

# On Reachability Analysis of Updatable Timed Automata with One Updatable Clock

Yunqing Wen<sup>1</sup>, Guoqiang Li<sup>1(✉)</sup>, and Shoji Yuen<sup>2</sup>

<sup>1</sup> School of Software, Shanghai Jiao Tong University, Shanghai, China  
{wyqwyq,li.g}@sjtu.edu.cn

<sup>2</sup> Graduate School of Information Science, Nagoya University, Nagoya, Japan  
yuen@is.nagoya-u.ac.jp

**Abstract.** As an extension of *Timed Automata (TAs)*, *Updatable Timed Automata (UTAs)* proposed by Bouyer et al. have the ability to update clocks in a more elaborate way than simply reset them to zero. The reachability of general UTAs is undecidable, by regarding a pair of updatable clocks as counters updatable with incrementation and decrementation operations. This paper investigates the model of subclass of UTAs by restricting the number of updateable clocks. It is shown that the reachability of *UTAs with one updatable clock (UTA1s)* under diagonal-free constraints is decidable. The decidability is proved by treating a region of a UTA1 as an unbounded digiword, and encoding sets of digiwords that are accepted by a pushdown system where regions are generated on-the-fly on the stack.

## 1 Introduction

*Timed Automata (TAs)*, introduced by Alur and Dill [1], are one of the most-studied and most-established models for real-time systems. Lots of work has been devoted to extensions of timed automata, with much interest for classes whose emptiness problem remains decidable.

*Updatable Timed Automata (UTAs)* [2,3] are extensions of timed automata, based on the possibility to update the clocks in an elaborate way such as increment and decrement operations and assignments to arbitrary values. Their decidability have been investigated in [4], and also a quite precise way the thin frontier between decidable and undecidable classes of updatable timed automata has been described. The undecidability is technically shown to simulate the Minsky machine in general, while the decidability is shown by the fact that the constructed regions are finitely many. Our motivation is to investigate an interesting subclass of UTAs existing between them.

This paper gives a positive answer for the reachability of *UTAs with one updatable clock (UTA1s)* with diagonal-free time constraints by constructing regions on-the-fly over the stack of *pushdown systems (PDSs)*. Our result expands the thin frontier between decidable and undecidable classes of updatable timed automata. The model can be effectively used into soft real-time system modelling and analysis, where the updatable clock is used to depict relative

deadline, which can be flexibly modified according to different conditions and environments.

The rest of the paper is organized as follows. Section 2 gives an introduction of UTAs and PDSs. Section 3 introduces UTAs with one updatable clock. Section 4 introduces the notion of *digiwords*, which is equivalent to region [1], but provides us a more concise description. Section 5 shows the decidable reachability of UTAs by encoding them to PDSs, based on the notion of digiwords. The related work is presented in Sects. 6 and 7 concludes the paper.

## 2 Preliminaries

For finite words  $w = \gamma_1\gamma_2\dots\gamma_n$ , we denote  $\gamma_i \in w$  for  $0 \leq i \leq n$ .

Let  $\mathbb{R}^{\geq 0}$  and  $\mathbb{N}$  denote the sets of non-negative real numbers and natural numbers respectively. Let  $\mathbb{N}^\omega = \mathbb{N} \cup \{\omega\}$ , where  $\omega$  is the first limit ordinal. Let  $\mathcal{I}$  denote the set of *intervals* over  $\mathbb{N}^\omega$ . An interval can be written as a pair of a lower limit and an upper limit in the form of either  $(a, b)$ ,  $[a, b)$ ,  $[a, c]$ ,  $(a, c]$ , where  $a, c \in \mathbb{N}$ ,  $b \in \mathbb{N}^\omega$ , ‘(’ and ‘)’ denote open limits, and ‘[’ and ‘]’ denote closed limits. For a number  $r \in \mathbb{R}^{\geq 0}$  and an interval  $I \in \mathcal{I}$ , we use  $r \in I$  to denote that  $r$  belongs to  $I$ . Let  $I \setminus I' = \{r \mid r \in I \wedge r \notin I'\}$ .

Let  $X = \{x_1, \dots, x_n\}$  be a finite set of *clocks*. A *clock valuation*  $\nu : X \rightarrow \mathbb{R}^{\geq 0}$ , assigns a value to each clock  $x \in X$ .  $\nu_0$  represents all clocks in  $X$  assigned to zero. Given a clock valuation  $\nu$  and a time  $t \in \mathbb{R}^{\geq 0}$ ,  $(\nu + t)(x) = \nu(x) + t$ , for  $x \in X$ . A clock assignment function  $\nu[y \leftarrow b]$  is defined by  $\nu[y \leftarrow b](x) = b$  if  $x = y$ , and  $\nu(x)$  otherwise.  $Val(X)$  is used to denote the set of clock valuations of  $X$ .

**Definition 1 (Clock Constraint).** *Given a finite set of clocks  $X$ , we define diagonal-free constraints  $con_{df}$  and diagonal constraints  $con$ , respectively as follows:*

$$\begin{aligned} con_{df} &::= x \in I? \\ con &::= x \in I? \mid x - y \in I? \end{aligned}$$

where  $x, y \in X$  and  $I \in \mathcal{I}$ .

### 2.1 Updatable Timed Automata

*Updatable Timed Automata (UTAs)* [2–4], extended from TAs, provide a more flexible way to adjust the value of clock during location switches.

**Definition 2 (Updatable Timed Automata).** *A UTA is a tuple  $\mathcal{A} = \langle Q, q_0, F, X, \Delta \rangle$ , where*

- $Q$  is a finite set of control locations, with the initial location  $q_0 \in Q$ ,
- $F \subseteq Q$  is the set of final locations,
- $X$  is a finite set of clocks,
- $\Delta \subseteq Q \times \mathcal{O} \times Q$ , where  $\mathcal{O}$  is a set of operations. A transition  $(q_1, \phi, q_2) \in \Delta$  is written as  $q_1 \xrightarrow{\phi} q_2$ , in which  $\phi$  is either

**Local**  $\epsilon$ , an empty operation,

**Test**  $x \in I?$  or  $x - y \in I?$ , where  $x, y \in X$  and  $I \in \mathcal{I}$ ,

**Assignment**  $x \leftarrow I$ , where  $x \in X$  and  $I \in \mathcal{I}$ ,

**Increment**  $x := x + 1$ , where  $x \in X$  or

**Decrement**  $x := x - 1$ , where  $x \in X$ .

Given a UTA  $\mathcal{A}$ , we use  $Q(\mathcal{A})$ ,  $q_0(\mathcal{A})$ ,  $F(\mathcal{A})$ ,  $X(\mathcal{A})$  and  $\Delta(\mathcal{A})$  to represent its set of control locations, initial location, set of final locations, set of clocks and set of transitions, respectively. We will use similar notations for other models.

