Updatable Timed Automata with Additive and Diagonal Constraints

Lakshmi Manasa, Shankara Narayanan Krishna, and Kumar Nagaraj

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay,
Powai, Mumbai, India 400 076
{manasa,krishnas,kumar}@cse.iitb.ac.in

Abstract. Timed automata [1] are a well established theoretical framework used for modeling real time systems. In this paper, we introduce a class of timed automata with updates of the form $x :\sim d - y$. We show that these updates preserve the decidability of emptiness of classical timed automata with diagonal constraints for $\sim \in \{=, <, \leq\}$, while emptiness is undecidable for $\sim \in \{>, \geq, \neq\}$. When used along with additive constraints, these updates give decidable emptiness with 2 or lesser number of clocks, and become undecidable for 3 clocks. This provides a partial solution to a long standing open problem [5].

1 Introduction

Timed automata, introduced in [1] are a well established model for the verification of real time systems. Since their introduction, several variants of timed automata have been studied. Some of them are : [4], wherein new operations on clock updates were considered, [5] where new kinds of guards were introduced, [7] where a set of clocks could be freezed, and [6], where silent transitions were studied. A very interesting result in the theory of timed automata is that the emptiness problem is decidable. This has paved the way for using timed automata in verification and many tools viz., UPPAAL [8], KRONOS [9] and HyTech [2] were built using this.

In this paper, we consider a variant of timed automata having new update operations. The updates we consider are of the form $x :\sim d - y$, $\sim \in \{<, \leq, =, >, \geq, \neq\}$. It has been shown [4] that updating a clock based on another clock's value ($x :\sim y, x :\sim y + c$) is very powerful and often leads to undecidability of emptiness. We show here that updates of the kind $x :\sim d - y$ $\sim \in \{=, \leq, <\}$ preserve the decidability, and thus behave very differently from the known set of updates. However, when these kinds of updates are used along with additive constraints, emptiness remains decidable only when the number of clocks used are two or less. In fact, our proof uses only deterministic updates $x := d - y$. This also gives a partial solution to the problem of deciding emptiness for the class of timed automata with additive constraints and 3 clocks [5].

It is well known that the expressive power of classical timed automata [1] does not change when location invariants ($x \sim c$) are used. It can be shown that using constraints $x - y \sim c$ as location invariants and guards, emptiness remains decidable for the class of timed automata using $x :< c$ as updates ([4] has proved this without using location invariants; however, adapting the proof to the case of using $x - y \sim c$ as location invariants is not hard). However, if we allow additive constraints $x + y = c$ as location invariants and guards, then emptiness becomes undecidable when used along with updates $x :< c$, in the presence of 3 or more clocks.

2 Prerequisites

For any set S, S^* (S^ω) denotes the set of all finite (infinite) strings over S. $S^\infty = S^* \cup S^\omega$. We consider as time domain $\mathbf{T}$ the set $\mathbf{Q}^+$ or $\mathbf{R}^+$ of non-negative rationals or reals, and Σ a finite set of actions. A time sequence over $\mathbf{T}$ is a finite (infinite) non-decreasing sequence $\tau = (t_i)_{i \geq 1}$; for simplicity t_0 is taken to be zero always. A timed word over Σ is defined as $\rho = (\sigma, \tau)$, where $\sigma = (\sigma_i)_{i \geq 1}$ is a finite (infinite) sequence of symbols in Σ and $\tau = (t_i)_{i \geq 1}$ is a finite (infinite) sequence in $\mathbf{T}^\infty$. A *timed language* L is a set of timed words.

We consider a finite set of variables X called *clocks*. A clock valuation over X is a map $\nu : X \rightarrow \mathbf{T}$ mapping each clock $x \in X$ to a time value. $\nu(x)$ represents the value assigned to the clock x by ν and $frac(\nu(x))$ represents the fractional part of the value $\nu(x)$. For $t \in \mathbf{T}$, the valuation $\nu + t$ is defined as $(\nu + t)(x) = \nu(x) + t, \forall x \in X$. The set of all clock valuations over X is denoted by $\mathbf{T}^X$.

For the set of clocks X, the set of constraints (guards) over X, denoted by $C_-(X)$ is given as follows: $\varphi ::= x \sim c | x - y \sim c | \varphi \wedge \varphi | \varphi \vee \varphi$ where $x, y \in X, c \in \mathbf{Q}^+$ and $\sim \in \{<, \leq, >, \geq, =, \neq\}$. Constraints $x - y \sim c$ are called *diagonal constraints*. Clock constraints are interpreted over clock valuations. The relation $\nu \models \varphi$ (valuation ν satisfies constraint φ) is defined as follows: $\nu \models x \sim c$ if $\nu(x) \sim c$, $\nu \models x - y \sim c$ if $\nu(x) - \nu(y) \sim c$ and so on.

Clock constraints allow us to test the values of clocks. To change the value of a clock x we use a *simple update* of the form $up_x :: x := 0$. $U_0(X)$ denotes the set of updates $up \in U_0(X)$ defined as $up = \bigwedge_{x \in X} up_x$ where up_x are *simple updates*. These updates are functions from $\mathbf{T}^X$ to $\mathcal{P}(\mathbf{T}^X)$.

