Verifying Linear Duration Constraints of Timed Automata

Pham Hong Thai and Dang Van Hung

United Nations University,
International Institute for Software Technology,
P. O. Box 3058, Macau
{dvh, pht}@iist.unu.edu

Abstract. This paper aims at developing a technique for checking if a timed automaton satisfies a linear duration constraint on the automaton states. The constraints are represented in the form of linear duration invariants - a simple class of chop-free Duration Calculus (DC) formulas. We prove that linear duration invariants of timed automata are discretisable, and reduce checking if a timed automaton satisfies a linear duration invariant to checking if the integer timed region graph of the original automaton satisfies the same linear duration invariant. The latter can be done with exhaustive search on graphs. In comparison to the techniques in the literature, our method is more powerful: it works for the standard semantics of DC and the class of the closed timed automata while the others cannot be applied.

1 Introduction

Constraints on the durations of system states form a class of important properties of real-time systems. They can be formalised by a class of simple chop-free Duration Calculus formulas of the form $A \leq \ell \leq B \Rightarrow \sum_{s \in S} c_s \int s \leq M$. This class was first introduced with the name *linear duration invariants* and investigated in [14]. The duration of a state s is a mapping from time intervals to reals and is denoted by $\int s$. $\int s$, when applied to an observation time interval $[b, e]$ is the accumulated time for the presence of state s over $[b, e]$; and the term ℓ when applied to an observation time interval $[b, e]$ returns the length $e - b$ of the interval. A linear duration invariant $A \leq \ell \leq B \Rightarrow \sum_{s \in S} c_s \int s \leq M$ simply says that for any observation time interval $[b, e]$, if the length ℓ of the interval satisfies the constraint $A \leq \ell \leq B$ then the durations of the system states over that interval should satisfy the constraint $\sum_{s \in S} c_s \int s \leq M$. A desired property for a simple gas burner "for any observation interval that is longer than 60 seconds, the ratio between the duration of the state *leak* and the length of the interval should not be more than 5%" is represented as a linear duration invariant $\ell \geq 60 \Rightarrow \int leak \leq 5\% * (\int leak + \int nonleak)$ (here we have used the equation $\int leak + \int nonleak = \ell$). A system safety saying that an unsafe state s should not occur, can also be represented by a linear duration invariant as $\ell \geq 0 \Rightarrow \int s \leq 0$. The relative fairness of two processes p_1 and p_2 can be represented by two linear duration invariants $\ell \geq 0 \Rightarrow \int p_1.run - \int p_2.run \leq 1$ and $\ell \geq 0 \Rightarrow \int p_2.run - \int p_1.run \leq 1$. This says that the running time of processes p_1 and p_2 are almost the same for any observation interval.

Since timed automata are good models of real-time systems, and since linear duration invariants are important properties of real-time systems, it is interesting if verifying a linear duration invariant of a timed automata can be done automatically. In fact, this problem has attracted a great deal of attention during last decade, since the introduction of Duration Calculus in [13]. Many algorithms have been proposed in the literatures, but all of them have high complexity and do not work for the general case. Some restrictions are needed either on timed automata, or on linear duration invariants or on the meaning of the satisfaction of linear duration invariants by automata in order for those algorithms to apply.

For example, in [5], a solution for checking a LDP of a timed automaton LDI is given using mixed integer and linear programming techniques. The authors have to put restrictions on both linear duration invariants and the meaning of satisfaction: the premise of LDIs should be true, i.e. there is no constraint on the length of the observation intervals, the coefficients in LDIs should be integral, and the observation intervals should start at time 0. In [14], a nice solution to the problem is given using linear programming (techniques) only, but the authors had to restrict themselves on the *real-time* automata, i.e. timed automata with one clock which is reset by every transition. This solution is generalised in [7] and in [9] to a wider subclass of timed automata, but still cannot be used for the whole class of timed automata, and the restriction on the meaning of satisfaction still applies. In [3] the authors considered checking LDI for timed automata with observation intervals started at time 0 only, which is a restriction on the meaning of satisfaction. In general, these algorithms are based on symbolic representation of the behavior of the systems by extended time regular expressions, and hence, reduce the problem to a number of linear programming problems. In practice, the number of linear programming problems which have to be solved is very large, so the time complexity of these algorithms is very high.

For reducing the complexity of the problem, some other papers use a different approach. The authors of these papers consider those properties which are discretisable, i.e. they are satisfied by all the behaviours of a timed automata if and only if they are satisfied by all integral behaviors only. That means, we can check such kind of properties of a timed automaton by exploring the integral region graph of the automaton as in [12]. This technique is combined with linear programming technique in [8] for checking some other classes of discretisable properties. In these papers, the authors also had to enforce some restrictions on linear duration invariants and on observation intervals.

In this paper, we study if we can remove the restrictions mentioned as above and develop a general technique to solve the problem for the general case. The idea on discretisability of LDP in [12] is the motivation for the discretisability of LDIs in this paper. We prove that LDI is also a discretisable property for timed automata. However, the discretisability of LDIs is used in this paper in a manner that is different from the one in [12]. In the following, we call a LDI having the premise equivalent to "true" (i.e. the premise can be removed) a linear duration property (LDP). In [12] the discretisability of LDP is used to reduce a region graph to an integral region graph, but in this paper the discretisability of LDI is used to approximate a real-time interval by integral-time interval as well.

Our results are summarised as follows. We first define the different semantics for the satisfaction of a LDI by a timed automaton. We do this by introducing the different classes of Duration Calculus models defined by a timed automaton $\mathcal{A}$: $\mathcal{M}_0(\mathcal{A})$ is the set of DC models generated by $\mathcal{A}$ with the observation intervals of the form $[0, t]$, where t is a non negative real; $\mathcal{M}(\mathcal{A})$ is the set of DC models generated by $\mathcal{A}$ with no restriction on the observation intervals; $\mathcal{M}_{uv}(\mathcal{A})$ is the set of DC models generated by $\mathcal{A}$ with the observation intervals of the form $[t_p, t_q]$, where t_p and t_q are the times the automaton enters states p and q, respectively, in the corresponding behaviour; and $\mathcal{M}_I(\mathcal{A})$ is the set of DC models corresponding to the integral behaviours $\mathcal{A}$ with the integral observation intervals. Then, we prove that given a LDI D, $\mathcal{M}(\mathcal{A}) \models D$ if and only if $\mathcal{M}_I(\mathcal{A}) \models D$. Based on these results we reduce the problem of checking $\mathcal{M}(\mathcal{A}) \models P$ to the one of checking $p \models P$ for all paths p in the region graph of $\mathcal{A}$, and we reduce the problem of checking $\mathcal{M}(\mathcal{A}) \models D$ to the one of checking $p \models D$ for all path p in the integral region graph of $\mathcal{A}$. The resulting problem can be solved by standard exhaustive search techniques.

