Checking Temporal Duration Properties of Timed Automata

LI Yong[⋆]
Dept. of Computer Sci&Tech, Nanjing University, P.R. China 210093
liyong@seg.nju.edu.cn
Dang Van Hung
International Institute for Software Technology,
The United Nations University, P.O. Box 3058, Macau SAR, P.R. China
dvh@iist.unu.edu

Abstract. In this paper, the problem of checking a timed automaton for a Duration Calculus formula of the form Temporal Duration Property [12] is addressed. We show that Temporal Duration Properties are in the class of discretisable real-time properties of Timed Automata, and give an algorithm to solve the problem based on linear programming techniques and the depth-first search method in the integral region graph of the automaton. The complexity of the algorithm is in the same class as for the solution of the reachability problem of timed automata.

1 Introduction

Model checking for real-time systems has been a great deal of attention for years. Many new model checking techniques have been developed and implemented for solving the problem (see, e.g. [10, 9]). We made two observations about them: First, most of these techniques are for checking a property of a real-time system at an *instant* of time; second, they are based on searching in the regional graph of the timed automata, and hence are in the same complexity class as the techniques for solving the reachability problem of timed automata.

In practice, sometimes we have to deal with a property of a system for *intervals* of time as well. For instance, one of the safety requirements for a gas burner is that for any interval of time that is more than one minute long, the duration for gas leaking should not exceed 5% of the length of the interval. This requirment is about time interval, not just time points. Duration Calculus [2] was introduced for specifying and reasoning about this kind of system properties. Some algorithms have been proposed to check a subclass of the timed automata for a subclass of Duration Calculus formulas called Linear Duration Invariants [3, 11, 7, 6]. Although these algorithms are based on the linear programing techniques, they have very high complexity because of the nature of the problem.

⋆ Supported by National Natural Science Foundation of China (Grant No. 69703009) and 863 High Technology Research Development Project (Grant No. 863-306-ZT06-04-4)

In this paper, we use timed automata as the standard model for real-time systems, but restrict ourselves to a smaller class of system properties for time intervals. Consider the following refinement for the requirement of the gas burner: (1) each gas leaking should not last longer than one second, and (2) any interval for which if the gas burner is traced as leaking, then not leaking and then leaking again, should be longer than 30 seconds. (1) and (2) are duration properties for a short system trace. These formulas are not in the class of linear duration invariants, but in the class of *temporal duration properties*.

We develop a technique to check a timed automaton for a temporal duration property. First, we show that a temporal duration property is satisfied by all the timed behaviors of a timed automaton if and only if it is just satisfied by all the integer timed behaviors of the timed automaton. Then, we propose an algorithm to check if the integer timed behaviors of a timed automaton satisfies a temporal duration property based on the linear programming techniques and the depth-first search method in the integral region graph of the automaton. The complexity of the algorithm is in the same class for the reachability problem of timed automata.

The paper is organized as follows. In the next section we recall some notations of timed automata [1], and introduce a class of so-called temporal duration properties for timed automata. In Section 3, we present our algorithm for checking the temporal duration properties of timed automata. The last section is the conclusion of the paper with some discussion about related work.

2 Timed Automata and Temporal Duration Properties

2.1 Timed Automata

In this section we recall some notations and related results from the theory of timed automata which will be used in this paper. The readers are referred to [1] for their more details.

A timed automaton is a finite state machine with an additional set of clock variables X and an additional set of clock constraints. A clock constraint ϕ over X is defined by the following grammar:

$$\phi \;\widehat{=}\; x \leq a \;\mid\; x \geq a \;\mid\; \phi_1 \wedge \phi_2,$$

where $x \in X$ and a stands for a natural number. Let $\Phi(X)$ denote the set of all clock constraints over X.

Definition 1. A timed automaton M is a tuple $\langle L, s_I, \Sigma, X, E\rangle$, where

- L is a finite set of locations,
- $s_I \in L$ is an initial location,
- Σ is a finite set of labels,
- X is a finite set of clocks,

– $E \subseteq L \times \Sigma \times \Phi(X) \times 2^X \times L$ is a finite set of transitions. An $e = \langle s, a, \phi, \lambda, s' \rangle \in E$ represents a transition from location s to location s', labeled with a; s and s' are called source and target locations of e, and denoted by $\overleftarrow{e}$ and $\overrightarrow{e}$ respectively; ϕ is a clock constraint over X that must be specified when the transition e is enabled, and $\lambda \subseteq X$ is the set of clocks to be reset by e when it takes place. In the sequel, we will use the subscript e with ϕ and λ to indicate that ϕ and λ are associated to e.

In this paper, we only consider the deterministic timed automata, i.e. those timed automata which have not more than one a-labeled edge starting from a location s for any label $a \in \Sigma$.

A clock interpretation ν for a set of clock X is a mapping $\nu : X \to \mathbf{Reals}$, i.e. ν assigns to each clock $x \in X$ the value $\nu(x)$. A clock interpretation represents the values of all clocks in X at a time point. We adopt the following denotations. ν_0 always denotes the clock interpretation which maps from X to $\{0\}$. For a clock interpretation ν and for $t \in R$, $\nu + t$ denotes the clock interpretation which maps each clock $x \in X$ to the value $\nu(x) + t$. For $\lambda \subseteq X$, $[\lambda \mapsto 0]\nu$ is the clock interpretation which assigns 0 to each $x \in \lambda$ and agrees with ν over the rest of the clocks.

A state of a timed automaton M is a pair $\langle s, \nu \rangle$, where $s \in L$ and ν is a clock interpretation for X. The fact that M is in a state $\langle s, \nu \rangle$ at a time instant means that M stays in location s with all clock values agreeing with ν at that instant.

