \le a \qquad \text{(Linear Duration Properties)} \\ D_2 \mathrel{\widehat{=}} \Box(\lceil s \rceil^0 \frown \lceil \neg s \rceil \frown \lceil s \rceil^0 \Rightarrow \\ \qquad \Rightarrow \sum_{s'\in S} c_{s'} \int s' \le a \text{ (Inter-State Duration Properties)}\,, \end{array}$$

where $S$ is the set of locations of $M$, $s \in S$, and all $c_{s'}$ and $a$ are reals. Then, $D_1$ and $D_2$ are discretizable w.r.t. $M$.

*Proof.* For a behaviour $\tau = (e_1,t_1)\frown(e_2,t_2)\frown,\ldots,\frown(e_m,t_m)$ let $S_\tau = \{frac(t_i) \mid 0 \le i \le m\} \cup \{0,1\}$, where *frac(x)* is the fraction of $x$, *frac(x)* $= x - \lfloor x \rfloor$ ($\lfloor x \rfloor$ is the biggest integer that not greater than $x$). We define $rank(\tau)$ as the number of the elements of the set $S_\tau$. It follows from the definition of $rank(\tau)$ that $\tau$ is an integral behaviour iff $rank(\tau) = 2$.

Now we show that given an arbitrary behaviour

$$\tau = (e_1,t_1)\frown(e_2,t_2)\frown,\ldots,\frown(e_m,t_m)\,, \quad rank(\tau) > 2,$$

in which the fragment between $e_k$ ($k \ge 0$) and $e_m$ is the effective fragment, we can construct a behaviour $\tau'$ such that $rank(\tau') = rank(\tau) - 1$ and $\Psi_k^m(\tau') \ge \Psi_k^m(\tau)$. By applying this step repeatedly, starting from the behaviour $\tau$ we can get an integral behaviour $\tau''$ satisfying $\Psi_k^m(\tau'') \ge \Psi_k^m(\tau)$. Thus, if $D_1$ ($D_2$) is not satisfied by $M$, there is an integral behaviour which does not satisfy $D_1$ ($D_2$).

Let

$$\begin{array}{l} \mathcal{S}_\tau = \{\zeta_0,\zeta_1,\zeta_2,\ldots,\zeta_q,\zeta_{q+1}\}\,, \quad (\zeta_i < \zeta_{i+1})\,, \zeta_0 = 0\,, \zeta_{q+1} = 1\,, q > 0\,, \\ \mathcal{I}_{\zeta_1} = \{i \mid (1 \le i \le m) \wedge (frac(t_i) = \zeta_1)\}\,. \end{array}$$

We construct the following two sequences,

$$\begin{array}{l} \tau_1 = (e_1,t_1')\frown(e_2,t_2')\frown\ldots\frown(e_m,t_m')\,, \\ \tau_2 = (e_1,t_1'')\frown(e_2,t_2'')\frown\ldots\frown(e_m,t_m'')\,, \end{array}$$

where

$$t_i' = \begin{cases} t_i & i \notin \mathcal{I}_{\zeta_1} \\ t_i - \zeta_1 & i \in \mathcal{I}_{\zeta_1} \end{cases} \qquad t_i'' = \begin{cases} t_i & i \notin \mathcal{I}_{\zeta_1} \\ t_i - \zeta_1 + \zeta_2 & i \in \mathcal{I}_{\zeta_1} \text{ ($\zeta_2$ may be 1)}\,. \end{cases}$$

We claim that:

1. for all integers $i,j,a$ $(1 \le i < j \le m)$, it holds $(t_j - t_i \ge a) \Rightarrow (t_j' - t_i' \ge a \wedge t_j'' - t_i'' \ge a)$.
2. for all integers $i,j,b$ $(1 \le i < j \le m)$, it holds $(t_j - t_i \le b) \Rightarrow (t_j' - t_i' \le b \wedge t_j'' - t_i'' \le b)$.
3. for all integers $i$ $(1 \le i < m-1)$ it holds $t_i' \le t_{i+1}' \wedge t_i'' \le t_{i+1}''$.
4. $\tau_1$ and $\tau_2$ are behaviours, and either $\Psi(\tau_1) \ge \Psi_k^m(t)$ or $\Psi_k^m(\tau_2) \ge \Psi(t)$.