The paper is organized as follows. In the next section we recall some basic notions of timed automata and Duration Calculus formulas. In Section 3 we prove the discretisability of LDIs for timed automata. In Section 4, we propose an algorithm for checking a LDI of a timed automata by searching on the weighted graph constructed from the integral region graph of the automaton. Finally, Section 5 is the conclusion of this paper.

2 Preliminaries

In this section, we recall some notions that will play the basic role in defining the problem in this paper. They are timed automata, region graphs, and Duration Calculus formulas in the form of Linear Duration Invariants (LDI).

2.1 Timed Automata

Timed automata was introduced in [1, 2] as formal models for real-time systems. Here we only give a brief description for timed automata and their behavior. Readers are referred to [1] for their more details. As usual, we denote by $\mathbf{R}^+$ and $\mathbf{N}$ the sets of nonnegative real numbers and natural numbers, respectively.

For a finite set of clock variables X, let $\Phi(X)$ be the set of clock constraints on X, which are conjunctions of the formulas of the form $x \leq c$ or $c \leq x$, where $x \in X$ and $c \in \mathbf{N}$. A timed automaton is a finite state machine equipped with the set of clock variables X, and is defined as follows.

Definition 1. *A timed automaton $\mathcal{A}$ is a tuple $\langle L, s_0, \Sigma, X, E, I\rangle$, where*

- *L is a finite set of locations,*
- *$s_0 \in L$ is an initial location,*
- *Σ is a finite set of symbols (action names),*
- *X is a finite set of clocks,*
- *I is a mapping that assigns to each location $s \in L$ a clock constraint $I(s) \in \Phi(X)$ which is called invariant of location s. Intuitively, the timed automaton only stays at s when the values of the clocks satisfy the invariant $I(s)$.*
- *$E \subseteq L \times \Phi(X) \times \Sigma \times 2^X \times L$ is a set of switches. A switch $\langle s, \varphi, a, \lambda, s'\rangle$ represents a transition from location s to location s' with symbol a, where φ is a clock constraint over X that specifies the enabling condition of the switch, and $\lambda \subseteq X$ gives the set of clocks to be reset with this switch.*

For convenient to our method, here we consider the class of the closed timed automata, i.e. timed automata that do not include clock constraints of the form $x < c$ or $c < x$.

A clock interpretation ν for the set of clocks X is a mapping that assigns a nonnegative real value to each clock. For $\delta \in \mathbf{R}$, let $\nu + \delta$ denote the clock interpretation which maps every clock $x \in X$ to the value $\nu(x) + \delta$. For $\lambda \subseteq X$, let $\nu[\lambda := 0]$ denote the clock interpretation which assigns 0 to each $x \in \lambda$ and agrees with ν over the rest of the clocks.

A state of automaton $\mathcal{A}$ is a pair (s, ν) where s is a location of $\mathcal{A}$ and ν is a clock interpretation which satisfies invariant $I(s)$. State (s_0, ν_0) is the initial state where s_0 is the initial location of $\mathcal{A}$ and ν_0 is the clock interpretation for which $\nu_0(x) = 0$ for all clocks x.

A transition of $\mathcal{A}$ can:

- change state by letting time elapse: For a state (s, ν) and a real-valued time increment $\delta \geq 0$, $(s, \nu) \xrightarrow{\delta} (s, \nu + \delta)$ if for all $0 \leq \delta' \leq \delta$, $\nu + \delta'$ satisfies invariant $I(s)$.
- change state by taking a location switch : For a state (s, ν) and a switch $\langle s, \varphi, a, \lambda, s'\rangle$ such that ν satisfies φ and $\nu[\lambda := 0]$ satisfies $I(s')$ then $(s, \nu) \xrightarrow{a} (s', \nu[\lambda := 0])$.

A time elapsing transition and a following location switching transition can be combined into one transition and denoted by $(s, \nu) \xrightarrow{\delta, a} (s', \nu')$. That means the system stays at location s with the current clock interpretation ν, after δ time units, the clock interpretation $\nu + \delta$ satisfies the enabling condition (time constraint) φ of switch $e = \langle s, \varphi, a, \lambda, s'\rangle$, and the system transits to location s' by taking e with label a and resets the clocks in the λ to 0, and the new state of the system is (s', ν').

In this paper we consider only *nonZeno* behaviours of automata ([1, 2]), that is those behaviours for which in any finite time interval there is only a finite number of transition occurrences.

Example 1. The timed automaton in Figure 1, taken from [4], have two clocks x and y. The set of locations is $\{s_0, s_1\}$, where invariants are $I(s_0) = (y \leq 5)$ and $I(s_1) = (x \leq 8 \wedge y \leq 10)$. The set of switches is

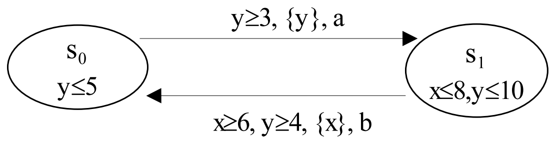

Fig. 1. A timed automaton

$\{\langle s_0, y \geq 3, a, \{y\}, s_1\rangle, \langle s_1, x \geq 6 \wedge y \geq 4, b, \{x\}, s_0\rangle\}$. The clock y is reset to 0 each time the automaton transits from s_0 to s_1 and the clock x is reset to 0 when the automaton transits from s_1 to s_0.

Definition 2. *Let $\mathcal{A}$ be a timed automaton.*

1. *A* run *or an* execution *r of $\mathcal{A}$ is an infinite sequence of state transitions: $(s_0, \nu_0) \overset{\delta_1, a_1}{\rightarrow} (s_1, \nu_1) \overset{\delta_2, a_2}{\rightarrow} \ldots$, where (s_0, ν_0) is an initial state of $\mathcal{A}$.*
2. *A* behavior *ρ correspondents to above run r, is the infinite sequence of timed locations*

$$\rho : (s_0, t_0)(s_1, t_1) \ldots (s_m, t_m) \ldots$$

that satisfies the following conditions
- *$t_0 = 0$.*
- *for any $T \in \mathbf{R}$, there is some $i \geq 0$ such that $t_i \geq T$.*
- *t_i is the moment that system enters to s_i, for all $i \geq 0$. That means, $\delta_i = t_i - t_{i-1}$ and $\mathcal{A}$ stays in state s_{i-1} for $t_i - t_{i-1}$ time units and then transits to s_i in the run r.*

Note that in this paper, a behavior of a timed automaton is a sequence of time stamped locations instead of a sequence of timed stamped switches as in other papers. However the semantics of timed automata is not changed. This way of representation of behaviours shows the DC models generated by them more explicitly. A run or a behavior is said to be *integral* iff for all $i \geq 0$, the values of clock variables in ν_i, the time delay δ_i, and time stamps t_i are integral.