The behavior of timed automata can be represented by time-stamped transition sequences. A behavior σ is a time-stamped transition sequence $\sigma = (e_1, \tau_1)(e_2, \tau_2) \ldots (e_m, \tau_m)$, where $m \geq 1$ and $e_i \in E$, $\overrightarrow{e_{i-1}} = \overleftarrow{e_i}$ for $1 \leq i \leq m$ (with the convention $\overrightarrow{e_0} = s_I$), and where $0 = \tau_0 \leq \tau_1 \leq \tau_2 \leq \ldots \leq \tau_m$, such that $(\nu_{i-1} + \tau_i - \tau_{i-1})$ satisfies ϕ_{e_i} for all $1 \leq i \leq m$, where $\nu_i = [\lambda_{e_i} \mapsto 0](\nu_{i-1} + \tau_i - \tau_{i-1})$ for $1 \leq i \leq m$.

So, a behavior σ expresses that M starts from the initial location s_I, transits to $\overrightarrow{e_1}$ by taking e_1 at time τ_1, then transits to $\overrightarrow{e_2}$ by taking e_1 at time τ_2, and so on, and at last transits to $\overrightarrow{e_m}$ at time τ_m. Note that $(\nu_{i-1} + \tau_i - \tau_{i-1})$ is the value of the clock variables just before e_i's taking place, and ν_i is the value of the clock variables just after e_i's taking place. The behavior σ expresses also that the system M stays in the location $\overleftarrow{e}_i$ for $\tau_i - \tau_{i-1}$ time units, and then transits to $\overleftarrow{e}_{i+1}$ for $(1 \leq i \leq m)$.

If $(e_1, \tau_1)(e_2, \tau_2) \ldots (e_m, \tau_m)$ is a behavior of timed automaton M, we call $\overrightarrow{e_m}$ a reachable location of M and $\langle \overrightarrow{e_m}, \nu_m \rangle$ a (discrete) reachable state of M.

In order to solve the emptiness problem for a timed automaton, Alur and Dill [1] have introduced a finite index equivalence relation over the state space of the automaton. The idea is to partition the set of the clock interpretations into a number of regions so that two clock interpretations in the same region will satisfy the same set of clock constraints.

For each $x \in X$, let K_x be the largest integer constant occurring in a clock constraint for the clock variable x of the timed automaton M, i.e. $K_x = \max\{a \mid \text{either } x \leq a \text{ or } x \geq a \text{ occurs in a clock constraint } \phi \text{ of a transition } e\}$. Let $K_X = \max_{x \in X} K_x$. For a real number r, let $\mathit{frac}(r) = r - \lfloor r \rfloor$ ($\lfloor r \rfloor$ is the maximal integer number which is not greater than r) be the fractional part of x. The equivalence relation $\cong$ over the set of clock interpretations is defined as follows: for two clock interpretations ν and ν', $\nu \cong \nu'$ iff the following three conditions are satisfied:

1. For all $x \in X$ either $\nu(x) > K_x \wedge \nu'(x) > K_x$ or $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$.
2. For all $x, y \in X$ such that $\nu(x) \leq K_x$ and $\nu(y) \leq K_y$, $\mathit{frac}(\nu(x)) \leq \mathit{frac}(\nu(y))$ iff $\mathit{frac}(\nu'(x)) \leq \mathit{frac}(\nu'(y))$.
3. For all $x \in X$ such that $\nu(x) \leq K_x$, $\mathit{frac}(\nu(x)) = 0$ iff $\mathit{frac}(\nu'(x)) = 0$.

When $\nu \cong \nu'$, it is not difficult to see that for any clock constraint ϕ occurring in a transition $e = \langle s, a, \phi, \lambda, s' \rangle \in E$, ν satisfies ϕ iff ν' satisfies ϕ.

A clock region for M is an equivalence class of the clock interpretations induced by $\cong$. We denote by $[\nu]$ the clock region to which a clock interpretation ν belongs. From the definition of $\cong$, a region is characterized by the integer part of the value of each clock x when it is not greater then K_x, by the order between the fraction part of the clocks when they are different from 0. Therefore, the number of clock regions is bounded by $|X|! \cdot 2^{|X|} \cdot \prod_{x \in X}(2K_x + 2)$. A configuration is defined as a pair $\langle s, \pi \rangle$ where $s \in L$ and π is a clock region. Based on the clock regions, the region automaton of M, whose states are configurations of M, can be defined. We will come back to this later.

2.2 Temporal Duration Properties

As said in the introduction of the paper, a temporal duration property is a constraint for state durations for a short trace of a certain pattern. It is defined formally in Duration Calculus [2] as follows.

Definition 2. A *temporal duration property* over L is a Duration Calculus formula of the form

$$\Box(\lceil s_{i_1} \rceil ^\frown \lceil s_{i_2} \rceil ^\frown \ldots ^\frown \lceil s_{i_k} \rceil \Rightarrow \textstyle\sum_{s \in L} c_s \int s \leq Q)$$

where L is a finite set of system locations, and c_s ($s \in L$) and Q are reals. For simplicity, let us denote

$$\begin{array}{ll} D & \widehat{=}\ \lceil s_{i_1} \rceil ^\frown \lceil s_{i_2} \rceil ^\frown \ldots ^\frown \lceil s_{i_k} \rceil \Rightarrow \sum_{s \in L} c_s \int s \leq Q, \\ \gamma(D) & \widehat{=}\ \lceil s_{i_1} \rceil ^\frown \lceil s_{i_2} \rceil \ldots ^\frown \lceil s_{i_k} \rceil. \end{array}$$

From now on in this paper, let $\Box D$ be a temporal duration property over L.

Although temporal duration properties are Duration Calculus formulas, the readers do not need to know about Duration Calculus for understanding this paper. For the comfort of the readers, who do not know about Duration Calculus, we will explain in the sequel the meaning of temporal duration properties and give a definition for the satisfaction of a temporal duration property by a timed automaton. Intuitively, a temporal duration property $\Box D$ says that for any time interval, in which the system evolves through the sequence of states $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$, the duration d_s of the states s's over that interval satisfies the constraint $\sum_{s\in L} c_s d_s \leq Q$) ($\int s$, when applied to an interval of time, is the accumulated time that the state s is present in the interval, and is called the duration of s over that interval).