Let ν be a valuation and let up_z be a simple update over clock z. A valuation ν' is in $up_z(\nu)$ if $\nu'(y) = \nu(y), y \neq z$ and $\nu'(z) = 0$. For two valuations $\nu, \nu' \in \mathbf{T}^X$, $\nu' \in up(\nu)$ if for every $x \in X$, ν' coincides with $\nu'' \in up_x(\nu)$ over the value of x.

2.1 Timed Automata

A *timed automaton* [1] is a tuple $\mathcal{A} = (L, L_0, \Sigma, X, E, F)$ where L is a finite set of locations; $L_0 \subseteq L$ is a set of initial locations; Σ is a finite set of symbols; X is a finite set of clocks; $E \subseteq L \times L \times \Sigma \times C_-(X) \times U_0(X)$ is the set of transitions and $F \subseteq L$ is a set of final locations. $C_-(X)$ and $U_0(X)$ are the set of clock constraints and clock updates as described above. An edge $e = (l, l', a, \varphi, \phi)$ represents a transition from l to l' on symbol a, with the valuation $\nu \in \mathbf{T}^X$ satisfying the guard φ, and then ϕ gives the updates of certain clocks.

A path is a finite (infinite) sequence of consecutive transitions. The path is said to be accepting if it starts in an initial location ($l_0 \in L_0$) and ends in a final location (or repeats a final location infinitely often). A run through a path from a valuation ν'_0 (with $\nu'_0(x) = 0$ for all x) is a sequence $(l_0, \nu'_0) \xrightarrow{t_1} (l_0, \nu_1) \overset{(\sigma_1,\varphi_1,\phi_1)}{\longrightarrow} (l_1, \nu'_1) \xrightarrow{t_2} (l_1, \nu_2) \overset{(\sigma_2,\varphi_2,\phi_2)}{\longrightarrow} (l_2, \nu'_2) \cdots (l_n, \nu'_n)$. Note that $\nu_i = \nu'_{i-1} + (t_i - t_{i-1}), \nu_i \models \varphi_i$, and that $\nu'_i \in up(\nu_i), i \geq 1$. A timed word ρ is accepted by $\mathcal{A}$ iff there exists an accepting run (through an accepting path) over $\mathcal{A}$, the word corresponding to which is ρ. The timed language $L(\mathcal{A})$ accepted by $\mathcal{A}$ is defined as the set of all timed words accepted by $\mathcal{A}$.

2.2 Region Automata

Given a set X of clocks, let $\mathcal{R}$ be a finite partitioning of $\mathbf{T}^X$. Each partition contains a set (possibly infinite) of clock valuations. Given $\alpha \in \mathcal{R}$, the successors of α represented by $Succ(\alpha)$ are defined as

$$\alpha' \in Succ(\alpha) \text{ if } \exists \nu \in \alpha, \exists t \in \mathbf{T} \text{ such that } \nu + t \in \alpha'$$

The finite partition $\mathcal{R}$ is said to be a *set of regions* iff

$$\alpha' \in Succ(\alpha) \iff \forall \nu \in \alpha, \exists t \in \mathbf{T} \text{ such that } \nu + t \in \alpha'.$$

A set of regions is consistent with time elapse if two valuations which are equivalent (within the same partition) stay equivalent with time elapse. A region $\alpha \in \mathcal{R}$ is said to satisfy a clock constraint $\varphi \in C_-(X)$ denoted as $\alpha \models \varphi$, if $\forall \nu \in \alpha, \nu \models \varphi$. A clock update $\phi \in U_0(X)$ maps a region α to a set of regions $\phi(\alpha) = \{\alpha' \mid \alpha' \cap \phi(\nu) \neq \emptyset \text{ for some } \nu \in \alpha\}$.

A set of regions $\mathcal{R}$ is said to be *compatible* with a finite set of clock constraints $C_-(X)$ iff $\forall\ \varphi \in C_-(X)$ and $\forall \alpha \in \mathcal{R}$ either $\alpha \models \varphi$ or $\alpha \models \neg\varphi$. A set of regions $\mathcal{R}$ is said to be *compatible* with a finite set of clock updates $U_0(X)$ iff $\alpha' \in \phi(\alpha) \Rightarrow \forall \nu \in \alpha, \exists \nu' \in \alpha'$ such that $\nu' \in \phi(\nu)$.

Given a timed automaton $\mathcal{A}$, and a set of regions $\mathcal{R}$ compatible with $C_-(X)$ and $U_0(X)$, the *region automaton* $\mathcal{R}(\mathcal{A}) = (Q, Q_0, \Sigma, E', F')$ is defined as follows: $Q = L \times \mathcal{R}$ the set of locations; $Q_0 = L_0 \times \mathcal{R} \subseteq Q$ the set of initial locations; $F' = F \times \mathcal{R} \subseteq Q$ the set of final locations; $E' \subseteq (Q \times \Sigma \times Q)$ is the set of edges. $(l, \alpha) \xrightarrow{a} (l', \alpha')$ if $\exists \alpha'' \in \mathcal{R}$ and a transition $(l, l', a, \varphi, \phi) \in E$ such that (a) $\alpha'' \in Succ(\alpha)$, (b) $\alpha'' \models \varphi$ and (c) $\alpha' \in \phi(\alpha'')$.