2.2 Linear Duration Invariants and Duration Properties

Models in Duration Calculus. Duration Calculus (DC) was introduced by Zhou Chaochen et al. in [13] as a logic to reason about the state duration of real-time systems. A comprehensive introduction to *DC* can be found in the recent monograph [15]. In DC, a state is viewed as a boolean-valued function of the continuous time that has the value true (denoted by 1) at time t iff the state is present at t. Otherwise it takes the value 0. An interpretation $\mathcal{I}$ of the system is an assignment that assigns to each system state s a boolean-valued function $\mathcal{I}_s$. A DC model consists an interpretation $\mathcal{I}$ and a time interval $[b, e]$.

It represents an observation of the behavior of the system states in an interval of time $[b, e]$. Given an interpretation $\mathcal{I}$, the duration of a state s over time interval $[b, e]$ is defined as $\int_b^e \mathcal{I}_s(t)dt$, which is exactly the accumulated present time of s in the interval $[b, e]$ by the interpretation $\mathcal{I}$.

In this paper we consider the set of DC models that express all the observations of the behaviours of a timed automaton. Each behavior $\rho = (s_0, t_0)(s_1, t_1)(s_2, t_2)\ldots$ of timed automaton $\mathcal{A}$ defines uniquely an interpretation $\mathcal{I}$ in DC by: for any $s \in L$, $\mathcal{I}_s(t) = 1$ iff $\exists i \bullet (s_i = s \wedge t \in [t_i, t_{i+1}))$. We also denote such $\mathcal{I}$ by $(\overline{s}, \overline{t})$ in which $\overline{s} = (s_0, s_1, \ldots)$ and $\overline{t} = (t_0, t_1, \ldots)$ express a sequence of s_i and t_i from behaviour ρ. Hence, $(\overline{s}, \overline{t}, [b, e])$ is a DC model representing the observation of ρ in the interval $[b, e]$, which is an observation of the timed automaton $\mathcal{A}$ over interval $[b, e]$. We also call $(\overline{s}, \overline{t}, [b, e])$ a DC model of $\mathcal{A}$.

Let $\mathcal{M}(\mathcal{A})$ denote set of DC models of $\mathcal{A}$. To cope with the different meanings of the satisfaction of a DC formula by $\mathcal{A}$ as said in the introduction of the paper, we introduced the following classes of DC models of $\mathcal{A}$:

$$\begin{aligned}
\mathcal{M}_0(\mathcal{A}) &= \{\sigma \mid \sigma = (\overline{s}, \overline{t}, [0, T]) \in \mathcal{M}(\mathcal{A}),\ T \geq 0\},\\
\mathcal{M}_{uv}(\mathcal{A}) &= \{\sigma \mid \sigma = (\overline{s}, \overline{t}, [t_u, t_v]) \in \mathcal{M}(\mathcal{A}),\ t_u, t_v \text{ occur in } \overline{t} \text{ and } t_u \leq t_v\},\\
\mathcal{M}_I(\mathcal{A}) &= \{\sigma \mid \sigma = (\overline{s}, \overline{t}, [b, e]) \in \mathcal{M}(\mathcal{A}) \text{ and } t_i, b, e \in \mathbf{N}, \forall i \geq 0\}.
\end{aligned}$$

In the other word, $\mathcal{M}_0(\mathcal{A})$ is the set of models representing the observations starting from 0 and ending at any time point. $\mathcal{M}_{uv}(\mathcal{A})$ consists of models that representing the observations starting and ending at those time points at which the automaton transits to a location, i.e. the observations between two location switching transitions. A DC model of $\mathcal{A}$ in $\mathcal{M}_I(\mathcal{A})$ represents an observation of an integral behavior of $\mathcal{A}$ (i.e behavior in which transitions take place only at integer time) from an integer time point to an integer time point.

Linear Duration Properties and Linear Duration Invariants. Given a timed automaton $\mathcal{A} = \langle L, s_0, \Sigma, X, E, I\rangle$. A linear duration invariant over L is a DC formula of the form

$$\mathcal{D} : A \leq \ell \leq B \Rightarrow \sum_{s \in L} c_s \int s \leq M.$$

where c_s, A, B and M are real numbers, $A \leq B$ (B may be ∞). In $\mathcal{D}$ DC term $\int s$ is a duration term denoting the duration of location s, and ℓ is a DC term denoting the interval length. LDI $\mathcal{D}$ evaluates over a DC model $(\mathcal{I}, [b, e])$ as tt and denoted by $(\mathcal{I}, [b, e]) \models \mathcal{D}$ iff $A \leq e - b \leq B \Rightarrow \sum_{s \in L} c_s \int_b^e \mathcal{I}_s(t)dt \leq M$ evaluates to true (in the predicate calculus). Here we define the satisfaction of $\mathcal{D}$ by $\mathcal{A}$ directly on the behaviours of $\mathcal{A}$ as follows.

Definition 3. *Let $\mathcal{D}$ be a LDI as above. For each $\sigma = (\overline{s}, \overline{t}, [b, e]) \in \mathcal{M}(\mathcal{A})$ we define $l(\sigma)$ and $\theta(\sigma)$ as*

$$l(\sigma) = e - b \text{ and } \theta(\sigma) = \sum_{s \in L} c_s P_s$$

where P_s is the accumulated time for the presence of location s in the interval $[b, e]$ and is calculated as follows. Let u and v be the indexes in $\bar{t}$ such that $t_{u-1} < b \leq t_u$ and $t_v \leq e < t_{v+1}$. For $s \neq s_v$ and $s \neq s_{u-1}$, $P_s = \sum_{u \leq j \leq v-1 \wedge s_j = s}(t_{j+1} - t_j)$. $P_{s_{u-1}} = \sum_{u \leq j \leq v-1 \wedge s_j = s_{u-1}}(t_{j+1} - t_j) + (t_u - b)$, and $P_{s_v} = \sum_{u \leq j \leq v-1 \wedge s_j = s_v}(t_{j+1} - t_j) + (e - t_v)$.

Hence, $\theta(\sigma)$ evaluates over model $\sigma = (\bar{s}, \bar{t}, [b, e])$ as

$$\theta(\sigma) = c_{s_{u-1}}(t_u - b) + \sum_{u \leq j \leq v-1 \wedge s_j = s} c_s(t_{j+1} - t_j) + c_{s_v}(e - t_v) \qquad (1)$$

By expanding the sum and letting t_i's be common factors, we have

$$\theta(\sigma) = \sum_{i=u}^{v} a_i t_i + c_{s_v} e - c_{s_{u-1}} b \qquad (2)$$

where a_i's are real numbers that are derivable from c_s's.