Temporal duration properties form a class of Duration Calculus formulas that are often encountered in the development of real-time systems using Duration Calculus. For example, design decisions for the simple gas burner in [2]:

$$\Box(\lceil leak \rceil \Rightarrow \ell \leq 1),$$
$$\Box(\lceil leak \rceil ^\frown \lceil nonleak \rceil ^\frown \lceil leak \rceil \Rightarrow \ell \geq 30)$$

are temporal duration properties because $\ell = \int leak + \int nonleak$.

For any timed transition sequence $\sigma = (e_1, \tau_1)(e_2, \tau_2) \ldots (e_m, \tau_m)$, for $i \geq 1$, $j \geq 0$ such that $j + i \leq m$, let us denote by $\sigma(j, i)$ the subsequence $(e_{j+1}, \tau_{j+1}) \ldots (e_{j+i}, \tau_{j+i})$.

Definition 3. For a timed transition sequence $\sigma = (e_1, \tau_1)(e_2, \tau_2) \ldots (e_m, \tau_m)$, for any $j \geq 0$ such that $j+k \leq m$, we say $\sigma(j, k)$ matches $\gamma(D)$ (or $\gamma(D)$ matches σ) iff $\overleftarrow{e_{j+l}} = s_{i_l}$ for any l such that $1 \leq l \leq k$.

So, the fact '$\sigma(j, k)$ matches $\gamma(D)$' means that the temporal order of the location occurrences in $\sigma(j, k)$ is defined by $\gamma(D)$.

For a subsequence $\sigma(j, k)$ that matches $\gamma(D)$, the value of $\sum_{s\in L} c_s \int s$ over $\sigma(j, k)$ is defined by $\sum_{l=1}^{k} c_{s_{i_l}}(\tau_{j+l} - \tau_{j+l-1})$ and is denoted by $\theta(\sigma_\delta(j, k))$. Note that $\sum_{s_{i_l}=s, l\leq k}(\tau_{j+l} - \tau_{j+l-1})$ is the duration of the state s over $\sigma(j, k)$.

Definition 4. 1. A time-stamped transition sequence σ satisfies the temporal duration property $\Box D$, denoted by $\sigma \models \Box D$, iff for any subsequence $\sigma(j, k)$ of σ that matches $\gamma(D)$, the condition $\theta(\sigma(j, k)) \leq Q$ holds.
2. A timed automaton M satisfies the temporal duration property $\Box D$, denoted by $M \models \Box D$, iff for any behavior σ of M, $\sigma \models \Box D$ holds.

3 Checking Timed Automata for Temporal Duration Properties

3.1 Discretisable Properties

A behavior $\sigma = (e_1, \tau_1)(e_2, \tau_2) \ldots (e_m, \tau_m)$ of timed automaton M is said to be an integral behavior iff τ_i is an integer for all $1 \leq i \leq m$.

Definition 5. Let M be a timed automaton, and let P be a predicate over the behaviors of M. P is said to be discretisable (w.r.t. M) iff P is satisfied by all the behaviors of M exactly when P is satisfied by all the integral behaviors of M.

Therefore, if P is discretisable (w.r.t. M), verifying that P is satisfied by all the behaviors of M is reduced to verifying that P is satisfied by all the integral behaviors of M only, and hence can be done by using the integral-time verification methods presented in [6].

Now, we prove that temporal duration properties for a timed automaton M are discretisable. Let $\sigma = (e_1, \tau_1)(e_2, \tau_2) \dots (e_m, \tau_m)$ be a behavior of the automaton M. Let $F_\sigma = \{\mathit{frac}(\tau_i) \mid 1 \le i \le m\} \cup \{0, 1\}$ and $\#(F_\sigma)$ be the number of the elements of F_σ. So σ is an integral behavior iff $\#(F_\sigma) = 2$. Let $f_0, f_1, \dots, f_q, f_{q+1}$ be the sorted sequence of all the elements of F_σ in the ascending order, i.e.

$$F_\sigma = \{f_0, f_1, \dots, f_q, f_{q+1}\},\ f_0 = 0, f_{q+1} = 1, q \ge 0, f_i < f_{i+1} (0 \le i \le q).$$

Let $F_\sigma^{-1}(f_1) = \{i \mid 1 \le i \le m \wedge \mathit{frac}(\tau_i) = f_1\}$.

Lemma 6. *Let $\#(F_\sigma) > 2$ (i.e. if $q > 0$). Then the timed transition sequences σ' and σ'' are also behaviors of M, where*

$\sigma' = (e_1, \tau_1')(e_2, \tau_2') \dots (e_m, \tau_m')$,
$\sigma'' = (e_1, \tau_1'')(e_2, \tau_2'') \dots (e_m, \tau_m'')$, *and*
$\tau_i' = \begin{cases} \tau_i & i \notin F_\sigma^{-1}(f_1) \\ \tau_i - f_1 & i \in F_\sigma^{-1}(f_1) \end{cases} \quad \tau_i'' = \begin{cases} \tau_i & i \notin F_\sigma^{-1}(f_1) \\ \tau_i - f_1 + f_2 & i \in F_\sigma^{-1}(f_1)\ (f_2 \mathit{may\ be}\ 1). \end{cases}$

Proof. We first prove that for all i, j such that $0 \le i < j \le m$, for all natural numbers a, b, $a \le \tau_j - \tau_i \le b$ implies that $a \le \tau_j' - \tau_i' \le b$ and $a \le \tau_j'' - \tau_i'' \le b$. Let $\tau_j - \tau_i \ge a$. Then