**Definition 3 (Semantics of UTAs).** *Given a UTA  $\mathcal{A} = \langle Q, q_0, F, X, \Delta \rangle$ , a configuration is a pair  $(q, \nu)$  of a control location  $q \in Q$ , and a clock valuation  $\nu$  on  $X$ . The transition relation of the UTA is represented as follows,*

- Progress transition:  $(q, \nu) \xrightarrow{t}_{\mathcal{A}} (q, \nu + t)$ , where  $t \in \mathbb{R}^{\geq 0}$ .
- Discrete transition:  $(q_1, \nu_1) \xrightarrow{\phi}_{\mathcal{A}} (q_2, \nu_2)$ , if  $q_1 \xrightarrow{\phi} q_2 \in \Delta$ , and one of the following holds,
  - **Local**  $\phi = \epsilon$ , then  $\nu_1 = \nu_2$ .
  - **Test**  $\phi = x \in I?$  or  $\phi = x - x' \in I?$ ,  $\nu_1 = \nu_2$  and  $\nu_2(x) \in I$  holds or respectively  $\nu_2(x) - \nu_2(x') \in I$  holds.
  - **Assignment**  $\phi = x \leftarrow I$ ,  $\nu_2 = \nu_1[x \leftarrow r]$  where  $r \in I$ .
  - **Increment**  $\phi = x := x + 1$ ,  $\nu_2 = \nu_1[x \leftarrow \nu_1(x) + 1]$ .
  - **Decrement**  $\phi = x := x - 1$ ,  $\nu_2 = \nu_1[x \leftarrow \nu_1(x) - 1]$  and  $\nu_1(x) \geq 1$  holds.

The initial configuration is  $(q_0, \nu_0)$ . The transition relation is  $\rightarrow$  and we define  $\rightarrow = \xrightarrow{t}_{\mathcal{A}} \cup \xrightarrow{\phi}_{\mathcal{A}}$ , and define  $\rightarrow^*$  to be the reflexive and transitive closure of  $\rightarrow$ . Without confusion, we will later leave out the subscript  $\mathcal{A}$ .

If only diagonal-free constraints appear in test transitions, the corresponding UTAs are called diagonal-free UTAs. Different from TAs, diagonal or diagonal-free constraints heavily affect decidability results of UTAs. We list a few undecidable subclasses of UTAs.

**Proposition 1.** *(Proposition 2 in [2]) UTAs without increment rules under diagonal-free constraints are undecidable.*

**Proposition 2.** *(Proposition 3 in [2]) UTAs without decrement rules under diagonal constraints are undecidable.*

## 2.2 Pushdown Systems

A *pushdown system* [5] is a transition system equipped with a finite set of control locations and a stack. The stack contains a word over some finite stack alphabet, whose length is unbounded. Hence, a pushdown system may have infinitely many reachable states.

**Definition 4 (Pushdown Systems).** *A pushdown system (PDS) is a quadruple  $\langle P, \Gamma, \Delta \rangle$  where*

- $P$  is a finite set of states,
- $\Gamma$  is finite stack alphabet,
- $\Delta \subseteq P \times \Gamma^{\leq 2} \times P \times \Gamma^{\leq 2}$  is a finite set of transitions, where  $(p, v, q, w) \in \Delta$  is denoted by  $\langle p, v \rangle \hookrightarrow \langle q, w \rangle$ , and

We use  $\alpha, \beta, \gamma, \dots$  to range over  $\Gamma$ , and  $w, v, \dots$  over words in  $\Gamma^*$ .

A configuration of  $\mathcal{P}$  is a pair  $\langle q, w \rangle$ , where  $q \in Q$  and  $w \in \Gamma^*$ . A transition relation  $\Rightarrow$  between configurations of  $\mathcal{P}$  is defined by

$$\langle q, \gamma w' \rangle \Rightarrow \langle q', ww' \rangle \text{ if } \langle q, \gamma \rangle \hookrightarrow \langle q', w \rangle$$

The reflective and transitive closure of  $\Rightarrow$  is denoted by  $\Rightarrow^*$ , and we write  $c \xRightarrow{\sigma^n} c'$  if  $c \xrightarrow{r_1} c_1 \xrightarrow{r_2} \dots c_n \xrightarrow{r_n} c'$  for any  $n \in \mathbb{N}$  and  $c, c', c_i \in Q \times \Gamma^*$  with  $1 \leq i \leq n$  and  $\sigma = [r_1, r_2, \dots, r_n]$ .

### 3 Updatable Timed Automata with One Updatable Clock

We propose a subclass of UTAs in a different facet by restricting the number of updatable clocks to be one, rather than restricting the ability of updates.

**Definition 5 (Updatable Timed Automata with One Updatable Clock).** A UTA with one updatable clock (UTA1) is a tuple  $\mathcal{A} = \langle Q, q_0, F, X, c, \Delta \rangle$ , where

- $Q$  is a finite set of control locations, with the initial location  $q_0 \in Q$ ,
- $F \subseteq Q$  is the set of final locations,
- $X = \{x_1, \dots, x_k\}$  is a finite set of clocks, and  $c$  is the singleton updatable clock,
- $\Delta \subseteq Q \times \mathcal{O} \times Q$ , where  $\mathcal{O}$  is a set of operations. A transition  $(q_1, \phi, q_2) \in \Delta$  is written as  $q_1 \xrightarrow{\phi} q_2$ , in which  $\phi$  is either
  - Local**  $\epsilon$ , an empty operation,
  - Test**  $x \in I?$ , where  $x \in X \cup \{c\}$  is a clock and  $I \in \mathcal{I}$  is an interval,
  - Assignment**  $x \leftarrow I$ , where  $x \in X \cup \{c\}$  and  $I \in \mathcal{I}$ , or
  - Increment**  $c := c + 1$ , or
  - Decrement**  $c := c - 1$ .

*Remark 1.* UTA1s defined in Definition 5 are diagonal-free. We do *not* discuss diagonal constraints between clocks, and our following proofs cannot be naturally extended to cover them.

**Definition 6 (Semantics of UTA1s).** Given a UTA1  $\mathcal{A} = \langle Q, q_0, F, X, c, \Delta \rangle$ , a configuration(state) is a pair  $(q, \nu)$  of a control location  $q \in Q$ , and a clock valuation  $\nu$  on  $X \cup \{c\}$ . The transition relation of the UTA1 is represented as follows,

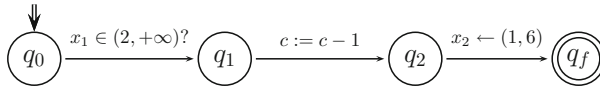
- Progress transition:  $(q, \nu) \xrightarrow{t} (q, \nu + t)$ , where  $t \in \mathbb{R}^{\geq 0}$ .

- Discrete transition:  $(q_1, \nu_1) \xrightarrow{\phi} (q_2, \nu_2)$ , if  $q_1 \xrightarrow{\phi} q_2 \in \Delta$ , and one of the following holds,
  - **Local**  $\phi = \epsilon$ , then  $\nu_1 = \nu_2$ .
  - **Test**  $\phi = x \in I?$ ,  $\nu_1 = \nu_2$  and  $\nu_2(x) \in I$  holds.
  - **Assignment**  $\phi = x \leftarrow I$ ,  $\nu_2 = \nu_1[x \leftarrow r]$  where  $r \in I$ .
  - **Increment**  $\phi = c := c + 1$ ,  $\nu_2 = \nu_1[c \leftarrow \nu_1(c) + 1]$ .
  - **Decrement**  $\phi = c := c - 1$ ,  $\nu_2 = \nu_1[c \leftarrow \nu_1(c) - 1]$  and  $\nu_1(c) \geq 1$  holds.

The initial configuration is  $(q_0, \nu_0)$ .

*Remark 2.* Although only increment ( $c := c + 1$ ) and decrement ( $c := c - 1$ ) are considered in the Definition 5, arbitrary decrement operation can also be easily encoded by some adjustment to the UTA1s we defined. For example,  $p \xrightarrow{c:=c-d} q$ , where  $d$  is some fixed positive integer, can be encoded in the following way, by introducing an extra clock  $x'$  and more locations  $p_1, p_2, \dots, p_d, p_{d+1}$ :  $p \xrightarrow{x' \leftarrow [0,0]} p_1 \xrightarrow{c:=c-1} p_2 \xrightarrow{c:=c-1} p_3 \cdots p_d \xrightarrow{c:=c-1} p_{d+1} \xrightarrow{x' \in [0,0] ?} q$ .