Definition 4. *Given a LDI $\mathcal{D}$.*

- *A DC model $\sigma = (\mathcal{I}, [b, e]) \in \mathcal{M}(\mathcal{A})$ is said to satisfy $\mathcal{D}$, denoted by $\sigma \models \mathcal{D}$, iff $A \leq l(\sigma) \leq B$ implies $\theta(\sigma) \leq M$.*
- *Timed automaton $\mathcal{A}$ is said to satisfy $\mathcal{D}$, denoted by $\mathcal{A} \models \mathcal{D}$, iff $\sigma \models \mathcal{D}$, for all $\sigma \in \mathcal{M}(\mathcal{A})$.*

When a LDI $\mathcal{D}$ has the premise equivalent to true, i.e. equivalent to $0 \leq \ell \leq \infty$, we say that $\mathcal{D}$ is a linear duration property (LDP) ([12]). So, a LDP is a special LDI which do not have premise. Hence, checking a LDP is normally simpler than checking a LDI.

Similarly, for any class $\mathcal{M}_x(\mathcal{A})$, $x \in \{uv, I, 0\}$, we define $\mathcal{M}_x(\mathcal{A}) \models \mathcal{D}$ iff $\sigma \models \mathcal{D}$ for all $\sigma \in \mathcal{M}_x(\mathcal{A})$.

The model-checking problem in this paper is formulated as: given a timed automaton $\mathcal{A} = \langle L, s_0, \Sigma, X, E, I \rangle$, given a LDI $\mathcal{D}$ over L; find an algorithm to decide whether $\mathcal{A} \models D$.

3 Discretisability of Linear Duration Invariants with Respect to Timed Automata

3.1 ϵ-Digitising

The concept of ϵ-digitising was first introduced in [6]. We recall here the definition of ϵ-digitisation given by them.

Definition 5. *Given a positive real x and ϵ, $(0 \leq \epsilon < 1)$. Let x_ϵ be an integer defined as*

$$x_\epsilon = \begin{cases} \lfloor x \rfloor & \textit{if fraction of } x \textit{ is less than or equal to } \epsilon \\ \lceil x \rceil & \textit{otherwise} \end{cases}$$

The number x_ϵ is called ϵ-digitisation of x.

Some properties of ϵ-digitisation needed in the development of our techniques are listed in the following lemmas. Proving of these lemmas is easy so reader be referred [10].

Lemma 1. *Given two integer numbers $a \le b$, given two nonnegative real numbers $t_i \ge t_j$. Then for all $\epsilon \in [0,1)$ we have*

$$a \le t_i - t_j \le b \Leftrightarrow a \le t_{i\epsilon} - t_{j\epsilon} \le b\,.$$

As a consequence of the lemma, if $t_i \ge t_j$ then $t_{i\epsilon} \ge t_{j\epsilon}$ for all $\epsilon \in [0,1)$ (apply the lemma with $a = 0$). This means that under the ϵ-digitisation, the order of a sorted sequence of real numbers is unchanged.

Lemma 2. *Let $\{a_i\}$, $\{t_i\}$, $(i = 1..m)$ be two sequences of real numbers, where $t_i \ge 0$ for all $i = 1, \ldots, m$. Then we can always find a real number $\epsilon \in [0,1)$ such that*

$$\sum_{i=1}^{m} a_i t_i \le \sum_{i=1}^{m} a_i t_{i\epsilon}$$

Lemma 3. *Let $\sigma = (\overline{s}, \overline{t}, [b,e]) \in \mathcal{M}(\mathcal{A})$ be a DC model of timed automaton $\mathcal{A}$. Let $\overline{s} = s_0, s_1, \ldots$; $\overline{t} = t_0, t_1 \ldots$ and $t_{u-1} < b \le t_u$, $t_v \le e < t_{v+1}$. Then for all $\epsilon \in [0,1)$, $\sigma_\epsilon = (\overline{s}, \overline{t}_\epsilon, [b_\epsilon, e_\epsilon])$ is an integral model of $\mathcal{A}$, i.e. $\sigma_\epsilon \in \mathcal{M}_I(\mathcal{A})$, where $\overline{t}_\epsilon = t_{0\epsilon}, t_{1\epsilon}, \ldots$.*

3.2 Discretisability of LDI

Definition 6. *Given a timed automaton $\mathcal{A}$ and a linear duration invariant $\mathcal{D}$. $\mathcal{D}$ is said to be* discretisable *with respect to $\mathcal{A}$ if $\mathcal{A} \models \mathcal{D}$ exactly when $\mathcal{M}_I(\mathcal{A}) \models \mathcal{D}$.*

Theorem 1. *Any linear duration invariant $\mathcal{D}$ which has the premise $A \le \ell \le B$ in which A and B are integral, is discretisable with respect to timed automaton $\mathcal{A}$ (here we consider ∞ as an integer by our convention).*

Proof. We have to prove that $\mathcal{M}(\mathcal{A}) \models \mathcal{D} \Leftrightarrow \mathcal{M}_I(\mathcal{A}) \models \mathcal{D}$.

The "only if" part is obvious because $\mathcal{M}_I(\mathcal{A}) \subseteq \mathcal{M}(\mathcal{A})$.

The "if" part is proved as follows. Let $\sigma \in \mathcal{M}(\mathcal{A})$ such that $\sigma \not\models \mathcal{D}$. We prove that there exists $\epsilon \in [0,1)$ such that $\sigma_\epsilon \not\models \mathcal{D}$, where σ_ϵ is the digitisation of σ w.r.t. ϵ.

Assume that $\sigma = (\overline{s}, \overline{t}, [b,e])$ with $\overline{s} = s_0, s_1, \ldots$ and $\overline{t} = t_0, t_1, \ldots$. Let indexes u and v be such that $t_{u-1} < b \le t_u$, $t_v \le e < t_{v+1}$. $\sigma \not\models \mathcal{D}$ implies that $A \le e - b \le B$ and $\theta(\sigma) > M$. By the definition of LDI, it follows from equation (2):

$$\theta(\sigma) = \sum_{i=u}^{v} a_i t_i + c_{s_v} e - c_{s_{u-1}} b > M$$

From Lemma 2 (note that coefficients a_i's in the lemma are any reals), $\exists \epsilon \in [0,1)$ such that $\sum_{i=u}^{v} a_i t_{i\epsilon} + c_{s_v} e_\epsilon - c_{s_{u-1}} b_\epsilon \geq \theta(\sigma) > M$. By Lemma 1 it follows from $A \leq e - b \leq B$ that $A \leq e_\epsilon - b_\epsilon \leq B$ (notice that A, B are integers). By Lemma 3 σ_ϵ is an integral DC model of $\mathcal{A}$, and $\theta(\sigma_\epsilon) = \sum_{i=u}^{v} a_i t_{i\epsilon} + c_{s_v} e_\epsilon - c_{s_{u-1}} b_\epsilon$. Hence, $\theta(\sigma_\epsilon) > M$.

Thus, we have obtained an integral model σ_ϵ which does not satisfy $\mathcal{D}$.

