SCIENCE CHINA
Information Sciences

Updatable timed automata with one updatable clock

Guoqiang LI[1,*], Yunqing WEN[1] and Shoji YUEN[2]

Citation: SCIENCE CHINA Information Sciences **61**, 012102 (2018); doi: 10.1007/s11432-016-9027-y

View online: https://engine.scichina.com/doi/10.1007/s11432-016-9027-y

View Table of Contents: https://engine.scichina.com/publisher/scp/journal/SCIS/61/1

Published by the Science China Press

SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

January 2018, Vol. 61 012102:1–012102:14
doi: 10.1007/s11432-016-9027-y

Updatable timed automata with one updatable clock

Guoqiang LI[1*], Yunqing WEN[1] & Shoji YUEN[2]

[1]*BASICS, School of Software, Shanghai Jiao Tong University, Shanghai* 200240*, China;*
[2]*Graduate School of Information Science, Nagoya University, Nagoya* 464-8601*, Japan*

Received 7 November 2016/Revised 15 December 2016/Accepted 9 February 2017/Published online 1 September 2017

Abstract Updatable timed automata (UTAs) proposed by Bouyer et.al., is an extension of timed automata (TAs) having the extra ability to update clocks in a more elaborate way than simply reset them to zero. The reachability of UTAs is generally undecidable, which can be easily gained by regarding a pair of clocks as updatable counters. This paper investigates a subclass of UTAs by restricting the number of updatable clocks to be one. We will show that (1) the reachability of general UTAs with one updatable clock (UTA1s) is still undecidable, and (2) that of UTA1s under diagonal-free constraints is decidable, and the complexity is PSPACE-complete. The former is achieved by encoding Minsky machines to the general UTA1s, where two counters are simulated by the updatable clock. The latter is gained by regarding a region of a UTA1 to be an unbounded digiword, and encoding sets of digiwords that are accepted by a one counter automaton where regions are generated as the value of the counter.

Keywords updatable timed automata, one counter automata, digiword, reachability problem

Citation Li G Q, Wen Y Q, Yuen S. Updatable timed automata with one updatable clock. Sci China Inf Sci, 2018, 61(1): 012102, doi: 10.1007/s11432-016-9027-y

1 Introduction

Timed automata (TAs), which are introduced by Alur and Dill [1], are one of the most-famous and most-established models for analyzing complex real-time systems. Quite a lot of researches have been devoted to extensions of timed automata, with much interest for classes whose emptiness problems are decidable.

Updatable timed automata (UTAs) [2, 3], which are an extension of timed automata, are based on the allowance to update the value of clocks in an elaborate way such as assignments to arbitrary values, and increment and decrement operations. Their decidability results have been deeply investigated in [4]. Furthermore, by restricting the behaviours of update clocks, a quite precise way the thin frontier between decidable and undecidable classes of updatable timed automata has been described. The undecidability is shown by simulating the Minsky machine [5] in general, while the decidability is shown by the fact that the number of constructed regions is finite. Recently, a forward analysis of UTAs has been proposed in [6] for a special subclass of UTAs that do not use comparisons between clocks. A refined algorithm for a special subclass of UTAs with diagonal constraints has also been proposed in [7].

In this paper, We will investigate an interesting subclass of UTAs which lies between them. We restrict the number of updatable clocks in a UTA to be one (UTA1s). The updatable clock has the full behaviours of addition and subtraction. The paper firstly proves that the reachability of general UTA1s is still undecidable, by encoding a Minsky machine into a UTA1 in a more elaborate way, such

* Corresponding author (email: li.g@sjtu.edu.cn)

that two counters of a Minsky machine are encoded by one updatable clock. The technique combines two traditional methods, the N-wrapping technique [8] and the digital encoding [4] together, which is quite new. Afterwards, a positive answer for the reachability of UTA1s with diagonal-free constraints is given by constructing unbounded number of regions on-the-fly over the counter of one counter automata (OCAs), instead of pushdown systems [9], to gain a finer complexity result. PSPACE-complete complexity for the reachability of the decidable UTA1 subclass is thus proved. Our result expands the thin frontier between undecidable and decidable subclasses of updatable timed automata. Furthermore, the decidable model is also useful for soft real-time systems verification and modelling, since the singleton updatable clock can be used to describe the deadline of a task that may update due to different environments.

Besides their theoretical interests, UTA1s can be effectively used in analysis and modelling of soft real-time system, where the singleton updatable clock can be used to simulate the relative deadline, which is flexibly modified depending on different conditions and environments. These applications cannot be modeled and analyzed by the existing decidable UTA subclasses since each of them is less of some update abilities to guarantee decidability.

The rest of this paper is organized as follows. Section 2 gives an introduction of UTAs and OCAs. Section 3 introduces the model UTAs with one updatable clock. Section 4 gives an undecidability proof of the general UTA1s, and Section 5 gives a decidability proof and complexity result of the UTA1s with diagonal-free constraints. Section 6 talks about related work and Section 7 concludes the paper.

2 Preliminaries

For finite words $w = \gamma_1\gamma_2\dots\gamma_n$, we denote $\gamma_i \in w$ for $0 \leqslant i \leqslant n$. Let $\mathbb{R}^{\geqslant 0}$, $\mathbb{N}$ and $\mathbb{Z}$ denote the sets of non-negative real numbers, natural numbers and integral numbers, respectively. Let $\mathbb{N}^\omega = \mathbb{N} \cup \{\omega\}$, where ω 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 $(a, b), [a, b), [a, c], (a, c]$, where $a, c \in \mathbb{N}$, $b \in \mathbb{N}^\omega$, '(' and ')' denote open limits, and '[' and ']' denote closed limits. For a number $r \in \mathbb{R}^{\geqslant 0}$ 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, and let a clock valuation $\nu : X \to \mathbb{R}^{\geqslant 0}$, assigns a value to each clock $x \in X$. We define $\nu_{\mathbf{0}}$ represents all clocks in X are assigned to zero. Given a clock valuation ν and a time $t \in \mathbb{R}^{\geqslant 0}$, $(\nu + t)(x) = \nu(x) + t$, 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. $\mathrm{Val}(X)$ is the set of clock valuations of X.

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

$$\mathrm{con}_{\mathrm{df}} ::= x \in I?$$
$$\mathrm{con} ::= x \in I? \mid x - y \in I?$$

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

2.1 Updatable timed automata

Updatable timed automata (UTAs) [2–4], which is an extendention of TAs, provide a more powerful way to adjust the value of clocks 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 ϕ is one of the following:

Local ϵ, 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 $x \in X$.

For 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. In this paper, we will use similar notations for other models.

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

- Progress transition: $(q, \nu) \xrightarrow{t}_{\mathscr{A}} (q, \nu + t)$, where $t \in \mathbb{R}^{\geqslant 0}$.
- Discrete transition: $(q_1, \nu_1) \xrightarrow{\phi}_{\mathscr{A}} (q_2, \nu_2)$, 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 $\nu_1(x) \geqslant 1$ holds.

The initial configuration of a UTA is $(q_0, \nu_{\mathbf{0}})$.

Remark 1 (Sound simulation)**.** The UTA definition in Definition 2 follows the style in [10,11]. In [4], a UTA allows a logical connection of several constraint tests, e.g. $x \leqslant 15 \wedge y > 20$, and several resets operations of different clocks during one discrete transition. During one discrete transition only one test or assignment (which is a generalization of the reset) is allowed in this definition. Since a discrete transition is followed by a progress transition in which time elapses, the main ambiguity of two definitions is whether a conjunction of two tests can be checked one by one, between which the time elapses. It can be shown that the TA with our definition simulates the timed traces in the original definition soundly, as follows:

For $\geqslant$ or $>$, $c \xrightarrow{x\in[a,+\infty)?}_{\mathscr{A}} c' \xrightarrow{t}_{\mathscr{A}} c'$ can be safely converted to $c \xrightarrow{t}_{\mathscr{A}} c \xrightarrow{[a,+\infty)?}_{\mathscr{A}} c'$, for some configurations c and c'.

For $\leqslant$ or $<$, $c \xrightarrow{t}_{\mathscr{A}} c \xrightarrow{x\in[0,a]?}_{\mathscr{A}} p'$ can be safely converted to $c \xrightarrow{x\in[0,a]?}_{\mathscr{A}} c' \xrightarrow{t}_{\mathscr{A}} c'$, for some configurations c and c'.