- when $i, j \in F_\sigma^{-1}(f_1)$, we have $\tau_j' - \tau_i' = \tau_j'' - \tau_i'' = \tau_j - \tau_i \ge a$;
- when $i, j \notin F_\sigma^{-1}(f_1)$, we have $\tau_j' - \tau_i' = \tau_j'' - \tau_i'' = \tau_j - \tau_i \ge a$;
- when $i \in F_\sigma^{-1}(f_1) \wedge j \notin F_\sigma^{-1}(f_1)$, we have $\tau_j' - \tau_i' = \tau_j - \tau_i + f_1 > \tau_j - \tau_i \ge a$ for the case $f_1 > f_0 = 0$, and we have $\tau_j'' - \tau_i'' = \tau_j - \tau_i - (f_2 - f_1) = \lfloor \tau_j - \tau_i \rfloor + \mathit{frac}(\tau_j - \tau_i) - (f_2 - f_1) \ge \lfloor \tau_j - \tau_i \rfloor \ge a$ for the case $\mathit{frac}(\tau_j - \tau_i) \ge f_2 - f_1$;
- when $i \notin F_\sigma^{-1}(f_1) \wedge j \in F_\sigma^{-1}(f_1)$, we have $\tau_j'' - \tau_i'' = \tau_j - \tau_i + (f_2 - f_1) > \tau_j - \tau_i \ge a$ for the case $f_2 > f_1$, and we have $\tau_j' - \tau_i' = \tau_j - f_1 - \tau_i = \lfloor \tau_j - \tau_i \rfloor + \mathit{frac}(\tau_j - \tau_i) - f_1 \ge \lfloor \tau_j - \tau_i \rfloor \ge a$ for the case $\mathit{frac}(\tau_j - \tau_i) \ge f_1$.

Therefore, in all cases, $\tau_j' - \tau_i' \ge a$ and $\tau_j'' - \tau_i'' \ge a$ hold.

The case $\tau_j - \tau_i \le b$ is proved similarly. □

Let $\#(F_\sigma) > 2$ and σ' and σ'' be as in Lemma 6.

Lemma 7. *Let $\sigma(j, k)$ be a subsequence of σ that matches $\gamma(D)$. Then the subsequences $\sigma'(j, k)$ of σ' and $\sigma''(j, k)$ of σ'' match $\gamma(D)$, too. Furthermore, either $\theta(\sigma'(j, k)) \ge \theta(\sigma(j, k))$ or $\theta(\sigma''(j, k)) \ge \theta(\sigma(j, k))$ holds.*

Proof. Since σ' and σ'' have the same transition sequence as σ has, and since the definition of 'matching' depends only on the transition sequence of the behavior, the first statement of the lemma is obvious. The second statement of the lemma is proved as follows. By the definition of the function θ:

$$\theta(\sigma(j,k)) = \textstyle\sum_{i=j+1}^{j+k} c_{s_i}(\tau_i - \tau_{i-1}),$$
$$\theta(\sigma'(j,k)) = \textstyle\sum_{i=j+1}^{j+k} c_{s_i}(\tau_i' - \tau_{i-1}'),$$
$$\theta(\sigma''(j,k)) = \textstyle\sum_{i=j+1}^{j+k} c_{s_i}(\tau_i'' - \tau_{i-1}'').$$

According to the construction of σ' and σ'',

$$\theta(\sigma'(j,k)) = \theta(\sigma(j,k)) + f_1\delta,$$
$$\theta(\sigma''(j,k)) = \theta(\sigma(j,k)) + (f_1 - f_2)\delta,$$

where $\delta = (\sum_{i\in F_\sigma^{-1}(f_1), j+1\le i\le j+k} c_{s_i} - \sum_{i+1\in F_\sigma^{-1}(f_1), j\le i\le j+k-1} c_{s_i})$.

Since $f_1 > 0$ and $f_1 - f_2 < 0$, we have either $\theta(\sigma'(j,k)) \geq \theta(\sigma(j,k))$ or $\theta(\sigma''(j,k)) \geq \theta(\sigma(j,k))$. $\square$

Theorem 8. *Temporal duration property $\Box D$ for timed automaton M is discretisable.*

Proof. For any behavior $\sigma = (e_1,\tau_1)(e_2,\tau_2)\ldots(e_m,\tau_m)$ of M, for any subsequence $\sigma(j,k)$ matching $\gamma(D)$, if $\#(F_\sigma) > 2$, we can find two behaviors σ' and σ'' as in Lemma 6. By Lemma 7, among σ' or σ'' there is $\sigma^{(1)}$ such that $\theta(\sigma^{(1)}(j,k)) \geq \theta(\sigma(j,k))$. Because $\#(F_{\sigma'}) = \#(F_{\sigma''}) = \#(F_\sigma) - 1$, we have $\#(F_{\sigma^{(1)}}) < \#(F_\sigma)$. By repeating this process, we can find a behavior $\sigma^{(i)}$ such that $\sigma^{(i)}(j,k)$ matches $\gamma(D)$, $\theta(\sigma^{(i)}(j,k)) \geq \theta(\sigma(j,k))$ and $\#(F_{\sigma^{(i)}}) = 2$. Therefore $\sigma^{(i)}$ is an integer behavior with a subsequence $\sigma^{(i)}(j,k)$ matching $\gamma(D)$ for which $\theta(\sigma^{(i)}(j,k)) \geq \theta(\sigma(j,k))$ (i.e. $\theta(\sigma^{(i)}(j,k)) \leq Q$ implies that $\theta(\sigma(j,k)) \leq Q$). Hence, if $\Box D$ is satisfied by all integral behaviors of M, then it is satisfied by all behaviors of M. $\square$

3.2 Algorithm

Now, we develop an algorithm to check if all integral behaviors of M satisfy $\Box D$.

By the exhausted investigation in the finite set of transitions E of M, we can find all the transition sequences $\omega = e_{i_1}e_{i_2}\ldots e_{i_k}$ for which $\overleftarrow{e_{i_{j+1}}} = \overrightarrow{e_{i_j}}$ for $1 \leq j < k$ (consecutive sequence), and $\overleftarrow{e_{i_j}} = s_{i_j}$ for $1 \leq j \leq k$ (i.e. ω matches $\gamma(D)$). Let Ω be the set of all such transition sequences ω's. For a reachable integral state $\langle s_{i_1}, \nu\rangle$ (a reachable state in which $\nu(x)$ is an integer for any clock variable x), there exists an integral behavior $\sigma_1 = (e_1,\tau_1)\ldots(e_m,\tau_m)$ such that $\overrightarrow{e_m} = s_{i_1}$ and $\nu_m = \nu$. The integral behavior σ_1 can be expand to an integral behavior $\sigma_1\sigma_2$ (see Figure 1), where $\sigma_2 = (e_{i_1},\tau_{m+1})(e_{i_2},\tau_{m+2})\ldots(e_{i_k},\tau_{m+k})$, τ_{m+j}'s are integers, such that σ_2 matches $\gamma(D)$ if and only if $\omega = e_{i_1}e_{i_2}\ldots e_{i_k} \in \Omega$ and the sequence of clock interpretations defined as:

