On Termination and Boundedness of Nested Updatable Timed Automata

Yuwei Wang, Xiuting Tao, and Guoqiang Li[✉]

School of Software, Shanghai Jiao Tong University, Shanghai, China
{wangywgg,xiutingtao,li.g}@sjtu.edu.cn

Abstract. We introduce a model named *nested updatable timed automata (NeUTAs)*, which can be regarded as a combination of *nested timed automata (NeTAs)* and *updatable timed automata with one updatable clock (UTA1s)*. The model is suitable for soft real-time system analysis, since the updatable clock representing a deadline can be updated due to environments. A NeUTA behaves as a UTA1, in which all clocks can be tested/updated and a special clock can be incremented/decremented. It also behaves as a pushdown system, in which a UTA1 can be pushed to a stack or popped from a stack. When time elapses, all clocks (clocks in the current running UTA1 or in the stack) proceed uniformly. We show the termination and boundedness of NeUTAs are decidable.

1 Introduction

Recently, numerous extensions of Alur and Dill's *timed automata (TAs)* have been proposed to model and reason real-time systems. They increase the expressive power of TAs in various ways, such as augmenting a stack, adding more operations on clocks, and extending the original deterministic reset operation to a non-deterministic update operation.

Nested timed automata (NeTAs) [1,2] are pushdown systems whose control locations and stack alphabet are TAs. A control location describes a working TA, and the stack presents a pile of interrupted TAs. A NeTA can either behaves as the top TA in the stack, or switches from one TA to another by pushing, popping, or changing the top TA of the stack. It is a natural model for analyzing real-time systems with context switches, e.g., interrupt systems. The reachability problem for NeTAs is shown to be decidable.

Updatable timed automata (UTAs) [3] are extensions of TAs having the ability to update clocks in a more elaborate way (i.e. increment and decrement) besides the normal operations. The reachability problem of UTAs is undecidable, which can be easily verified by encoding two counter machine with UTAs. However, there are still many decidable subclasses of UTAs by restricting the expressive power. Among them, *updatable timed automata with one updatable clock (UTA1s)* [4] is an interesting decidable subclass by restricting the number of updatable clocks to be one.

The aim of this paper is to combine NeTAs and UTA1s as a new model and to study the verification problems. More precisely, we replace TAs in NeTAs with

UTA1s and thus get *nested updatable timed automata (NeUTAs)*. Such kind of model is suitable for soft real-time system analysis, since the updatable clock of each UTA1 is used to describe the soft deadline, which is adjusted due to different environment. Termination and boundedness of the model are proved to be decidable, by encoding NeUTAs to *vector pushdown systems*, after digitizing dense time. The properties of the latter model are proved to be decidable by extending the reduced reachability tree proof technique of *pushdown vector addition systems* [5]. Note that pushdown vector addition systems assume the set of locations is a WQO, while the stack alphabet is finite. vector pushdown systems assume the set of locations is finite, while stack alphabet is a WQO.

Related Work. Timed automata (TAs) [6], proposed by Alur and Dill, are finite automata augmented with a finite set of clocks. Clocks can be used to record precisely how much time has elapsed in a dense manner and constrain the behaviour of the model. Although the theory of timed automata is successful in modeling and analyzing real-time systems with a large number of problems having been studied, it is so low-level that it is hard to apply it to verification of systems in reality directly. Actually many researchers are devoted to study subclasses or extensions of timed automata.

Hybrid automata [7–9] can be regarded as a generalization of timed automata. It is a mathematical model for mixed discrete-continuous systems, in which a discrete problem is embedded in continuous changing environments. The decidability of reachability problem is undecidable for general hybrid automata, while initialized rectangular automata form a maximum decidable subclass of hybrid automata that lies in the boundary of decidability.

Dense timed pushdown automata (DTPDAs) [10] play an essential role of the prove in this paper. DTPDAs extend timed automata with an additional stack, where a stack symbol with an age can be pushed to the stack. When time elapses, all clocks together with ages in the stack proceed uniformly. When popping, the value of the age in the top stack frame can be checked. The reachability problem for DTPDAs is shown to be EXPTIME-complete.

Interrupt timed automata (ITAs) [11,12] intend to model timed multi-task systems with different priority levels. As extensions of TAs, each control state in ITAs is in an interrupt level, ranged from 1 to n, with exactly one active clock recording time in each interrupt level. When ITAs are in a given interrupt level, all clocks of lower interrupt levels are suspended and those of higher interrupt levels are undefined. The reachability problem for ITAs is shown to be in NEX-PTIME and PTIME when the number of clocks is fixed. Though both ITAs and NeTAs can be used to model interrupt systems, they are different in what they are focus on. ITAs focus on the interrupt level, while NeTAs focus on context switches.

Paper Organization. The remainder of this paper is structured as follows: In Sect. 2 we introduce basic notations and models. Section 3 defines syntax and the semantics of UDTPDA1s. Section 4 shows that the termination and

boundedness of UDTPDA1s are decidable. Section 5 introduces NeUTAs and shows the decidability on termination and boundedness by encoding NeUTAs to UDTPDA1s. Section 6 concludes this paper with summarized results.