For test transitions, a very general simulation strategy is as follows. Firstly, check the $\geqslant$, and $>$ one by one. Then check $\leqslant$ and $<$ later. If there exists a "=" constraint, decompose it into $\geqslant \wedge \leqslant$. For example, a transition $p \xrightarrow{x\leqslant 15\wedge y>20} q$ in the original definition can be simulated by two transitions $p \xrightarrow{y\in(20,+\infty)?} p' \xrightarrow{x\in[0,15]?} q$ under the new definition, in which p' is a new control location.

For reset transitions, a group of clocks which are reseted simultaneously can be simulated by resetting those clocks one by one sequentially, with a zero test of the first reset clock on the tail. For example, a transition $p \xrightarrow{\{x,y\}} q$, resetting x and y simultaneously, in the original definition is simulated by $p \xrightarrow{x\leftarrow[0,0]} p' \xrightarrow{y\leftarrow[0,0]} p'' \xrightarrow{x\in[0,0]?} q$, in which p', p'' are fresh control locations.

The arbitrary update, $p \xrightarrow{x:=x-d} q$, in which d is some fixed positive integer, can be encoded by introducing an extra clock x' and locations $p_1, p_2, \ldots, p_d, p_{d+1}$: $p \xrightarrow{x'\leftarrow[0,0]} p_1 \xrightarrow{x:=x-1} p_2 \xrightarrow{x:=x-1} p_3 \cdots p_d \xrightarrow{x:=x-1} p_{d+1} \xrightarrow{x'\in[0,0]?} q$.

If a transition contains both test and reset operations, we simulate test operations, reset operations and update operations respectively, following the above rules.

If only diagonal-free constraints appear in test transitions, the corresponding UTAs are called diagonal-free UTAs. Actually diagonal or diagonal-free constraints affect decidability results of UTAs heavily, which is different from TAs. We list a few undecidable subclasses of UTAs as follows:

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 One counter automata

Counter automata are a fundamental computational model, known to be Turing-complete [5]. One counter automata (OCAs) [12, 13], is one interesting model in subclasses of counter automata in which reachability is decidable.

Definition 4 (One counter automata). A one counter automaton (OCA) is a quadruple $\mathcal{C} = \langle P, p_0, \Delta \rangle$ where

- P is a finite set of control states,
- $p_0 \in P$ is the initial control state, and
- $\Delta \subseteq P \times Op \times P$ is a finite set of transitions, where $Op = \{\mathtt{zero}, \mathtt{nonzero}\} \cup \{\mathtt{add(z)} \mid \mathtt{z} \in \{1, -1\}\}$.

$(p, op, p') \in \Delta$ is written as $p \overset{op}{\hookrightarrow} p'$.

A configuration of the OCA $\mathcal{C}$ is a pair $(p, \mathtt{n})$, where $p \in P$ is a control state and $\mathtt{n} \in \mathbb{N}$ is the value of the counter. The transition relation on the locations of $\mathcal{C}$ induces an unlabelled transition relation on configurations as follows:

- An edge $p \overset{\mathtt{zero}}{\hookrightarrow} p'$ yields a single transition $\langle p, \mathtt{0} \rangle \hookrightarrow \langle p', \mathtt{0} \rangle$.
- An edge $p \overset{\mathtt{nonzero}}{\hookrightarrow} p'$ yields a single transition $\langle p, \mathtt{n} \rangle \hookrightarrow \langle p', \mathtt{n} \rangle$, where $\mathtt{n} > 0$.
- An edge $p \overset{\mathtt{add(z)}}{\hookrightarrow} p'$ yields a transition $\langle p, \mathtt{n} \rangle \hookrightarrow \langle p', \mathtt{n} + \mathtt{z} \rangle$, provided that both $\mathtt{n}$ and $\mathtt{n} + \mathtt{z}$ are both non-negative.

An OCA can be seen as a special case of a pushdown system [14] which has only one stack symbol. So the configuration reachability problem is naturally guaranteed.

Fact 1. The configuration reachability problem of OCAs is decidable [13, 15].

Remark 2. We restrict the values to added on the counter as ± 1, instead of arbitrary integer number in $\mathbb{Z}$ to gain the complexity for reachability of OCAs as in NLOGSPACE, as well as to properly encode. The latter will result in the complexity as NP-completeness [13].

3 General UTAs with one updatable clock

We propose a subclass of UTAs in a different facet by restricting the number of updatable clocks to be one, instead of by 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,
- $X = \{x_1, \ldots, x_k\}$ is a finite set of clocks, c is the singleton updatable clock, and
- $\Delta \subseteq Q \times \mathcal{O} \times Q$, where $\mathcal{O}$ is a set of operations. A transition $(q_1, \phi, q_2) \in \Delta$ is written as $q_1 \overset{\phi}{\rightarrow} q_2$, in which ϕ is one of the following:

Local ϵ, an empty operation,

Test $x \in I?$ or $x - x' \in I?$, where $x, x' \in X \cup \{c\}$ are two clocks 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 $c := c - 1$.

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, ν) of a control location $q \in Q$, and a clock valuation ν on $X \cup \{c\}$. The transition relation of the UTA1 is represented as follows:

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

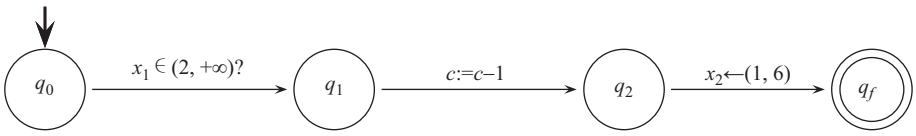

Figure 1 An Example for UTA1s.

- 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 $\phi = x \in I?$ 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 $\phi = c := c + 1$, $\nu_2 = \nu_1[c \leftarrow \nu_1(c) + 1]$.
Decrement $\phi = c := c - 1$, $\nu_2 = \nu_1[c \leftarrow \nu_1(c) - 1]$ and $\nu_1(c) \geqslant 1$ holds.

The initial configuration is $(q_0, \nu_{\mathbf{0}})$. The transition relation is $\rightarrow$ and we define $\rightarrow = \xrightarrow{t}_{\mathscr{A}} \cup \xrightarrow{\phi}_{\mathscr{A}}$, and define $\rightarrow^*$ to be the reflexive and transitive closure of $\rightarrow$.

Example 1. The UTA1 illustrated in Figure 1 has three clocks, c, x_1 and x_2, among which c is the special singleton updatable clock allowed to be updated in a different way.

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

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

The first transition is a progress transition in which time elapses 2.5 units, and each clock are increased by 2.5. The second transition tests whether the value of clock x_1 is greater than 2, and since it is true, it moves from location q_0 to location q_1. The third transition updates the updatable clock c by decreasing one time unit and make a move from location q_1 to location q_2. The last but one transition picks a value (here 5.7 is picked) from the range $(1, 6)$ randomly and assigns it to the clock x_2. The last transition is a progress transition in which time elapses 5 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 $(q_0, \nu_{\mathbf{0}}) \rightarrow^* (q, \nu)$, for some ν.

4 UTA1 with diagonal constraints is undecidable

Generally, to prove the undecidable result on reachability problems of a timed model, there are two methods including the N-wrapping technique [8], and the digital encoding [4], both of which encode Minsky machine into the target timed model. The former includes, Hybrid automata [8], task automata [16], NeTA-F [11], etc., while the latter includes updatable time automata [2,3], etc.. For general undecidable result of the UTA1, we need to combine the two methods together.

4.1 Value passing

Fact 2. The reachability problem of a UTA1 with value passing between the updatable clock and normal clocks is undecidable.

The fact is quite trivial if value passing is allowed. All normal clocks are essentially updatable via value passing through the special updatable clock.

Firstly, we will show that we can implement value passing between the updatable clock and normal clocks, although the operation is imperfect. That is, when value passing happens, values of other clocks will change.

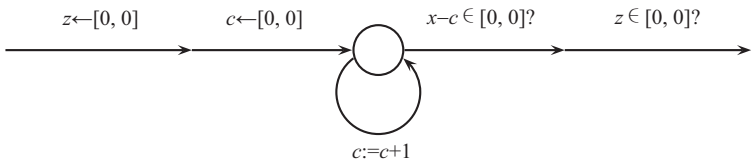

Figure 2 Value passing from a normal clock to the updatable clock.