*Example 1.* The UTA1 illustrated in Fig. 1 has three clocks,  $c$ ,  $x_1$  and  $x_2$ , among which  $c$  is the singleton updatable clock allowed to update in a different way.



**Fig. 1.** An Example for UTA1s

One run of the UTA1 is as follows:  $(q_0, \nu_0) \xrightarrow{2.5} (q_0, \nu_1) \xrightarrow{x_1 \in (2, +\infty) ?} (q_1, \nu_2) \xrightarrow{c:=c-1} (q_2, \nu_3) \xrightarrow{x_2 \leftarrow (1,6)} (q_f, \nu_4) \xrightarrow{5} (q_f, \nu_5)$ , where

- $\nu_0 = \{\nu_0(c) = \nu_0(x_1) = \nu_0(x_2) = 0\}$ ,
- $\nu_1 = \nu_2 = \{\nu_1(c) = \nu_1(x_1) = \nu_1(x_2) = 2.5\}$ ,
- $\nu_3 = \{\nu_3(c) = 1.5, \nu_3(x_1) = \nu_3(x_2) = 2.5\}$ ,
- $\nu_4 = \{\nu_4(c) = 1.5, \nu_4(x_1) = 2.5, \nu_4(x_2) = 5.7\}$ , and
- $\nu_5 = \{\nu_5(c) = 6.5, \nu_5(x_1) = 7.5, \nu_5(x_2) = 10.7\}$ .

The first transition is a progress transition which elapses 2.5 time units, and each clock grows older for 2.5. The second transition tests whether the value of clock  $x_1$  is greater than 2, and since it is so, a move from location  $q_0$  to location  $q_1$  happens. The sequent transition updates the particular clock  $c$  by decreasing one time unit and make a move from location  $q_1$  to location  $q_2$ . The last but one transition randomly picks a value(in this run, 5.7 is picked) from the range  $(1, 6)$  and assigns it to the clock  $x_2$ . The last transition is a progress transition which elapses 5 time units.

**Definition 7 (Reachability Problem).** Given a UTA1  $\mathcal{A} = \langle Q, q_0, F, X, c, \Delta \rangle$  and a state  $q$ , decide whether there exists a path from the initial configuration such as  $(q_0, \nu_0) \rightarrow^* (q, \nu)$ , for some  $\nu$ .

## 4 Digiword and Its Operations

We denote the powerset of  $D$  by  $\mathcal{P}(D)$ . Let  $\mathcal{A} = (Q, q_0, F, X, c, \Delta)$  be a UTA1, and let  $n$  be the largest integer appearing in  $\Delta$ . For  $v \in \mathbb{R}^{\geq 0}$ ,  $\text{proj}(v) = \mathbf{r}_i$  if  $v \in \mathbf{r}_i \in \text{Intv}(n)$ , where

$$\text{Intv}(n) = \{\mathbf{r}_{2i} = [i, i] \mid 0 \leq i \leq n\} \cup \{\mathbf{r}_{2i+1} = (i, i+1) \mid 0 \leq i < n\} \cup \{\mathbf{r}_{2n+1} = (n, \omega)\}$$

The idea of the following digitization is inspired by [6–10].

**Definition 8.** Let  $\text{frac}(x, t) = t - \text{floor}(t)$  for  $(x, t) \in ((X \cup \{c\}) \times \mathbb{R}^{\geq 0})$ , where  $t$  is clock  $x$ 's value. A digitization  $\text{digi} : \text{Val}(X \cup \{c\}) \rightarrow (\mathcal{P}((X \cup \{c\}) \times \text{Intv}(n)))^*$  is as follows. For  $\nu \in \text{Val}(X \cup \{c\})$ , let  $Y_0, Y_1, \dots, Y_m$  be sets that collect  $(x, \text{proj}(t))$ 's having the same  $\text{frac}(x, t)$  for  $(x, t) \in (X \cup \{c\}) \times \mathbb{R}^{\geq 0}$ . Among them,  $Y_0$  (which is possibly empty) is reserved for the collection of  $(x, \text{proj}(t))$  with  $\text{frac}(t) = 0$ . We assume  $Y_i$ 's except for  $Y_0$  is non-empty, and  $Y_i$ 's are sorted by the increasing order of  $\text{frac}(x, t)$  (i.e.,  $\text{frac}(x, t) < \text{frac}(x', t')$  for  $(x, \text{proj}(t)) \in Y_i$  and  $(x', \text{proj}(t')) \in Y_j$ , where  $0 \leq i < j \leq m$ ).

*Example 2.* In example 1,  $n = 6$  and we have 13 intervals illustrated below.

$$\begin{array}{ccccccccccccccc} 0 & \mathbf{r}_1 & 1 & \mathbf{r}_3 & 2 & \mathbf{r}_5 & 3 & \mathbf{r}_7 & 4 & \mathbf{r}_9 & 5 & \mathbf{r}_{11} & 6 & \mathbf{r}_{13} \\ \hline & & & & & & & & & & & & & \\ \mathbf{r}_0 & & \mathbf{r}_2 & & \mathbf{r}_4 & & \mathbf{r}_6 & & \mathbf{r}_8 & & \mathbf{r}_{10} & & \mathbf{r}_{12} & \end{array}$$

For the clock valuation  $\nu_4$  in Example 1,  $\text{digi}(\nu_4)$  is  $\{(c, \mathbf{r}_3), (x_1, \mathbf{r}_5)\}\{(x_2, \mathbf{r}_{11})\}$ . Note that, for convenience we do not show the empty set  $Y_0$  in  $\text{digi}(\nu_4)$ .

A word in  $(\mathcal{P}((X \cup \{c\}) \times \text{Intv}(n)))^*$  is called *digiword*. If for a digiword  $\bar{Y}$  there exists an clock valuation  $\nu \in \text{Val}(X \cup \{c\})$  such that  $\text{digi}(\nu) = \bar{Y}$ , we call the word *well-formed digiword*.

*Remark 3.* For a finite set of clocks  $X \cup \{c\}$ , the set of *well-formed digiword* is finite. This is obvious, since there are a fixed number of clocks, which leads to a fixed number of combinations of digiword. This play an essential role in our encoding, ensuring the PDS has finite states.

**Definition 9.** Let  $\bar{Y} = Y_0 \cdots Y_m \in (\mathcal{P}((X \cup \{c\}) \times \text{Intv}(n)))^*$ . We define digiword operations as follows.

– **Insert<sub>I</sub>**  $\text{insert}(\bar{Y}, (x, \mathbf{r}_i))$  for  $x \in X \cup \{c\}$  inserts  $(x, \mathbf{r}_i)$  to  $\bar{Y}$  at

$$\begin{cases} \text{either put into } Y_j \text{ for } j > 0, \text{ or} \\ \text{put the singleton set } \{(x, \mathbf{r}_i)\} \text{ at any place after } Y_0 & \text{if } i \text{ is odd} \\ \text{put into } Y_0 & \text{if } i \text{ is even} \end{cases}$$

– **Delete**  $\text{delete}(\bar{Y}, x)$  for  $x \in X \cup \{c\}$  is obtained from  $\bar{Y}$  by deleting the element  $(x, \mathbf{r})$  indexed by  $x$ .

– **Increase**  $\text{increase}(\bar{Y}, c)$  is obtained from  $\bar{Y}$  by replacing the element  $(c, \mathbf{r}_i)$  indexed by  $c$  with element  $(c, \mathbf{r}_{\min\{i+2, 2n+1\}})$ .