2   Preliminaries

Let $\mathbb{R}^{\geq 0}$ and $\mathbb{N}$ be the sets of non-negative real and natural numbers, respectively. Let $\mathbb{N}_\omega := \mathbb{N} \cup \{\omega\}$, where ω is the least limit ordinal. $\mathcal{I}$ denotes the set of *intervals*, which are (a, b), $[a, b]$, $[a, b)$ or $(a, b]$ for $a \in \mathbb{N}$ and $b \in \mathbb{N}_\omega$.

Let $X = \{x_1, \ldots, x_n\}$ be a finite set of *clocks*. A *clock valuation* $\nu : X \to \mathbb{R}^{\geq 0}$, assigns a value to each clock $x \in X$. ν_0 denotes the clock valuation assigning each clock in X to 0. Given a clock valuation ν 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. Further, multiple clock assignment function $\nu[y_1 \leftarrow b_1, \cdots, y_n \leftarrow b_n]$ is defined by $\nu[y_1 \leftarrow b_1, \cdots, y_n \leftarrow b_n](x) = b_i$ if $x = y_i$ for $1 \leq i \leq n$, and $\nu(x)$ otherwise. $\mathcal{V}al(X)$ is used to denote the set of clock valuation of X.

For finite words $w = aw'$, we denote $a = head(w)$ and $w' = tail(w)$. The concatenation of two words w, v is denoted by $w.v$, and ϵ is the empty word. we denote the set of finite multisets over D by $\mathcal{MP}(D)$, and the union of two multisets M, M' by $M \uplus M'$. We regard a finite set as a multiset with the multiplicity 1, and a finite word as a multiset by ignoring the ordering.

2.1   Updatable Timed Automata

Updatable timed automata (UTAs) [13,14] are extensions of timed automata, based on the possibility to update the clocks in a elaborate way such as increment and decrement operations and assignments to arbitrary values. However, generally, the reachability problem of updatable timed automata is undecidable. Several decidable subclasses are investigated, based on restriction on the update abilities [3]. In [4], we proposed another decidable subclass by restricting the number of updatable clocks to be one. We will adopt the restriction in the following content paper.

Definition 1 (Updatable Timed Automata with One Updatable Clock). *An updatable timed automaton with one updatable clock (UTA1) is a tuple* $\mathcal{A} = \langle Q, q_0, X, c, \Delta \rangle$, *where*

- *Q is a finite set of control locations, with the initial control location $q_0 \in Q$,*
- *$X = \{x_1, \ldots, x_k\}$ is a finite set of normal clocks, and c is the singleton updatable clock,*
- *$\Delta \subseteq Q \times Actions_{\mathcal{A}}^+ \times Q$ is a finite set of actions. A (discrete) transition $\delta \in \Delta$ is a sequence of actions $(q_1, \phi_1, q_2) \ldots (q_i, \phi_i, q_{i+1})$, written as $q_1 \xrightarrow{\phi_1; \ldots; \phi_i} q_{i+1}$, in which ϕ_j (for $1 \leq j \leq i$) is one of the following*

Local ϵ, 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* $I \in \mathcal{I}$,
Increment $c := c + 1$, *and*
Decrement $c := c - 1$.

Similar to the definitions in [2], for an easier encoding later, a transition as a sequence of actions $q_1 \xrightarrow{\phi_1; \cdots; \phi_i} q_{i+1}$ prohibits interleaving time progress. This can be encoded with an extra clock by resetting it to 0 and checking it still 0 after transitions, and introducing fresh control states.

Given a UTA1 $\mathcal{A} \in \mathscr{A}$, we use $Q(\mathcal{A})$, $q_0(\mathcal{A})$, $X(\mathcal{A})$, $c(\mathcal{A})$ and $\Delta(\mathcal{A})$ to represent the set of control locations, the initial location, the set of normal clocks, the updatable clock and the set of actions, respectively. We will use similar notations throughout the paper.

Definition 2 (Semantics of UTA1s). *Given a UTA1* $\mathcal{A} = \langle Q, q_0, X, c, \Delta \rangle$, *a configuration 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: $(q, \nu) \xrightarrow{t} (q, \nu + t)$, *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* $\nu_1 = \nu_2$. *This operation only changes the control location and leaves the clock valuation unaltered.*
  - **Test** $\phi = x \in I$?, $\nu_1 = \nu_2$ *and* $\nu_2(x) \in I$ *holds. The transition can be performed only if the value of* x *belongs to* I.
  - **Assignment** $\phi = x \leftarrow I$, $\nu_2 = \nu_1[x \leftarrow r]$ *where* $r \in I$. *Clock* x *is assigned to a non-deterministic value in* I.
  - **Increment** $\phi = c := c + 1$, $\nu_2 = \nu_1[c \leftarrow \nu_1(c) + 1]$. *The value of the updatable clock* c *is incremented by 1.*
  - **Decrement** $\phi = c := c - 1$, $\nu_2 = \nu_1[c \leftarrow \nu_1(c) - 1]$ *and* $\nu_1(c) \geq 1$ *holds. The value of the updatable clock* c *is decremented by 1.*

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

Proposition 1. *The reachability problem of UTA1 under diagonal-free constraints is decidable* [4].