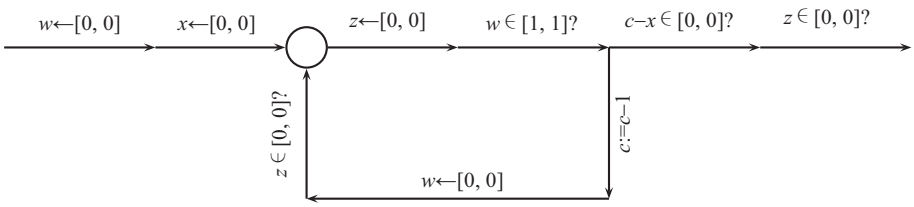

Figure 3 Value passing from the updatable clock to a normal clock.

Value passing from a normal clock to the updatable clock: Given a normal clock x and an updatable clock c, the value passing from x to c is simple, which in shown in Figure 2, where z is a normal clock used to prevent time passage through transition. During the value passing, the value of other clocks will not change. We use ValuePassingXC(x) to represent the function while x is parameter of normal clock.

Value passing from the updatable clock to a normal clock: Given the updatable clock c, and a normal clock x, the value passing from c to x is shown in Figure 3, which firstly resets the x , and then every 1 time unit (captured by the clock of w) judges whether value of x is equal to c. If not, the value of c will be decreased 1 time unit and repeatedly check after 1 time unit time passage (which means, value of c will be remain unchanged). Similarly, z is a normal clock to prevent time passage through transitions. Note that after value passing from c to x, the value of other clocks will change, which makes our encoding difficult. We use ValuePassingCX(x) to represent the function while x is parameter of normal clock.

4.2 Undecidability proof

For showing the undecidability, we encode the halting problem of a Minsky machine [5] to that of a UTA1. A Minsky machine $\mathcal{M}$ is a tuple (L, C, D) where,

- L is a finite set of states, and $l_f \in L$ is the terminal state,
- $C = \{ct_1, ct_2\}$ is the set of two counters,
- D is the finite set of transition rules of the following types,

Increment counter $d_i : ct := ct + 1$, goto l_k,

Test-and-decrement counter d_i : if $(ct > 0)$ then $(ct := ct - 1, \text{goto} l_k)$ else goto l_m, where $ct \in C$, $d_i \in D$ and l_k, $l_m \in L$.

Since the value passing encoding is imperfect, we will use another way to encode two counters, borrowing some ideas from N-wrapping techniques.

A Minsky machine can be encoded into a UTA1 with two normal clocks z and w, where z is used to prevent time passage during transitions and w is adopted to capture exact 1 time unit. The value of unique updatable clock is used to represent the value of two counters. That is, $\mu(c) = 2^{ct_1} \cdot 3^{ct_2}$. Incrementing and decrementing the counter ct_1 are simulated by doubling and halving of the value of the clock c, respectively, while those for ct_2 are simulated by tripling and thirding the value of clock c, respectively. These operations are shown in Figure 4, respectively, and described below formally.

Doubling. Firstly reset x to zero. Then repeatedly decrease one time units of c after two time units time passage. When the value of c is equal to the value of x, it will be $2d$.

Halving. Firstly rese x to zero. Then repeatedly decrease two time units of c after one time unit time passage. When value of c is equal to value of x, then the value of c will be $d/2$. If after these operations, value of c is smaller than that of x, it means that the value of the first counter is zero (zero-test).

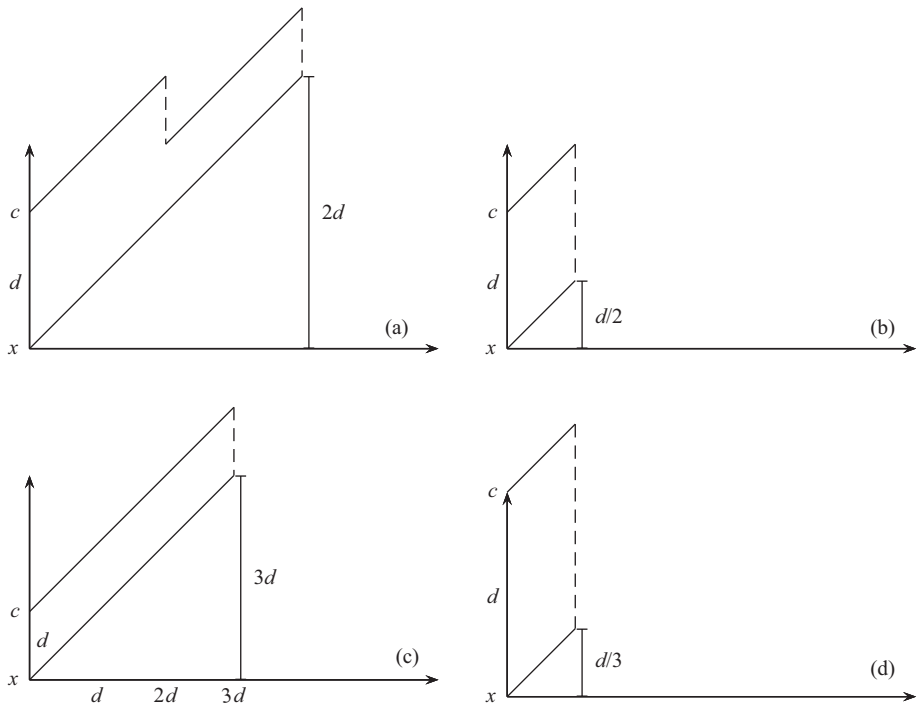

Figure 4 (a) Doubling d to $2d$; (b) halving d to $d/2$; (c) tripling d to $3d$; (d) thirding d to $d/3$.

Tripling. Firstly rese x to zero. Then repeatedly decrease one time unit of c after three time units time passage. When value of c is equal to the value of x, it will be $3d$.

Thirding. Firstly rese x to zero. Then repeatedly decrease three time units of c after one time unit time passage. When value of c is equal to value of x, then the value of c will be $d/3$. If after these operations, value of c is smaller than that of x, it means that the value of the second counter is zero (zero-test).

For the zero-test, we will apply halving (and thirding) repeatedly until $x - c \in (0, \omega)$, then the value of ct_1 (and ct_2) is zero. After that the value of c should be recovered, which can be done as follows. For ct_1, we keep decreasing two time units of c after one time unit time passage until c is zero. At that time, the value of x is the same as the original value of c. ValuePassingXC(y) will be applied to restore value of c. Similar procedure can be applied to ct_2.

By the above encoding, the reachability problem of Minsky machine can be encoded to the reachability problem of general UTA1, and thus we know that the latter is undecidable.

5 UTA1 with diagonal-free constraints is decidable

In this section, the reachability problem of UTA1s with diagonal-free constraint ($\mathrm{UTA1_{df}}$) is shown to be decidable. The main idea is to encode $\mathrm{UTA1s_{df}}$ to OCAs, whose configuration reachability is nicely decidable. The key simulation is that when the particular clock's value is greater than the the maximum integer n and meanwhile its value changes between integer and non-integer with time elapsing, we add the counter to some certain value to record the time interval.

5.1 Digiword and its operations

The powerset of D is denoted by $\mathcal{P}(D)$. Let $\mathcal{A} = (Q, q_0, F, X, c, \Delta)$ be a $\mathrm{UTA1_{df}}$, and n be the maximum integer appearing in Δ. For $v \in \mathbb{R}^{\geqslant 0}$, $\mathrm{proj}(v) = \mathtt{r}_i$ if $v \in \mathtt{r}_i \in \mathrm{Intv}(n)$, where

$$\mathrm{Intv}(n) = \{\mathtt{r}_{2i} = [i, i] \mid 0 \leqslant i \leqslant n\} \cup \{\mathtt{r}_{2i+1} = (i, i+1) \mid 0 \leqslant i < n\} \cup \{\mathtt{r}_{2n+1} = (n, \omega)\}.$$

The following idea of digitization is inspired by [11, 17–20].

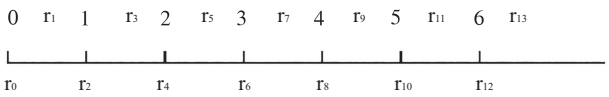

Figure 5 The intervals when $n = 6$.