Now that the assumption that two integral bounds A and B in the premise of LDIs is not too restricted, and the result can be extended to the case that A and B are rationals using the well-known technique.

From this theorem, from now on we will consider only the integral DC models of timed automaton $\mathcal{A}$, i.e. models $\sigma = (\overline{s}, \overline{t}, [b,e]) \in \mathcal{M}_I(\mathcal{A})$.

4 Checking Linear Duration Invariants of Timed Automata with Graph Search

4.1 Integral Reachability Graph of Timed Automata

In this section, we shortly recall about integral region graph which is a part of region graph. Region graph was presented by Alur and Dill in ([1]) and has become well-known.

Let K_x be the largest constant compared with the clock $x \in X$ in the time constraints and the invariants of $\mathcal{A}$ and let $K = \max\{K_x \mid x \in X\} + 1$. An equivalence relation restricted into the set of all integral clock interpretations of $\mathcal{A}$ is defined as follows. Let ν_1, ν_2 be two integer clock interpretations. We say that ν_1 is equivalent to ν_2 and denoted by $\nu_1 \cong \nu_2$ iff for all $x \in X$ either $\nu_1(x) = \nu_2(x)$ or $\nu_1(x) \geq K_x + 1 \;\wedge\; \nu_2(x) \geq K_x + 1$. The equivalence class containing ν is denoted by $[\nu]$ and is called integral clock region. It is easy to see that number of integral clock regions is bounded by $(K+1)^k$ (k is number of clocks).

The equivalence relation $\cong$ is also extended to an equivalence relation on state space of timed automata. we call two states $q_1 = (s_1, \nu_1)$ and $q_2 = (s_2, \nu_2)$ of timed automaton $\mathcal{A}$ be region-equivalent (denoted by $q_1 \equiv q_2$) iff $\nu_1 \cong \nu_2$ and $s_1 = s_2$. The equivalence relation $\equiv$ partitions space of states of $\mathcal{A}$ into classes of states, each class is characterized by a couple of a location s and a clock region π and is denoted by $\langle s, \pi \rangle$. We also call $\langle s, \pi \rangle$ a region. It is obvious that the number of regions is bounded by $|L|(K+1)^k$.

A region $\langle s', [\nu'] \rangle$ is called be successor of $\langle s, [\nu] \rangle$ if $\exists d \geq 0$ and an transition $e = \langle s, \varphi, a, \lambda, s' \rangle$ such that $(s, \nu) \overset{d,a}{\to} (s', \nu')$. Then we write $\langle s, [\nu] \rangle \overset{d,a}{\to} \langle s', [\nu'] \rangle$

We can easily prove the following lemma.

Lemma 4. *If $(s, \nu) \overset{d,a}{\to} (s', \nu')$ then $\langle s, [\nu] \rangle \overset{d,a}{\to} \langle s', [\nu'] \rangle$, and reversely, if $\langle s, \pi \rangle \overset{d,a}{\to} \langle s', \pi' \rangle$ then for each $\nu \in \pi$, there exists $\nu' \in \pi'$ such that $(s, \nu) \overset{d,a}{\to} (s', \nu')$.*

From the lemma 4, the integral reachability graph $\mathcal{RG} = (\mathsf{V}, \mathsf{E})$ of the timed automaton $\mathcal{A}$ is built as follows. Each vertex $\mathsf{v} \in \mathsf{V}$ is a region $\langle s, \pi \rangle$. E is initialised to $\emptyset$, and V is initialised to $\{\langle s_0, \pi_0 \rangle\}$, where s_0 is initial location of $\mathcal{A}$ and π_0 is region with 0 as the value of all of clocks. Then, V is expanded as follows. If a vertex $\langle s, \pi \rangle \in \mathsf{V}$ has a successor $\langle s', \pi' \rangle$ then $\langle s', \pi' \rangle$ is added into V and $\mathsf{e} = (\langle s, \pi \rangle, \langle s', \pi' \rangle)$ is an edge in E. Besides, each edge e is labelled by an interval $[\mathsf{l}(\mathsf{e}), \mathsf{u}(\mathsf{e})]$, where $\mathsf{l}(\mathsf{e})$ and $\mathsf{u}(\mathsf{e})$ are the minimal and maximal integer time delay that automaton can stay at location s before it transits into location s'. $\mathsf{l}(\mathsf{e})$ and $\mathsf{u}(\mathsf{e})$ are defined as:

$$\begin{aligned}\mathsf{l}(\mathsf{e}) &= \inf\left\{d \geq 0 \mid d \in \mathbf{N},\ \langle s, \pi \rangle \overset{d,a}{\to} \langle s', \pi' \rangle\right\},\\ \mathsf{u}(\mathsf{e}) &= \sup\left\{d \geq 0 \mid d \in \mathbf{N},\ \langle s, \pi \rangle \overset{d,a}{\to} \langle s', \pi' \rangle\right\}.\end{aligned}$$

From the definition of $\langle s, \pi \rangle$ and $\langle s', \pi' \rangle$, either $\mathsf{l}(\mathsf{e}) = \mathsf{u}(\mathsf{e})$ or $\mathsf{u}(\mathsf{e}) = \infty$. We will denote a labelled edge e by $(\mathsf{v}, \mathsf{v}', [\mathsf{l}, \mathsf{u}])$.

An detailed algorithm was also constructed in [12] and also in [10].

4.2 Relationship Between $\mathcal{M}_{uv}(\mathcal{A}) \cap \mathcal{M}_I(\mathcal{A})$ and Reachability Graph $\mathcal{RG}$ w.r.t. LDI $\mathcal{D}$

As mentioned above, in this section we consider only integral models. The restriction of $\mathcal{M}_{uv}(\mathcal{A})$ and $\mathcal{M}(\mathcal{A})$ on the integral DC models for $\mathcal{A}$ are $\mathcal{M}_{uv}(\mathcal{A})^I \mathrel{\widehat{=}} \mathcal{M}_{uv}(\mathcal{A}) \cap \mathcal{M}_I(\mathcal{A})$ and $\mathcal{M}_I(\mathcal{A})$, respectively.

Let $\mathcal{RG}$ be the reachability graph of $\mathcal{A}$.

Definition 7. *Let $\mathsf{p} = \mathsf{v}_1\mathsf{v}_2 \ldots \mathsf{v}_m$ be a path in $\mathcal{RG}$, and let $\overline{d} = d_1, d_2, \ldots, d_{m-1}$ be a sequence of integers, where $d_i \in [\mathsf{l}(\mathsf{v}_i, \mathsf{v}_{i+1}), \mathsf{u}(\mathsf{v}_i, \mathsf{v}_{i+1})]$, for $i = 1..m-1$. The sequence $\wp = \mathsf{v}_1 d_1 \mathsf{v}_2 d_2 \ldots \mathsf{v}_{m-1} d_{m-1} \mathsf{v}_m$ (written as $\wp = (\mathsf{p}, \overline{d})$ for short) is called* **weighted interpretation** *of* p.