- **Decrease**  $decrease(\bar{Y}, c, d)$  and  $d \in \mathbb{N}$  is obtained from  $\bar{Y}$  by replacing the element  $(c, \mathbf{r}_i)$  indexed by  $c$  with element  $(c, \mathbf{r}_{\max\{i-d, 0\}})$ .
- **Shift**. Let  $j \in [0..m]$  and  $0 \leq i \leq 2n + 1$ . A shift  $\bar{Y} = Y_0 Y_1 \cdots Y_m \Rightarrow \bar{Y}' = Y'_0 Y'_1 \cdots Y'_{m'}$  is defined as follows.

$$\left\{ \begin{array}{l} \text{either } \bar{Y}' = Y'_0, Y'_1, \dots, Y'_{m+1} \text{ if } Y_0 \neq \emptyset, Y'_0 = \emptyset, Y'_1 = \{(x, \mathbf{r}_{\min\{i+1, 2n+1\}}) \\ \quad \mid (x, \mathbf{r}_i) \in Y_0\} \text{ and } Y'_j = Y_{j-1} \text{ for } j \in [2..m+1]. \\ \text{or } \bar{Y}' = Y'_0, Y'_1, \dots, Y'_{m-1} \text{ otherwise, } Y'_0 = \{(x, \mathbf{r}_{\min\{i+1, 2n+1\}}) \mid (x, \mathbf{r}_i) \in Y_m\}, \\ \quad \text{and } Y'_j = Y_j \text{ for } j \in [1..m-1]. \end{array} \right.$$

As convention, we define  $\Rightarrow^*$  as reflexive transitive closure of  $\Rightarrow$ .

*Example 3.* Consider the digiword in Example 1,  $digi(\nu_0) = \{(c, \mathbf{r}_0), (x_1, \mathbf{r}_0), (x_2, \mathbf{r}_0)\}$ .

- after finite times shifts,  
 $digi(\nu_1) = digi(\nu_2) = \{(c, \mathbf{r}_5), (x_1, \mathbf{r}_5), (x_2, \mathbf{r}_5)\}$
- after  $decrease(digi(\nu_2), c, 2)$   
 $digi(\nu_3) = \{(c, \mathbf{r}_3), (x_1, \mathbf{r}_5), (x_2, \mathbf{r}_5)\}$
- after  $insert(delete(digi(\nu_3), x_2), (x_2, \mathbf{r}_{11}))$   
 $digi(\nu_4) = \{(c, \mathbf{r}_3), (x_1, \mathbf{r}_5)\} \{(x_2, \mathbf{r}_{11})\}$
- after finite times shifts,  
 $digi(\nu_5) = \{(c, \mathbf{r}_{13}), (x_1, \mathbf{r}_{13})\} \{(x_2, \mathbf{r}_{13})\}$

Equivalent to the *regions* [1] in nature, well formed digiwords are bisimilar to the clock valuations in the following sense:

$$\bar{Y}_1 \Rightarrow^* \bar{Y}_2 \text{ if and only if } \forall \nu \in [\bar{Y}_1], \exists t \in \mathbb{R}^{\geq 0} \text{ s.t. } \nu + t \in [\bar{Y}_2]$$

where,  $[\bar{Y}] = \{\nu \mid \bar{Y} = digi(\nu)\}$ .

## 5 Reachability for UTA1s

In this section, we show that the reachability problem of UTA1s is decidable. discretePDSs, which nicely enjoy decidable property of configuration reachability. The key idea is that when the updatable clock's value exceeds the maximum integer  $n$ , we push a special symbol into the stack to record the updates.

**Definition 10.** For a UTA1  $\mathcal{A} = \langle Q, q_0, F, X, c, \Delta \rangle$ , there is a PDS  $\mathcal{P} = \langle P \times (\mathcal{P}((X \cup \{c\}) \times Intv(n)))^*, \{\bullet, \perp\}, \Delta_d \rangle$ , where the set of states  $P = Q \cup \{p' \mid p \in Q\}$  and the initial configuration  $\kappa_0 = \langle (q_0, \{(x, \mathbf{r}_0) \mid x \in X \cup \{c\}\}), \perp \rangle$ .  $\Delta_d$  consists of:

- **Time Progress**
  1.  $\langle (p, \bar{Y}), \perp \rangle \hookrightarrow \langle (p, \bar{Z}), \perp \rangle$ , where  $\bar{Y} \Rightarrow \bar{Z}$ , for  $(c, \mathbf{r}_i) \in Y_j \in \bar{Y}$  and  $i \leq 2n - 1$ .
  2.  $\langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (p, \bar{Z}), \bullet \rangle$ , where  $\bar{Y} \Rightarrow \bar{Z}$ , for  $(c, \mathbf{r}_i) \in Y_j \in \bar{Y}$  and  $2n \leq i \leq 2n + 1$ .

- **Local** ( $p \xrightarrow{\epsilon} q \in \Delta$ )  
 $\langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (q, \bar{Y}), \epsilon \rangle$ .
- **Test** ( $p \xrightarrow{x \in I?} q \in \Delta$ )  
 $\langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (q, \bar{Y}), \epsilon \rangle$  if  $r_i \subseteq I$  for  $(x, \mathbf{r}_i) \in Y_j \in \bar{Y}$ .
- **Assignment** ( $p \xrightarrow{c \leftarrow I} q \in \Delta$ )
  1.  $\langle (p, \bar{Y}), \bullet \rangle \hookrightarrow \langle (p, \bar{Y}), \epsilon \rangle$ .
  2.  $\langle (p, \bar{Y}), \perp \rangle \hookrightarrow \langle (p', \bar{Y}), \perp \rangle$ .
  3.  $\forall \mathbf{r}_i \subseteq I \setminus \mathbf{r}_{2n+1}, \langle (p', \bar{Y}), \perp \rangle \hookrightarrow \langle (q, \text{insert}(\text{delete}(\bar{Y}, c), (c, \mathbf{r}_i))), \perp \rangle$ .
  4.  $\langle (p', \bar{Y}), \epsilon \rangle \hookrightarrow \langle (p', \bar{Y}), \bullet \rangle$ , if  $\mathbf{r}_{2n+1} \subseteq I$ .
  5.  $\langle (p', \bar{Y}), \bullet \rangle \hookrightarrow \langle (q, \text{insert}(\text{delete}(\bar{Y}, c), (c, \mathbf{r}_{2n+1}))), \bullet \rangle$ , if  $\mathbf{r}_{2n+1} \subseteq I$ .
- **Assignment** ( $p \xrightarrow{x \leftarrow I} q \in \Delta$ , where  $x \in X$ )  
 $\forall \mathbf{r}_i \subseteq I, \langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (q, \text{insert}(\text{delete}(\bar{Y}, x), (x, \mathbf{r}_i))), \epsilon \rangle$ .
- **Increment** ( $p \xrightarrow{c := c+1} q \in \Delta$ )
  1.  $\langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (q, \text{increase}(\bar{Y}, c)), \epsilon \rangle$ , for  $(c, \mathbf{r}_i) \in Y_j \in \bar{Y}$  and  $i \leq 2n - 2$ .
  2.  $\langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (q, \text{increase}(\bar{Y}, c)), \bullet \rangle$ , for  $(c, \mathbf{r}_{2n-1}) \in Y_j \in \bar{Y}$ .
  3.  $\langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (q, \text{increase}(\bar{Y}, c)), \bullet \bullet \rangle$ , for  $(c, \mathbf{r}_i) \in Y_j \in \bar{Y}$  and  $i \geq 2n$ .
- **Decrement** ( $p \xrightarrow{c := c-1} q \in \Delta$ )
  1.  $\langle (p, \bar{Y}), \bullet \bullet \bullet \rangle \hookrightarrow \langle (q, \bar{Y}), \bullet \rangle$ .
  2.  $\langle (p, \bar{Y}), \bullet \bullet \perp \rangle \hookrightarrow \langle (q, \text{decrease}(\bar{Y}, c, 1)), \perp \rangle$ .
  3.  $\langle (p, \bar{Y}), \bullet \perp \rangle \hookrightarrow \langle (q, \text{decrease}(\bar{Y}, c, 2)), \perp \rangle$ .
  4.  $\langle (p, \bar{Y}), \perp \rangle \hookrightarrow \langle (q, \text{decrease}(\bar{Y}, c, 2)), \perp \rangle$  for  $(c, \mathbf{r}_i) \in Y_j \in \bar{Y}$  and  $i \geq 2$ .