Definition 8. For $(x,t) \in ((X \cup \{c\}) \times \mathbb{R}^{\geqslant 0})$, where t is the value of clock x, let $\mathrm{frac}(x,t) = t - \mathrm{floor}(t)$. A digitization $\mathrm{digi} : \mathrm{Val}(\{c\} \cup X) \to (\mathcal{P}((\{c\} \cup X) \times \mathrm{Intv}(n)))^*$ is defined as follows. For $\nu \in \mathrm{Val}(X \cup \{c\})$, let $Y_0, Y_1, \ldots, Y_m$ be sets that collect $(x, \mathrm{proj}(t))$'s having the identical $\mathrm{frac}(x,t)$ for $(x,t) \in (\{c\} \cup X) \times \mathbb{R}^{\geqslant 0}$. Among them, Y_0 (which is likely to be empty) is reserved to collect all $(x, \mathrm{proj}(t))$ with $\mathrm{frac}(t) = 0$. Let us assume that Y_i's except for Y_0 is not empty, and Y_i's are arranged by the ascending order of $\mathrm{frac}(x,t)$ (i.e., for $(x, \mathrm{proj}(t)) \in Y_i$ and $(x', \mathrm{proj}(t')) \in Y_j$, $\mathrm{frac}(x,t) < \mathrm{frac}(x',t')$, where $m \geqslant j > i \geqslant 0$.

Example 2. For example 1, we have $n = 6$ and 14 intervals. These intervals are illustrated as Figure 5.

For the clock valuation ν_4 in Example 1, $\mathrm{digi}(\nu_4)$ is $\{(c, \mathtt{r}_3), (x_1, \mathtt{r}_5)\}\{(x_2, \mathtt{r}_{11})\}$. For convenience we do not explicitly show the empty set Y_0 of $\mathrm{digi}(\nu_4)$.

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

Remark 3. The set of well-formed digiword is obviously finite, for a finite set of clocks $\{c\} \cup X$. This is trivial, since there are a constant number of clocks, which implies that there are a constant number of combinations of digiword. In our following encoding this plays a key role, ensuring the fact that the OCA has bounded states.

Definition 9. Let $\bar{Y} = Y_0 \cdots Y_m \in (\mathcal{P}((\{c\} \cup X) \times \mathrm{Intv}(n)))^*$. Digiword operations is defined as follows.

- **Insert**$_I$ $\mathrm{insert}(\bar{Y}, (x, \mathtt{r}_i))$ for $x \in X \cup \{c\}$ inserts $(x, \mathtt{r}_i)$ to $\bar{Y}$ at

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

- **Delete** $\mathrm{delete}(\bar{Y}, x)$ for $x \in X \cup \{c\}$ is obtained from $\bar{Y}$ by deleting the element $(x, \mathtt{r})$ indexed by x.
- **Increase** $\mathrm{increase}(\bar{Y}, c)$ is obtained from $\bar{Y}$ by replacing the element $(c, \mathtt{r}_i)$ indexed by c with element $(c, \mathtt{r}_{\min\{i+2, 2n+1\}})$.
- **Decrease** $\mathrm{decrease}(\bar{Y}, c, d)$ where $d \in \mathbb{N}$ is obtained from $\bar{Y}$ by replacing the element $(c, \mathtt{r}_i)$ indexed by c with element $(c, \mathtt{r}_{\max\{i-d, 0\}})$.
- **Shift** Let $j \in [0..m]$ and $0 \leqslant i \leqslant 2n+1$. 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.

$$\begin{cases} \bar{Y}' = Y_0', Y_1', \ldots, Y_{m+1}', & \text{if } Y_0 \neq \emptyset,\ Y_0' = \emptyset,\ Y_1' = \{(x, \mathtt{r}_{\min\{i+1, 2n+1\}}) \mid (x, \mathtt{r}_i) \in Y_0\}, \\ & \text{and } Y_j' = Y_{j-1} \text{ for } j \in [2..m+1], \\ \bar{Y}' = Y_0', Y_1', \ldots, Y_{m-1}', & \text{if } Y_0' = \{(x, \mathtt{r}_{\min\{i+1, 2n+1\}}) \mid (x, \mathtt{r}_i) \in Y_m\}, \\ & \text{and } Y_j' = Y_j \text{ for } j \in [1..m-1]. \end{cases}$$

If $\{c, \mathtt{r}_i\} \in Y_0$ or, $Y_0 = \emptyset$ and $\{c, \mathtt{r}_i\} \in Y_m$ for a shift $\Rightarrow$, we call it a critical shift and denote it as $\Rightarrow^{\uparrow}$, otherwise call it non-critical shift and denote it as $\Rightarrow^{\rightarrow}$. Define $\Rightarrow^*$ as reflexive transitive closure of $\Rightarrow$ as convention.

Example 3. Take the digiword in Example 1 into consideration, $\mathrm{digi}(\nu_0) = \{(c, \mathtt{r}_0), (x_1, \mathtt{r}_0), (x_2, \mathtt{r}_0)\}$.

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

Similar to the regions [1], well formed digiwords are bisimilar to the clock valuations:

$$\bar{Y_1} \Rightarrow^* \bar{Y_2} \text{ if and only if for any } \nu \in [\bar{Y_1}], \text{ exists } t \in \mathbb{R}^{\geqslant 0} \text{ such that } \nu + t \in [\bar{Y_2}],$$

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

5.2 Decidability proof

Definition 10. For a UTA1$_{\text{df}}$ $\mathcal{A} = \langle Q, q_0, F, X, c, \Delta\rangle$, there is an OCA $\mathcal{C} = \langle P \times (\mathcal{P}((X \cup \{c\}) \times \text{Intv}(n)))^*, p_0, \Delta_d\rangle$, where the set of states $P = Q \cup \{p', p'', p''' \mid p \in Q \wedge p', p'', p''' \notin Q\}$ and $p_0 = (q_0, \{(x, \mathtt{r}_0) | x \in X \cup \{c\}\})$. Δ_d consists of the following:

- **Time progress**

(1) $(p, \bar{Y}) \overset{\texttt{add}(1)}{\longrightarrow} (p, \bar{Z})$, where $\bar{Y} \Rightarrow^{\uparrow} \bar{Z}$, for $(c, \mathtt{r}_i) \in Y_j \in \bar{Y}$ and $i \geqslant 2n$.
(2) $(p, \bar{Y}) \overset{\texttt{nonzero}}{\longrightarrow} (p, \bar{Z})$, where $\bar{Y} \Rightarrow^{\rightarrow} \bar{Z}$, for $(c, \mathtt{r}_i) \in Y_j \in \bar{Y}$ and $i \geqslant 2n$.
(3) $(p, \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (p, \bar{Z})$, where $\bar{Y} \Rightarrow \bar{Z}$, for $(c, \mathtt{r}_i) \in Y_j \in \bar{Y}$ and $i < 2n$.

- **Local** $(p \xrightarrow{\epsilon} q \in \Delta)$

(1) $(p, \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (q, \bar{Y})$, $(p, \bar{Y}) \overset{\texttt{nonzero}}{\longrightarrow} (q, \bar{Y})$.

- **Test** $(p \xrightarrow{x \in I?} q \in \Delta)$

(1) $(p, \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (q, \bar{Y})$, $(p, \bar{Y}) \overset{\texttt{nonzero}}{\longrightarrow} (q, \bar{Y})$, if $r_i \subseteq I$ for $(x, \mathtt{r}_i) \in Y_j \in \bar{Y}$.

- **Assignment**$_c$ $(p \xrightarrow{c \leftarrow I} q \in \Delta)$

(1) $(p, \bar{Y}) \overset{\texttt{add}(-1)}{\longrightarrow} (p, \bar{Y})$.
(2) $(p, \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (p', \bar{Y})$.
(3) $\forall \mathtt{r}_i \subseteq I \setminus \mathtt{r}_{2n+1}$, $(p', \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (q, \text{insert}(\text{delete}(\bar{Y}, c), (c, \mathtt{r}_i)))$.
(4) $(p', \bar{Y}) \overset{\texttt{add}(1)}{\longrightarrow} (p', \bar{Y})$, if $\mathtt{r}_{2n+1} \subseteq I$.
(5) $(p', \bar{Y}) \overset{\texttt{nonzero}}{\longrightarrow} (q, \text{insert}(\text{delete}(\bar{Y}, c), (c, \mathtt{r}_{2n+1})))$, if $\mathtt{r}_{2n+1} \subseteq I$.

- **Assignment**$_x$ $(p \xrightarrow{x \leftarrow I} q \in \Delta$, where $x \in X)$

(1) $\forall \mathtt{r}_i \subseteq I$, $(p, \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (q, \text{insert}(\text{delete}(\bar{Y}, x), (x, \mathtt{r}_i)))$.