*Proof of (1)*: From the definition of $t_i'$ and $t_i''$, for any integers $i,j$, $(1 \le i < j \le m)$,

– If $i, j \in \mathcal{I}_{\zeta_1}$ then $t'_j - t'_i = t''_j - t''_i = t_j - t_i \geq a$;
– If $i, j \notin \mathcal{I}_{\zeta_1}$ then $t'_j - t'_i = t''_j - t''_i = t_j - t_i \geq a$;
– If $i \in \mathcal{I}_{\zeta_1} \wedge j \notin \mathcal{I}_{\zeta_1}$ then $t'_j - t'_i \geq t_j - t_i \geq a$, and because $frac(t_j - t_i) \geq \zeta_2 - \zeta_1$ we have $t''_j - t''_i = t_j - t''_i = t_j - t_i - (\zeta_2 - \zeta_1) = \lfloor t_j - t_i \rfloor + frac(t_j - t_i) - (\zeta_2 - \zeta_1) \geq \lfloor t_j - t_i \rfloor \geq a$;
– If $i \notin \mathcal{I}_{\zeta_1} \wedge j \in \mathcal{I}_{\zeta_1}$, then $t''_j - t''_i > t_j - t_i \geq a$, and because $frac(t_j - t_i) \geq \zeta_1$, we have $t'_j - t'_i = t_j - \zeta_1 - t_i = \lfloor t_j - t_i \rfloor + frac(t_j - t_i) - \zeta_1 \geq \lfloor t_j - t_i \rfloor \geq a$.

Therefore, (1) is proved. The proof of (2) is similar to the proof of (1), and (3) follows immediately from (1) and $t_{i+1} - t_i \geq 0$.
*Proof of (4)*: It is obvious that from the previous items it follows that $\tau_1$ and $\tau_2$ are behaviours of $\mathcal{N}$. From (3), each untimed state just after the occurrence of $i$th transition ($(1 \leq i \leq m)$ are the same for the sequences $\tau$, $\tau_1$ and $\tau_2$. We denote this state by $\overline{s}_i$. The value of $\Psi(\tau), \Psi(\tau_1)$ and $\Psi(\tau_2)$ are calculated as follows.

$$\Psi_k^m(\tau) = \sum_{i=k}^{m-1} C_{\overline{s}_i}(t_{i+1} - t_i)\,,$$
$$\Psi_k^m(\tau_1) = \sum_{i=k}^{m-1} C_{\overline{s}_i}(t'_{i+1} - t'_i)\,,$$
$$\Psi_k^m(\tau_2) = \sum_{i=k}^{m-1} C_{\overline{s}_i}(t''_{i+1} - t''_i)\,.$$

From the definition of $\tau_1$ and $\tau_2$,

$$\Psi_k^m(\tau_1) = \Psi_k^m(\tau) + \zeta_1(\sum_{i \in \mathcal{I}_{\zeta_1}, k<i<m} C_{\overline{s}_i} - \sum_{i+1 \in \mathcal{I}_{\zeta_1}, k \leq i} C_{\overline{s}_i})\,,$$
$$\Psi_k^m(\tau_2) = \Psi_k^m(\tau) + (-\zeta_2 + \zeta_1)(\sum_{i \in \mathcal{I}_{\zeta_1}, k<i<m} C_{\overline{s}_i} - \sum_{i+1 \in \mathcal{I}_{\zeta_1}, k \leq i} C_{\overline{s}_i})\,.$$

Because $\zeta_1 > 0$ and $(-\zeta_2 + \zeta_1) < 0$, we have that either $\Psi_k^m(\tau_1) \geq \Psi_k^m(\tau)$ or $\Psi_k^m(\tau_2) \geq \Psi_k^m(\tau)$.