2.2   Dense Timed Pushdown Automata

Dense timed pushdown automata [2,10] extend timed pushdown automata with time updating in the stack. Each symbol in the stack is equipped with a local clock named an *age*, and all ages in the stack proceed uniformly. An age in each context is assigned to the value of a clock when a push action occurs. A pop action pops the top symbol to assign the value of its age to a specified clock.

Definition 3 (Dense Timed Pushdown Automata). *A dense timed pushdown automaton is a tuple* $\mathcal{D} = \langle Q, q_0, \Gamma, X, \Delta \rangle \in \mathscr{D}$, *where*

– Q is a finite set of control locations with the initial control location $q_0 \in Q$,
– Γ is a finite stack alphabet,
– X is a finite set of clocks, and
– $\Delta \subseteq Q \times Actions_{\mathcal{D}}^{+} \times Q$ is a finite set of actions.

A (discrete) transition $\delta \in \Delta$ is a sequence of actions $(q_1, \varphi_1, q_2), \cdots, (q_i, \varphi_i, q_{i+1})$ written as $q_1 \xrightarrow{\varphi_1; \cdots; \varphi_i} q_{i+1}$, in which φ_j (for $1 \leq j \leq i$) is one of the followings,

– **Local** ϵ, an empty operation,
– **Test** $x \in I?$, where $x \in X$ is a clock and $I \in \mathcal{I}$ is an interval,
– **Assign** $x \leftarrow I$ where $x \in X$ and $I \in \mathcal{I}$,
– **Push** $push(\gamma, x)$, where $\gamma \in \Gamma$ is a stack symbol and $x \in X$, and
– **Pop** $pop(\gamma, x)$, where $\gamma \in \Gamma$ is a stack symbol and $x \in X$.

Definition 4 (Semantics of DTPDAs). *For a dense timed pushdown automaton $\langle Q, q_0, \Gamma, X, \Delta \rangle$, a configuration is a triplet (q, w, ν) with a control location $q \in Q$, a stack $w \in (\Gamma \times \mathbb{R}^{\geq 0})^*$, and a clock valuation ν on X. In a stack $w = (\gamma_1, t_1). \cdots .(\gamma_n, t_n)$, γ_i is a stack symbol and t_i is an age. t-time passage on the stack increases all ages in the stack by the same value, which is denoted by $w + t = (\gamma_1, t_1 + t). \cdots .(\gamma_n, t_n + t)$. The transition relation of a DTPDA is represented as follows.*

– Progress transition: $(q, w, \nu) \xrightarrow{t}_{\mathcal{D}} (q, w + t, \nu + t)$, where $t \in \mathbb{R}^{\geq 0}$. When time elapses, all clocks together with all ages in the stack proceed uniformly.
– Discrete transition: $(q_1, w_1, \nu_1) \xrightarrow{\varphi}_{\mathcal{D}} (q_2, w_2, \nu_2)$, if $q_1 \xrightarrow{\varphi} q_2$, and one of the following holds,
   • **Local** $\varphi = \epsilon$, then $w_1 = w_2$, and $\nu_1 = \nu_2$.
   • **Test** $\varphi = x \in I?$, then $w_1 = w_2$, $\nu_1 = \nu_2$ and $\nu_1(x) \in I$ holds.
   • **Assign** $\varphi = x \leftarrow I$, then $w_1 = w_2$, $\nu_2 = \nu_1[x \leftarrow r]$ where $r \in I$.
   • **Push** $\varphi = push(\gamma, x)$, then $\nu_1 = \nu_2$, $w_2 = (\gamma, \nu_1(x)).w_1$. The stack symbol γ and an age of the value of clock x are pushed to the stack.
   • **Pop** $\varphi = pop(\gamma, x)$, then $\nu_2 = \nu_1[x \leftarrow t]$, $w_1 = (\gamma, t).w_2$. The top stack frame (γ, t) is popped from the stack and the clock x is assigned with the value of the age t.

   The initial configuration $\varrho_0 = (q_0, \epsilon, \nu_{\mathbf{0}})$.

Remark 1. For simplicity of the later proofs, the definition of DTPDAs follows Definition 1 in [2], slightly modified from the original [10]. The former can encode the later easily.

3   Updatable Dense Timed Pushdown Automata

An updatable dense timed pushdown automaton with one updatable clock (UDTPDA1) is different from Definition 3 at:

– besides the set X of normal clocks (of the fixed number k), an updatable clock c is introduced. We refer to the normal clocks as $x_1, x_2, ..., x_k$ and sometimes we refer to the updatable clock c as x_0 for simplicity.
– a tuple of ages (for simplicity, we fix the length of a tuple to be $k + 1$) is pushed on the stack and/or popped from the stack.