- **Increment** $(p \xrightarrow{c := c+1} q \in \Delta)$

(1) $(p, \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (q, \text{increase}(\bar{Y}, c))$, for $(c, \mathtt{r}_i) \in Y_j \in \bar{Y}$ and $i \leqslant 2n - 2$.
(2) $(p, \bar{Y}) \overset{\texttt{add}(1)}{\longrightarrow} (q, \text{increase}(\bar{Y}, c))$, for $(c, \mathtt{r}_{2n-1}) \in Y_j \in \bar{Y}$.
(3) $(p, \bar{Y}) \overset{\texttt{add}(1)}{\longrightarrow} (p'', \bar{Y})$, for $(c, \mathtt{r}_i) \in Y_j \in \bar{Y}$ and $i \geqslant 2n$.
(4) $(p'', \bar{Y}) \overset{\texttt{add}(1)}{\longrightarrow} (q, \text{increase}(\bar{Y}, c))$.

- **Decrement** $(p \xrightarrow{c := c-1} q \in \Delta)$

(1) $(p, \bar{Y}) \overset{\texttt{add}(-1)}{\longrightarrow} (p''', \bar{Y})$.
(2) $(p''', \bar{Y}) \overset{\texttt{add}(-1)}{\longrightarrow} (q''', \bar{Y})$.
(3) $(q''', \bar{Y}) \overset{\texttt{nonzero}}{\longrightarrow} (q, \bar{Y})$.
(4) $(q''', \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (q, \text{decrease}(\bar{Y}, c, 1))$.
(5) $(p''', \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (q, \text{decrease}(\bar{Y}, c, 2))$.
(6) $(p, \bar{Y}) \overset{\texttt{zero}}{\longrightarrow} (q, \text{decrease}(\bar{Y}, c, 2))$ for $(c, \mathtt{r}_i) \in Y_j \in \bar{Y}$ and $i \geqslant 2$.

Given a UTA1$_{\text{df}}$ $\mathcal{A}$, for all kinds of transitions, we have their corresponding rules in the $\Delta_d(\mathcal{C})$. Some transitions of $\mathcal{C}$ seem hard to understand. The following gives a simple explanation for how to simulate assignment and decrement.

For an assignment in the form of $x \leftarrow I$, the OCA $\mathcal{C}$ proceeds with two different subcases: 1) x is the updatable clock; 2) x is a normal clock. For the former case, $\rfloor$ firstly needs to decrease the counter to zero, then add some value whenever needed, and finally perform delete and insert operations. It is much simpler for the latter case, merely performing delete and insert operations.

For a decrement in the form of $c := c - 1$, the OCA $\mathcal{C}$ proceeds with four different subcases: 1) $\nu(c) > n + 1$; 2) $\nu(c) = n + 1$; 3) $n < \nu(c) < n + 1$; 4) $1 \leqslant \nu(c) \leqslant n$. For the first case, the

counter in $\mathcal{C}$ only performs $\mathtt{add}(-1)$ twice. For the second and third cases, the counter in $\rfloor$ performs $\mathtt{add}(-1)$ twice and once, respectively, and further performs decrease operation. The difference between them is the unit it decreases. For the last case, the value of the counter in $\mathcal{C}$ is zero. The system only performs decrease operation.

The following example illustrates the simulation of assignment transitions and decrement transitions.

Example 4. Take the run in the Example 1 into consideration.

• $(q_1, \nu_2) \xrightarrow{c:=c-1} (q_2, \nu_3)$, 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: $(q_1, \mathrm{digi}(\nu_2)) \xhookrightarrow{\mathtt{zero}} (q_2, \mathrm{decrease}(\mathrm{digi}(\nu_2), c, 2))$.

• $(q_2, \nu_3) \xrightarrow{x_2 \leftarrow (1,6)} (q_f, \nu_4)$, where ν_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: $(q_2, \mathrm{digi}(\nu_3)) \xhookrightarrow{\mathtt{zero}} (q_f, \mathrm{insert}(\mathrm{delete}(\mathrm{digi}(\nu_3), x_2), (x_2, \mathtt{r}_{11})))$.

Definition 11. Let ϱ be any configuration of a $\mathrm{UTA1_{df}}$ such that $\varrho_0 = (q_0, \nu_0) \hookrightarrow^* \varrho = (q, \nu)$. Define $[\![\varrho]\!]$ to be $\langle (q, \mathrm{digi}(\nu)), \mathtt{k}\rangle$, where $\mathtt{k} = 2 \times \mathrm{floor}(\nu(c) - n) + \mathrm{ceiling}(\mathrm{frac}(\nu(c)))$ if $\nu(c) > n$; otherwise $\mathtt{k} = 0$. A configuration κ of OCA $\mathcal{C}$ with some ϱ and $\kappa = [\![\varrho]\!]$ is named an encoded configuration.

Example 5. Consider the configuration (q_f, ν_5) in Example 1, where $\nu_5 = \{c = 6.5, x_1 = 7.5, x_2 = 10.7\}$. By Definition 11, $[\![(q_f, \nu_5)]\!] = \langle (q_f, \mathrm{digi}(\nu_5)), 1\rangle$, where $\mathrm{digi}(\nu_5) = \{(c, \mathtt{r}_{13}), (x_1, \mathtt{r}_{13})\}\{(x_2, \mathtt{r}_{13})\}$.

Remark 4. For a time progress transition $(p, \nu) \xrightarrow{t} (q, \nu')$ of $\mathrm{UTA1_{df}}$, where $t \in \mathbb{R}^{\geqslant 0}$, there exists a transition sequence of $[\![(p, \nu)]\!] \hookrightarrow \langle (p, \bar{Z}_1), \mathtt{k}_1\rangle \hookrightarrow \langle (p, \bar{Z}_2), \mathtt{k}_2\rangle \cdots \hookrightarrow [\![(q, \nu')]\!]$ to faithfully simulate it, including finite time progress transitions of OCA.

Lemma 1. Given a $\mathrm{UTA1_{df}}$ $\mathcal{A}$, its associated OCA $\mathcal{C}$, and any configuration ϱ, ϱ' of $\mathcal{A}$.

(**Preservation**) If $\varrho \to \varrho'$ then $[\![\varrho]\!] \hookrightarrow^* [\![\varrho']\!]$.

(**Reflection**) If $[\![\varrho]\!] \hookrightarrow^* \kappa$,

(1) there exists ϱ' such that $\kappa = [\![\varrho']\!]$ and $\varrho \to^* \varrho'$, or

(2) κ is not an encoded configuration, and there exists ϱ' such that $\kappa \hookrightarrow^* [\![\varrho']\!]$ by transitions (of $\mathcal{C}$) and $\varrho \to^* \varrho'$.

Proof. Let $\varrho = (q, \nu)$. We assume $[\![\varrho]\!] = \langle (q, \mathrm{digi}(\nu)), \mathtt{k}\rangle$.

For preservation part, we discuss by case analysis of $\varrho \to \varrho'$.

(1) **Time progress** $\varrho \xrightarrow{t} \varrho'$. We have $\mathrm{digi}(\nu) \Rightarrow^* \mathrm{digi}(\nu + t)$, by the digiword's region-like property. Discuss with two different subcases:

(a) If $\nu(c) + t \leqslant n$, by applying the third transition rule of time progress rules finite times we have $[\![\varrho]\!] = \langle (q, \mathrm{digi}(\nu)), 0\rangle \hookrightarrow^* [\![\varrho']\!] = \langle (q, \mathrm{digi}(\nu + t)), 0\rangle$.

(b) If $\nu(c) + t > n$, we get $[\![\varrho]\!] = \langle (q, \mathrm{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow^* [\![\varrho']\!] = \langle (q, \mathrm{digi}(\nu + t)), \mathtt{k} + \mathtt{k}'\rangle$ by applying the third time progress rule finitely many times (zero times if $\nu(c) \geqslant n$) and then iteratively applying the first time progress rule one time, the second time progress rule finitely many times, and another first time progress rule one time. Note that it is possible that fist time progress rule is applied zero times at the first iteration.

(2) **Local** $\varrho = (p, \nu) \xrightarrow{\epsilon} \varrho' = (q, \nu)$. Then we have $[\![\varrho]\!] = \langle (p, \mathrm{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow [\![\varrho']\!] = \langle (q, \mathrm{digi}(\nu)), \mathtt{k}\rangle$ with the local transition of OCA.

(3) **Test** $\varrho = (p, \nu) \xrightarrow{x \in I?} \varrho' = (q, \nu)$. Then $[\![\varrho]\!] = \langle (p, \mathrm{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow [\![\varrho']\!] = \langle (q, \mathrm{digi}(\nu)), \mathtt{k}\rangle$ with the test transition of OCA, since there exists $(x, \mathtt{r}_i) \in Y_j \in \mathrm{digi}(\nu)$ such that $\mathtt{r}_i \subseteq I$, where $\mathrm{digi}(\nu) = Y_1 Y_2 \cdots Y_j \cdots Y_m$.

