CHECKING PARALLEL REAL-TIME SYSTEMS FOR TEMPORAL DURATION PROPERTIES BY LINEAR PROGRAMMING*

Pham Hong Thai

Faculty of Technology, Vietnam National University, Hanoi

Abstract. Model checking for Duration Properties of real-time systems has been a great deal of attention for recent years. In some works, duration properties as Linear Duration Invariants, Linear Duration Properties, Temporal Duration Properties have been checked for systems, which are expressed by timed automata or restricted classes of timed automata. Up to now, one of such properties, Temporal Duration Properties have not dealed directly for parallel real-time systems. In this paper, we propose an algorithm for checking TDP for such systems, i.e. for systems which are expressed by networks of timed automata. The algorithm is based on depth-first searching on the region graph of networks and reduces checking problem to solving a set of linear programming problems. So, the complexity of the algorithm is acceptable.

1. Introduction

Model checking for real-time systems, i.e. given a real-time system and a real-time property and check whether the system satisfies the property. Instant properties have been studied extensively and some verifying tools have been implemented for checking them. In the recent few years, duration properties, i.e. properties which concern to intervals of time, were considered more and more. Those are certain predicates on accumulated time during of locations of systems and often are represented formally by formulas of Duration Calculus [2]. Checking duration properties is difficult because the duration of locations are defined on the history of the systems, especially for the concurrent and distributed real-time systems.

A solution in general case is given in [3] using mixed integer and linear programming techniques that as known complexity of this problem is in class NP. To get effective algorithms, many authors have dealed about more restricted systems and/or properties, for example, Linear Duration Invariant property (LDI) was checked for real-time automata using linear programming techniques (polynomial time) in [4]. The technique have been

*A preliminary version of this paper was presented at and published in the proceedings of *The First International Workshop for Computer, Information and Communication Technologies*, Hanoi, Feb. 2003, pp. 18-24.

generalized to check LDI for extended subclass of timed automata [5] and for parallel compositions of real-time automata [6]. However, for general cases (timed automata, networks of timed automata) LDI have not checked by algorithms with accepted complexity.

Recently, discretisable formulas is more considered. Those are formulas which their satisfiable for real time behaviors of the system is the same for integer time behaviors. For such properties, algorithms based on exploring region graph will have accepted complexity, because they only search on integral region graph. In [8] authors considered to a subclass of duration formulas named temporal duration properties (TDP), and showed that they are discretisable. Combining depth-first search method with linear programming technique, authors proposed an algorithm to check TDP for timed automata.

In this paper, we show that the techniques introduced in [8] can be applied to solve the problem for timed automaton networks and propose an algorithm for checking satisfiable of TDP. Although a parallel composition of timed automata can be viewed as a restricted production timed automaton, but from discretisable of general property for timed automata, it is not obviously to deduce that the property is also discretisable for networks. This is appeared by two reasons : by synchronization of component automata and by durations of time are calculated on local locations but not on global locations of systems. For example, Linear Duration Invariant can be checked by linear programming for real-time automata [4], but it have to use mixed integer programming in the case of system is extended to networks of real-time automata [6]. So an effective algorithm for checking TDP for networks of timed automata is necessary.

The paper is organized as follows. In the next section we recall some notations of timed automata, parallel composition of them and integral region graph. Defining and proving TDP be discretisable is given in section 3. In section 4 we present algorithm using linear programming technique to check TDP for timed automaton network. And, finally in section 5, we give a short discussion about some directions to reduce the complexity of algorithm.

2. Parallel composition of timed automata

2.1. Timed automata [1]

A timed automaton is a finite state machine combined with a set of clock variables X. We use $\Phi(X)$ to denote the set of time constrains which are conjunctions of the formulas of the form : $x \leq a$ or $x \geq a$, where $x \in X$ and a is a natural constant.

Definition 1. A timed automaton is a tuple $A = \langle S, s_0, \Sigma, X, \Phi(X), E \rangle$, where

- S is a finite set of locations,
- $s_0 \in S$ is an initial location,
- Σ is a finite set of labels,

- X is a finite set of clocks,
- $\Phi(X)$ is a finite set of time constrains of clock variables,
- $E \subseteq S \times \Phi(X) \times \Sigma \times 2^X \times S$ is a finite set of transitions.

An $e = (s, \psi_e, a, r_e, s') \in E$ (so-called an a-labelled edge) represents a transition from location s to location s' with label a; s and s' are called source and target location of e, and denoted by source(e) and target(e) respectively. ψ_e is a clock constraint that is satisfied when the transition e is enabled and r_e is the set of clock variables to be reset to 0 by e when it takes place. For simplicity, in this paper we only consider deterministic timed automata, i.e. automata which have not more than one a-labelled edge for any $a \in \Sigma$. A_1 and A_2 in figure 1 are an example of two timed automata.

2.2. Parallel composition of timed automata

In general, a system is often a set of timed automata running in parallel and communicating with each other. These timed automata can be synchronously composed into a global timed automata as follows : transitions of timed automata that do not execute a share event (label) are interleaved and transitions using a share event are synchronized.