Definition 5 (UDTPDA1s). *A UDTPDA1 is a tuple* $\mathcal{U} = \langle S, s_0, \Gamma, X, c, \Delta \rangle \in \mathscr{U}$, *where*

– *S is a finite set of control locations with the initial control location $s_0 \in S$,*
– *Γ is a finite stack alphabet,*
– *X is a finite set of local clocks (with $|X| = k$),*
– *c is the singleton updatable clock and*
– *$\Delta \subseteq S \times Action_{\mathcal{U}}^+ \times S$ is a finite set of actions.*

A (discrete) transition $\delta \in \Delta$ is a sequence of actions $(s_1, \varphi_1, s_2), \cdots, (s_i, \varphi_i, s_{i+1})$ written as $s_1 \xrightarrow{\varphi_1; \cdots ; \varphi_i} s_{i+1}$, in which φ_j (for $1 \leq j \leq i$) is one of the followings,

– **Local** ϵ, *an empty operation,*
– **Test** $x \in I?$, *where $x \in X \cup \{c\}$ is a clock and $I \in \mathcal{I}$ is an interval,*
– **Assign** $x \leftarrow I$ *where $x \in X \cup \{c\}$ and $I \in \mathcal{I}$,*
– **Increment** $c := c + 1$,
– **Decrement** $c := c - 1$,
– **Push** $push(\gamma)$, *where $\gamma \in \Gamma$, and*
– **Pop** $pop(\gamma)$, *where $\gamma \in \Gamma$.*

Definition 6 (Semantics of UDTPDA1s). *For a UDTPDA1 $\langle S, s_0, \Gamma, X, c, \Delta \rangle$, a configuration is a triplet (s, w, ν) with a control location $s \in S$, a stack $w \in (\Gamma \times (\mathbb{R}^{\geq 0})^{k+1})^*$, and a clock valuation ν on $X \cup \{c\}$. In a stack $w = (\gamma_1, \bar{t}_1). \cdots .(\gamma_n, \bar{t}_n)$, γ_i is a stack symbol and $\bar{t}_i = (t_i^0, \cdots, t_i^k)$ is a $k + 1$-tuple of ages. t-time passage on the stack increases all ages in the stack by the same value t, which is denoted by $w + t = (\gamma_1, \bar{t}_1 + t). \cdots .(\gamma_n, \bar{t}_n + t)$ where $\bar{t}_i + t = (t_i^0 + t, \cdots, t_i^k + t)$.*
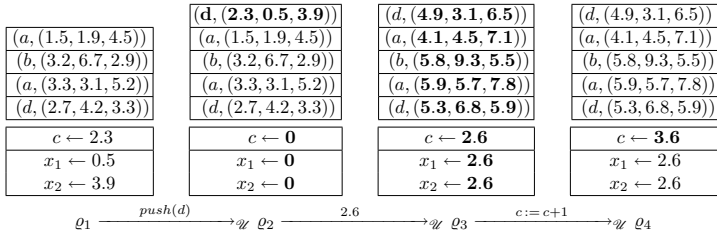
The transition relation of a UDTPDA1 is represented as follows.

– Time progress: $(s, w, \nu) \xrightarrow{t}_{\mathcal{U}} (s, w + t, \nu + t)$, where $t \in \mathbb{R}^{\geq 0}$.
– Discrete transition: $(s_1, w_1, \nu_1) \xrightarrow{\varphi}_{\mathcal{U}} (s_2, w_2, \nu_2)$, if $s_1 \xrightarrow{\varphi} s_2$, and one of the following holds,
  • **Local** $\varphi = \epsilon$, then $w_1 = w_2$, and $\nu_1 = \nu_2$.
  • **Test** $\varphi = x \in I?$, then $w_1 = w_2$, $\nu_1 = \nu_2$, and $\nu_1(x) \in I$ holds.
  • **Assign** $\varphi = x \leftarrow I$, then $w_1 = w_2$, $\nu_2 = \nu_1[x \leftarrow r]$ where $r \in I$.
  • **Increment** $c := c + 1$, then $w_1 = w_2$, and $\nu_2 = \nu_1[c \leftarrow \nu_1(c) + 1]$,
  • **Decrement** $c := c - 1$, then $w_1 = w_2$, $\nu_2 = \nu_1[c \leftarrow \nu_1(c) - 1]$ and $\nu_1(c) \geq 1$ holds,
  • **Push** $\varphi = push(\gamma)$, then $\nu_2 = \nu_0$, $w_2 = (\gamma, (\nu_1(c), \nu_1(x_1), \cdots, \nu_1(x_k))).w_1$ for $X = \{x_1, \cdots, x_k\}$. The values of $k+1$ clocks are pushed as ages in the stack.