(4) **Assignment**$_c$ $\varrho = (p, \nu) \xrightarrow{c \leftarrow I} \varrho' = (q, \nu[c \leftarrow d])$, where $d \in I$. We discuss with two different subcases, $d \leqslant n$ and $d > n$.

(a) $d \leqslant n$: We first need to substract the counter to zero, by repeatedly applying $\mathtt{add}(-1)$ from the first assignment$_c$ rule, which has $[\![\varrho]\!] = \langle (p, \mathrm{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow^* \kappa = \langle (p, \mathrm{digi}(\nu)), 0\rangle$. Then we have $\kappa \hookrightarrow \kappa' = \langle (p', \mathrm{digi}(\nu)), 0\rangle \hookrightarrow [\![\varrho']\!] = \langle (q, \mathrm{digi}(\nu[x \leftarrow d])), 0\rangle$, by applying the second and third assignment rules.

(b) $d > n$: Similar to the above subcase, We firstly need to substract the counter to zero, by repeatedly applying $\mathtt{add}(-1)$ from the first assignment$_c$ rule, having $[\![\varrho]\!] = \langle (p, \mathrm{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow^* \kappa = \langle (p, \mathrm{digi}(\nu)), 0\rangle$. Secondly, by the second assignment$_c$ rule we have $\kappa \hookrightarrow \kappa' = \langle (p', \mathrm{digi}(\nu)), 0\rangle$. Then, we have $\kappa' \hookrightarrow^* \kappa'' = \langle (p', \mathrm{digi}(\nu)), \mathtt{k}\rangle$ by repeatedly applying the fourth rule of $\mathcal{C}$ until the counter $\mathtt{k} = 2 \times \mathrm{floor}(d - n) +$

ceiling(frac(d)). Finally, we have $\kappa'' \hookrightarrow [\![\varrho']\!] = \langle(q, \text{digi}(\nu[c \leftarrow d])), \mathtt{k}\rangle$, by applying the last assignment rule.

(5) **Assignment**$_x$ $\varrho = (p, \nu) \xrightarrow{x \leftarrow I} \varrho' = (q, \nu[x \leftarrow d])$, where $x \in X$, and $d \in I$. By applying the first assignment rule of OCA, $(p, \bar{Y}), \epsilon\rangle \hookrightarrow \langle(q, \text{insert}(\text{delete}(\bar{Y}, x), (x, \mathtt{r}_i))), \epsilon\rangle$, where $\bar{Y} = \text{digi}(\nu)$ and $d \in \mathtt{r}_i$ We have $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow [\![\varrho']\!] = \langle(q, \text{digi}(\nu[x \leftarrow d])), \mathtt{k}\rangle$.

(6) **Increment** $\varrho = (p, \nu) \xrightarrow{c:=c+1} \varrho' = (q, \nu[c \leftarrow \nu(c) + 1])$. Discuss with three different subcases:

(a) If $\nu(c) \leqslant n-1$, then by applying the first transition increment rule of $\mathcal{C}$ we have $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{0}\rangle \hookrightarrow [\![\varrho']\!] = \langle(q, \text{increase}(\text{digi}(\nu), c)), \mathtt{0}\rangle = \langle(q, \text{digi}(\nu[c \leftarrow \nu(c) + 1])), \mathtt{0}\rangle$.

(b) If $n - 1 < \nu(c) < n$, then by applying the second increment transition rule of $\mathcal{C}$ we have $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow [\![\varrho']\!] = \langle(q, \text{increase}(\text{digi}(\nu), c)), \mathtt{k} + 1\rangle = \langle(q, \text{digi}(\nu[c \leftarrow \nu(c) + 1])), \mathtt{k} + 1\rangle$.

(c) If $\nu(c) \geqslant n$, then by applying the third and the fourth increment transition rules of $\mathcal{C}$ we have $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow [\![\varrho']\!] = \langle(q, \text{increase}(\text{digi}(\nu), c)), \mathtt{k} + 2\rangle = \langle(q, \text{digi}(\nu[c \leftarrow \nu(c) + 1])), \mathtt{k} + 2\rangle$.

(7) **Decrement** $\varrho = (p, \nu) \xrightarrow{c:=c-1} \varrho' = (q, \nu[c \leftarrow \nu(c) - 1])$. Note that only when $\nu(c) \geqslant 1$, can this transition happen. We proceed with 4 subcases:

(a) If $\nu(c) > n + 1$, then we have $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow^* [\![\varrho']\!] = \langle(q, \text{digi}(\nu)), \mathtt{k} - 2\rangle = \langle(q, \text{digi}(\nu[c \leftarrow \nu(c) - 1])), \mathtt{k} - 2\rangle$ by applying the 1–3 decrement transition rules of $\mathcal{C}$.

(b) If $\nu(c) = n + 1$, then we have $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{2}\rangle \hookrightarrow^* [\![\varrho']\!] = \langle(q, \text{decrease}(\text{digi}(\nu), c, 1)), \mathtt{0}\rangle = \langle(q, \text{digi}(\nu[c \leftarrow \nu(c) - 1])), \mathtt{0}\rangle$ by applying the 1,2,4 decrement transition rules of $\mathcal{C}$.

(c) If $n < \nu(c) < n + 1$, then we have $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{1}\rangle \hookrightarrow^* [\![\varrho']\!] = \langle(q, \text{increase}(\text{digi}(\nu), c, 2)), \mathtt{0}\rangle = \langle(q, \text{digi}(\nu[c \leftarrow \nu(c) - 1])), \mathtt{0}\rangle$ by applying the 1,5 decrement transition rules of $\mathcal{C}$.

(d) If $1 \leqslant \nu(c) \leqslant n$, then we have $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{0}\rangle \hookrightarrow [\![\varrho']\!] = \langle(q, \text{increase}(\text{digi}(\nu), c, 2)), \mathtt{0}\rangle = \langle(q, \text{digi}(\nu[c \leftarrow \nu(c) - 1])), \mathtt{0}\rangle$ by applying the 6 decrement transition rule of $\mathcal{C}$.

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

Base step. Consider the case of $[\![\varrho]\!] \hookrightarrow \kappa$,

(1) **Time progress** Apparently, we have $[\![\varrho]\!] \hookrightarrow \kappa$ by one of three time progress rules of OCA $\mathcal{C}$. $\text{digi}(\nu) \Rightarrow \bar{Y}$ means that there exists a clock valuation $\nu' \in \text{Val}(\{c\} \cup X)$ such that $\nu' = \nu + t$ and $\nu' \in [\bar{Y}]$ for a real number t ($0 < t < 1$). Now discuss with three cases:

(a) If $\nu(c) < n$, $[\![\varrho]\!] \hookrightarrow \kappa$ by using the third time progress rule of OCA $\mathcal{C}$. We get that $\kappa = \langle(p, \bar{Y}), 0\rangle = [\![\varrho']\!]$, where $\varrho = (p, \nu) \xrightarrow{t} \varrho' = (p, \nu')$.

(b) If $\nu(c) \geqslant n$ and $2 \times \text{floor}(\nu(c)) + \text{ceiling}(\text{frac}(\nu(c))) = 2 \times \text{floor}(\nu'(c)) + \text{ceiling}(\text{frac}(\nu'(c)))$,then by using the second time progress rule we get $[\![\varrho]\!] \hookrightarrow \kappa$. We have $\kappa = \langle(p, \bar{Y}), \mathtt{k}\rangle = [\![\varrho']\!]$, where $\varrho = (p, \nu) \xrightarrow{t} \varrho' = (p, \nu')$.

(c) If $\nu(c) \geqslant n$ and $2 \times \text{floor}(\nu(c)) + \text{ceiling}(\text{frac}(\nu(c))) + 1 = 2 \times \text{floor}(\nu'(c)) + \text{ceiling}(\text{frac}(\nu'(c)))$, then by using the first time progress rule we have $[\![\varrho]\!] \hookrightarrow \kappa$. We get $\kappa = \langle(p, \bar{Y}), \mathtt{k} + 1\rangle = [\![\varrho']\!]$, where $\varrho = (p, \nu) \xrightarrow{t} \varrho' = (p, \nu')$.