The region automaton is an abstraction of the timed automaton accepting Untime($L(\mathcal{A})$) [1]. As a consequence, the following theorem was proved [1].

Theorem 1. *Let $\mathcal{A}$ be a timed automaton. Then the problem of checking emptiness of $L(\mathcal{A})$ is decidable.*

We say that a class of timed automata is *decidable* if there exists a *decider* for the emptiness problem.

3 Diagonal Constraints

In this section, we consider a class of timed automata called *updatable timed automata with diagonal constraints*. This class of automata has $C_-(X)$ as the set of clock constraints. The set of updates of this class of automata denoted by $U_{\leq}(X)$ are defined as $up = \bigwedge_{x \in X} up_x$ where up_x are *simple updates* as follows: $up_x ::= x := 0 \mid x :\sim d - y$, where $x, y \in X, d \in \mathbf{Q}^+$ and $\sim \in \{=, <, \leq\}$. Here, $d \leq c^{max}$ where $c^{max} = max\{c \mid x \sim c \in C_-(X)$ or $x - y \sim c \in C_-(X)\}$. Accordingly, we have $\nu' \in up_z(\nu)$ if $\nu'(y) = \nu(y), y \neq z$ and $\nu'(z) = 0$ if $up_z ::= z := 0$ and $\nu'(z) \sim d - \nu(y) \wedge \nu'(z) \geq 0$ if $up_z ::= z :\sim d - y$.

For every clock $x \in X$, define a constant $c_x^{max} = c^{max} + d_x^{max}$ where $d_x^{max} = max\{c \mid x - y \sim c \in C_-(X) \wedge y \in X\}$, and a set of intervals $\mathcal{I}_x$ as

$$\mathcal{I}_x = \{[c] | 0 \leq c \leq c_x^{max}\} \cup \{(c, c+1) | 0 \leq c < c_x^{max}\} \cup \{(c_x^{max}, \infty)\}.$$

For every pair of clocks $x, y \in X$, define $D_{xy} = max\{c \mid x - y \sim c \in C_-(X)\}$, and the set of intervals $\mathcal{J}_{xy}$ as

$\mathcal{J}_{xy} = \{(-\infty, -d_{yx})\} \cup \{[d]\} \cup \{(d', d'+1)\} \cup \{(d_{xy}, +\infty)\}$ where $-d_{yx} \leq d \leq d_{xy}$ and $-d_{yx} \leq d' < d_{xy}$.

Let α be a tuple $((I_x)_{x \in X}, (J_{xy})_{x,y \in X}, \prec)$ where

1. $I_x \in \mathcal{I}_x$, $J_{xy} \in \mathcal{J}_{xy}$, and
2. $\prec$ is a total preorder on $X_0 = \{x \in X \mid I_x$ is of the form $(c, c+1)\}$.

α defines the following subset of $\mathbf{T}^X$: valuations $\nu \in \mathbf{T}^X$ such that $\nu(x) \in I_x$ for all $x \in X$, $\nu(x) - \nu(y) \in J_{xy}$ for all $x, y \in X$, and $frac(\nu(x)) \leq frac(\nu(y))$ if $x \prec y$ for all $x, y \in X_0$. The set of all such tuples α partitions $\mathbf{T}^X$ and is represented by $\mathcal{R}^-$.

Remark 1. When using updates $x :\sim d - y$, the value $\nu'(x)$ must be non-negative, as mentioned above. This can be ensured by adding a guard $y \leq d$? while making the transition.

The following are easy to observe.

Lemma 1. *Set $\mathcal{R}^-$ forms a set of regions.*

Proof. Let $\alpha = ((I_x)_{x \in X}, (J_{xy})_{x,y \in X}, \prec) \in \mathcal{R}^-$. If for all x, $I_x = (c_x^{max}, \infty)$, then $Succ(\alpha) = \{\alpha\}$ as time elapse would not change J_{xy}. If there is atleast one I_x such that $I_x \neq (c_x^{max}, \infty)$, then there exists a region $\alpha' \neq \alpha$ such that $\alpha' \in Succ(\alpha)$. We define the closest successor of α to be α' such that $\forall \nu \in \alpha, \forall t \in \mathbf{T}$, if $\nu + t \notin \alpha$, then $\exists t' \leq t$ such that $\nu + t' \in \alpha'$.

Let $Z = \{x \in X \mid I_x = [c]\}$. In this case, $I'_x = (c, c+1)$ for $x \in Z$ and $c < c_x^{max}$; $I'_x = (c_x^{max}, \infty)$ for $x \in Z$ and $c = c_x^{max}$ while $I'_x = I_x$ for $x \notin Z$. $x \prec' y$ if $x \prec y$ or if $I_x = [c], c < c_x^{max}$ and $I_y = (d, d+1)$. $J'_{xy} = J_{xy}$ for all x, y.