Note that from the definition of $\tau_1$, it is obvious that $rank(\tau_1) = rank(\tau) - 1$. From the definition of $\tau_2$, for all $i \in \mathcal{I}_{\zeta_1}$ $t''_i$ will have the fraction $\zeta_2$, and hence it also holds that $rank(\tau_2) = rank(\tau) - 1$.

Now, by choosing $\tau' = \tau_1$ if $\Psi_k^m(\tau_1) \geq \Psi_k^m(\tau)$, or $\tau' = \tau_2$ if $\Psi_k^m(\tau_2) \geq \Psi_k^m(\tau)$, we have that $rank(\tau') = rank(\tau) - 1$ and $\Psi_k^m(\tau') \geq \Psi_k^m(\tau)$.

**Proposition 4**. Let $M$ be a timed automaton. Let $\tau$ be an integral behaviour of $M$. The behaviour $\tau$ has no FIS iff for each non-zero index $i$ of $\tau$, $0 \overset{\tau}{\leftrightarrow} i$.

*Proof.* First we prove that if for each non-zero index $i$ $0 \overset{\tau}{\leftrightarrow} i$, then $\tau$ has no FIS. In fact, for any index $i$, because $i \overset{\tau}{\leftrightarrow} 0$ implies that for any FIS $A$ $i$ is in $A$ iff 0 is also in $A$, so $i$ can not be in any FIS because 0 is not in any FIS. So, this part of the proposition is proved.

The other part is proved as follows. We will prove that if there is an index $i$ for which $0 \overset{\tau}{\not\leftrightarrow} i$, the set $T = \{j \mid j \overset{\tau}{\leftrightarrow} i\} \cup \{i\}$ is an FIS. We use $t'_k$ and $t''_k$ to denote the time stamps of the $k$th element of the sequence $\tau^l(T)$ and $\tau^r(T)$ respectively. It directly follows the definition of $\tau^l(T)$ and $\tau^r(T)$ that for all $k, l$, $-1 \leq (t'_l - t'_k) - (t_l - t_k) \leq 1$ and $-1 \leq (t''_l - t''_k) - (t_l - t_k) \leq 1$. Let $k$ and $l$ $(k < l)$ be two arbitrary indices such that there is a clock $x$ which is reset by $e_k$ and not reset by $e_p$ for each $p (k < p < l)$, and $x \backsim c$ is a formula in $I(\overleftarrow{e}_l)$ or $\gamma_{e_l}$. There are two cases to consider.

1. If $k, l \in T$ or $k, l \notin T$, $t'_l - t'_k = t''_l - t''_k = t_l - t_k$, so $t'_l - t'_k \backsim c$ and $t''_l - t''_k \backsim c$.
2. For $k \in T \wedge l \notin T$ or $k \notin T \wedge l \in T$, from the definition $t_l - t_k \neq c$ because $k \overset{\tau}{\not\mapsto} l$. Together with the fact that $-1 \leq (t'_i - t'_j) - (t_i - t_j) \leq 1$ and $-1 \leq (t''_i - t''_j) - (t_i - t_j) \leq 1$, we can conclude that $t'_l - t'_k \backsim c$ and $t''_l - t''_k \backsim c$.

Therefore, $\tau^l(T)$ and $\tau^r(T)$ are behaviours. This implies that the set $T$ is an FIS.

**Proposition 5**. Let $D$ be a linear duration property or inter-state duration property. If the automaton $M$ does not satisfy $D$, there must be a behaviour $\tau$ violating $D$ and satisfying that

1. either $\tau$ has no FIS, or
2. there is an index $i$ in the effective segment satisfying $c_{\overleftarrow{e}_i} > 0 \wedge t_i - t_{i-1} > K$ ($K$ is the largest constant appeared in the time constraints and state invariants), and furthermore, for each index $j$ it holds that $j > i \Rightarrow i \overset{\tau}{\leftrightarrow} j$ and $j < i \Rightarrow 0 \overset{\tau}{\leftrightarrow} j$.