Definition 8

- *Let $\wp = (\mathsf{p}, \overline{d})$ be a weighted interpretation of path p. We define $l(\wp) \mathrel{\widehat{=}} \sum_{i=1}^{m-1} d_i$ and $\theta(\wp) \mathrel{\widehat{=}} \sum_{i=0}^{m-1} c_{\mathsf{v}_i} d_i$ and call them length and cost of $\wp$ respectively, where c_{v_i} is the coefficient c_{s_i} in formula $\mathcal{D}$ when s_i is the location of v_i.*
- *A weighted interpretation $\wp$ is said to satisfy $\mathcal{D}$, denoted by $\wp \models \mathcal{D}$, iff*

$$A \leq l(\wp) \leq B \Rightarrow \theta(\wp) \leq M$$

- *The graph $\mathcal{RG}$ is said to satisfy LDI $\mathcal{D}$ and is denoted by $\mathcal{RG} \models \mathcal{D}$ iff $\wp \models \mathcal{D}$ for all weighted interpretations $\wp$ of $\mathcal{RG}$.*

The following lemma plays a key role for our checking technique.

Lemma 5. *For any DC model $\sigma \in \mathcal{M}_{uv}(\mathcal{A})^I$, there exists a weighted interpretation $\wp$ of $\mathcal{RG}$ such that $l(\sigma) = l(\wp)$ and $\theta(\sigma) = \theta(\wp)$, and vice versa.*

Proof. Let $\sigma = (\overline{s}, \overline{t}, [t_u, t_v]) \in \mathcal{M}_{uv}(\mathcal{A})^I$. Then,

$$\begin{cases} \overline{s} = s_0 \dots s_u \dots s_v \dots , \\ \overline{t} = t_0 \dots t_u \dots t_v \dots , \\ l(\sigma) = t_v - t_u = \sum_{i=u}^{v-1}(t_{i+1} - t_i), \\ \theta(\sigma) = \sum_{i=1}^{m} c_{s_i} \sum_{\substack{j=u \\ s_j = s_i}}^{v-1} (t_{j+1} - t_j). \end{cases}$$

From the definition of model σ, σ corresponds to the sequence of transitions $(s_u, \nu_u) \overset{\delta_u, a_u}{\rightarrow} (s_{u+1}, \nu_{u+1}) \overset{\delta_{u+1}, a_{u+1}}{\rightarrow} \dots \overset{\delta_{v-1}, a_{v-1}}{\rightarrow} (s_v, \nu_v)$, where $\delta_i = t_{i+1} - t_i$, for all $i = u..v-1$. By Lemma 4 we have: $\langle s_u, [\nu_u]\rangle \overset{\delta_u, a_u}{\rightarrow} \langle s_{u+1}, [\nu_{u+1}]\rangle \overset{\delta_{u+1}, a_{u+1}}{\rightarrow} \dots \overset{\delta_{v-1}, a_{v-1}}{\rightarrow} \langle s_v, [\nu_v]\rangle$. Consequently, the weighted interpretation $\wp = (\mathsf{p}, \overline{d})$, where $\mathsf{p} = \langle s_u, [\nu_u]\rangle \langle s_{u+1}, [\nu_{u+1}]\rangle \dots \langle s_v, [\nu_v]\rangle$ and $\overline{d} = \delta_1, \delta_2, \dots, \delta_v$, satisfies the requirement of the lemma, i.e. $l(\wp) = l(\sigma)$, $\theta(\wp) = \theta(\sigma)$.

To prove the reverse direction, assume that $\wp = (\mathsf{p}, \overline{d})$ is a weighted interpretation of $\mathcal{RG}$, where $\mathsf{p} = \mathsf{v}_u \mathsf{v}_{u+1} \dots \mathsf{v}_v$, $\overline{d}$ is a sequence of integers $d_u, d_{u+1} \dots, d_v$, and $\mathsf{v}_i = \langle s, \pi_i \rangle$ for $i = u..v$. Due to the fact that $\mathcal{RG}$ is a reachability graph of $\mathcal{A}$, there exists a sequence of switches e_i $(i = u..v-1)$ such that $\langle s_u, \pi_u\rangle \overset{\delta_u, a_u}{\rightarrow} \langle s_{u+1}, \pi_{u+1}\rangle \overset{\delta_{u+1}, a_{u+1}}{\rightarrow} \dots \overset{\delta_{v-1}, a_{v-1}}{\rightarrow} \langle s_v, \pi_v\rangle$. By Lemma 4 we can find a model $\sigma \in \mathcal{M}_{uv}(\mathcal{A})$, i.e sequence of clock interpretations $\nu_i \in \pi_i$ such that $(s_u, \nu_u) \overset{\delta_u, a_u}{\rightarrow} (s_{u+1}, \nu_{u+1}) \overset{\delta_{u+1}, a_{u+1}}{\rightarrow} \dots \overset{\delta_{v-1}, a_{v-1}}{\rightarrow} (s_v, \nu_v)$. Hence, $l(\sigma) = l(\wp)$ and $\theta(\sigma) = \theta(\wp)$.

This lemma allows us, instead of checking $\mathcal{M}_{uv}(\mathcal{A}) \models \mathcal{D}$, to check $\mathcal{RG} \models \mathcal{D}$ which can be done by using popular searching techniques.

Removing Infinitive Edges

We now give some lemmas to simplify $\mathcal{RG}$ before doing search. Lemmas 6 and 7 say that the label $[\mathsf{l}, \infty)$ of an edge in $\mathcal{RG}$ either makes $\mathcal{RG}$ not satisfy $\mathcal{D}$ or can be replaced by a finite label $[\mathsf{l}, \mathsf{u}]$ without any change to the result of checking $\mathcal{RG} \models \mathcal{D}$. Recall that the premise of LDI $\mathcal{D}$ is $A \le \ell \le B$.

Lemma 6. *Assume that* $\mathsf{e} = (\mathsf{v}, \mathsf{v}', [\mathsf{l}, \infty))$ *is an infinite edge of region graph* $\mathcal{RG}$. *Then, if* $B = \infty$ *and* $c_{\mathsf{v}} > 0$ *then* $\mathcal{RG} \not\models \mathcal{D}$.

Lemma 7. *Assume that* $\mathsf{e} = (\mathsf{v}, \mathsf{v}', [\mathsf{l}, \infty))$ *is an infinite edge of* $\mathcal{RG}$. *Then label* $[\mathsf{l}, \infty)$ *can be replaced as follows without any change to the result of checking* $\mathcal{RG} \models \mathcal{D}$.