In case $Z = \emptyset$, then pick the clock(s) $x \in X_0$ having the maximal fractional part. For these, $I'_x = [c+1]$ if $I_x = (c, c+1), c < c_x^{max}$, and for the rest of clocks $y \in X$, $I'_y = I_y$. Here, $\prec'$ is the restriction of $\prec$ to clocks x such that $I'_x = (d, d+1)$. Again, $J'_{xy} = J_{xy}$ for all x, y.

It is easy to see that for all $\nu \in \alpha$, there exists a $t \in \mathbf{T}$ such that $\nu + t \in \alpha'$. $\square$

Lemma 2. *The set of regions $\mathcal{R}^-$ is compatible with the set of clock constraints $C_-(X)$.*

Proof. It is easy to see that with the choice of regions $\mathcal{R}^-$, any $\alpha \in \mathcal{R}^-$ satisfies either φ or $\neg\varphi$. □

Lemma 3. *The set of regions $\mathcal{R}^-$ is compatible with any simple update $z := 0$ or $z :\sim d - y$, $\sim \in \{=, <, \leq\}$.*

Proof. Let $\alpha = ((I_x)_{x \in X}, (J_{xy})_{x,y \in X}, \prec)$ be a region in $\mathcal{R}^-$ where $\prec$ is a total preorder on $X_0 = \{x \mid I_x \text{ is of the form } (c, c+1)\}$. Let up_z be a simple update over clock $z \in X$. Let $\alpha' = ((I'_x)_{x \in X}, (J'_{xy})_{x,y \in X}, \prec')$. Then, $\alpha' \in up_z(\alpha)$ if $I'_x = I_x \forall x \in X \backslash \{z\}$, $J'_{xy} = J_{xy} \forall x, y \in X \backslash \{z\}$, and

1. $\phi = z := 0$
   - $I'_z = [0]$, $X'_0 = X_0 \backslash \{z\}$, $\prec' = \prec \cap (X'_0 \times X'_0)$,
   - $J'_{xz} = \left\{ \begin{array}{lll} [c] & \text{if } I_x = [c] & \text{and } c \leq d_{xz}, \\ (c, c+1) & \text{if } I_x = (c, c+1) & \text{and } c < d_{xz}, \\ (d_{xz}, \infty) & \text{otherwise} & \end{array} \right\}$
   - $J'_{zx} = \left\{ \begin{array}{lll} -[c] & \text{if } I_x = [c] & \text{and } c \leq d_{xz}, \\ (-c-1, -c) & \text{if } I_x = (c, c+1) & \text{and } c < d_{xz}, \\ (-\infty, -d_{xz}) & \text{otherwise} & \end{array} \right\}$
2. $\phi = z := d - y$
   Since $d - y \geq 0 \wedge d \leq c^{max}$, $I_y \neq (c_y^{max}, \infty)$, $I'_z \neq (c_z^{max}, \infty)$.
   - If $I_y = [c] \wedge 0 \leq c \leq d \leq c^{max}$ then $I'_z = [d - c]$
   - If $I_y = (c, c+1) \wedge 0 \leq c < d \leq c^{max}$ then $I'_z = (d - c - 1, d - c)$

   If $I'_z = [d - c]$ and let $e = d - c$ then
   - $X'_0 = X_0 \backslash \{z\}$, $\prec' = \prec \cap (X'_0 \times X'_0)$
   - $J'_{xz} =$
     $\left\{ \begin{array}{ll} (-\infty, -d_{zx}) & \text{if } I_x = [c'] \text{ or } (c', c'+1) \text{ and } c' - e < -d_{zx}, \\ [c' - e] & \text{if } I_x = [c'] \text{ and } -d_{zx} \leq c' - e \leq d_{xz}, \\ (c' - e, c' - e + 1) & \text{if } I_x = (c', c'+1) \text{ and } -d_{zx} \leq c' - e < d_{xz}, \\ (d_{xz}, \infty) & \text{otherwise} \end{array} \right\}$
   - J'_{zx} can be calculated similarly.