*Proof.* Let $\tau = (e_0, 0), (e_1, t_1), \ldots, (e_n, t_n)$ be a behaviour of $M$ which does not satisfy $D$. Let $B_\tau$ be the set of behaviours satisfying that every behaviour in $B_\tau$ has the same untimed transition sequence as that of $\tau$. We will prove that there is a behaviour in $B_\tau$ satisfying the conditions in this proposition and violating $D$. There are two cases.

*Case (a).* If there is an index $i$ in an effective fragment of $\tau$ such that

1. $c_{\overleftarrow{e}_i} > 0$ and
2. for each clock $x$ in $C$, there is no time constraint of the form $x \leq c$ associated with a transition after $e_{i-1}$ and before $x$ being reset,
3. for each clock $x$ in $C$, there is no formula of the form $x \leq c$ in the state invariant $I(\overleftarrow{e}_j)$, where $j \geq i$ and $e_j$ is a transition before $x$ is reset,

then for a positive integer $d$ ($d > K$), the time stamped transition sequence

$$\tau' = (e_0, 0), (e_1, t_1), \ldots, (e_i, t_i), (e_{i+1}, t_{i+1} + d), (e_{i+2}, t_{i+2} + d), \ldots, (e_n, t_n + d)$$

is still a behaviour. For any index $j$ other than $i$ and satisfying $0 \overset{\tau'}{\not\mapsto} j$ and $i \overset{\tau'}{\not\mapsto} j$, let $I_j = \{j' \mid j \overset{\tau'}{\leftrightarrow} j'\} \cup \{j\}$. From the proof of Proposition 4, $I_j$ is an FIS. Hence, $\tau'^l(I_j)$ is also a behaviour. Because each time applying this step makes the time-stamp of some transitions decrease, repeating this step on the newly derived behaviour for finite times, we can get a behaviour $\tau''$ satisfying that for each index $j$ other than $i$ and 0, $0 \overset{\tau''}{\leftrightarrow} j \vee i \overset{\tau''}{\leftrightarrow} j$. Since the $i$th time stamp is at least $K$ greater than the $(i-1)$th time stamp, for any index $p, q$ satisfying $p \geq i$ and $q < i$, it holds $p \overset{\tau''}{\not\mapsto} q$. Therefore, for each index $j > i$, $i \overset{\tau''}{\leftrightarrow} j$ and for each index $j < i$, $0 \overset{\tau''}{\leftrightarrow} j$. Consequently, $\tau''$ satisfies the second condition in this proposition. Based on $\tau''$, we can increase the time stamps after the $(i+1)$th

position to get a behaviour violating $D$. It is obvious that this behaviour still satisfies the second condition.

*Case (b).* There is no index $i$ in $\tau$ satisfying the conditions in the case (a). For each index $j$, if $t_{j+1} - t_j > K + 1$, then there is no clock $x$ such that $x \leq c$ is a formula in $I(\overleftarrow{e}_k)$ or $\gamma_{e_k}$, where $e_k (k > j)$ is a transition before $x$ is reset in the sequence and $c$ is an integer constant. (It is because that $K + 1 > c$, and the $x \leq c$ cannot be satisfied.) From the assumption for this case, it follows $c_{\overleftarrow{e}_j} \leq 0$. The time stamped transition sequence

$$\tau' = (e_0, 0), (e_1, t_1), \ldots, (e_j, t_j), (e_{j+1}, t_{j+1} - \Delta), (e_{j+2}, t_{j+2} - \Delta), \ldots, (e_n, t_n - \Delta)\,,$$

where $\Delta = t_{j+1} - t_j - K - 1$, is also a behaviour. The value of $\Psi$ along $\tau'$ is not less than the value along $\tau$ because $c_{\overleftarrow{e}_i} \leq 0$ and $t_{j+1} - t_j > K + 1$. Repeating this step, we can get a behaviour $\tau''$ violating $D$ and satisfying that for each index $j$, $t''_j - t''_{j-1} \leq K + 1$, where $t''_j$ and $t''_{j-1}$ are $j$th and $(j-1)$th time stamps of $\tau''$.