– *If* $B = \infty$ *and* $c_{\mathsf{v}} < 0$, *replace* $[\mathsf{l}, \infty)$ *by* $[\mathsf{l}, \mathsf{u}]$ *with* $\mathsf{u} = \max\{\mathsf{l}, A\}$.
– *If* $B < \infty$, *replace* $[\mathsf{l}, \infty)$ *by* $[\mathsf{l}, \mathsf{u}]$ *with* $\mathsf{u} = \max\{\mathsf{l}, B\}$.

The proof of the above lemmas is simple and is omitted here.

In summary, for checking $\mathcal{M}_{uv}(\mathcal{A}) \models \mathcal{D}$, we can apply the above lemmas first and either we discover $\mathcal{M}_{uv}(\mathcal{A}) \not\models \mathcal{D}$ early or we can convert the infinite edges of $\mathcal{RG}$ into finite ones. From now on, we assume that $\mathcal{RG}$ does not contain infinite edges.

4.3 Weighted Graph for Checking LDI

Similarly to checking LDP ([12]), we can also construct a weighted graph G from the reachability graph $\mathcal{RG}$ (not containing infinite edges) such that $\mathcal{RG} \models \mathcal{D}$ if and only if $\mathsf{G} \models \mathcal{D}$.

The weighted graph $\mathsf{G} = (\mathsf{V}, \mathsf{E}, \omega)$ is constructed from $\mathcal{RG} = (\mathsf{V}_R, \mathsf{E}_R)$ by the following procedure:

Step 1. $\mathsf{V} := \mathsf{V}_R$, $\mathsf{E} := \mathsf{E}_R$.
Step 2. For each edge $\mathsf{e} = ((\mathsf{v}_i, \mathsf{v}_j), [\mathsf{l}_{ij}, \mathsf{u}_{ij}]) \in E_R$,

1. $\mathsf{V} := \mathsf{V} \cup \left\{\mathsf{v}_{ij}^1, \mathsf{v}_{ij}^2, \dots, \mathsf{v}_{ij}^{\mathsf{u}_{ij}-1}\right\}$ and $\omega(\mathsf{v}_{ij}^k) := c_{\mathsf{v}_i}$ for all $k = 0..\mathsf{u}_{ij} - 1$ (where $\mathsf{v}_{ij}^0 = \mathsf{v}_i$ and $\mathsf{v}_{ij}^{\mathsf{u}_{ij}} = \mathsf{v}_j$),
2. $\mathsf{E} := \mathsf{E} \setminus \{\mathsf{e}\}$,
3. $\mathsf{E} := \mathsf{E} \cup \{(\mathsf{v}_{ij}^k, \mathsf{v}_{ij}^{k+1}) \mid k = 0..\mathsf{u}_{ij} - 1\}$, and $\omega(\mathsf{v}_{ij}^k, \mathsf{v}_{ij}^{k+1}) := 1$ for all $k = 0..\mathsf{u}_{ij} - 1$,
4. $\mathsf{E} := \mathsf{E} \cup \{(\mathsf{v}_{ij}^k, \mathsf{v}_j) \mid k = \mathsf{l}_{ij}..\mathsf{u}_{ij} - 1\}$, and $\omega(\mathsf{v}_{ij}^k, \mathsf{v}_j) := 0$ for all $k = \mathsf{l}_{ij}..\mathsf{u}_{ij} - 1$.

Roughly speaking, G is built by "splitting" each edge $\mathsf{e} = (\mathsf{v}, \mathsf{v}', [\mathsf{l}, \mathsf{u}])$ of $\mathcal{RG}$ into u small edges with the length (weight) 1 by adding $\mathsf{u} - 1$ sub-vertices. All of these sub-vertices and v are assigned a weight as the coefficient c_s in LDI, where s is location of vertex v ($s \in \mathsf{v}$). On the other hand, from sub-vertices v^{l} to $\mathsf{v}^{\mathsf{u}-1}$ there are edges joining these sub-vertices to v' of the edge e with length 0. Hence, from v we can reach v' of the edge e through a path passing through only sub-vertices in G with the integer lengths between l and u. For the simplicity of presentation we call vertices v and v' of edge e mother vertices and call the sub-vertices in e child vertices. Besides, all the paths joining v and v' that go through only child vertices of e (in $\mathcal{RG}$) are also called paths belongs to e.

Figure 2 gives an example how to build graph G' from simple graph G with 2 edges.

In order to make use of G, we have to show that G is compatible to $\mathcal{RG}$ w.r.t checking LDI. First, we define length, cost and satisfaction of a path p in G w.r.t LDI $\mathcal{D}$.

Definition 9. *Let $\mathsf{p} = \mathsf{v}_1\mathsf{v}_2\dots\mathsf{v}_m$ be a path in G. The length $l(\mathsf{p})$ and the cost $\theta(\mathsf{p})$ of p are defined as*

$$l(\mathsf{p}) \mathrel{\widehat{=}} \sum_{i=1}^{m-1} \omega(\mathsf{v}_i, \mathsf{v}_{i+1}), \quad \theta(\mathsf{p}) \mathrel{\widehat{=}} \sum_{i=1}^{m-1} \omega(\mathsf{v}_i)\omega(\mathsf{v}_i, \mathsf{v}_{i+1}).$$

A path p satisfies $\mathcal{D}$ iff $A \leq l(\mathsf{p}) \leq B \Rightarrow \theta(\mathsf{p}) \leq M$.

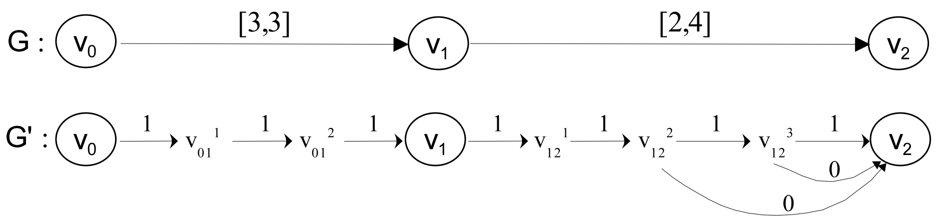

Fig. 2. "Discretising" graph of region graph

Lemma 8. *Each integral weighted interpretation $\wp = (\mathsf{p}, \overline{d})$ of $\mathcal{RG}$ corresponds to a path p' in G such that $l(\wp) = l(\mathsf{p}')$, $\theta(\wp) = \theta(\mathsf{p}')$, and reversely, each path p' in G corresponds to an integral weighted interpretation $\wp = (\mathsf{p}, \overline{d})$ of $\mathcal{RG}$ such that $l(\wp) = l(\mathsf{p}')$, $\theta(\wp) = \theta(\mathsf{p}')$.*

Proof. It is obvious from the definition of G that for each edge $(\mathsf{v}_i, \mathsf{v}_j, [\mathsf{l}, \mathsf{u}])$ of $\mathcal{RG}$ and an integer $d \in [\mathsf{l}, \mathsf{u}]$ there exists a path $\mathsf{p} = \mathsf{v}_i \mathsf{v}_{ij}^1 \ldots \mathsf{v}_{ij}^d \mathsf{v}_j$ of G such that $l(\mathsf{p}) = d$, $\theta(\mathsf{p}) = c_{\mathsf{v}_i} d$ and vice-versa. Hence, the lemma is correct.