In our encoding, we abuse the symbol  $\epsilon$  in the left hand of the transition rule of PDS  $\mathcal{P}$  to indicate that whatever the topmost symbol in the stack is, the transition can occur. This leads to a concise description for our encoding, which is equivalent to its counterpart with only standard transition rules. For example,  $\langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (q, \bar{Y}), \epsilon \rangle$  can be replaced with two transition rules since there are only 2 symbols in the alphabet: (1)  $\langle (p, \bar{Y}), \perp \rangle \hookrightarrow \langle (q, \bar{Y}), \perp \rangle$  and (2)  $\langle (p, \bar{Y}), \bullet \rangle \hookrightarrow \langle (q, \bar{Y}), \bullet \rangle$ .

Given a UTA1  $\mathcal{A}$ , for each kind of its transitions, we have its counterpart in the  $\Delta_d(\mathcal{P})$ . Some transitions of  $\mathcal{P}$ , at the first look, may seem hard to understand. The following give a simple explanation for simulating *assignment* and *decrement*.

For an assignment of the form  $x \leftarrow I$ , PDS  $\mathcal{P}$  proceed with two different cases: (1)  $x = c$ ; (2)  $x \neq c$ . For the first case,  $\mathcal{P}$  first need to pop all symbol  $\bullet$  out of stack, then push a certain number of symbols  $\bullet$  if needed, and finally perform *delete* and *insert* operations. For the latter case, much simpler, just directly perform *delete* and *insert* operations.

For a decrement of the form  $c := c - 1$ , PDS  $\mathcal{P}$  proceed with four different cases: (1)  $\nu(c) > n + 1$ ; (2)  $\nu(c) = n + 1$ ; (3)  $n < \nu(c) < n + 1$ ; (4)  $1 \leq \nu(c) \leq n$ . For the first case,  $\mathcal{P}$  merely pops out two symbols of  $\bullet$ . For the second and third cases,  $\mathcal{P}$  pops out one or two symbol(s) of  $\bullet$  and performs *decrease* operation. The difference between them is how much it decreases. For the last case,  $\mathcal{P}$  merely performs *decrease* operation.

The following example shows the discretization of assignment transitions, decrement transitions.



*Example 4.* Consider the run in the Example 1.

- $(q_1, \nu_2) \xrightarrow{c:=c-1} (q_2, \nu_3)$ , where  $\nu_2 = \{\nu_2(c) = \nu_2(x_1) = \nu_2(x_2) = 2.5\}$  and  $\nu_3 = \{\nu_3(c) = 1.5, \nu_3(x_1) = \nu_3(x_2) = 2.5\}$ . Corresponding simulation:  $\langle (q_1, \text{digi}(\nu_2)), \perp \rangle \hookrightarrow \langle (q_2, \text{decrease}(\text{digi}(\nu_2), c, 2)), \perp \rangle$ .
- $(q_2, \nu_3) \xrightarrow{x_2 \leftarrow (1,6)} (q_f, \nu_4)$ , where  $\nu_3$  is defined above and  $\nu_4 = \{\nu_4(c) = 1.5, \nu_4(x_1) = 2.5, \nu_4(x_2) = 5.7\}$ . Corresponding simulation:  $\langle (q_2, \text{digi}(\nu_3)), \perp \rangle \hookrightarrow \langle (q_f, \text{insert}(\text{delete}(\text{digi}(\nu_3), x_2), (x_2, \mathbf{r}_{11}))), \perp \rangle$ .

**Definition 11.** Let  $\varrho$  be any configuration of a UTA1 such that  $\varrho_0 = (q_0, \nu_0) \hookrightarrow^* \varrho = (q, \nu)$ . Define  $\llbracket \varrho \rrbracket$  to be  $\langle (q, \text{digi}(\nu)), \bullet^k \perp \rangle$ , where  $k = 2 \times \text{floor}(\nu(c) - n) + \text{ceiling}(\text{frac}(\nu(c)))$  if  $\nu(c) > n$  otherwise  $k = 0$ . A configuration  $\kappa$  of PDS  $\mathcal{P}$  with some  $\varrho$  and  $\kappa = \llbracket \varrho \rrbracket$  is called an encoded configuration.

*Example 5.* Consider the configuration  $(q_f, \nu_5)$  in Example 1, where  $\nu_5 = \{c = 6.5, x_1 = 7.5, x_2 = 10.7\}$ . By Definition 11,  $\llbracket (q_f, \nu_5) \rrbracket = \langle (q_f, \text{digi}(\nu_5)), \bullet \perp \rangle$ , where  $\text{digi}(\nu_5) = \{(c, \mathbf{r}_{13}), (x_1, \mathbf{r}_{13})\} \{(x_2, \mathbf{r}_{13})\}$ .

*Remark 4.* For a UTA1's time progress transition  $(p, \nu) \xrightarrow{t} (q, \nu')$ , where  $t \in \mathbb{R}^{\geq 0}$ , there has a transition sequence of  $\llbracket (p, \nu) \rrbracket \hookrightarrow \langle (p, \bar{Z}_1), w_1 \rangle \hookrightarrow \langle (p, \bar{Z}_2), w_2 \rangle \cdots \hookrightarrow \llbracket (q, \nu') \rrbracket$  to simulate it, which consists of finite many time *time progress* transitions of PDS.

**Lemma 1.** Given a UTA1  $\mathcal{A}$ , its associated PDS  $\mathcal{P}$ , and any configuration  $\varrho$ ,  $\varrho'$  of  $\mathcal{A}$ .

**(Preservation)** If  $\varrho \rightarrow \varrho'$  then  $\llbracket \varrho \rrbracket \hookrightarrow^* \llbracket \varrho' \rrbracket$ .

**(Reflection)** If  $\llbracket \varrho \rrbracket \hookrightarrow^* \kappa$ ,

1. there exists  $\varrho'$  such that  $\kappa = \llbracket \varrho' \rrbracket$  and  $\varrho \rightarrow^* \varrho'$ , or
2.  $\kappa$  is not an encoded configuration, and there exists  $\varrho'$  such that  $\kappa \hookrightarrow^* \llbracket \varrho' \rrbracket$  by transitions (of  $\mathcal{P}$ ) and  $\varrho \rightarrow^* \varrho'$ .

With Lemma 1, we have the following theorem.

**Theorem 1.** The reachability of a UTA1 is decidable.

The proof is given in Appendix A.

## 6 Related Work

After *timed automata (TAs)* [1] had been proposed by Alur and Dill, a lot of work has been devoted to extensions of TAs.