Let $B'_\tau$ be the subset of $B_\tau$ satisfying that for each index $j$, $t_j - t_{j-1} \leq K + 1$. Since the set $B'_\tau$ is finite (we remind that we consider only the integral behaviours), we can select a behaviour $\tau_1$ from $B'_\tau$ such that the value of $\Psi$ along $\tau_1$ is not less than the values along other behaviours in $B'_\tau$. From above, for any behaviour $\tau_2$ in $B_\tau$, the value of $\Psi$ along $\tau_2$ is not greater than the value along $\tau_1$. It is obvious that $\tau_1$ violates $D$ because $\tau''$ is in $B'_\tau$. We show that we can construct a behaviour having no FIS and violating $D$ starting from $\tau_1$. Suppose $\tau_1$ has an FIS $T$. From the definition of $\tau_1^l(T)$ and $\tau_1^r(T)$, for each $k$ $(0 \leq k \leq n)$, it holds that $t'_k + t''_k = 2t_k$, where $t_k, t'_k$ and $t''_k$ are respectively the time stamps associated with the $k$th transitions of $\tau$, $\tau_1^l(T)$, and $\tau_1^r(T)$. So, for any fragment from $e_k$ to $e_n$,

$$\begin{aligned}
&\Psi_k^n(\tau_1^l(T)) + \Psi_k^n(\tau_1^r(T)) \\
&\quad = \textstyle\sum_{j=k+1}^{n} C_{\overleftarrow{e}_j}(t'_j - t'_{j-1}) + \sum_{j=k+1}^{n} C_{\overleftarrow{e}_j}(t''_j - t''_{j-1}) \\
&\quad = \textstyle\sum_{j=k+1}^{n} C_{\overleftarrow{e}_j}(t'_j + t''_j - (t'_{j-1} + t''_{j-1})) \\
&\quad = \textstyle\sum_{j=k+1}^{n} C_{\overleftarrow{e}_j}(2t_j - 2t_{j-1}) \\
&\quad = 2\textstyle\sum_{j=k+1}^{n} C_{\overleftarrow{e}_j}(t_j - t_{j-1}) \\
&\quad = 2\Psi(\tau)\,.
\end{aligned}$$

Because of the assumption that the value of $\Psi$ along $\tau_1$ is maximum, $\Psi_k^n(\tau_1^l(T)) \leq \Psi_k^n(\tau)$ and $\Psi_k^n(\tau_1^l(T)) \leq \Psi_k^n(\tau)$. Consequently, $\Psi_k^n(\tau_1^l(T)) = \Psi_k^n(\tau_1^r(T)) = \Psi_k^n(\tau)$. Hence, the following procedure gives a behaviour having no FIS and violating $D$. Let $\overline{\tau}$ be a variable.

$\overline{\tau} = \tau_1$;
`while` $\overline{\tau}$ has an FIS $T$ `do` $\overline{\tau} := \tau_1^l(T)$;
`return` $\overline{\tau}$.

This procedure will stop because the time stamps can not be decreased infinitely.

**Proposition 6**. Let $\tau = (e_0, 0), (e_1, t_1), \ldots, (e_n, t_n)$ be a behaviour. Let $i$ be an index of $\tau$. If there is an index $j$ satisfying $j < i$ and $j \overset{\tau}{\leftrightarrow} i$, there must be an

index $j'$ satisfying $(e_{j'}, e_i, t_i - t_{j'}) \in \mathcal{D}$ and $j' < i$, and furthermore, if $t_i - t_{j'} > 0$, there must be a clock $x \in C$ which is reset by $e_{j'}$ and not reset by $e_k$ for all $j' < k < i$.

*Proof.* From the definition of $\overset{\tau}{\leftrightarrow}$, there must exists a sequence of indices $J = j_1, j_2, \ldots, j_k$ ($k \geq 2$, $1 \leq j_i \leq m$) satisfying

1. $j_1 = i \wedge j_k < i \wedge (j_2, j_3, \ldots, j_{k-1} > i)$, and
2. for each $p$ ($1 \leq p \leq k-1$), $j_p \overset{\tau}{\leftrightarrow}_l j_{p+1}$ or $j_p \overset{\tau}{\leftrightarrow}_r j_{p+1}$ , and
3. $(p \neq q) \Rightarrow (j_p \neq j_q)$ .