   If $I'_z = (d - c - 1, d - c)$ then
   - $X'_0 = X_0 \cup \{z\}$, $\prec'$ is same as $\prec$ except that if $frac(v(y)) = 0.5$ then $y \prec' z \wedge z \prec' y$, if $frac(v(y)) < 0.5$ then $y \prec' z$. Otherwise, $z \prec' y$.
   - J'_{xz} and J'_{zx} calculated similar to the case $I'_z = [d - c]$.
3. $\phi = z :< d - y$
   Note that as $d - y > 0 \wedge d \leq c^{max}$, $I_y \neq (c_y^{max}, \infty)$ and also $I'_z \neq (c_z^{max}, \infty)$.
   - If $I_y = [e] \wedge 0 \leq e \leq d \leq c^{max}$ then $I'_z = [e'] \vee I'_z = (e', e'+1)$ such that $e' < d - e$. If $I'_z = (e', e'+1)$, then $X'_0 = X_0 \cup \{z\}$, and $\prec'$ is any total preorder that coincides with $\prec$ on $X'_0 \backslash \{z\}$. Otherwise, $X'_0 = X_0 \backslash \{z\}$ and $\prec' = \prec \cap (X'_0 \times X'_0)$.
   - If $I_y = (e, e+1) \wedge 0 \leq e < d \leq c^{max}$ then $I'_z = [e']$ such that $e' < d - e$. $X'_0 = X_0 \backslash \{z\}$ and $\prec' = \prec \cap (X'_0 \times X'_0)$, or $I'_z = (e'', e''+1)$ such that $e'' \leq d - e - 1$. If $I'_z \cap (d - I_y) = \emptyset$, then $\prec'$ is any total preorder that coincides with $\prec$ on $X'_0 \backslash \{z\}$. If $I'_z \cap (d - I_y) \neq \emptyset$, $\prec'$ coincides with $\prec$ on $X'_0 \backslash \{z\}$ and $z \prec' y$, $y \not\prec' z$.

   J'_{xz} and J'_{zx} can be calculated in a similar manner to the above case.
4. $\phi = z :\leq d - y$. This is similar to the above case.

It is easy to see that for any $\nu \in \alpha$, and any $\alpha' \in up_z(\alpha)$, there exists $\nu' \in \alpha' \cap up_z(\nu)$. □

Lemma 4. *The set of regions $\mathcal{R}^-$ is compatible with updates $U_{\leq}(X)$.*

Proof. Let $\alpha = ((I_x)_{x\in X}, (J_{xy})_{x,y\in X}, \prec)$ and $\alpha_z = ((I^z_x)_{x\in X}, (J^z_{xy})_{x,y\in X}, \prec_z)$ be two regions of $\mathcal{R}^-$ and up_z be a simple update such that $\alpha_z \in up_z(\alpha)$. Let ν be a valuation in α. By Lemma 3, $\mathcal{R}^-$ is compatible with up_z. Thus, for any valuation $\nu \in \alpha$, there exists some valuation $\nu_z \in up_z(\nu) \cap \alpha_z$. We need to show that for $up \in U_{\leq}(X)$, and region α, we can find $\alpha' = ((I'_x)_{x\in X}, (J'_{xy})_{x,y\in X}, \prec') \in up(\alpha)$ such that for any $\nu \in \alpha$, there exists $\nu' \in \alpha' \cap up(\nu)$.

Given $\nu \in \alpha$, define a valuation ν' as : (i) $\nu'(y) = \nu_y(y)$ for any clock y; (ii) for a pair (y, z) of clocks, calculate $\nu'(y) - \nu'(z)$ as in Lemma 3, and (iii) $\prec'$ can be calculated from $\prec_z, z \in X$ as follows: Let $X' = \{x' \mid x \in X\}$ be a copy of clocks in X. Define $\prec'_x$ from $\prec_x$ by replacing x with x'. $\prec'_x$ is a preorder on $X \cup X'$. Then $\prec'$ is obtained by taking the union of all $\prec'_x, x \in X$; restricting it to X'; and then replacing X' with X. Then $\nu' \in \alpha' \cap up(\nu)$.

Thus, from $\alpha_z \in up_z(\alpha), z \in X$, we can obtain $\alpha' \in up(\alpha)$ such that for any $\nu \in \alpha$, there exists $\nu' \in up(\nu) \cap \alpha'$. □

Theorem 2. *The class of updatable timed automata is decidable.*

Proof. Lemmas 1, 2, 3, 4 indicate that $\mathcal{R}^-$ forms a set of regions and is compatible with $C_-(X)$ and $U_{\leq}(X)$ and a region automaton can be constructed as explained in Section 2.1 and hence according to Theorem 1 this class is decidable. □

Remark 2. Note that if we consider updates $U_{\nleq}(X)$ of the form $up = \bigwedge_{x\in X} up_x$ with $up_x ::= x := 0 \mid x :\sim d - y$, where $x, y \in X, d \in \mathbf{Q}^+$ and $\sim \in \{\neq, >, \geq\}$, $d \leq c^{max}$, then emptiness is no longer decidable (The Undecidability result of section 4 in [4] can be modified [10]). Recall that allowing updates of the kind $x :< y$ or $x :> y$ or $x :\sim y + c$, $c \in \mathbf{Q}^+$, with constraints from $C_-(X)$, emptiness is not decidable [4]. It is interesting to note here that the updates $x :< d - y$ and $x :> d - y$ behave differently.

4 Additive Constraints

In this section, we consider *updatable timed automata with additive constraints.* The constraints $C_+(X)$ of the form $\varphi ::= x \sim c | x+y \sim c | \varphi \wedge \varphi | \varphi \vee \varphi$ where $x, y \in X, c \in \mathbf{Q}^+$ and $\sim \in \{<, \leq, >, \geq, =, \neq\}$. Constraints $x + y \sim c$ are called *additive constraints.* Timed automata with constraints $C_+(X)$ and updates $U_0(X)$ have been studied in [5]. It has been shown that with 2 or lesser number of clocks, emptiness is decidable, while 4 clocks gives undecidability. The case of 3 clocks has been open since.