Definition 2. Given a set of timed automaton $A_i = \langle S_i, s_{0i}, \Sigma_i, X_i, \Phi_i(X_i), E_i \rangle$ (i = 1..n). It does not loss generality, assumed that $X_i \cap X_j = \emptyset, \forall i \neq j$. A system can be expressed as an parallel composition of A_i's, i.e. an global timed automaton $A = \langle S, s_0, \Sigma, X, \Phi(X), E \rangle$, where

- $S = S_1 \times S_2 \times \ldots \times S_n$,
- $s_0 = (s_{01}, s_{02}, \ldots, s_{0n})$,
- $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \ldots \cup \Sigma_n$,
- $X = X_1 \cup X_2 \cup \ldots \cup X_n$,
- $\Phi(X) = \Phi_1(X_1) \cup \Phi_2(X_2) \cup \ldots \cup \Phi_n(X_n)$,

With $n = 2$, let $(s_i, \psi_i, a_i, r_i, s_i')(i = 1, 2)$ are transitions of E_1, E_2.

- If $a_1 = a_2$ then $((s_1, s_2), \psi_1 \cup \psi_2, a, r_1 \cup r_2, (s_1', s_2')) \in E$, where $a = a_1 = a_2$,
- If $a_1 \notin \Sigma_1 \cap \Sigma_2$ then $((s_1, s_2), \psi_1, a_1, r_1, (s_1', s_2)) \in E$,
- f $a_2 \notin \Sigma_1 \cap \Sigma_2$ then $((s_1, s_2), \psi_2, a_2, r_2, (s_1, s_2')) \in E$,

Similarly, we can easily extend definition of E for $n > 2$. For example, the automaton A in figure 1 is a part of parallel composition of A_1 and A_2 which is reachable from initial location $u_0 = (h_0, k_0)$ of system.

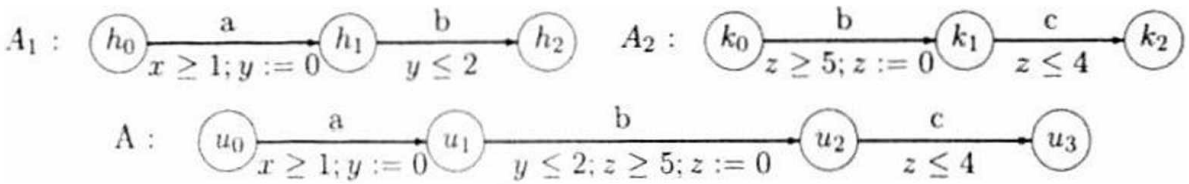

Figure 1. Paraiiel compositional automaton of A_1 and A_2
$u_0 = (h_0, k_0), u_1 = (h_1, k_0), u_2 = (h_2, k_1), u_3 = (h_2, k_2)$

2.3. Behaviors of timed automaton networks

Let A be parallel composition of timed automata A_i. Let $\overline{s}$ be a location of A, i.e. $\overline{s}$ be a vector $(s_1, s_2, \ldots, s_n)$, where s_i $(i = 1..n)$ is the location of timed automaton A_i that is called local location, $\overline{s}$ is called global location or location for short. Formally, if s is a local location which occurs in $\overline{s}$ then we denote $s \in \overline{s}$. If e is a transition of network, which transits from location $\overline{s}$ to $\overline{s'}$, then we also use denotations source(e), target(e) to denote $\overline{s}$ and $\overline{s'}$, respectively. A clock valuation v is function $v : x \mapsto R$. That is, v assigns to each clock $x \in X$ the value $v(x) \in R$. We denote v_0 as an initial clock valuation, i.e. $v_0(x) = 0, \forall x \in X$. For $d \in$ R, $v + d$ (time elapse by d) is the valuation v' such that $\forall x \in X.\ v'(x) = v(x) + d$. For $r \subseteq X$, $v[r \mapsto 0]$ (reset r to zero) is the valuation v' such that $\forall x \in r.\ v'(x) = 0$ and $\forall x \notin r.\ v'(x) = v(x)$. A clock valuation v is called integral clock valuation iff $v(x) \in N, \forall x \in X$.

A state of A is a pair $(\overline{s}, v)$, where $\overline{s}$ is a location of A and v is a clock valuation. A state $(\overline{s}, v)$ expresses system is staying at the location $\overline{s}$ and all clock values agrees with v at that time point. As number of valuations v is infinite, the number of states of system is infinite, too.

Given two states $(\overline{s}, v)$, $(\overline{s'}, v')$ of A, and a non-negative real d. Then, we define state transition from $(\overline{s}, v)$ to $(\overline{s'}, v')$ denoted by $(\overline{s}, v) \xrightarrow{d,e} (\overline{s'}, v')$ as follows.

Definition 3. $(\overline{s}, v) \xrightarrow{d,e} (\overline{s'}, v')$ iff : $\exists e = (\overline{s}, \psi_e, a, r_e, \overline{s'}) \in E$, such that

- $v + d$ satisfies ψ_e,
- $v' = (v + d)[r_e \mapsto 0]$.