- **Pop** $\varphi = pop(\gamma)$, then $\nu_2 = \nu_1[c \leftarrow t_0, x_1 \leftarrow t_1, \cdots, x_k \leftarrow t_k]$, $w_1 = (\gamma, (t_0, \cdots, t_k)).w_2$. The values of $k+1$ clocks are recovered with ages in the top stack frame.

The initial configuration $\varrho_0 = (s_0, \epsilon, \nu_0)$. We use $\hookrightarrow$ to range over these transitions, and $\hookrightarrow^*$ is the reflexive and transitive closure of $\hookrightarrow$.

Example 1. The figure shows transitions $\varrho_1 \hookrightarrow \varrho_2 \hookrightarrow \varrho_3 \hookrightarrow \varrho_4$ of a UDTPDA1 with $S = \{\bullet\}$ (omitted in the figure), $X = \{x_1, x_2\}$ and $\Gamma = \{a, b, d\}$. All values which are changed in a transition are in bold. At $\varrho_1 \hookrightarrow \varrho_2$, the values of c, x_1 and x_2 (2.3, 0.5 and 3.9) are pushed to the stack with d. After pushing, value of c, x_1 and x_2 will be reset to zero, At $\varrho_2 \hookrightarrow \varrho_3$, time elapses 2.6. At $\varrho_3 \hookrightarrow \varrho_4$, a increment occurs which increases the value of c from 2.6 to 3.6.

$(a, (1.5, 1.9, 4.5))$	$(\mathbf{d}, (\mathbf{2.3}, \mathbf{0.5}, \mathbf{3.9}))$	$(d, (\mathbf{4.9}, \mathbf{3.1}, \mathbf{6.5}))$	$(d, (4.9, 3.1, 6.5))$
$(b, (3.2, 6.7, 2.9))$	$(a, (1.5, 1.9, 4.5))$	$(a, (\mathbf{4.1}, \mathbf{4.5}, \mathbf{7.1}))$	$(a, (4.1, 4.5, 7.1))$
$(a, (3.3, 3.1, 5.2))$	$(b, (3.2, 6.7, 2.9))$	$(b, (\mathbf{5.8}, \mathbf{9.3}, \mathbf{5.5}))$	$(b, (5.8, 9.3, 5.5))$
$(d, (2.7, 4.2, 3.3))$	$(a, (3.3, 3.1, 5.2))$	$(a, (\mathbf{5.9}, \mathbf{5.7}, \mathbf{7.8}))$	$(a, (5.9, 5.7, 7.8))$
	$(d, (2.7, 4.2, 3.3))$	$(d, (\mathbf{5.3}, \mathbf{6.8}, \mathbf{5.9}))$	$(d, (5.3, 6.8, 5.9))$

$c \leftarrow 2.3$	$c \leftarrow \mathbf{0}$	$c \leftarrow \mathbf{2.6}$	$c \leftarrow \mathbf{3.6}$
$x_1 \leftarrow 0.5$	$x_1 \leftarrow \mathbf{0}$	$x_1 \leftarrow \mathbf{2.6}$	$x_1 \leftarrow 2.6$
$x_2 \leftarrow 3.9$	$x_2 \leftarrow \mathbf{0}$	$x_2 \leftarrow \mathbf{2.6}$	$x_2 \leftarrow 2.6$

$$\varrho_1 \xrightarrow{\;push(d)\;}_{\mathscr{U}} \varrho_2 \xrightarrow{\;2.6\;}_{\mathscr{U}} \varrho_3 \xrightarrow{\;c:=c+1\;}_{\mathscr{U}} \varrho_4$$

4 Termination and Boundedness of UDTPDA1s

In this section, we show that the termination and boundedness of UDTPDA1s are decidable. We first introduce vector pushdown systems and prove its decidability on termination and boundedness. Then we describe the digitization technique in UDTPDA1s using *digitized configuration* and its operations, which intend to simulate configurations and transitions of UDTPDA1s, respectively. Finally, the specific encoding from a UDTPDA1 to a snapshot vector pushdown system is given.

4.1 Vector Pushdown Systems

Definition 7 (Vector Pushdown Systems). *A vector pushdown system is a tuple* $\mathcal{P} = (Q, \Gamma, \mathbb{N}^k, \Delta)$, *where* Q *is a finite set of states,* Γ *is a finite stack alphabet,* $\mathbb{N}^k$ *is k-dimension natural number vectors, and* $\Delta \subseteq P \times (\Gamma \times \mathbb{N}^k)^{\leq 2} \times P \times (\Gamma \times \mathbb{N}^k)^{\leq 2}$. *We use* $\alpha, \beta, \gamma, \cdots$ *to range over* $\Gamma \times \mathbb{N}^k$, *and* $w, v, \cdots$ *over words in* $(\Gamma \times \mathbb{N}^k)^*$.