In this section, we give a partial answer to this open problem by looking at the class of timed automata with constraints $C_+(X)$ and updates $\mathcal{U}_=(X)$ consisting of updates $up = \bigwedge_{x\in X} up_x$ where $up_x ::= x := 0 | x := d - y,\ x, y \in X, d \in \mathbf{Q}^+$. Here, $d \leq c^{max}$ where $c^{max} = max\{c \mid x \sim c \in C_+(X)$ or $x + y \sim c \in C_+(X)\}$.

We show that emptiness is undecidable if 3 or more clocks are used, while it is decidable with 2 or lesser number of clocks.

For every clock $x \in X$, define $c_x^{max} = c^{max}$. $\mathcal{I}_x$ is defined as done before. For every pair of clocks $x, y \in X$, define $i_{xy} = i_{yx} = max\{c \mid x + y \sim c \in \mathcal{C}_+(X)\}$ and the set of intervals $\mathcal{K}_{xy}$ as

$$\mathcal{K}_{xy} = \{[d] \mid 0 \leq d \leq i_{xy}\} \cup \{(d, d+1) \mid 0 \leq d < i_{xy}\} \cup \{(i_{xy}, +\infty)\}$$

Let α be a tuple $((I_x)_{x \in X}, (K_{xy})_{x,y \in X}, \prec)$ where $I_x \in \mathcal{I}_x, K_{xy} \in \mathcal{K}_{xy}$ and $\prec$ is a total preorder on $X_0 = \{x \in X \mid I_x \text{ is of the form } (c, c+1)\}$. The region associated with α is defined as the set of all valuations ν such that $\nu(x) \in I_x$ for all $x \in X$, $\nu(x) + \nu(y) \in K_{xy}$ and $frac(\nu(x)) \leq frac(\nu(y))$ if $x \prec y$, for all $x, y \in X_0$. Let $\mathcal{R}^+$ represent the set of all tuples α. $\mathcal{R}^+$ forms a finite partition of $\mathbf{T}^X$.

Theorem 3. *The class of timed automata with 2 clocks, updates $U_=(X)$ and clock constraints $C_+(X)$ are decidable.*

Proof. The proof of this theorem follows by the extension of the classical region construction as given above. It is easy to see that the partition $\mathcal{R}^+$ is a set of regions (consistent with time in the sense that from two equivalent valuations, the same set is reached as time progresses), consistent with the set of constraints (2 equivalent valuations satisfy the same constraints), and updates (updates from 2 equivalent valuations yield the same region). $\square$

Theorem 4. *The class of timed automata with 3 clocks, updates $U_=(X)$ and clock constraints $C_+(X)$ is undecidable.*

Proof. The proof lies in the simulation of a deterministic two counter machine. Such a machine consists of a finite sequence of labeled instructions which handle two counters i and j and end at a special instruction labeled *Halt*. The instructions are as follows:

1. $l_k : \; x = x + 1;$ *goto* l_{k+1};
2. $l_k : \; if \; x = 0$ *goto* l_{k+1} *else* $x = x - 1;$ *goto* l_{k+2}.

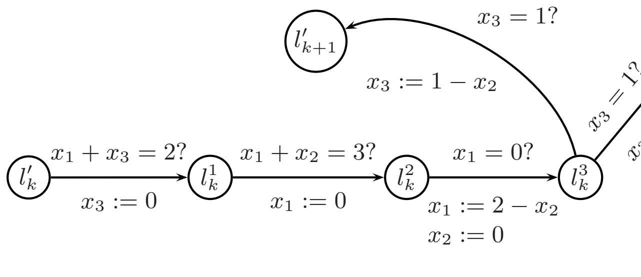
Without loss of generality, assume that the instructions are labeled $l_0, \ldots, l_n$ where $l_n = Halt$ and that in the initial configuration, both counters have value zero. Further, assume that the first instruction increments the counter i. The behaviour of the machine is described by a possibly infinite sequence of configurations $< l_0, 0, 0 >, < l_1, i_1, j_1 >, \cdots < l_k, i_k, j_k > \ldots$ where i_k and j_k are the respective counter values and l_k is the label of the kth instruction. The halting problem of such a machine has been shown to be undecidable [11].