From the assumption and the definition of $\mathcal{D}$, the following property for the sequence $J$ holds.

*For any two neighbouring indices $p, q$ in the sequence of indices, either $(e_p, e_q, t_q - t_p) \in \mathcal{D}$ or $(e_q, e_p, t_p - t_q) \in \mathcal{D}$.*

We will show that we can remove an index from $J$ with this property being preserved.

Let $j_q$ be the largest index in $J$ ($q \neq 1, k$ because $j_1$ and $j_k$ are two least indices in $J$). Because $t_{j_q} \geq t_{j_{q-1}}$ and $t_{j_q} \geq t_{j_{q+1}}$ $(e_{j_{q-1}}, e_{j_q}, t_{j_q} - t_{j_{q-1}}) \in \mathcal{D}$ and $(e_{j_{q+1}}, e_{j_q}, t_{j_{q+1} - t_{j_q}}) \in \mathcal{D}$. From the definition of $\mathcal{D}$, either $(e_{j_{q-1}}, e_{j_{q+1}}, t_{j_{q+1}} - t_{j_{q-1}}) \in \mathcal{D}$ (when $t_{j_{q+1}} \geq t_{j_{q-1}}$) or $(e_{j_{q+1}}, e_{j_{q-1}}, t_{j_{q-1}} - t_{j_{q+1}}) \in \mathcal{D}$ (when $t_{j_{q-1}} \geq t_{j_{q+1}}$).

Let $J' = j'_1, j'_2, \ldots, j'_{k-1}$ be the derived sequence of integers after $q$ is removed from $J$. It can be seen easily that the property described above holds for $J'$ as well.

We can repeat this step until we get a two-element sequence $j_1, j_k$ ($j_1 = i$) for which $(e_{j_k}, e_i, t_i - t_{j_k})$ is still in $\mathcal{D}$.

From the definition of $J$, $j_k \overset{\tau}{\leftrightarrow}_l j_{k-1}$. If $t_i > t_{j_k}$ then $t_{j_{k-1}} \geq t_i > t_{j_k}$. From the definition of $\overset{\tau}{\leftrightarrow}_l$, there must be a clock $x$ which is reset by $e_{j_k}$ and not reset by $e_k$ ($j_k \leq k < j_{k-1}$).

**Proposition 7**. Let $M$ be a timed automaton. Let $\Psi$ be a linear duration expression. Let $G'$ be the graph generated by the above algorithm. Let $D$ be a linear duration property $\Psi \leq a$ or an inter-state duration property $\Box(\llbracket s \rrbracket^0 \frown \llbracket \neg s \rrbracket \frown \llbracket s \rrbracket^0 \Rightarrow \Psi \leq a)$. The algorithms proposed in Section 3 working on $G'$ (instead of the reachability graph) can decide whether the timed automaton $M$ satisfies $D$.

*Proof.* To prove this proposition, we need to prove that each behaviour satisfying the conditions described in Proposition 5 is the corresponding behaviour of a path of the reduced reachability graph generated by the algorithm.

Let $\tau$ be such a behaviour and $\alpha = \mathcal{C}_0 \overset{e_1, d_1}{\rightarrow} \mathcal{C}_2 \overset{e_1, d_1}{\rightarrow} \ldots \overset{e_n, d_n}{\rightarrow} \mathcal{C}_m$ be its corresponding run. From Propositions 4 and 5, for any index $i$ of $\tau$, either there is an index $j$ ($j < i$) such that $j \overset{\tau}{\leftrightarrow} i$, or $c_{\overleftarrow{e}_i} > 0 \wedge t_i - t_{i-1} > K$. The initial node $[\mathcal{C}_0]$ is obvious in the reduced reachability graph. From Proposition 6, $[\mathcal{C}_0] \rightarrow [\mathcal{C}_1] \rightarrow \ldots \rightarrow [\mathcal{C}_n]$ is a path of the graph.