A configuration of $\mathcal{P}$ is a pair $\langle q, w \rangle$, where a state $q \in Q$ and a stack $w \in (\Gamma \times \mathbb{N})^*$. A transition relation $\Longrightarrow$ between configurations of $\mathcal{P}$ is defined by

$$\frac{(p, \gamma \rightarrow p', \gamma') \in \Delta}{\langle p, \gamma w \rangle \hookrightarrow \langle p', \gamma' w \rangle} \quad \text{INTER} \qquad \frac{(p, \gamma \rightarrow p', \alpha\beta) \in \Delta}{\langle p, \gamma w \rangle \hookrightarrow \langle p', \alpha\beta w \rangle} \quad \text{PUSH}$$

$$\frac{(p, \gamma \rightarrow p', \epsilon) \in \Delta}{\langle p, \gamma w \rangle \hookrightarrow \langle p', w \rangle} \quad \text{POP} \qquad \frac{(p, \epsilon \rightarrow p', \alpha) \in \Delta}{\langle p, w \rangle \hookrightarrow \langle p', \alpha w \rangle} \quad \text{SIMPLE-PUSH}$$

$$\frac{(p, \alpha\beta \to p', \gamma) \in \Delta}{\langle p, \alpha\beta w \rangle \hookrightarrow \langle p', \gamma w \rangle} \quad \text{NONSTANDARD-POP}$$

Remark 2. In a pushdown system, the SIMPLE-PUSH and NONSTANDARD-POP rules can be encoded by other three rules. However, in a vector pushdown system, since the stack symbols in $\Gamma \times \mathbb{N}^k$ are essential unbounded, thus all of the five rules are necessary.

Let $|w|$ denotes the length of word w and $w[i]$ denotes the i-th symbol in w. The *head* $h(p, w)$ of a configuration $\langle p, w \rangle$ is $(p, w[1])$ if $w \neq \epsilon$; otherwise, $h(p, w) = (p, \bot)$. Since a finite set is well-quasi-ordered and $(\mathbb{N}^k, \leq)$ is well-quasi-ordered, by Dickson's Lemma, we can obtain the set of heads of configurations is well-quasi-ordered.

Besides, we denote $a_1 a_2 \ldots a_m \lll b_1 b_2 \ldots b_n$, if $m = n$ and, for each i, $a_i \leq b_i$ holds, and $w \lll v$ if $w \lll v$ and $w \neq v$.

The *reachability tree* of a VPS $\mathcal{V} = (Q, \Gamma, \mathbb{N}^k, \Delta)$ with an initial configuration c_0 is a rooted unordered tree defined as follows. Each node of the tree is labeled by a configuration of $\mathcal{V}$. The root r is labeled by the initial configuration c_0, denoted by $r : c_0$. Each node $n : c_n$ has a child $m : c_m$ when $c_n \hookrightarrow c_m$. Note that the reachability tree of $\mathcal{V}$ is finitely branching since Δ is finite.

Termination Problem. The termination problem asks whether all runs of a given system are finite, we have the following definition.

Definition 8. *A node $s : \langle p, w \rangle$ pumps a node $t : \langle q, v \rangle$ if*

- *there is a path from s to t, and every node $t' : \langle p', w' \rangle$ on it satisfies $|w'| \geq |w|$.*
- *$h(\langle p, w \rangle) \trianglelefteq h(\langle q, v \rangle)$, i.e., $p \preceq q$ and either $w = \epsilon$ or $w[1] \leq v[1]$.*

We call a node *pumpable* if there exists a node pumping it. The notion of pumpable nodes is similar to *subsumed nodes* in [5], but we consider the increase of heads instead of states. Let *the reduced reachability tree* be the largest prefix of the reachability tree such that every *pumpable node* has no children.

The intuition of pumpable nodes is that if the run from $\langle p, w \rangle$ to $\langle q, v \rangle$ only changes the top element of w, then we can simulate this run from $\langle q, v \rangle$ to some $\langle q', v' \rangle$ by monotonicity, satisfying $p \preceq q \preceq q'$, and $w[1] \leq v[1] \leq v'[1]$. We can construct an infinite run by repeating this process.

Conversely, assume $\langle p_0, w_0 \rangle \hookrightarrow \langle p_1, w_1 \rangle \cdots$ is an infinite run, we can extract an infinite subsequence, say $\langle p_{i_0}, w_{i_0} \rangle, \langle p_{i_1}, w_{i_1} \rangle, \cdots$, such that each node is chosen if it has the minimal depth of the stack in its suffix run. Note that each pair of $\langle p_{i_k}, w_{i_k} \rangle$ and $\langle p_{i_j}, w_{i_j} \rangle$ with $k < j$ in this subsequence satisfies the first condition of *pumpable nodes*. By the fact that the set of heads is well-quasi-ordered, it must contain a *pumpable node*.

Theorem 1. *A VPS has an infinite run if, and only if, its reduced reachability tree contains a pumpable node.*

Boundedness Problem. The boundedness asks whether the reachability set is finite. We know that any infinite run has a pumpable node. If a pumpable node is exactly the same as the one that pumps it, still an infinite run keeps the reachability set finite. Otherwise, a VPS enlarges reachable configurations infinitely.