(2) **Local** If $[\![\varrho]\!] \hookrightarrow \kappa$ for $p \xrightarrow{\epsilon} q$ being a transition in $\mathcal{A}$, then $\kappa = \langle(q, \text{digi}(\nu)), \mathtt{k}\rangle = [\![\varrho']\!]$, where $\varrho' = (q, \nu)$.

(3) **Test** Similar with the case for **Local**.

(4) **Assignment**$_x$ If $[\![\varrho]\!] \hookrightarrow \kappa$, κ is an encoded configuration. Then $[\![\varrho]\!] = \langle(p, \text{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow \kappa = [\![\varrho']\!] = \langle(q, \text{insert}(\text{delete}(\text{digi}(\nu), x), (x, \mathtt{r}_i))), \mathtt{k}\rangle$, where for $d \in \mathtt{r}_i \in I$, $\varrho = (p, \nu) \xrightarrow{x \leftarrow I} \varrho' = (q, \nu[x \leftarrow d])$.

(5) **Assignment**$_c$ We only need to consider the first two rules since the configurations are encoded configurations (i.e. $[\![\varrho]\!]$).

(a) If by applying the first assignment transition rule of $\mathcal{C}$ $[\![\varrho]\!] \hookrightarrow \kappa$, the clock involved is c and κ may not be an encoded configuration. However, by applying the same rules finitely many times and then the third assignment rule, $\kappa \hookrightarrow^* \kappa_1 = \langle(p, \text{digi}(\nu)), \mathtt{0}\rangle \hookrightarrow \kappa_2 = \langle(p', \text{digi}(\nu)), \mathtt{0}\rangle$, and then if $\mathtt{r}_i \in I$ for $i \leqslant 2n$, we have $\kappa_2 \hookrightarrow [\![\varrho']\!] = \langle(q, \text{insert}(\text{delete}(\text{digi}(\nu), c), (c, \mathtt{r}_i)), \mathtt{0}\rangle$, by applying the third assignment rule; Otherwise, we have $\kappa_2 \hookrightarrow^* \kappa_3 = \langle(p', \text{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow [\![\varrho']\!] = \langle(q, \text{insert}(\text{delete}(\text{digi}(\nu), c), (c, \mathtt{r}_i)), \mathtt{k}\rangle$ by applying finitely many times of the fourth assignment rule firstly and then the last assignment rule.

(b) If by applying the first assignment transition rule of $\mathcal{C}$ we have $[\![\varrho]\!] \hookrightarrow \kappa$, then the proof is similar to the above case.

(6) **Increment** Discuss with three different cases:

(a) If the first increment transition rule is taken, then $\nu(c) <= n-1$ can be inferred. Hence we get $[\![\varrho]\!] \hookrightarrow \kappa = \langle(q, \text{increase}(\text{digi}(\nu), c)), \mathtt{0}\rangle = [\![\varrho']\!]$, where $\varrho \xrightarrow{c:=c+1} \varrho'$.

(b) If the second increment transition rule is taken, then $n-1 < \nu(c) < n$ can be inferred. Then $[\![\varrho]\!] \hookrightarrow \kappa = \langle(q, \text{increase}(\text{digi}(\nu), c)), \mathtt{k}\rangle = [\![\varrho']\!]$ for $\mathtt{k} > 0$, where $\varrho \xrightarrow{c:=c+1} \varrho'$.

(c) If the third increment transition rule is taken, then κ is not an encoded confirguration, then by the fourth rule $\kappa \hookrightarrow \kappa'$, $\nu(c) \geqslant n$ can be inferred. Then we have $[\![\varrho]\!] \hookrightarrow^* \kappa' = \langle(q, \text{increase}(\text{digi}(\nu), c)), \mathtt{k}\rangle = [\![\varrho']\!]$ for $\mathtt{k} \geqslant 2$, where $\varrho \xrightarrow{c:=c+1} \varrho'$.

(7) **Decrement** In order to get an encoded configuration, we only have to discuss with four cases:

(a) If the 1-3 decrement transition rules are taken sequentially, $\nu(c) > n+1$ can be inferred. Then $[\![\varrho]\!] = \langle(q, \text{digi}(\nu)), \mathtt{k}\rangle \hookrightarrow^* \kappa = \langle(q, \text{digi}(\nu)), \mathtt{k}-2\rangle = [\![\varrho']\!]$ for $\mathtt{k} > 2$, where $\varrho \xrightarrow{c:=c-1} \varrho'$. since $\nu(c) > n+1$, we have $\text{digi}(\nu) = \text{digi}(\nu[\nu(c) \leftarrow \nu(c)-1])$.

(b) If the 1,2,4 decrement transition rules are taken sequentially, $\nu(c) = n+1$ can be inferred. Then $[\![\varrho]\!] = \langle(q, \text{digi}(\nu)), \mathtt{2}\rangle \hookrightarrow^* \kappa = \langle(q, \text{decrease}(\text{digi}(\nu), c, 1)), \mathtt{0}\rangle = [\![\varrho']\!]$, where $\nu'(c) = n$ and $\varrho \xrightarrow{c:=c-1} \varrho' = (q, \nu')$.

(c) If the 1,5 decrement transition rules are taken, $n < \nu(c) < n+1$ can be inferred. Then $[\![\varrho]\!] = \langle(q, \text{digi}(\nu)), \mathtt{1}\rangle \hookrightarrow^* \kappa = \langle(q, \text{decrease}(\text{digi}(\nu), c, 2)), \mathtt{0}\rangle = [\![\varrho']\!]$, where $n-1 < \nu'(c) < n$ and $\varrho \xrightarrow{c:=c-1} \varrho' = (q, \nu')$.

(d) If the last decrement transition rule is taken, $1 <= \nu(c) <= n$ can be inferred. Then $[\![\varrho]\!] = \langle(q, \text{digi}(\nu)), \mathtt{0}\rangle \hookrightarrow \kappa = \langle(q, \text{decrease}(\text{digi}(\nu), c, 2)), \mathtt{0}\rangle = [\![\varrho']\!]$, where $\varrho \xrightarrow{c:=c-1} \varrho' = (q, \nu')$.

Induction step. We assume $[\![\varrho]\!] \hookrightarrow^* \kappa' \hookrightarrow \kappa$. Proceed with two different cases:

(1) κ' is an encoded configuration, and we have $[\![\varrho]\!] \hookrightarrow^* \kappa' = [\![\varrho']\!]$ by induction hypothesis. The remaining proof is similar to the base step.

(2) κ' is not an encoded configuration. The the rule is one of the first second and the fourth rules of assignment$_c$, the third rule of increment, and the first and second rules of the decrement. A proof for the fourth assignment$_c$ rule is given here. The other cases are similar. We assume $\kappa' = \langle(p', \bar{Y}), \mathtt{k}\rangle$ is obtained by applying the fourth assignment rule, then by applying the last assignment rule, we have $\kappa' \hookrightarrow \kappa = \langle(q, \text{digi}(\nu')), \mathtt{k}\rangle = [\![\varrho']\!] = [\![(q, \nu')]\!]$, where $\exists \nu \in [\bar{Y}]$ such that $\nu' = \nu[c \leftarrow d]$ for $d \in \mathtt{r}_{2n+1}$. Finally, put together we have $\varrho \rightarrow^* \varrho'$.

With Lemma 1, we have the following main results.

Theorem 1. The reachability problem of a UTA1$_{\text{df}}$ is decidable.

5.3 Complexity discussion

Similar to TAs, the complexity of the algorithm for checking the reachability of UTA1s$_{\text{df}}$ is exponential in the number of clocks and the length of the constants in the timing constraints. This blow-up in complexity is unavoidable. Since UTA1s$_{\text{df}}$ is a superclass of TAs, which is PSPACE lower bound [1].

The reachability of OCAs is in NLOGSPACE, if we bounds the value that added to the counter as ± 1 [15]. Let now $\mathcal{A}$ be a UTA1 and $\mathcal{D}$ be a set of digiwords. As explained, the reachability of $\mathcal{A}$ can be checked by the reachability problem of digiword OCAs $\mathcal{C}$. If we apply the algorithm recalled above and if we want to compute its complexity, we have to compute the space needed to encode a state of $\mathcal{C}$. Such a state is a pair $(q, \bar{Y})$ where q is a control location of $\mathcal{A}$ and $\bar{Y} \in \mathcal{D}$ a digiword. For encoding a digiword, it is sufficient to store a word with each of elements a pair, and with the length of the number of clocks, $|X(\mathcal{A})|$. Therefore, a control state of $\mathcal{C}$ can be encoded in polynomial space and reachability of UTA1s is in PSPACE, and hence the PSPACE-completeness.