$$\nu_m = \nu$$
$$\nu_{m+j} = [\lambda \mapsto 0](\nu_{m+j-1} + t_j), \text{ where } t_j = \tau_{m+j} - \tau_{m+j-1}, 1 \leq j \leq k,$$

verifies that $\nu_{m+j-1} + \tau_{m+j} - \tau_{m+j-1}$ $(= \nu_{m+j-1} + t_j)$ satisfies $\phi_{e_{i_j}}$ for all $1 \leq j \leq k$. The fact that $\nu_{m+j-1} + t_j$ satisfies $\phi_{e_{i_j}}$ corresponds exactly to a linear constraint C_j on t_l's from the definition of $\nu m + j - 1$ and $\phi_{e_{i_j}}$. Note that $\theta(\sigma_2) = \sum_{j=1}^{k} c_{s_{i_j}} t_{m+j}$.

Therefore, all the parts σ_2 of a behavior that have ω an their untimed sequence and that starts from the integral reachable state $\langle s_{i_1}, \nu\rangle$ satisfy the inequality $\sum_{j=1}^{k} c_{s_{i_j}} t_{m+j} \leq Q$ if and only if the optimal value for the following linear integer problem (with k integer variables) is not greater than Q:

$$\begin{array}{l}\sup \sum_{j=1}^{k} c_{s_{i_j}} t_j \\ \text{subject to the constraints} \\ C_1, C_2, \ldots, C_k, t_1 \geq 0, \ldots, t_k \geq 0\end{array}$$

(by our convention, the optimal value is $-\infty$ when the constraint set is unfeasible). This problem depends only on the integral clock interpretation ν and the sequence ω. It is well-known that the complexity of the integer linear programming problems is NP. Fortunately, we can take t_{m+j}'s ($1 \leq j \leq k$) to be real variables to convert it to a linear programing $\mathcal{P}(\nu, \omega)$. By Theorem 8, the results of the two problems are the same.

So, checking $M \models \Box D$ can be done by solving the linear programing problem $\mathcal{P}(\nu, \omega)$ of k variables and verifying if the result is not greater then Q for each integral reachable states $\langle s_{i_1}, \nu\rangle$ of time automaton M. Since the set of all integral reachable states $\langle s_{i_1}, \nu\rangle$ may be infinite, the number of problems to be solve may be infinite, too. Fortunately, we only have to solve a finite number of them to come to conclusion. This is because of the help of the region automata.

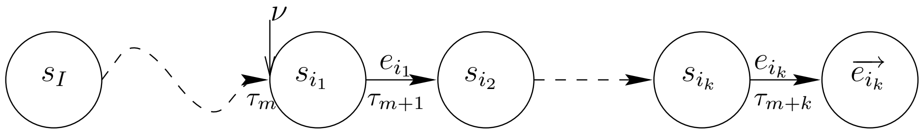

Fig. 1. Cases for the successive sequence from $\langle s_{i_1}, \nu\rangle$ that matches $\gamma(D)$

According to our definition, an integral clock interpretation ν is a mapping from the set of clock variables X to the set of natural numbers N. The restriction of the equivalence relation $\cong$ to the set of integral clock interpretations (also denoted by $\cong$) is much simpler than its original: for any integral clock interpretations ν and ν', $\nu \cong \nu'$ if and only if for each $x \in X$, either $\nu(x) = \nu'(x)$ or $\nu(x) > K_x \wedge \nu'(x) > K_x$. Let Π be the set of all integral clock regions induced by this restriction of $\cong$. Note that $|\Pi| = \prod_{x \in X}(K_x + 2)$.

An integral clock region $\pi \in \Pi$ can be characterized by a set of simple clock constraints $C(\pi)$ of the form $x = c$ or $x > K_x$. For each clock $x \in X$, there is one and only one constraint in $C(\pi)$ from the set $\{x = c \mid c = 0, 1, \ldots, K_x\} \cup \{x > K_x\}$. So, $C(\pi)$ if of the form $\bigcup_{x \in X}\{x = c_{\pi,x} \text{ or } x > K_x\}$ where $c_{\pi,x} \in N \wedge 0 \leq c_{\pi,x} \leq K_x$. If all clock constraints in $C(\pi)$ are of the form $x = c_{\pi,x}$, π is said to be bounded. Otherwise π is said to be unbounded. When $C(\pi) = \bigcup_{x \in X}\{x > K_x\}$, π is said to be entirely unbounded and is denoted by π_K. It is obvious that there is only one integral clock interpretation in any bounded region while there are infinite integral clock interpretations in an unbounded region. It is easy to see that $\nu + t \cong \nu' + t$ if $\nu, \nu' \in \pi$. So, we can define $\pi + t$ as $[\nu + t]$ with any $\nu \in \pi$.

And for every $x \in X$, if $x = c \in C(\pi)$ then if $c+t \leq K_x$ then $x = c+t \in C(\pi+t)$, otherwise $x > K_x \in C(\pi + t)$. Note that $\pi_K = \pi_K + t$ for any $t \in N$. Similarly, $[\lambda \mapsto 0]\nu \cong [\lambda \mapsto 0]\nu'$ if $\nu, \nu' \in \pi$. So, we define $[\lambda \mapsto 0]\pi$ as $[[\lambda \mapsto 0]\nu]$ with any $\nu \in \pi$. For every $x \in X$, if $x \in \lambda$ then $x = 0 \in C([\lambda \mapsto 0]\pi)$, and if $x \notin \lambda$ then when $x = c \in C(\pi)$ we have $x = c \in C([\lambda \mapsto 0]\pi)$, and when $x > K_x \in C(\pi)$ we have $x > K_x \in C([\lambda \mapsto 0]\pi)$.