Definition 9. *A node $s : \langle p, w \rangle$ strictly pumps a node $t : \langle q, v \rangle$ if s pumps t, and either $|w| < |v|$ or $h(\langle p, w \rangle) \lhd h(\langle q, v \rangle)$.*

Theorem 2. *A VPS has an infinite reachability set if, and only if, its reduced reachability tree contains a strictly pumpable node.*

Proof. **(Only-if)** Assume a VPS $\mathcal{V}$ has an infinite reachability set. Let $\mathcal{T}$ be the largest prefix of its reachability tree such that, on each branch, all nodes have distinct labels. The tree $\mathcal{T}$ is infinite since every configuration in the reachability set is a node in $\mathcal{T}$.

By König's lemma, it follows that $\mathcal{T}$ contains finitely many branches in which all nodes are distinct. Since the reduced reachability tree of $\mathcal{V}$ is finite, among finitely many branches, there are two nodes $n : (p, w)$ and $m : (q, v)$ such that they are in the reduced reachability tree and n pumps m.

Thus, $(p, w) \neq (q, v)$ and (p, w) pumps (q, v). By definition of pumpable nodes, we have two cases: (1) $|w| < |v|$, and (2) $|w| = |v|$. In case (2), either $w \ll v$ or $p \prec q$ holds. $w[2, |w|] = v[2, |v|]$ implies either $w[1] < v[1]$ or $p \prec q$. Thus, both cases, n strictly pumps m.

(If) Similar to that of Theorem 1. The path from the root to a strictly pumpable node yields a run

$$(p_0, w_0) \xrightarrow{op_1} \ldots \xrightarrow{op_k} (p_k, w_k) \xrightarrow{op_{k+1}} \ldots \xrightarrow{op_l} (p_l, w_l)$$

such that (p_k, w_k) strictly pumps (p_l, w_l), which leads an infinite run by iterating the sequence of operations $op_{k+1}, ..., op_l$. As the case analysis, if $|w| < |v|$, the resulting infinite run enlarges the length of the stack infinitely; if $w[1] < v[1]$, the resulting infinite run enlarges the top element of the stack infinitely.

4.2 Digitized Configuration and Its Operations

Let $\langle S, s_0, \Gamma, X, c, \Delta \rangle$ be a UDTPDA1, and let n be the largest integer (except for ω) appearing in Δ. For $v \in \mathbb{R}^{\geq 0}$, $proj(v) = \mathtt{r}_i$ if $v \in \mathtt{r}_i \in Intv(n)$, where

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

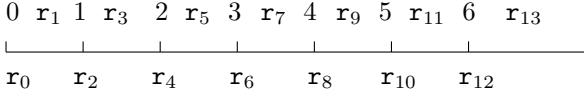The idea of the next digitization is inspired by [15–17].

Definition 10 (Digitization). *Let $frac(t) = t - floor(t)$ for $t \in \mathbb{R}^{\geq 0}$. A digitization function $\mathtt{digi} : \mathcal{MP}((\{c\} \cup X \cup \Gamma) \times \mathbb{R}^{\geq 0}) \to \mathcal{MP}((\{c\} \cup X \cup \Gamma) \times Intv(n))^*$ is defined as follows.*

For $\bar{\mathcal{Y}} \in \mathcal{MP}((\{c\} \cup X \cup \Gamma) \times \mathbb{R}^{\geq 0})$, let $Y_0, Y_1, \cdots, Y_m$ be multisets that collect $(x, proj(t))$'s having the same $frac(x, t)$ for $(x, t) \in \bar{\mathcal{Y}}$. Among them,

Y_0 (which is possibly empty) is reserved for the collection of $(x, proj(t))$ with $frac(t) = 0$. We assume that Y_i's except for Y_0 is non-empty (i.e., $Y_i = \emptyset$ with $i > 0$ is omitted), and Y_i's are sorted by the increasing order of $frac(t)$ (i.e., $frac(t) < frac(t')$ for $(x, proj(t)) \in Y_i$ and $(x', proj(t')) \in Y_{i+1}$). Thus, $\texttt{digi}(\bar{\mathcal{Y}})$ is a word $\bar{Y} = Y_0 Y_1 \cdots Y_m$.

For a stack frame $v = (\gamma, (t_0, \cdots, t_k))$ of a UDTPDA1, we denote a word $(\gamma, t_0) \cdots (\gamma, t_k)$ by $dist(v)$. Given a clock valuation ν, we denote a clock word $(c, \nu(c))(x_1, \nu(x_1)) \ldots (x_n, \nu(x_k))$ by $time(\nu)$ where c is the singleton updatable clock and $x_i \in X$ for $1 \le i \le k$.