The state transition means system staying at location $\overline{s}$ in time interval d from time point t (with clock valuation v) to $t' = t + d$ (with clock valuation $v + d$). At time point t', some components of system take transition e with event a, when $v + d$ satisfies any constraints on a-labelled edge of these components. The system transits to $\overline{s'}$. Then $v + d$ is changed to v' by some clock variables are reset to 0.

In order to represent behavior of A, we use sequence of time-stamped transitions. A time-stamped transition is a pair (e, t), where e is a transition of A and t is a non-negative real number that expresses time point transition e be take place.

Definition 4. A time-stamped transition sequence $\sigma = (e_0, t_0)(e_1, t_1)(e_2, t_2)...(e_m, t_m)$ $(m \geq 1)$ is a behavior of A iff:

- target(e_i) = source(e_{i+1}) , $\forall i = 1..m-1$ (with the convention target(e_0) = source(e_1) = s_0).
- $0 = t_0 \leq t_1 \leq t_2 \leq \ldots \leq t_m$, such that $(v_{i-1} + t_i - t_{i-1})$ satisfies all constraints in ψ_{ei}, where $v_i = (v_{i-1} + t_i - t_{i-1})[r_{ei} \mapsto 0], \forall i = 1..m$.

If $(e_0, t_0)(e_1, t_1)(e_2, t_2)\dots(e_m, t_m)$ is a behavior of A, we call $\bar{s}_m = \text{target}(e_m)$ a reachable location and $(\bar{s}_m, v_m)$ a reachable state of A.

A behavior $\sigma = (e_0, t_0)(e_1, t_1)(e_2, t_2)\dots(e_m, t_m)$ is called integral behavior iff $t_i \in N$. $\forall i = 1..m$.

Example 1: Let us consider behavior : $\sigma_1 = (e_0, 0), (a, 4.5), (b, 5.7)$ of network given in figure 1. It expresses the system staying at initial location (h_0, k_0) up to time point 4.5 with clock valuation $v_0 + 4.5 = (4.5, 4.5, 4.5)$. The clock valuation satisfies time constraint $x \geq 1$, so the system executes the event a and transits to location (h_1, k_0). After transition of system, the clock y is reset to 0 and clock valuation becomes $v_1 = (4.5, 0, 4.5)$. At next time point 5.7, clock valuation $v_1 + 1.2 = (5.7, 1.2, 5.7)$ satisfying time constraint $y \leq 2 \wedge z \leq 5$, the system executes the event b and transits to location (h_2, k_1) and $v_1 + 1.2$ becomes $v_2 = (5.7, 1.2, 0)$. Similarly, $\sigma_2 = (e_0, 0), (a, 5), (b, 6)$ is an integral behavior of the network (For short, in the example we identified the name of transitions to their labels).

2.4. Region graph

Checking a property for timed automata, that is in principle solving problem based on the corresponding valuation graph, which is in general infinite. However, instead of using the valuation graph, it is sufficient to use the region graph, which is presented in this section.

Now, we summarily present about technique partitioning the space of clock valuations which have been proposed by Alur and Dill [1] and well-known. Main idea of the technique is grouping clock valuations into regions such that all valuations in a region will satisfy the same set of clock constraints. Hence, states of system are also grouped into equivalence classes which is named a region. These regions will be nodes of region graph and the number of nodes is finite. In the case values of clock is natural numbers, the number of region is much smaller than the case of real time.

In this paper, we consider only integral clock valuation, so we represent only integral region graph. Besides, some known properties will be not proved. For detail, the reader can refer in [1].

Definition 5. Let v_1, v_2 be two integer clock valuations, let K_x be the largest integer appearing in a clock constraints $a \leq x$ or $x \leq a$ of clock x. Relation $\cong$ is defined as follows : $v_1 \cong v_2$ iff $\forall x \in X$: either $v_1(x) = v_2(x)$ or $\min(v_1(x), v_2(x)) > K_x$. It is easily to prove that $\cong$ is an equivalence relation. An equivalence class of v is denoted by [v] and called a integral clock region. Let Π be the set of all integral clock regions, we have $|\Pi| \leq \prod_{x \in X}(K_x + 2)$.

We have some following properties.

Property 1. $v \cong v'$ implies v satisfies clock constraint ψ iff v' satisfies, too.

Property 2. Every clock region $\pi \in \Pi$ can be characterized by a set of simple clock constraint $C(\pi)$ of the form $x = c$ or $x > K_x$. That is $C(\pi) = \bigcup_{x \in X}\{x = c_{\pi,x}, x > K_x\}$.

Property 3. Let v, v' be integral valuations. If $v \cong v'$ then

- $v + d \cong v' + d, \forall d \in \mathbb{N}$. So, we can define clock region $\pi + d$ as $[v + d]$ with any $v \in \pi$. Besides, for every $x \in X$, if $x = c \in C(\pi)$ then if $c + d \leq K_x$ then $x = c + d \in C(\pi + d)$, otherwise $x > K_x \in C(\pi + d)$. Note that $\pi_K = \pi_K + d$ for any $d \in \mathbb{N}$.
- $v[r \mapsto 0] \cong v'[r \mapsto 0]$. So we can define $\pi[r \mapsto 0]$ as $[v[r \mapsto 0]]$ with any $v \in \pi$. For every $x \in X$, if $x \in r$ then $x = 0 \in C(\pi[r \mapsto 0])$, and if $x \notin r$ then when $x = c \in C(\pi)$ we have $x = c \in C(\pi[r \mapsto 0])$, and when $x > K_x \in C(\pi)$ we have $x > K_x \in C(\pi[r \mapsto 0])$.