An integral configuration is a pair of $\langle s, \pi \rangle$ where $s \in L$ and π is an integral clock region. So, the number of integral configurations is bounded by $|L| \cdot |\mathit{\Pi}| = |L| \cdot \prod_{x \in X}(K_x + 2)$. For an integral state $\langle s, \nu \rangle$ we write $\langle s, \nu \rangle \in \langle s, \pi \rangle$ iff $\nu \in \pi$. The following lemma plays a key role in reducing the number of linear problems to be solved to finite.

Lemma 9. *Let ν and ν' be clock interpretations such that the states $\langle s_{i_1}, \nu \rangle$ and $\langle s_{i_1}, \nu' \rangle$ are reachable and $\nu \cong \nu'$ (i.e. $[\nu] = [\nu'] = \pi$). Then for any $\omega \in \mathit{\Omega}$ the linear programing problems $\mathcal{P}(\nu, \omega)$ and $\mathcal{P}(\nu', \omega)$ give the same result.*

Proof. Let $\omega = e_{i_1} e_{i_2} \dots e_{i_k} \in \mathit{\Omega}$. Let σ_ν and $\sigma_{\nu'}$ be the behaviors of M that lead M to the states $\langle s_{i_1}, \nu \rangle$ and $\langle s_{i_1}, \nu' \rangle$, respectively. Let (e_ν, τ_ν) and $(e_{\nu'}, \tau_{\nu'})$ be the last elements of σ_ν and $\sigma_{\nu'}$, respectively. We will prove that for any non-negative real numbers $t_1, t_2, \dots, t_k$, the sequence $\sigma_\nu (e_{i_1}, \tau_\nu + t_1)(e_{i_2}, \tau_\nu + t_1 + t_2) \dots (e_{i_k}, \tau_\nu + t_1 + \dots + t_k)$ is a behavior of M if and only if the sequence $\sigma_{\nu'} (e_{i_1}, \tau_{\nu'} + t_1)(e_{i_2}, \tau_{\nu'} + t_1 + t_2) \dots (e_{i_k}, \tau_{\nu'} + t_1 + \dots + t_k)$ is a behavior of M. This means that the set of the constraints of the problem $\mathcal{P}(\nu, \omega)$ has the same solutions as the set of the constraints of the problem $\mathcal{P}(\nu', \omega)$. Since the two problems have the same objective function, they have the same optimal value.

Since the two implications of the statement to be proved are symmetric, we have to prove only one of them. Namely, we assume that $\sigma_\nu (e_{i_1}, \tau_\nu + t_1)(e_{i_2}, \tau_\nu + t_1 + t_2) \dots (e_{i_k}, \tau_\nu + t_1 + \dots + t_k)$ is a behavior of M. We have to prove that $\sigma_{\nu'} (e_{i_1}, \tau_{\nu'} + t_1)(e_{i_2}, \tau_{\nu'} + t_1 + t_2) \dots (e_{i_k}, \tau_{\nu'} + t_1 + \dots + t_k)$ is also a behavior of M. Let $\nu_0 = \nu$, $\nu_l = [\lambda_{e_{i_l}} \mapsto 0](\nu_{l-1} + t_l)$, $1 \leq l \leq k$. By our assumption, $(\nu_{l-1} + t_l)$ satisfies $\phi_{e_{i_l}}$ for all $1 \leq l \leq k$. Now let $\nu'_0 = \nu'$, $\nu'_l = [\lambda_{e_{i_l}} \mapsto 0](\nu'_{l-1} + t_l)$, $1 \leq l \leq k$. All we have to prove is that $(\nu'_{l-1} + t_l)$ satisfies $\phi_{e_{i_l}}$ for all $1 \leq l \leq k$.

For every clock $x \in X$

- if $\nu(x) \leq K_x$, then $\nu'(x) = \nu(x)$ since $\nu \cong \nu'$. Therefore $\nu_l(x) = \nu'_l(x)$ for all $0 \leq l \leq k$. Consequently, $(\nu'_{l-1} + t_l)$ satisfies the constraint for the clock x in $\phi_{e_{i_l}}$ for all $0 \leq l \leq k$ as well.
- if $\nu(x) > K_x$, then $\nu'(x) > K_x$ since $\nu \cong \nu'$. From the definition of ν_l and ν'_l, for all $1 \leq l \leq k$ either $\nu'_l(x) > K_x$ and $\nu_l(x) > K_x$, or $\nu'_l(x) = \nu_l(x)$. Since K_x is the maximal value that x is compared with in M, and since $(\nu_{l-1} + t_l)$ satisfies $\phi_{e_{i_l}}$, $(\nu'_{l-1} + t_l)$ satisfies the constraint for the clock x in $\phi_{e_{i_l}}$ as well. □

Lemma 9 means that in order to check $M \models D$, we have to solve at most one linear programing problem for each integral integral reachable configuration $\langle s_{i_1}, [\nu] \rangle$ for each $\omega \in \mathit{\Omega}$. An algorithm to produce the set of linear programming problems for a reachable integral configuration $\langle s_{i_1}, \pi \rangle$ is shown in Figure 2.

For simplicity, in the algorithm, because π is characterized by a set of clock constraints $C(\pi)$ as mentioned earlier, we identify π with a mapping from X to the set of expressions (strings) as: for each clock variable x if $x = c \in C(\Pi)$ then $\pi(x) = c$, otherwise $(x > K_x \in C(\Pi))$, $\pi(x) = -1$. As usual, we denote $\phi[x \backslash Region(x)]$ the formula obtained from ϕ by replacing all occurrences of x by the expression $Region(x)$.

```
$ProblemSet := \emptyset$;  $TransitionSequenceSet := \Omega$;
while $TransitionSequenceSet \neq \emptyset$ do
begin
  $TransitionSequence :=$ a sequence $e_{i_1} \dots e_{i_k}$ in $TransitionSequenceSet$;
  $TransitionSequenceSet := TransitionSequenceSet - TransitionSequence$;
  $Region := \pi$;  $ConstraintSet := \{t_1 \leq 0, \dots, t_k \leq 0\}$;  $Infeasible := False$;
  for $j := 1$ to $k$ do
  begin
    for every clock $x \in X$ do
    begin
      if $Region(x) \neq -1$
      then begin
             $Region(x) := Region(x) + t_j$;
             For every constraint $\phi$ on $x$ in $\phi_{e_{i_j}}$
             do begin
                  $Constraint := \phi[x \backslash Region(x)]$;
                  $ConstraintSet := ConstraintSet + Constraint$;
                end;
           end;
      else if there exists a constraint $\phi$ on $x$ in $\phi_{e_{i_j}}$ of the form $x \leq d$
               then begin
                      $Infeasible := True$;
                      break;
                    end;
      if $x \in \lambda_{e_{i_j}}$ then $Region(x) = 0$;
    end;
    if $\neg Infeasible$
    then begin
           $NewProblem := \max(\sum_{j=1}^{k} c_{s_{i_j}} t_j)$  subject to  $ConstraintSet$;
           $ProblemSet := ProblemSet + NewProblem$;
         end;
  end;
end;
```