The extension that allows to compare the sum of two clocks with a constant has also been investigated in [1]. It leads to an undecidable class of automata. Periodic clock constraints defined in [11], can express properties like “the value of a clock is even” or “the value of a clock is of the form  $0.5 + 3n$  where  $n$  is some integer. The corresponding class of automata is strictly more powerful

than TAs if silent transitions are not allowed but otherwise coincides with the original model.

*Controlled real-time automata*, a parameterized family of TAs with some additional features like clock stopping, variable clock velocities and periodic tests has been proposed in [12]. Due to carefully chosen restrictions, controlled real-time automata remains decidable.

*Recursive timed automata (RTAs)* [13] is an extension of TAs with recursive structure. It has clocks by the mechanism of “pass-by-value”. When the condition of “glitch-freeness”, i.e. all the clocks of components are uniformly either by “pass-by-value” or by “pass-by-reference”, the reachability is shown to be decidable.

*Nested timed automata (NeTAs)* [9, 10, 14] extend TAs with recursive structure in another way, which allow clocks of some TAs in the stack elapse simultaneously with the current running clocks during time passage. Those clocks are named *local clocks*, while clocks in other TAs kept unaltered clocks during time passage are named *frozen clocks*. It is proved that the reachability of NeTAs with both types of clocks and a singleton *global clock* that can be observed by all TAs is decidable, while that with two or more global clocks is undecidable [10].

The *updatable timed automata (UTAs)* [4] is a natural syntactic extension of TA. It enjoyed the possibility of updating the clocks in a more elaborate way than just simple reset in TA. The value of a clock could be reassigned to a basic arithmetic computation result of values of other clocks. The paper gave undecidability and decidability results for several specific cases. The decidability results were obtained through a generalization of the region graph proposed by Alur and Dill, while the undecidability results were obtained by reducing an undecidable problem on Minsky Machine [15] to the emptiness problem for a subclass of UTAs. The expressiveness of the UTAs was also investigated in the paper. Our model UTA1s, is actually a specific subclass of general UTAs by restricting the number of updatable clock to one.

A forward analysis of UTAs has been proposed in [16] for specific subclass of UTAs that do not use comparisons between clocks. Recently, a refined algorithm for specific subclass of UTAs with diagonal constraints has been proposed in [17].

## 7 Conclusion

This paper has investigated the reachability of UTA1s, UTAs with one updatable clock. By restricting the number of updatable clocks to one, the reachability of UTA1s is decidable, under the diagonal-free constraints. The decidability is proved by encoding the clock behavior of UTA1s to PDSs based on the notion of *digiword*. The key idea is to use a stack to record the time interval exceeding the maximum constant integer. As a result, the transitions in pushdown systems are finer than that of UTA1s.

UTA1s defined in Definition 5 only allow diagonal-free clock constraints. Our proof can not be extended to cover the UTA1s with diagonal clock constraints. We will further investigate the reachability for the UTA1s with diagonal clock constraints as a future work.

**Acknowledgements.** This work is supported by the NSFC-JSPS bilateral joint research project (61511140100), the National Natural Science Foundation of China (No. 61472240, 91318301, 61261130589), and JSPS KAKENHI Grant-in-Aid for Scientific Research(B) (15H02684, 25280023) and Challenging Exploratory Research (26540026).

## A A Proof of Lemma 1

*Proof.* Let  $\varrho = (q, \nu)$ . Then  $\llbracket \varrho \rrbracket = \langle (q, \text{digi}(\nu)), w \rangle$ , where  $w = \bullet^k \perp$  for some  $k$ . For preservation part, By case analysis of  $\varrho \rightarrow \varrho'$ .

1. **Time Progress:**  $\varrho \xrightarrow{t} \varrho'$ . By the digiword's region-like property, we have  $\text{digi}(\nu) \Rightarrow^* \text{digi}(\nu + t)$ . Proceed with two subcases:
  - (a) If no stack operations involved (i.e.  $\nu(c) + t \leq n$ ), then we have  $\llbracket \varrho \rrbracket = \langle (q, \text{digi}(\nu)), w \rangle \hookrightarrow^* \llbracket \varrho' \rrbracket = \langle (q, \text{digi}(\nu + t)), w \rangle$  by applying the first transition rule of time progress rules finite times. Note that in this subcase,  $w = \perp$ .
  - (b) If  $\nu(c) + t > n$ , then we have  $\llbracket \varrho \rrbracket = \langle (q, \text{digi}(\nu)), w \rangle \hookrightarrow^* \llbracket \varrho' \rrbracket = \langle (q, \text{digi}(\nu + t)), w' \rangle$  by applying the first time progress rule finite times (maybe zero times if  $\nu(c) \geq n$ ) and then applying the second time progress rule finite times.
2. **Local:**  $\varrho = (p, \nu) \xrightarrow{\epsilon} \varrho' = (q, \nu)$ . Then with the *Local* transition of PDS,  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), w \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{digi}(\nu)), w \rangle$ .
3. **Test:**  $\varrho = (p, \nu) \xrightarrow{x \in I?} \varrho' = (q, \nu)$ . Then with the *Test* transition of PDS,  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), w \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{digi}(\nu)), w \rangle$ , since  $\exists (x, \mathbf{r}_i) \in Y_j \in \text{digi}(\nu)$  such that  $\mathbf{r}_i \subseteq I$ , where  $\text{digi}(\nu) = Y_1 Y_2 \cdots Y_j \cdots Y_m$ .
4. **Assignment:**  $\varrho = (p, \nu) \xrightarrow{x \leftarrow I} \varrho' = (q, \nu[x \leftarrow d])$ , where  $d \in I$ . We proceed with 2 cases:
  - (a) If  $x \neq c$ , then we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), w \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{digi}(\nu[x \leftarrow d])), w \rangle$  by applying the first assignment rule of PDS  $\mathcal{P}$ :  $\langle (p, \bar{Y}), \epsilon \rangle \hookrightarrow \langle (q, \text{insert}(\text{delete}(\bar{Y}, x), (x, \mathbf{r}_i))), \epsilon \rangle$ , where  $\bar{Y} = \text{digi}(\nu)$  and  $d \in \mathbf{r}_i$ .
  - (b) Otherwise,  $x = c$ , proceed with two subcases,  $d \leq n$  and  $d > n$ .
    - $d \leq n$ : If  $w = \bullet^k \perp$  for  $k > 0$ , we first need to pop all symbols of  $\bullet$  out of stack, by repeatedly applying the second assignment rule of PDS  $k$  times, having  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), \bullet^k \perp \rangle \hookrightarrow^* \kappa = \langle (p, \text{digi}(\nu)), \perp \rangle$ , otherwise define  $\kappa = \llbracket \varrho \rrbracket$  since the stack already has no symbols of  $\bullet$ . Then by applying the third and fourth assignment rule, we have  $\kappa \hookrightarrow \kappa' = \langle (p', \text{digi}(\nu)), \perp \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{digi}(\nu[x \leftarrow d])), \perp \rangle$ .
    - $d > n$ : If  $w = \bullet^k \perp$  for  $k > 0$ , we first need to pop all symbols of  $\bullet$  out of stack, by repeatedly applying the second assignment rule of PDS  $k$  times, having  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), \bullet^k \perp \rangle \hookrightarrow^* \kappa = \langle (p, \text{digi}(\nu)), \perp \rangle$ , otherwise define  $\kappa = \llbracket \varrho \rrbracket$ , since the stack already has no symbols of  $\bullet$ . Next, we have  $\kappa \hookrightarrow \kappa' = \langle (p', \text{digi}(\nu)), \perp \rangle$  by the third assignment rule. Then, by repeatedly applying the fifth assignment rule of  $\mathcal{P}$  until we have  $k = 2 \times \text{floor}(d - n) + \text{ceiling}(\text{frac}(d))$  symbols of  $\bullet$  in stack,

we have  $\kappa' \hookrightarrow^* \kappa'' = \langle (p', \text{digi}(\nu)), \bullet^k \perp \rangle$ . Finally, by applying the last assignment rule of *mathcal{P}*, we have  $\kappa'' \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{digi}(\nu[c \leftarrow d])), \bullet^k \perp \rangle$ .