We extend the region equivalence $\cong$ to the states of network A as follows.

Definition 6. Two states $u_1 = (\overline{s}, v_1)$ and $u_2 = (\overline{s}, v_2)$ are region-equivalence iff $v_1 \cong v_2$. Hence, the set of states of networks is partitioned by classes of states, each class is characterized by a couple of a location $\overline{s}$ and a clock region π. We call $< \overline{s}, \pi >$ be a configuration of network. These configurations will be nodes of region graph that is defined below.

Example 2 : With timed automaton A (which expresses parallel composition of A_1 and A_2) in figure 1, we have the set of clock $X = \{x, y, z\}$ and K_x, K_y, K_z is corresponding to 1, 2, 5. Two following clock valuations are equivalence : $v_1 = (2, 1, 4), v_2 = (3, 1, 4)$. They are in a region π with characterized set of constrains $C(\pi) = \{x > K_x, y = 1, z = 4\}$. There are infinite elements in π. Each element (a clock valuation) in π is a tuple (x, 1, 4), where $x > K_x = 1$. Hence, states (u_0, v_1) and (u_0, v_2) are equivalence and characterized by configuration $< u_0, \pi >$.

Definition 7. (Region Graph). Given network of timed automata $A = \langle S, s_0, \Sigma, X, \Phi(X), E \rangle$, the integral region graph IRG(A) is the transition system $(Q, q_0, \Sigma, \rightarrow)$, where

- the set of states $Q = S \times \Pi$,
- the initial state $q_0 = (s_0, [v_0])$,
- the set of labels Σ,
- the set of transitions $\rightarrow \in Q \times \Sigma \times Q$ is defined as $\alpha = ((\overline{s}, \pi), a, (\overline{s'}, \pi') \in \rightarrow$ iff there exists $e = (\overline{s}, \psi_e, a, r_e, \overline{s'})$ such that $\pi + d$ satisfies ψ_e and $\pi' = (\pi + d)[r_e \mapsto 0]$ for some natural number d

3. Duration temporal properties

3.1. Definition

A temporal duration property is a constraint for location durations for a short trace of a certain pattern. It is defined formally in Duration Calculus [2] as follows.

Definition 8. A temporal duration property over A is a Duration Calculus formula of the form

$$\Box([[\overline{s}_{i_1}]] \frown [[\overline{s}_{i_2}]] \frown \ldots \frown [[\overline{s}_{i_k}]] \Rightarrow \sum_{s\in\Omega} c_s \int s \leq M)$$

where $\overline{s}_{i_j}$'s are locations and Ω is a finite set of local locations of systems (i.e. $\Omega = S_1 \cup S_2 \cup \ldots \cup S_n$), and $c_s (s \in \Omega)$, M are reals. For simplicity, let us denote

$$D \hat{=} [[\overline{s}_{i_1}]] \frown [[\overline{s}_{i_2}]] \frown \ldots \frown [[\overline{s}_{i_k}]] \Rightarrow \sum_{s\in\Omega} c_s \int s \leq M,$$

$$\gamma(D) \hat{=} [[\overline{s}_{i_1}]] \frown [[\overline{s}_{i_2}]] \ldots \frown [[\overline{s}_{i_k}]].$$

Hence, temporal duration property over Ω is denoted as $\Box D$. The above form of formula is expressed in syntax of Duration Calculus. In the semantics, intuitively, a temporal duration property $\Box D$ says that for any time interval, in which if the system runs through the sequence of global locations $\overline{s}_{i_1}, \overline{s}_{i_2}, \ldots, \overline{s}_{i_k}$, then duration $\int s$ of the local locations s over that interval satisfies the constraint $\sum_{s\in\Omega} c_s \int s \leq M$ ($\int s$, when applied to an interval of time, is the accumulated time that the location 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].

For any timed transition sequence $\sigma = (e_1, t_1)(e_2, t_2)\ldots(e_m, t_m)$, for $u \geq 1, l \geq 0$ such that $u + l \leq m$, let us denote by $\sigma(u, l)$ the subsequence $(e_{u+1}, t_{u+1})\ldots(e_{u+l}, t_{u+l})$. That means $\sigma(u, l)$ is a subsequence of σ from index $u + 1$ with l timed-stamp transitions.

Definition 9. For a timed transition sequence $\sigma = (e_1, t_1)(e_2, t_2)\ldots(e_m, t_m)$, for any $u \geq 0$ such that $u + k \leq m$, we say $\sigma(u, k)$ matches $\gamma(D)$ (or $\gamma(D)$ matches σ) iff source$(e_{u+j}) = \overline{s}_{i_j}$ for any j such that $1 \leq j \leq k$.