From Lemma 8 we can conclude that if there exists an integral model $\sigma \in \mathcal{M}_{uv}(\mathcal{A})$ not satisfying LDI $\mathcal{D}$ then there exists a path that joins mother vertices of G and does not satisfy LDI and reversely. A similar result for any integral model of $\mathcal{A}$ and any path (joining two child vertices) of G is formulated by following lemma.

Lemma 9. *Given a timed automaton $\mathcal{A}$, a LDI $\mathcal{D}$ and weighted graph G as above. Then if there exists a path $\mathsf{p} \in \mathcal{P}(\mathsf{G})$ such that $\mathsf{p} \not\models LDI$ then there exists an integral model $\sigma \in \mathcal{M}_I(\mathcal{A})$ such that $\sigma \not\models LDI$ and vice-versa.*

Proof. See [10].

Theorem 2. *Checking the satisfaction of LDI $\mathcal{D}$ by timed automaton $\mathcal{A}$ is equivalent to checking the satisfaction of LDI $\mathcal{D}$ by the set of paths $\mathcal{P}(\mathsf{G})$. That is, $\mathcal{A} \models \mathcal{D}$ if and only if $\mathcal{P}(\mathsf{G}) \models \mathcal{D}$.*

This theorem follows immediately from Lemma 9 and the discretisability of LDI w.r.t timed automata $\mathcal{A}$.

4.4 Algorithm for Checking LDI

In this section we present the idea an algorithm for checking $\mathcal{A} \models \mathcal{D}$ based on traversing the weighted graph G. The algorithm uses procedure **Traverse**(vstart) and procedure **Checking-LDI**. The procedure **Traverse**(vstart) explores every path starting from fixed vertex vstart to see if it satisfies $\mathcal{D}$, and the procedure **Checking-LDI** calls procedure **Traverse**(vstart) for all vertices vstart $\in \mathsf{V}$ for deciding satisfaction of $\mathcal{D}$ by $\mathcal{A}$.

The procedure **Traverse**(vstart) uses the backtracking technique to explore the graph. Starting from vertex vstart of G, the procedure constructs the current path p ($l(\mathsf{p})$ and $\theta(\mathsf{p})$ initialised to 0) while going along out-going edges to their destination vertices at which $l(\mathsf{p})$ and $\theta(\mathsf{p})$ are re-calculated and $A \leq l(\mathsf{p}) \leq B \Rightarrow \theta(\mathsf{p}) < M$ is verified. The procedure goes back when the current path p cannot be expanded (see the next paragraph how a path can be expanded) or the length of current path exceeds B, and terminates when either $A \leq l(\mathsf{p}) \leq B \Rightarrow \theta(\mathsf{p}) < M$ is violated or it goes back to the starting vertex vstart with no more out-going edge to go.

The current path p cannot be expanded when there is no new out-going edge to go when the number of repetitions of cycles has reached the limit. This applies only when $B = \infty$. When a positive cycle is discovered, i.e. a cycle p' (which is a sub-path of p) with $\theta(p') > 0$, the procedure returns $\mathcal{A} \not\models \mathcal{D}$. When there is no positive cycle in p, p cannot be expanded when the number of cycle repetitions has reached $k = \lceil \frac{A-c}{\triangle c} \rceil$, where c is the length of the shortest cycles in p. Any more repetition of a cycle will make $\theta(\mathsf{p})$ smaller. So, there is no need to check with expansion of p by more repetitions of cycles.

So, either there is a positive cycle in G, or eventually, either p will become not expandable, or $l(\mathsf{p}) > B$ will be reached for a path p starting from vstart. So, the procedure **Traverse**(vstart) will terminate eventually.

The detailed technical construction of the algorithm can be worked out easily, and is omitted here.

5 Conclusion

Exploring reachability graphs is one of popular methods for checking reachability property and some properties concerning time instants of real time systems. However, paths in the reachability graph do not preserve time durations of system locations, and hence, cannot be used for checking duration properties. By equipping edges of integral reachability graphs with the minimal and maximal bounds of state transitions, we are able to use this technique for checking duration properties. We have proposed an algorithm for checking LDI of timed automata using this technique. In this paper we have proved the discretisability of LDI, and proposed an algorithm based on this result to check if a timed automaton satisfies a LDI in the general semantics. Although the complexity of this algorithm is high, it can serve, at least, for showing that checking LDI of closed time automata is decidable. We do believe that checking is feasible for some specific LDI and abstract timed automata.

References

1. R. Alur and D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, pp. 183–235, 1994.
2. R. Alur. Timed Automata. Proceedings of *11th International Conference on Computer-Aided Verification,* LNCS 1633, pp. 8–22, Springer-Verlag, 1999.

3. Victor A. Braberman and Dang Van Hung. On Checking Timed Automata for Linear Duration Invariants. Technical Report 135, UNU/IIST, P.O.Box 3058, Macau, February 1998. Proceedings of *the 19th Real-Time Systems Symposium RTSS'98*, December 2–4, 1998, Madrid, Spain, IEEE Computer Society Press 1998, pp. 264–273.
4. E. Clarke, O. Grumberg, and D. Peled. *Model Checking.* The MIT Press, Cambridge, Massachusetts, 1999.
5. 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.
6. Thomas A. Henzinger, Zohar Manna and Amir Pneuli. Towards Refining Temporal Specifications into Hybrid Systems, Hybrid Systems I, LNCS 736,Springer-Verlag, 1993.
7. 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.
8. Li Yong and 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, Nov. 2002. pp. 689 – 698.
9. 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. Proceedings of the *International Symposium on Software Engineering for Parallel and Distributed Systems* (PDSE'98), Kyoto, Japan, 20 – 21, April 1998. Bernd Kramer, Naoshi Uchihira, Peter Croll and Stefano Russo (Eds). IEEE Press 1998, pp. 61 – 71.
10. Pham Hong Thai and Dang Van Hung. Verifying Linear Duration Constraints of Timed Automata. Technical Report 306, UNU/IIST, P.O.Box 3058, Macau, June 2004.
11. S. Tripakis, S. Yovine. Analysis of timed systems based on time-abstracting bisimulations. *Formal Methods in System Design*, 18, 25–68, 2001. Kluwer Academic Publishers, Boston.
12. 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.
13. Zhou Chaochen , C.A.R. Hoare, and Anders P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269–276, 1991.
14. 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.
15. Zhou Chaochen and Hansen M. R., *Duration Calculus. A Formal Approach to Real-Time Systems*, Springer, 2004.