5. **Increment:**  $\varrho = (p, \nu) \xrightarrow{c:=c+1} \varrho' = (q, \nu[c \leftarrow \nu(c) + 1])$ . We proceed with 3 subcases:
- (a) If  $\nu(c) \leq n - 1$ , then we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), w \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{increase}(\text{digi}(\nu), c)), w \rangle = \langle (q, \text{digi}(\nu[c \leftarrow \nu(c) + 1])), w \rangle$  by applying the first transition increment rule of PDS  $\mathcal{P}$ .
  - (b) If  $n - 1 < \nu(c) < n$ , then we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), w \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{increase}(\text{digi}(\nu), c)), \bullet w \rangle = \langle (q, \text{digi}(\nu[c \leftarrow \nu(c) + 1])), \bullet w \rangle$  by applying the second transition increment rule of PDS  $\mathcal{P}$ .
  - (c) If  $\nu(c) \geq n$ , then we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), w \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{increase}(\text{digi}(\nu), c)), \bullet \bullet w \rangle = \langle (q, \text{digi}(\nu[c \leftarrow \nu(c) + 1])), \bullet \bullet w \rangle$  by applying the third transition increment rule of PDS  $\mathcal{P}$ .
6. **Decrement:**  $\varrho = (p, \nu) \xrightarrow{c:=c-1} \varrho' = (q, \nu[c \leftarrow \nu(c) - 1])$ . Note that only when  $\nu(c) \geq 1$ , can this transition happen. We proceed with 4 subcases:
- (a) If  $\nu(c) > n + 1$ , then we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), \bullet \bullet \bullet w' \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{digi}(\nu)), \bullet w' \rangle = \langle (q, \text{digi}(\nu[c \leftarrow \nu(c) - 1])), \bullet w' \rangle$  by applying the first transition decrement rule of PDS  $\mathcal{P}$ .
  - (b) If  $\nu(c) = n + 1$ , then we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), \bullet \bullet \perp \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{decrease}(\text{digi}(\nu), c, 1)), \perp \rangle = \langle (q, \text{digi}(\nu[c \leftarrow \nu(c) - 1])), \perp \rangle$  by applying the second transition decrement rule of PDS  $\mathcal{P}$ .
  - (c) If  $n < \nu(c) < n + 1$ , then we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), \bullet \perp \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{increase}(\text{digi}(\nu), c, 2)), \perp \rangle = \langle (q, \text{digi}(\nu[c \leftarrow \nu(c) - 1])), \perp \rangle$  by applying the third transition decrement rule of PDS  $\mathcal{P}$ .
  - (d) If  $1 \leq \nu(c) \leq n$ , then we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), \perp \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{increase}(\text{digi}(\nu), c, 2)), \perp \rangle = \langle (q, \text{digi}(\nu[c \leftarrow \nu(c) - 1])), \perp \rangle$  by applying the fourth transition decrement rule of PDS  $\mathcal{P}$ .

For *reflection* part, by induction on the steps of  $\hookrightarrow^*$ .

**Base step:** Consider the case of  $\llbracket \varrho \rrbracket \hookrightarrow \kappa$ :

1. **Time Progress.** Obviously,  $\llbracket \varrho \rrbracket \hookrightarrow \kappa$  by one of two time progress rules of PDS  $\mathcal{P}$ . Since digiwords have the region-like property,  $\text{digi}(\nu) \Rightarrow \bar{Y}$  implies that there exists a clock valuation  $\nu' \in \text{Val}(X \cup \{c\})$  such that  $\nu' = \nu + t$  and  $\nu' \in [\bar{Y}]$  for a real number  $t$  (more precisely,  $0 < t < 1$ ). Proceed with two cases:
  - (a) If  $\nu(c) < n$ , then  $\llbracket \varrho \rrbracket \hookrightarrow \kappa$  by using the first time progress rule of PDS  $\mathcal{P}$ . In such case, we have  $\kappa = \langle (p, \bar{Y}), \perp \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho = (p, \nu) \xrightarrow{t} \varrho' = (p, \nu')$ .
  - (b) If  $\nu(c) \geq n$ , then  $\llbracket \varrho \rrbracket \hookrightarrow \kappa$  by using the second time progress rule of PDS  $\mathcal{P}$ . In such case, we have  $\kappa = \langle (p, \bar{Y}), \bullet w \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho = (p, \nu) \xrightarrow{t} \varrho' = (p, \nu')$ .
2. **Local.** If  $\llbracket \varrho \rrbracket \hookrightarrow \kappa$  for  $p \xrightarrow{\epsilon} q$  being a transition in  $\mathcal{A}$ , then  $\kappa = \langle (q, \text{digi}(\nu)), w \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho' = (q, \nu)$ .
3. **Test.** Similar with the case for **Local**.