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

For a subsequence $\sigma(u, k)$ that matches $\gamma(D)$, the duration of the local location s over $\sigma(u, k)$ is defined

$$\int s = \sum_{s\in\overline{s}_{i_j}, j\leq k} (t_{u+j} - t_{u+j-1})$$

hence, the value $\sum_{s\in\Omega} c_s \int s$ of over $\sigma(u,k)$ is defined by

$$\theta(\sigma(u,k)) = \sum_{j=1}^{k} \sum_{s\in\overline{s}_{i_j}} c_s(t_{u+j} - t_{u+j-1})$$

Definition 10.

1. A behavior σ satisfies the temporal duration property $\Box D$, denoted by $\sigma \models \Box D$, iff for any subsequence $\sigma(u,k)$ for σ that matches $\gamma(D)$, the condition $\theta(\sigma(u,k)) \leq M$ holds.
2. A time automaton network A satisfies the temporal duration properties $\Box D$, denotes by $A \models \Box D$, iff for any behavior σ of A, $\sigma \models \Box D$ holds.

3.2. Discretising TDP

Definition 11. Let A be a timed automaton network, and let P be a predicate over the behaviors of A. P is said to be discretisable (w.r.t. A) if P is satisfied by all the behaviors of A iff P is satisfied by all the integral behaviors of A.

Therefore, if P is discretisable (w.r.t. A), verifying that P is satisfied by all the behaviors of A is reduced to verifying that P is satisfied by all the integral behaviors of A only.

Theorem 1. *TDP is discretisable with respect to timed automaton networks.*

Proof. For any $t \in R^+$, let $int(t)$ and $frac(t)$ respectively be integral part and fractional part of t, i.e. $t = int(t) + frac(t)$ and $frac(t) = 0$ iff t is an integer number. Let $\sigma = (e_1,t_1)(e_2,t_2)\ldots(e_m,t_m)$ be a real behavior.

Let $F_\sigma = \{frac(t_i) \mid 1 \leq i \leq m\} \cup \{0,1\}$ and $\text{card}(F_\sigma)$ be the number of the elements of F_σ. So, σ is an integer behavior iff $\text{card}(F_\sigma) = 2$ ($F_\sigma = \{0,1\}$). Let $f_0, f_1, \ldots, f_q, f_{q+1}$ be the sorted sequence in the ascending order of all elements of F_σ, i.e. : $F_\sigma = \{f_0, f_1, \ldots, f_q, f_{q+1}\}$, where $f_0 = 0, f_{q+1} = 1, f_i \leq f_{i+1} (0 \leq i \leq q)$. Because σ is the real behavior so $\text{card}(F_\sigma) > 2$ (i.e. $q > 0$). Let $I_\sigma = \{i | frac(t_i) = f_1\ (i \in 1..m)\}$.

We construct behaviors $\sigma' = (e_1,t'_1)(e_2,t'_2)...(e_m,t'_m)$ and $\sigma'' = (e_1,t''_1)(e_2,t''_2)...(e_m,t''_m$ as follows.

- $t'_i = t''_i = t_i$ if $i \notin I_\sigma$
- $t'_i = t_i - f_1$ and $t''_i = t_i - f_1 + f_2$ (f_2 may be 1) if $i \in I_\sigma$

Lemma 1. *σ' and σ'' are behaviors of A.*

Proof. At first, we prove that if $t_j - t_i \propto a$ then so are $t'_j - t'_i \propto a$ and $t''_j - t''_i \propto a$, where $j > i, a \in N$, and $\propto \in \{\leq, \geq\}$. We prove only for the case $\propto$ being $\leq$. For the case $\geq$, the proof is similar.

- When $i,j \in I_\sigma$ or $i,j \notin I_\sigma$, we have $t'_j - t'_i = t''_j - t''_i = t_j - t_i \leq a$.

- When $i \in I_\sigma$ and $j \notin I_\sigma$, we have $frac(t_j) > frac(t_i)$, hence $t'_j - t'_i = t_j - (t_i - f_1) = (t_j - t_i) + f_1 = int(t_j - t_i) + frac(t_j) - frac(t_i) + f_1 = int(t_j - t_i) + frac(t_j) \leq a - 1 + frac(t_j) \leq a$. We have $t''_j - t''_i = t_j - (t_i - f_1 + f_2) = (t_j - t_i) - (f_2 - f_1) \leq t_j - t_i \leq a$.
- When $i \notin I_\sigma$ and $j \in I_\sigma$, we have $t'_j - t'_i = (t_j - f_1) - t_i = t_j - t_i - f_1 \leq t_j - t_i \leq a$. Consider that $frac(t_j) = f_1 < frac(t_i), frac(t''_j) = f_2 \leq frac(t_i)$ and $int(t''_j) = int(t_j)$. So, if $t_j - t_i \leq a$ then $t''_j - t''_i = t''_j - t_i \leq a$, too.

Value of any clock x at time point t_j is $t_j - t_i$ where t_i is last time point that clock variable x to be reset. Therefore, if x satisfies time constraints $a \leq x$ and/or $x \geq b$, at time points t'_j and t''_j, x satisfies these constraints, too. Hence, σ' and σ'' are also behaviors of A.