A timed automaton $\mathcal{A}_\mathcal{M}$ with 3 clocks can be built which simulates a two counter machine $\mathcal{M}$ and reaches a final state iff $\mathcal{M}$ halts. Thus the halting problem of two counter machine is reduced to the problem of emptiness of $L(\mathcal{A}_\mathcal{M})$. The alphabet of $\mathcal{A}_\mathcal{M}$ is $\{*\}$, it has 3 clocks $\{x_1, x_2, x_3\}$ and the set of locations of $\mathcal{A}_\mathcal{M}$ is $Q = \{l'_i, l''_i \mid 0 \leq i \leq n\} \cup \{l_i^k \mid 1 \leq k \leq 7, 0 \leq i < n\}$. The set of final states is $F = \{l'_n, l''_n\}$. There are 5 basic modules : (i)an initialization module, (ii) module to increment counter i, (iii) module to check for zero and decrement counter i, (iv)module to increment counter j and (v)module to check for zero and decrement counter j. Initially, all clocks have value 0. At the end of the initialization module[1], $x_1 = x_2 = 0$ while $x_3 = 1$. The control shifts to the location l'_0, which represents the label of the first instruction. The values of the counters i and j are encoded in values of the clocks. The normal form for the clock values is as follows: $x_1 = \left(1 - \frac{1}{2^i}\right) + \left(1 - \frac{1}{2^j}\right), x_2 = \left(1 - \frac{1}{2^j}\right), x_3 = \frac{1}{2^j}$ or 0. Each module simulating instruction $l_i, i \geq 0$ has a unique initial location labeled by either l'_i or l''_i, and has two last locations labeled by l'_{i+1}, l''_{i+1}, where l_{i+1} is the next instruction to be simulated. At the beginning of each module, the clocks are assumed to be in normal form, and in the last location of each module, clocks will be in normal form. At location l''_{i+1} in each module, the value of x_3 will be necessarily zero, whereas at l'_{i+1} it will be $\frac{1}{2^j}$ where j represents the value of counter j at that point of time.

Incrementing counter i: The module in Fig. 1 simulates the instruction $l_k:\ i = i + 1;$ *goto* l_{k+1}. It can be seen that if the clock values are $x_1 = \left(1 - \frac{1}{2^i}\right) + \left(1 - \frac{1}{2^j}\right), x_2 = \left(1 - \frac{1}{2^j}\right)$ and $x_3 = \frac{1}{2^j}$ while entering this module, then at the end, we will have the value of x_1 to be $\left(1 - \frac{1}{2^{i+1}}\right) + \left(1 - \frac{1}{2^j}\right)$, x_2 remains unchanged, and x_3 will retain its earlier value at l'_{k+1} and is reset to zero at l''_{k+1}.

Incrementing counter j: Fig. 2 depicts the module for incrementing counter j. At the initial state of this module, x_3 must be zero. It is clear that after the transition, $x_2 = \left(1 - \frac{1}{2^{j+1}}\right)$, $x_1 = \left(1 - \frac{1}{2^i}\right) + \left(1 - \frac{1}{2^{j+1}}\right)$ and $x_3 = \frac{1}{2^{j+1}}$ or $x_3 = 0$ depending on the location.

Decrementing counter i: The module in figure 3 decrements counter i. The module begins in location l''_k. The control reaches any of l'_{k+2} or l''_{k+2} only if value of counter i is 0. If counter i is non-zero, then the location l^2_k is reached. It can be seen that at l'_{k+1}, the value of x_2 will be unchanged, x_3 will be $1 - x_2$ and x_1 would be $\left(1 - \frac{1}{2^{i-1}}\right) + \left(1 - \frac{1}{2^j}\right)$. Similar is the case of l''_{k+1} ($x_3 = 0$ here).



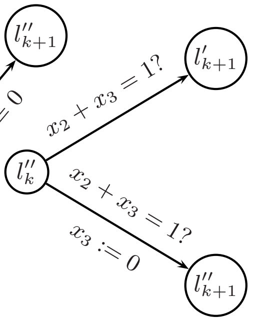


Fig. 1. Increment counter i

Fig. 2. Increment j

[1] The initialization module can be easily constructed by having a timed automaton with two locations with the constraint $x_3 = 1?$ on the transition between them. Clocks x_1 and x_2 are reset on transition.

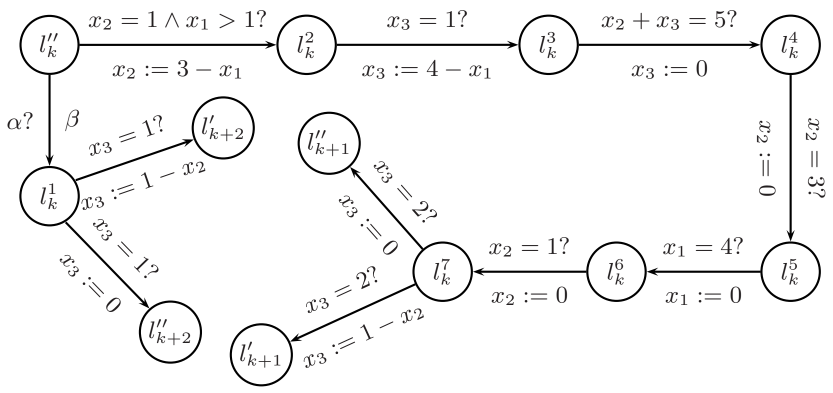


Fig. 3. Dec. counter i: $\alpha = (x_1 = 1 \wedge x_2 = 1)$ and $\beta = (x_1 := 0, x_2 := 0)$