4. **Assignment.** We only need to proceed with the first three cases, the other cases can not happen since the configuration is a *encoded configuration* (i.e.  $\llbracket \varrho \rrbracket$ ).
- (a) If  $\llbracket \varrho \rrbracket \hookrightarrow \kappa$  by applying the first assignment transition rule of PDS  $\mathcal{P}$ , then  $\kappa$  is a *encoded configuration*, since we have  $\llbracket \varrho \rrbracket = \langle (p, \text{digi}(\nu)), w \rangle \hookrightarrow \kappa = \llbracket \varrho' \rrbracket = \langle (q, \text{insert}(\text{delete}(\text{digi}(\nu), x), (x, \mathbf{r}_i))), w \rangle$ , where  $\varrho = (p, \nu) \xrightarrow{x \leftarrow I} \varrho' = (q, \nu[x \leftarrow d])$  for  $d \in \mathbf{r}_i \in I$ .
- (b) If  $\llbracket \varrho \rrbracket \hookrightarrow \kappa$  by applying the second assignment transition rule of PDS  $\mathcal{P}$ , then the clock involved is  $c$  and  $\kappa$  may not be a *encoded configuration*. However, by applying the same rules finite times and then third assignment rule,  $\kappa \hookrightarrow^* \kappa_1 = \langle (p, \text{digi}(\nu)), \perp \rangle \hookrightarrow \kappa_2 = \langle (p', \text{digi}(\nu)), \perp \rangle$ , and then if  $\mathbf{r}_i \in I$  for  $i \leq 2n$ , we have  $\kappa_2 \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{insert}(\text{delete}(\text{digi}(\nu), c), (c, \mathbf{r}_i)), \perp \rangle$ , by applying the fourth assignment rule, otherwise, we have  $\kappa_2 \hookrightarrow^* \kappa_3 = \langle (p', \text{digi}(\nu)), \bullet^k \perp \rangle \hookrightarrow \llbracket \varrho' \rrbracket = \langle (q, \text{insert}(\text{delete}(\text{digi}(\nu), c), (c, \mathbf{r}_i)), \bullet^k \perp \rangle$  by applying finite times of fifth assignment rule first and then the last assignment rule.
- (c) If  $\llbracket \varrho \rrbracket \hookrightarrow \kappa$  by applying the second assignment transition rule of PDS  $\mathcal{P}$ , the proof is similar to the above case.
5. **Increment.** We proceed with three cases:
- (a) If it takes the first increment transition rule, then we can infer that  $\nu(c) \leq n - 1$ . Thus we have  $\llbracket \varrho \rrbracket \hookrightarrow \kappa = \langle (q, \text{increase}(\text{digi}(\nu), c)), w \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho \xrightarrow{c := c+1} \varrho'$ .
- (b) If it takes the second increment transition rule, then we can infer that  $n - 1 < \nu(c) < n$ . Then we have  $\llbracket \varrho \rrbracket \hookrightarrow \kappa = \langle (q, \text{increase}(\text{digi}(\nu), c)), \bullet w \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho \xrightarrow{c := c+1} \varrho'$ .
- (c) If it takes the third increment transition rule, then we can infer that  $\nu(c) \geq n$ . Then we have  $\llbracket \varrho \rrbracket \hookrightarrow \kappa = \langle (q, \text{increase}(\text{digi}(\nu), c)), \bullet \bullet w \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho \xrightarrow{c := c+1} \varrho'$ .
6. **Decrement.** We proceed with four cases:
- (a) If it takes the first decrement transition rule, then we can infer that  $\nu(c) > n + 1$ . Then we have  $\llbracket \varrho \rrbracket = \langle (q, \text{digi}(\nu)), \bullet \bullet \bullet w' \rangle \hookrightarrow \kappa = \langle (q, \text{digi}(\nu)), \bullet w' \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho \xrightarrow{c := c-1} \varrho'$ . Note that  $\text{digi}(\nu) = \text{digi}(\nu[\nu(c) \leftarrow \nu(c) - 1])$ , since  $\nu(c) > n + 1$ .
- (b) If it takes the second decrement transition rule, then we can infer that  $\nu(c) = n + 1$ . Then we have  $\llbracket \varrho \rrbracket = \langle (q, \text{digi}(\nu)), \bullet \bullet \perp \rangle \hookrightarrow \kappa = \langle (q, \text{decrease}(\text{digi}(\nu), c, 1)), \perp \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho \xrightarrow{c := c-1} \varrho' = (q, \nu')$  and  $\nu'(c) = n$ .
- (c) If it takes the third decrement transition rule, then we can infer that  $n < \nu(c) < n + 1$ . Then we have  $\llbracket \varrho \rrbracket = \langle (q, \text{digi}(\nu)), \bullet \perp \rangle \hookrightarrow \kappa = \langle (q, \text{decrease}(\text{digi}(\nu), c, 2)), \perp \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho \xrightarrow{c := c-1} \varrho' = (q, \nu')$  and  $n - 1 < \nu'(c) < n$ .
- (d) If it takes the fourth decrement transition rule, then we can infer that  $1 \leq \nu(c) \leq n$ . Then we have  $\llbracket \varrho \rrbracket = \langle (q, \text{digi}(\nu)), \perp \rangle \hookrightarrow \kappa = \langle (q, \text{decrease}(\text{digi}(\nu), c, 2)), \perp \rangle = \llbracket \varrho' \rrbracket$ , where  $\varrho \xrightarrow{c := c-1} \varrho' = (q, \nu')$ .

**Induction step:** Assume  $\llbracket \varrho \rrbracket \hookrightarrow^* \kappa' \hookrightarrow \kappa$ . We proceed with two cases:

1.  $\kappa'$  is an *encoded configuration*, and by induction hypothesis  $\llbracket \varrho \rrbracket \hookrightarrow^* \kappa' = \llbracket \varrho' \rrbracket$ . Then the proof is similar to the base step.
2.  $\kappa'$  is not an *encoded configuration*. Note that for our encoding, we have encoded configurations except for the second, third and fifth assignment rule. Here, we give a proof for the fifth assignment rule. The other cases are similar. Assume  $\kappa' = \langle (p', \bar{Y}), w \rangle$  is obtained by applying the fifth assignment rule, we have  $\kappa' \hookrightarrow \kappa = \langle (q, \mathit{digi}(\nu')), w \rangle = \llbracket \varrho' \rrbracket = \llbracket (q, \nu') \rrbracket$  by applying the last assignment rule, where  $\exists \nu \in [\bar{Y}]$  such that  $\nu' = \nu[c \leftarrow d]$  for  $d \in \mathbf{r}_{2n+1}$ . Finally, put together  $\varrho \rightarrow^* \varrho'$ .

## References

1. Alur, R., Dill, D.L.: A theory of timed automata. *Theor. Comput. Sci.* **126**, 183–235 (1994)
2. Bouyer, P., Dufourd, C., Fleury, E., Petit, A.: Are timed automata updatable? In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855. Springer, Heidelberg (2000)
3. Bouyer, P., Dufourd, C., Fleury, É., Petit, A.: Expressiveness of updatable timed automata. In: Nielsen, M., Rovan, B. (eds.) MFCS 2000. LNCS, vol. 1893, pp. 232–242. Springer, Heidelberg (2000)
4. Bouyer, P., Dufourd, C., Fleury, E., Petit, A.: Updatable timed automata. *Theor. Comput. Sci.* **321**, 291–345 (2004)
5. Schwoon, S.: Model-checking pushdown system. Ph.D. thesis, Technical University of Munich (2000)
6. Ouaknine, J., Worrell, J.: On the language inclusion problem for timed automata: closing a decidability gap. In: Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS'04), IEEE Computer Society, pp. 54–63 (2004)
7. Abdulla, P.A., Jonsson, B.: Verifying Networks of Timed Processes (Extended Abstract). In: Steffen, B. (ed.) TACAS 1998. LNCS, vol. 1384, p. 298. Springer, Heidelberg (1998)
8. Abdulla, P., Jonsson, B.: Model checking of systems with many identical time processes. *Theor. Comput. Sci.* **290**, 241–264 (2003)
9. Li, G., Cai, X., Ogawa, M., Yuen, S.: Nested timed automata. In: Braberman, V., Fribourg, L. (eds.) FORMATS 2013. LNCS, vol. 8053, pp. 168–182. Springer, Heidelberg (2013)
10. Li, G., Ogawa, M., Yuen, S.: Nested timed automata with frozen clocks. In: Sankaranarayanan, S., Vicario, E. (eds.) FORMATS 2015. LNCS, vol. 9268, pp. 189–205. Springer, Heidelberg (2015)
11. Choffrut, C., Goldwurm, M.: Timed automata with periodic clock constraints. *J. Automata, Lang. Comb.* **5**, 371–404 (2000)
12. 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)
13. Trivedi, A., Wojtczak, D.: Recursive timed automata. In: Bouajjani, A., Chin, W.-N. (eds.) ATVA 2010. LNCS, vol. 6252, pp. 306–324. Springer, Heidelberg (2010)

14. Wen, Y., Li, G., Yuen, S.: An over-approximation forward analysis for nested timed automata. In: Liu, S., Duan, Z. (eds.) SOFL+MSVL 2014. LNCS, vol. 8979, pp. 62–80. Springer, Heidelberg (2015)
15. Minsky, M.: *Computation: Finite and Infinite Machines*. Prentice-Hall, Upper Saddle River (1967)
16. Bouyer, P.: Forward analysis of updatable timed automata. *Formal Methods in System Design* **24**, 281–320 (2004)
17. Fang, B., Li, G., Fang, L., Xiang, J.: A refined algorithm for reachability analysis of updatable timed automata. In: *Proceedings of the 1st IEEE International Workshop on Software Engineering and Knowledge Management (SEKM 2015 @ QRS 2015)*, IEEE Computer Society, pp. 230–236 (2015)