Lemma 2. *Let $\sigma(u,k)$ be a subsequence of σ that matches $\gamma(D)$. If $\theta(\sigma(u,k)) \geq M$ then either $\theta(\sigma'(u,k)) \geq M$ or $\theta(\sigma''(u,k)) \geq M$.*

Proof. It is easily to see that subsequences $\sigma'(u,k)$ of σ' and $\sigma''(u,k)$ of σ'' match $\gamma(D)$, too. By the definition of the function θ, we have

$$\theta(\sigma(u,k)) = \sum_{j=1}^{k} \sum_{s \in \overline{s}_{i_j}} c_s (t_{u+j} - t_{u+j-1})$$

$$\theta(\sigma'(u,k)) = \sum_{j=1}^{k} \sum_{s \in \overline{s}_{i_j}} c_s (t'_{u+j} - t'_{u+j-1})$$

$$\theta(\sigma''(u,k)) = \sum_{j=1}^{k} \sum_{s \in \overline{s}_{i_j}} c_s (t''_{u+j} - t''_{u+j-1})$$

hence, we easily calculate:

$$\theta(\sigma'(u,k)) = \theta(\sigma(u,k)) + f_1 \Delta$$
$$\theta(\sigma''(u,k)) = \theta(\sigma(u,k)) + (f_1 - f_2)\Delta$$

where $\Delta = \sum_{j \in I_\sigma} \sum_{s \in \overline{s}_{i_j}} c_s - \sum_{j+1 \in I_\sigma} \sum_{s \in \overline{s}_{i_j}}$.

Since $f_1 > 0$ and $f_1 - f_2 < 0$, we have either $\theta(\sigma'(u,k)) \geq \theta(\sigma(u,k))$ or $\theta(\sigma''(u,k)) \geq \theta(\sigma(u,k))$, so lemma was proved. From lemma 1 and lemma 2 we can construct consecutively behaviors σ^* by choosing σ' or σ'' compatible. After each time, $card(F_\sigma)$ decreasing by 1, and finally (after q times) $card(F_{\sigma^*}) = 2$, we reach a integral behavior σ^* satisfying $\theta(\sigma^*(u,k)) \geq \theta(\sigma^*(u,k))$. Hence, if there exists a real behaviors σ which fails $\Box D$ (i.e. $\theta(\sigma(u,k)) > M$ then we can get an integral behavior σ^* fails $\Box D$, too. Therefore, if $\Box D$ is satisfied by all integral behaviors of A, then it is satisfied by all (real) behaviors of A.

4. Algorithm

By theorem 1, checking A for TDP can reduce to checking whether all integer behaviors of M satisfy $\Box D$.

Denote the set of all integer transition sequences $\gamma = e_{i_1} e_{i_2} \dots e_{i_k}$ such that source(e_{i_j}) $= \overline{s}_{i_j}$ for $j = 1..k$ (i.e. the sequence matches $\gamma(D)$) by Γ. Constructing the set Γ is easily, so we do not present here. For each such integer transition sequence $\gamma \in \Gamma$, if γ appears in an integer behavior σ then σ will be of the form as in figure 2. Along integer behavior σ, system reaches to state $\overline{s}_{i_1}$ at time point $t_m \in N$ corresponding to integer clock valuation v_m and starting from $(\overline{s}_{i_1}, v_m)$ system continuously runs along γ and takes transitions $e_{i_1}, e_{i_2}, \dots, e_{i_k}$ at time points $t_{m+1}, t_{m+2}, \dots, t_{m+k}$ corresponding to clock valuation $v_m, v_{m+1}, \dots, v_{m+k}$. These clock valuation satisfy constraints $\psi_{e_{i_1}}, \psi_{e_{i_2}}, \dots, \psi_{e_{i_k}}$, where $v_{m+j} = (v_{m+j-1} + x_j)[r \mapsto 0], x_j = t_{m+j} - t_{m+j-1} (1 \le j \le k)$. Verifying $v_{m+j-1} + x_j$ satisfies $\psi_{e_{i_j}}$ corresponds to a linear constraint C_j on x_j from the definition of v_{m+j-1} and $\psi_{e_{i_j}}$ as in algorithm 1.

Therefore, all subsequences γ of a behavior σ and starts from the integral reachable location $(\overline{s}_{i_1}, v)$ satisfy the inequality $\sum_{j=1}^{k} \sum_{s \in \overline{s}_{i_j}} c_s (t_{m+j} - t_{m+j-1}) \le M$ if and only if the optimal value for the following linear integer problem (with k integer variables) is not greater than M.

$$sup \sum_{j=1}^{k} \sum_{s \in \overline{s}_{i_j}} c_s x_j$$

subject to the constraints

$$C_1, C_2, \dots, C_k, x_1 \ge 0, x_2 \ge 0, \dots, x_k, \ge 0$$

(by our convention, the optimal value is $-\infty$ when the constraint set is unfeasible). As above discussion, we see that this problem depends only on the integral clock interpretation v of reachable location $(\overline{s}_{i_1}, v)$ and the sequence γ. That is integer linear programming problem which is in NP. However, by theorem 1, we can take $t_{m+j}'s (1 \le j \le k)$ as real numbers, (thus $x_j's$ be real variables) to convert it to a linear programming $P(v, \gamma)$. The results of the two problems are the same.