Fig. 2. Algorithm for producing the set of linear programming problems for $\langle s_{i_1}, \pi \rangle$

Now we develop a technique to find all reachable integral configurations.

Definition 10. For a timed automaton $M = \langle L, s_I, \Sigma, X, E \rangle$, the integral region automaton $\mathcal{I}(M)$ is a transition system $\langle Q, q_I, \Sigma, \Psi \rangle$, where

- the set of states $Q = L \times \Pi$ (which is the set of all integral configurations of M),
- the initial state $q_I = \langle s_I, [\nu_0] \rangle \in Q$,
- the set of labels Σ is the same as that of M,
- the set of transitions $\Psi \subseteq Q \times \Sigma \times Q$ is defined as $\psi = \langle \langle s, \pi \rangle, a, \langle s', \pi' \rangle \rangle \in \Psi$ iff there exists $e = \langle s, a, \phi, \lambda, s' \rangle \in E$ such that $\pi + t$ satisfies ϕ and $\pi' = [\lambda \mapsto 0](\pi + t)$ for some natural number t.

In fact, the integral region automaton of M is similar to the region automaton for it, but has much smaller size.

To find all the transitions of $\mathcal{I}(M)$ from a given integral configurations $\langle s, \pi \rangle$ of M, we can compute $\pi + t$ for each natural number $t \leq \mu$, where μ is the minimal natural number for which $\pi + \mu = \pi_K$, and verify if there exists $e = \langle s, a, \phi, \lambda, s' \rangle \in E$ such that $\pi + t$ satisfies ϕ. The detail of the construction is given in Figure 3.

```
$SuccConfigurationSet := \emptyset$;  $t := 0$;
**repeat**
    $\pi' := \pi + t$;
    **if** there exists $e = \langle s, a, \phi, \lambda, s' \rangle \in E$ such that $\pi'$ satisfies $\phi$
    **then** $SuccConfigurationSet := SuccConfigurationSet + [\lambda \mapsto 0]\pi'$;
    $t := t + 1$;
**until** $\pi' = \pi_K$;
```

Fig. 3. Finding all the successive integral configurations of $\langle s, \pi \rangle$

Note that $\pi + K_X = \pi_K$ for any integral region π, so this algorithm must terminate within K_X times of repetitions.

Lemma 11. *An integral configuration $\langle s, \pi \rangle$ is a reachable state of the integral region automaton $\mathcal{I}(M)$ iff (s, ν) is a reachable integral state of timed automaton M for some $\nu \in \pi$.*

Proof. The lemma is proved directly from the definitions of the behaviors.

Proof of $\Rightarrow$: If $\langle s, \nu \rangle$ is a reachable integral state of timed automaton M for some $\nu \in \pi$, there exists an integral behavior $\sigma = (e_1, \tau_1)(e_2, \tau_2) \ldots (e_m, \tau_m)$ such that $\overrightarrow{e_m} = s$ and $\nu = \nu_m$, where for $1 \leq j \leq m$, $\nu_j = [\lambda_{e_j} \mapsto 0](\nu_{j-1} + \tau_j - \tau_{j-1})$, ν_0 assigns 0 to each clock variable, and $\nu_{j-1} + \tau_j - \tau_{j-1}$ satisfies ϕ_j. Let $\pi_j = [\nu_j]$ for $0 \leq j \leq m$. By the Definition 10, there exists a transition $\psi_j \in \Psi$ of $\mathcal{I}(M)$ from q_j to q_{j+1} for $1 \leq i \leq m$, where $q_1 = \langle s_I, [\nu_0] \rangle$, $q_l = \langle \overrightarrow{e_l}, \pi_l \rangle$ for $2 \leq l \leq m$. Therefore, $q_m = \langle s, \pi \rangle = \langle \overrightarrow{e_m}, \pi_m \rangle$ is reachable in $\mathcal{I}(M)$.

Proof of $\Leftarrow$:If the state $\langle s, \pi \rangle$ of the integral region automaton $\mathcal{I}(M)$ is reachable, there exists a run $\langle s_I, \pi_0 \rangle \xrightarrow{a_1} \langle s_1, \pi_1 \rangle \xrightarrow{a_2} \ldots \xrightarrow{a_m} \langle s_m, \pi_m \rangle$ of $\mathcal{I}(M)$, where $\pi_0 = [\nu_0]$, and $\langle s_m, \pi_m \rangle = \langle s, \pi \rangle$. Let $s_0 = s_I$. By Definition 10, for $1 \leq j \leq m$ there exist $e_j = \langle s_{j-1}, a_j, \phi_j, \lambda_j, s_j \rangle \in E$ and $t_j \in N$ such that $\pi_{j-1} + t_j$ satisfies

ϕ_j and $\pi_j = [\lambda_j \mapsto 0](\pi_{j-1} + t_j)$. Because $\nu + t \in \pi + t$ and $[\lambda \mapsto 0]\nu \in [\lambda \mapsto 0]\pi$ for any $\nu \in \pi$ and $t \geq 0$, we have $\nu_j = [\lambda_j \mapsto 0](\nu_{j-1} + t_j) \in \pi_j$ for $1 \leq j \leq m$. Of course $\nu_0 \in [\nu_0]$. Hence, $\nu_{j-1} + t_j$ satisfies ϕ_j. Hence, the time-stamped transition sequence $(e_1, \tau_1)(e_2, \tau_2) \ldots (e_m, \tau_m)$ where $\tau_j = \sum_{l=1}^{j} t_l$ is an integral behavior of timed automaton M and $\langle \overrightarrow{e_m}, \nu_m \rangle$ $(= \langle s, \nu_m \rangle)$ is a reachable integral state of M with $\nu_m \in \pi$. $\square$