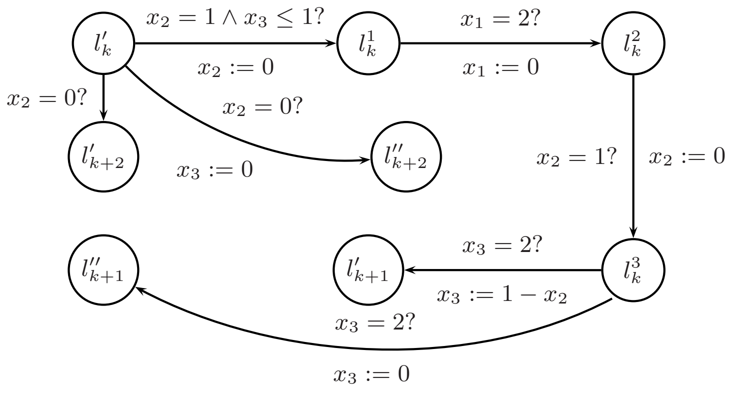


Fig. 4. Module to decrement counter j

Decrementing counter j: The module to decrement counter j is shown in Fig. 4. The value of x_3 is $\frac{1}{2^j}$ at the start of the module.

The simulation of the 2-counter machine is done as follows: The automaton $\mathcal{A}_\mathcal{M}$ starts in location q_0 with all clocks initialized to zero. When $x_3 = 1$, the control goes to l'_0, resetting x_1, x_2 to zero. Each instruction is implemented by the corresponding module. $\mathcal{A}_\mathcal{M}$ is built according to the sequence of instructions of the 2-counter machine. The different modules are linked by choosing the appropriate end state of a module to be the initial state of the next module. Modules can be linked this way until the *Halt* instruction is encountered. The set of final states is $\{l'_n, l''_n\}$. The language accepted by $\mathcal{A}_\mathcal{M}$ is empty iff $\mathcal{M}$ does not halt, which concludes the proof. $\square$

Remark 3. It should be noted that Theorems 3,4 are in agreement with [3] wherein it has been shown that reachability is decidable for dynamical systems with piecewise-constant derivatives in the case of 2 dimensions, while it is not for higher dimensions. Even though our systems allow much simpler guards and have a constant rate of evolution, they differ from [3] due to updates.

5 Location Invariants

It is well known that adding location invariants of the form $x \sim c$ does not increase the expressive power of classical timed automata. It is easy to see that the same holds good even when we allow as location invariants the elements of $C_-(X)$ or $C_+(X)$ (invariants can be removed and the same condition appended to all outgoing transitions [10]). [4] has considered a class of timed automata with $U_1(X)$ with constraints from $C_-(X)$ and proved that this class is decidable. This result will hold even when we allow elements of $C_-(X)$ as constraints and location invariants (region automtaton can include loc. invariants [10]). However, note that for $U_1(X)$ along with location invariants and constraints from $C_+(X)$, the decidability will hold good only when the number of clocks is 2 or less. The undecidability result given in Theorem 4 can easily be changed to prove this (replacing update $x := d - y$ with $x :< d + 1$ followed by the loc.invariant $x + y = d$ [10]).

References

1. Alur, R., Dill, D.L.: A Theory of Timed Automata. Theoretical Computer Science 126(2) (1994)
2. Alur, R., Henzinger, T.A., Ho, P.-H.: Automatic Symbolic Verification of Embedded Systems. IEEE Transactions on Software Engineering 22, 181–201 (1996)
3. Asarin, E., Maler, O., Pnueli, A.: Reachability analysis of dynamical sytems having piecewise-constant derivatives. Theoretical Computer Science 138, 35–65 (1995)
4. Bouyer, P., Duford, C., Fleury, E., Petit, A.: Updatable Timed Automata. Theoretical Computer Science 321(2-3), 291–345 (2004)
5. Bérard, B., Duford, C.: Timed automata and additive clock constraints. Information Processing Letters 75(1-2), 1–7 (2000)
6. Bérard, B., Gastin, P., Petit, A.: On the power of non observable actions in timed automata. In: Puech, C., Reischuk, R. (eds.) STACS 1996. LNCS, vol. 1046, pp. 257–268. Springer, Heidelberg (1996)
7. Demichelis, F., Zielonka, W.: Controlled Timed Automata. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 455–469. Springer, Heidelberg (1998)
8. Behrmann, G., David, A., Larsen, K.G., Mller, O., Pettersson, P., Yi, W.: Uppaal - Present and Future. In: Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, Florida, USA, December 4-7, 2001 (2001)
9. Daws, C., Olivero, A., Tripakis, S., Yovine, S.: The Tool KRONOS, Hybrid Systems, pp. 208–219 (1995)
10. Lakshmi Manasa, G., Krishna, S.N., Nagaraj, K.: Updatable Timed automata, Technical Report, `http://www.cse.iitb.ac.in/krishnas/uta.pdf`
11. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, USA (1967)