In a word, to check $M \models \Box D$, with each couple (v, γ), where v is integer valuation of integral reachable locations $(\overline{s}_{i_1}, v)$ and $\gamma \in \Gamma$, we have to construct and solve the linear programming problem $P(v, \gamma)$ of k variables and verifying if the result is not greater than M.

The number of integral reachable states is infinite, so the number of linear programming is also infinite. However, from the definition of equivalence relation on clock

valuations, we can easily prove following lemma which asserts that for each valuation region π and sequence γ, we have to solve at most one linear programming problem $P(v, \gamma)$, where v is any valuation of π.

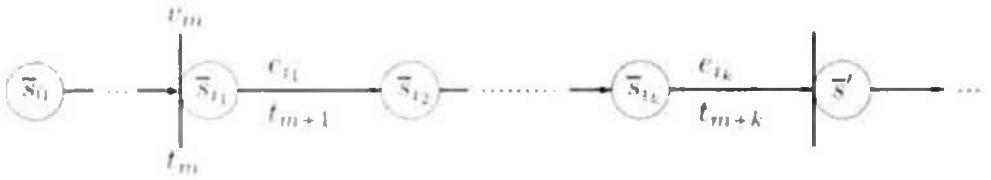

Figure 2. General behavior containing $\overline{s}_{i_1}, \overline{s}_{i_2}, \ldots, \overline{s}_{i_k}$. $\overline{s}' = \text{target}(e_{i_k})$

Lemma 3.. *Let $(\overline{s}_{i_1}, v), (\overline{s}_{i_1}, v')$ be integer reachable states of A and $v \cong v'$ (i.e. $[v] = [v'] = \pi$). Then, for any $\gamma \in \Gamma$ problems $P(v, \gamma)$ and $P(v', \gamma)$ give the same result.*

From the lemma, we can combine a region of clock valuation π with a transition sequence $\gamma \in \Gamma$ to generate a linear programming problem $P(\pi, \gamma)$ instead of set of problems $P(v, \gamma)$ with $\forall v \in \pi$. $P(\pi, \gamma)$ is generated by algorithm 1 below. In the algorithm, we call Cons be the set of constraints of P, it is generated step by step along γ by replacing each clock variable x in constraints of $\psi_{e_{i_j}}$ by $x_j + c_{\pi,x}$ if $x = c_{\pi,x} \in C(\pi)$. If $x > K_x \in C(\pi)$ then constraints of the form $x \geq d$ in $\psi_{e_{i_j}}$ will be reduced and constraints of the form $x \leq d \in \psi_{e_{i_j}}$ will make P infeasible. For example, assume that $\psi_{e_{i_j}} = \{x \geq 2, y \geq 4, z \leq 8\}$ and $\pi = \{x = 2, y > 5, z = 4\}$, then Cons $= \{x_j + 2 \geq 2, x_j + 4 \leq 8\}$ and if $y \leq 4 \in \psi_{e_{i_j}}$ then P is infeasible. As usual, we denote $\alpha[x/y]$ the formula obtained from α by replacing all occurrences of x by y.

Algorithm 1 : Generating linear programming $P(\pi, \gamma)$

Cons := $\{x_1 \geq 0, \ldots, x_k \geq 0\}$; Infeasible := *false*;
For $j := 1$ **to** k **do Begin**
  For every clock $x \in X$ **do Begin**
    If $x > K_x \notin \psi_{e_{i_j}}$ (i.e. $x = c_{\pi,x} \in \psi_{e_{i_j}}$) **then**
      For every constraint α on x in $\psi_{e_{i_j}}$ **do** Cons := Cons $\cup\, \alpha[x/x_j + c_{\pi,x}]$;
    Else If there exists a constraint α on x in $\psi_{e_{i_j}}$ of the form $x \leq d$ **then**
    Begin
      Infeasible := *true*; **break**;
    End;
    If $x \in r_{e_{i_j}}$ **then** $\pi := \pi[c_{\pi,x}/0]$;
  End;
End;
If $\neg$ Infeasible **then** P(π, γ) := sup $\sum_{j=1}^{k} \sum_{s \in \overline{s}_{i_j}} c_s x_j$ subject to Cons;

Idea of the main algorithm is as follows.

By depth-first techniques we sequentially generate reachable nodes $(\overline{s}_{i_1}, \pi)$ of integral region graph of A. Combine π with each γ and solve $P(\pi, \gamma)$.The process terminate

when all of reachable nodes were generated (i.e. $A \models TDP$) or there exists a problem $P(\pi, \gamma)$ give negative answer (i.e A do not satisfy TDP).

Basic steps of algorithm is given in algorithm 2. In the algorithm we use S denoting a stack saving current path lead to $\overline{s}_{i_1}$, n to current node of region graph of network and SN to the set of successive node of n which has been traversed.