By Lemmas 11 and 9, we can decide if $M \models \square D$ by: first, determine all the reachable states of the finite automaton $\mathcal{I}(M)$ of the form $\langle s_{i_1}, \pi \rangle$, and then for each of them generate the set of linear programming problems using the algorithm in Figure 2; last solve all these problems and compare the results with Q. Of course, when implementing these algorithms, we will use on-the-fly technique to reduce the complexity (e.g. we can stop checking once we discover that a linear programing problem results in a value that greater than Q). An algorithm to find all reachable integral configurations while generating the reachability integral region automaton $\mathcal{I}(M)$ by the depth-first method and to check the result of linear programming problems "on-the-fly" is shown in Figure 4. We can formulate the correctness of our algorithm by the following theorem which is an immediate consequence of Lemmas 11, 9, and Definition 4:

```
$CurrentPath := \{\langle s_I, \nu_0 \rangle\}$;  $ConfigurationSet := \emptyset$;
repeat
  $Configuration$ := the last configuration of $CurrentPath$;
  if $Configuration$ has no new successive configuration
  then begin
          if the location of $Configuration$ is $s_{i_1}$
          then produce linear programming problems of $Configuration$ to check;
          delete the last configuration i.e. $Configuration$, from $CurrentPath$;
        end
  else begin
          $Configuration$ := a new successive node of $Configuration$;
          if $Configuration$ is not in $ConfigurationSet$
          then begin
                  append $Configuration$ to $CurrentPath$;
                  put $Configuration$ into $ConfigurationSet$;
                end;
        end;
until $CurrentPath = \emptyset$;
```

Fig. 4. Algorithm for checking TDPs of timed automata

Theorem 12. *$M \models \square D$ iff every linear programming problem produced by the algorithm in Figure 4 either has no solution or has the optimal value not great than Q.*

3.3 Improvements and Complexity

Since the complexity of Linear Programming is in the class $\mathcal{P}$ and sine the number of variables of linear programming problems that we need to solve is fixed as k, the complexity of our algorithm is decided by the complexity of finding all reachable integral configurations of the integral region automaton of the input time automaton, which is in the same complexity class as for solving the reachability problem of timed automata. To improve our algorithm, we can use some well-established techniques to reduce the state space of the integral region automata. We don't discuss about these techniques here, and leave it to the implementation stage.

4 Conclusion

We have presented a technique for deciding whether a timed automaton satisfies a temporal duration property. Temporal duration properties form a new class of duration properties which are properties of behaviours of systems for a time interval. There is no doubt that checking real-time systems for a duration property is much more difficult than for a property of systems at a moment of time.

Some work on checking duration properties has been done. The earliest one, to our knowledge is [3], which proposed a technique to check the linear duration invariants, which require systems to satisfy some linear inequalities on integrated durations of system locations in any observation time interval whose length matches the premise. Temporal duration properties are different from linear duration properties. Temporal duration properties pay attention to the trace of system locations, while linear duration invariants pay attention to the length of the observation intervals during of real-time systems running. We have shown in this paper that verification of temporal duration properties is much simpler than that of linear duration invariants. In [8], the authors shows that the satisfaction problem of linear duration invariants for timed automata can be solved by mixed integer linear programming. In [4, 5, 11], we have restricted ourselves to study the problem of checking whether a real-time system whose behaviours could be represented by a timed regular expression, satisfies a Linear Duration Invariant. We have developed a technique to solve the problem and implemented it as a tool. In comparison to [8], our technique is simpler because it uses only linear programming techniques. However, when we tried our tool on some small sized practical examples, we discovered that the tool could give answers in few hours or ran out of the computer memory. We reconsidered our technique and found that the size of the linear programming problem we have to solve could be huge in some cases.

Some model checking tools are available now (see, e.g. [10, 9]), but most of them are for checking instant properties, and based on the algorithms for solving the reachability problems. In this paper, we have shown that the satisfaction problem of temporal duration properties for timed automata can be solved by linear programming and the complexity is in the same class as reachability problem. We think that it is acceptable, and are going to implement this model checking technique in our future work.

References

1. R. Alur and D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, pages 183–235, 1994.
2. Zhou Chaochen, C.A.R. Hoare, and Anders P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.
3. 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.
4. 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.
5. 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.
6. Zhao Jianhua and Dang Van Hung. Checking Timed Automata for Some Discretisable Duration Properties. Technical Report 145, UNU/IIST, P.O.Box 3058, Macau, August 1998. Published in "Journal of Computer Science and Technology", Volume 15, Number 5, September 2000, pp. 423–429.
7. 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. Proceedings of *Formal Techniques in Real-Time and Fault-Tolerant Systems* 5th International Symposium, Lyngby, Denmark, September 1998 (FTRTFT'98), Anders P. Ravn and Hans Rischel (Eds.), LNCS 1486, pp. 241–250, Springer-Verlag, 1998.
8. Y. Kesten, A. Pnueli, J Sifakis, and S. Yovine. Integration Graphs: A Class of Decidable Hybrid Systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 179–208. Springer Verlag, 1994.
9. K.G.Larsen and H. Huttel. UPPAAL-An Automatic Tool for Verification of Real Time and Hybrid Systems. In *Proc. of the 16th IEEE Real-Time Systems Symposium*, pages 76–87, Dec. 1997.
10. T.A.Henzinger, P.-H. Ho, and H. Wong-Toi. A Users Guide to HyTech. Technical report, Department of Computer Science, Cornell University, 1995.
11. 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.
12. Li Yong and Dang Van Hung. Checking History Properties of Real-Time Systems. Technical Report 214, UNU/IIST, P.O. Box 3058, Macau, October 2000.