6 Related work

Timed automata(TAs) [1] was proposed by Alur and Dill, after which lots of researchers had devoted to extensions of TAs. One of the extension that allowed to compare a constant with the sum of two

clocks, had also been investigated in [1]. This leaded to an undecidable class of automata. Periodic clock constraints defined in [21] can express properties like "the value of a clock is of the form $0.5 + 3n$ where n is some integer" or "the value of a clock is even".

Controlled real-time automata was proposed in [22]. It is a parameterized family of TAs with some extra features like variable clock velocities, clock stopping and periodic tests. Controlled real-time automata is still decidable thanks to carefully chosen restrictions.

Timed Counter Systems [23], was proposed by introducing counters into TAs. The reachability problem was generally undecidable, while three subclasses were deidable: timed VASS, bounded timed counter systems, and reversal-bounded timed counter systems. Priced timed automata [24] or weighted timed automata [25] extend timed automata with costs on both edges and locations. These costs increase while time elapses, but are never tested in one automaton. An interesting problem is then to compute the optimal cost for reaching a given state, which is proven decidable.

Updatable timed automata (UTAs) [2–4] failed remain some important theoretical properties of classical TAs. For instance, TAs with diagonal constraints are not more expressive, yet exponentially more concise than diagonal-free TAs [1,26]. Howver, as mentioned in [4], and in this paper, diagonal constraints affected expressiveness of UTAs quite a lot. In suspension automata [16], only bounded substraction clock updates were allowed. Such restriction guaranteed the decidability on both the reachability problem and the language inclusion problem. Integer reset TAs [27, 28] was a subclass of classical TAs since it can reset a clock only when it has integer value. Integer reset TAs were less expressive than classical TAs. It was decidable to check whether a timed regular language contains an integer reset timed regular language and integer reset TAs are closed under complementation.

An extension of TAs with recursive structure was recursive timed automata(RTAs) [29]. Clocks were divided by the mechanism of "pass-by-value". The reachability was proved to be decidable, when the condition of "glitch-freeness", that is all clocks of components were uniformly either by "pass-by-reference" or by "pass-by-value". Nested timed automata (NeTAs) [11,20,30] extended TAs with recursive structure in a different way. It allowed the current running clocks elapse simultaneously with clocks of some TAs in the stack during time passage. Those clocks of some TAs in the stack were called local clocks, while clocks of other TAs in the stack kept unaltered during time passage were called frozen clocks. It was shown that the reachability of NeTAs with a singleton global clock that can be observed by all TAs and both two types of clocks was decidable, while the reachability of NeTAs with two or more global clocks was proved to be undecidable [11].

7 Conclusion

The reachability of UTA1s, UTAs with one updatable clock, is investigated in this paper. Setting the number of updatable clocks to one, we shows a finer frontier between the UTA group: the reachability of general UTA1s is undecidable, while that of UTA1s with diagonal-constraints are decidable. The undecidability result is given by encoding Minsky machine to a general UTA1, whose the technique combines digital encoding and N-wrapping technique together. The decidability is proved by encoding the reachability problem of UTA1s_{df} to that of OCAs with the notion of digiword. The essential idea is to utilize a counter to record the time interval, which roughly measures how much the value of updatable clock exceeds the maximum integer.

Acknowledgements This work was supported by NSFC-JSPS Bilateral Joint Research Project (Grant No. 61511140100), and National Natural Science Foundation of China (Grant Nos. 61472240, 61602224, 91318301).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Alur R, Dill D L. A theory of timed automata. Theor Comput Sci, 1994, 126: 183–235

2 Bouyer P, Dufourd C, Fleury E, et al. Are timed automata updatable? In: Proceedings of the 12th International Conference on Computer Aided Verification, Chicago, 2000. 464–479
3 Bouyer P, Dufourd C, Fleury E, et al. Expressiveness of updatable timed automata. In: Proceedings of International Symposium on Mathematical Foundations of Computer, Bratislava, 2000. 232–242
4 Bouyer P, Dufourd C, Fleury E, et al. Updatable timed automata. Theor Comput Sci, 2004, 321: 291–345
5 Minsky M L. Computation: Finite and Infinite Machines. Upper Saddle River: Prentice-Hall, 1967
6 Bouyer P. Forward analysis of updatable timed automata. Form Method Syst Des, 2004, 24: 281–320
7 Fang B B, Li G Q, Fang L, et al. A refined algorithm for reachability analysis of updatable timed automata. In: Proceedings of IEEE International Conference on the Software Quality, Reliability and Security - Companion, Vancouver, 2015. 230–236
8 Henzinger T A, Kopke P W, Puri A, et al. What's decidable about hybrid automata? J Comput Syst Sci, 1998, 57: 94–124
9 Wen Y Q, Li G Q, Yuen S. On reachability analysis of updatable timed automata with one updatable clock. In: Proceedings of International Workshop on Structured Object-Oriented Formal Language and Method, Pairs, 2015, 147–161
10 Abdulla P A, Atig M F, Stenman J. Dense-timed pushdown automata. In: Proceedings of 27th Annual IEEE Symposium on Logic in Computer Science, Dubrovnik, 2012. 35–44
11 Li G Q, Ogawa M, Yuen S. Nested timed automata with frozen clocks. In: Proceedings of International Conference on Formal Modeling and Analysis of Timed Systems, Madrid, 2015, 189–205
12 Jancar P, Kucera A, Moller F, et al. DP lower bounds for equivalence-checking and model-checking of one-counter automata. Inform Comput, 2004, 188: 1–19
13 Haase C, Kreutzer S, Ouaknine J, et al. Reachability in succinct and parametric one-counter automata. In: Proceedings of the International Conference on Concurrency Theory, Bologna, 2009. 369–383
14 Schwoon S. Model-checking pushdown system. Dissertation for Ph.D. Degree. Munich: Technical University of Munich, 2000
15 Demri S, Gascon R. The effects of bounding syntactic resources on presburger LTL. J Logic Comput, 2009, 19: 1541–1575
16 Fersman E, Krcal P, Pettersson P, et al. Task automata: schedulability, decidability and undecidability. Inform Comput, 2007, 205: 1149–1172
17 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, Turku, 2004. 54–63
18 Abdulla P A, Jonsson B. Verifying networks of timed processes. In: Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Lisbon, 1998. 298–312
19 Abdulla P A, Jonsson B. Model checking of systems with many identical time processes. Theor Comput Sci, 2003, 290: 241–264
20 Li G Q, Cai X J, Ogawa M, et al. Nested timed automata. In: Proceedings of International Conference on Formal Modeling and Analysis of Timed Systems, Buenos Aires, 2013. 168–182
21 Choffrut C, Goldwurm M. Timed automata with periodic clock constraints. J Autom Lang Comb, 2000, 5: 371–404
22 Demichelis F, Zielonka W. Controlled timed automata. In: Proceedings of the 9th International Conference on Concurrency Theory, Nice, 1998. 455–469
23 Bouchy F, Finkel A, Sangnier A. Reachability in timed counter systems. Electr Notes Theor Comput Sci, 2009, 239: 167–178
24 Alur R, La Torre S, Pappas G J. Optimal paths in weighted timed automata. In: Proceedings of International Workshop on Hybrid Systems: Computation and Control, Rome, 2001 , 49–62
25 Behrmann G, Fehnker A, Hune T, et al. Minimum-cost reachability for priced timed automata. In: Proceedings of the International Workshop on Hybrid Systems: Computation and Control, Rome, 2001, 147–161
26 Bouyer P, Chevalier F. On conciseness of extensions of timed automata. J Autom Lang Combin 2005, 10: 393–405
27 Suman P V, Pandya P K, Krishna S N, et al. Timed automata with integer resets: language inclusion and expressiveness. In: Proceedings of International Conference on Formal Modeling and Analysis of Timed Systems, Saint Malo, 2008. 78–92
28 Suman P V, Pandya P K. Determinization and expressiveness of integer reset timed automata with silent transitions. In: Proceedings of International Conference on Language and Automata Theory and Applications, Tarragona, 2009, 728–739
29 Trivedi A, Wojtczak D. Recursive timed automata. In: Proceedings of International Symposium on Automated Technology for Verification and Analysis, Singapore, 2010. 306–324
30 Wen Y Q, Li G Q, Yuen S J. An over-approximation forward analysis for nested timed automata. In: Proceedings of International Workshop on Structured Object-Oriented Formal Language and Method, Luxembourg, 2014. 62–80