To find successive node of n we can use the algorithm 3.

Algorithm 2 : Checking TDP algorithm

```
S := {(s̄0, [v0])}; SN := ∅;
Repeat
    pop(S,n);
    If n = (s̄, π) have no new successive node then Begin
       If s̄ = s̄i1 then check P(π, γ), (∀γ ∈ Γ);
    End
    Else Begin
       push(n, S);
       n := a new successive node of n;
       If n ∉ SN then Begin
          push(n, S); SN := SN ∪ {n};
       End;
    End;
Until empty(S);
```

Algorithm 3 : Finding the set of successive of $(\overline{s}, \pi)$

```
Succ := ∅; d := 0;
Repeat
    π' := π + d;
    if ∃e = (s̄, ψe, a, re, s̄') ∈ E such that π' satisfies ψe
    then Succ := Succ + π'[r ↦ 0];
    d := d + 1;
Until π' = π;
```

5. Conclusion

In this paper, we applied and extended techniques in [8] to give an algorithm for deciding whether a timed automaton network satisfies a temporal duration property. Although, timed automaton was checked for TDP [8], however, it have not dealed directly in the case of network. So, we think that a such algorithm is necessary. Main our extension in this paper is showing that TDP is also discretisable for network of timed automata and re-arrangement linear programming problem producing from each fragment γ of behavior σ and each reachable region π.

In fact, complexity of our algorithm is still high. This depends on natural basis of problem. As behaviors of system have to run along sequence of global locations, the algorithm have to search reachable nodes on region graph of production automaton. That is item which decides the high complexity of the algorithm while linear programming problem is in class of polynomial complexity problems. However, there is still some techniques which can be used to reduce the complexity of the algorithm. For example, we can apply techniques proposing by Alur and Dill [1] for minimizing region graph or techniques "FIS : finite index sets" in [7] to generate integral abstract graph which have size smaller than region graph. We think that, techniques in [8] (basis of this report) is very useful for checking some another duration properties. Especially, we hope combining techniques discretising and linear programming for checking Linear Duration Invariant in future.

Acknowledgement The author would like to thank Dr. Dang Van Hung for his valuable comments and encouragement when writing this paper.

References

1. R. Alur, D.L. Dill, A Theory of timed automata, *Theoretical computer science*, 1994, pages 183-235.
2. Zhou Chaochen, C.A.R. Hoare, Anders P. Ravn, A calculus of durations, *Information processing letters*, 40 5(1991), pp 269–276.
3. Y. Kesten, A. Pnueli, J Sifakis, S. Yovine, Integration Graphs: A Class of decidable hybrid systems, In *Hybrid systems*, volume 736 of *Lecture notes in computer science*, Springer Verlag, 1994, pp 179-208.
4. Zhou Chaochen, Zhang Jingzhong, Yang Lu, 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.
5. Li Xuan Dong, 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.
6. Pham Hong Thai, 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. Zhao Jianhua, 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, N0 5(2000), pp. 423–429.
8. Li Yong, Dang Van Hung, Checking temporal duration properties of timed automata, Technical report 214, UNU/IIST, P.O.Box 3058, Macau, October 2001, Published in *Journal of computer science and technology*, Vol. 17, No. 6(2002) pp. 689 - 698
9. Pham Hong Thai, Discretising and verifying temporal duration properties for timed automaton networks, Proceedings of *The first international workshop for computer, information and communication technologies*, 2003, pp. 18 - 24.

TẠP CHÍ KHOA HỌC ĐHQGHN, KHTN & CN, T.XIX, N04, 2003

KIỂM TRA HỆ THỜI GIAN THỰC HOẠT ĐỘNG SONG SONG ĐỐI VỚI CÁC TÍNH CHẤT KHOẢNG TUẦN TỰ BẰNG QUY HOẠCH TUYẾN TÍNH

Phạm Hồng Thái

Khoa Công nghệ, ĐHQG Hà Nội

Bài toán kiểm chứng mô hình đối với công thức khoảng đã được quan tâm nhiều hơn trong những năm gần đây. Đã có một số công trình đề xuất thuật toán kiểm chứng cho các công thức khoảng như "Tính chất khoảng tuyến tính" (Linear Duration Properties), "Bất biến khoảng tuyến tính" (Linear Duration Invariant), "Tính chất khoảng tuần tự" (Temporal Duration Properties - TDP) đối với lớp các hệ thống biểu được bởi ôtômat thời gian. Trong đó, "Tính chất khoảng tuần tự" cho đến nay vẫn chưa được bàn chi tiết đối với hệ thống các hệ thời gian thực hoạt động song song. Trong bài báo này, chúng tôi đề nghị một thuật toán kiểm chứng TDP cho các hệ thống như vậy, tức đối với các hệ thống biểu diễn được bởi lưới ôtômat thời gian. Thuật toán được đặt trên cơ sở tìm kiếm theo độ sâu trên đồ thị phân vùng của lưới ôtômat và đưa bài toán kiểm chứng về việc giải một tập hợp các bài toán qui hoạch tuyến tính, do vậy độ phức tạp của thuật toán là chấp nhận được.