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

$$0 \ \mathbf{r_1} \ 1 \ \mathbf{r_3} \ \ 2 \ \mathbf{r_5} \ \ 3 \ \mathbf{r_7} \ \ 4 \ \mathbf{r_9} \ \ 5 \ \mathbf{r_{11}} \ 6 \ \ \ \mathbf{r_{13}}$$

$$\mathbf{r_0} \qquad \mathbf{r_2} \qquad \ \ \mathbf{r_4} \qquad \ \ \mathbf{r_6} \qquad \ \ \mathbf{r_8} \qquad \ \mathbf{r_{10}} \qquad \mathbf{r_{12}}$$

For the configuration $\varrho_1 = (\bullet, v_4 \cdots v_1, \nu)$ in Example 1, let $\bar{\mathcal{Y}} = dist(v_4) \uplus time(\nu)$ be a word, and $\bar{Y} = \texttt{digi}(\bar{\mathcal{Y}})$, then

$$\bar{\mathcal{Y}} = \{(a, 1.5), (a, 1.9), (a, 4.5), (c, 2.3), (x_1, 0.5), (x_2, 3.9)\}$$
$$\bar{Y} = \{\}\{(c, \mathbf{r_5})\}\{(x_1, \mathbf{r_1}), (a, \mathbf{r_3}), (a, \mathbf{r_9})\}\{(x_2, \mathbf{r_7}), (a, \mathbf{r_3})\}$$

Definition 11 (Digiword). *A word $\bar{Y} \in \mathcal{MP}((\{c\} \cup X \cup \Gamma) \times Intv(n))^*$ is a* digiword *if the following is satisfied:*

- *Let a $k+1$-pointer $\bar{\rho}$ of $\bar{U}$ is a tuple of $k+1$ pointers to mutually different $k+1$ elements in $\bar{U}$. Then there are a pair of $k+1$-pointers $(\bar{\rho}_1, \bar{\rho}_2)$ in $\bar{Y}$ that point to clocks and ages in the topmost stack frame, respectively.*
- *For every element in $\bar{Y}$, either $\bar{\rho}_1$ or $\bar{\rho}_2$ points to it.*

We refer the element $(\gamma, \mathbf{r})$ pointed by the i-th pointer by $\bar{\rho}[i]$ where $0 \le i \le k$. Let the set $Digi$ contains all digiwords. $Digi$ is a finite set by observing that at most $2k + 2$ elements exist in a digiword.

A digiword $\bar{Y}$ intends to be the digitization of the current clock valuation (pointed by $\bar{\rho}_1$) and the topmost stack frame (pointed by $\bar{\rho}_2$) in a UDTPDA1. More precisely, for $0 \le i \le k$, $\bar{\rho}_1[i]$ points to $(x_i, proj(\nu(x_i)))$ for $(x_i, \nu(x_i)) \in time(\nu)$ where ν is the clock valuation and $\bar{\rho}_2[i]$ points to $(\gamma, proj(t_i))$ for $(\gamma, t_i) \in dist(v)$ where v is the topmost stack frame.

Definition 12 (Digitized Configuration). *A* digitized configuration *is a tuple $\bar{U} \in Digi \times \mathbb{N}^2$, which contains a digiword and a pair of natural numbers. The pair of natural numbers intend to roughly record the time passage when the value of updatable clocks in the current clock valuation and the topmost stack frame exceed the maximum integer n. Each time the updatable clocks value exceeds n and meanwhile its value changes between integer and non-integer with time elapsing, we increase the corresponding natural number by one.*

Example 3. The word $\bar{Y}$ in Example 2 can be extended to a digitized configuration $\bar{U}$ by adding a pair of 3-pointers $(\bar{\rho}_1, \bar{\rho}_2)$ (pointers are marked with the numbered overlines and underlines). and a pair of numbers $(0,0)$.

$\bar{U} = (\{\}\overline{\{(c, \mathbf{r}_5)\}}^0\{\overline{(x_1, \mathbf{r}_1)}^1, \underline{(a, \mathbf{r}_3)}_0, \underline{(a, \mathbf{r}_9)}_2\}\overline{\{(x_2, \mathbf{r}_7)}^2, \underline{(a, \mathbf{r}_3)}_1\}, 0, 0)$

Pointers are given more explicitly below:

$\bar{\rho}_1(0) = (c, \mathbf{r}_5)$ $\bar{\rho}_1(1) = (x_1, \mathbf{r}_1)$ $\bar{\rho}_1(2) = (x_2, \mathbf{r}_7)$

$\bar{\rho}_2(0) = (a, \mathbf{r}_3)$ $\bar{\rho}_2(1) = (a, \mathbf{r}_3)$ $\bar{\rho}_2(2) = (a, \mathbf{r}_9)$

Definition 13 (Operations on Digitized Configuration). *Let* $\bar{U} = (Y_0 \cdots Y_m, num_1, num_2), \bar{U}' = (Y'_0 \cdots Y'_{m'}, num'_1, num'_2), \bar{V} = (Y''_0 \cdots Y''_m, num''_1, num''_2) \in Digi \times \mathbb{N}^2$ *are digitized configurations such that* $\bar{U}$ *(resp.* $\bar{U}'$ *and* $\bar{V}$*) has a pair of* $k + 1$*-pointers* $(\bar{\rho}_1, \bar{\rho}_2)$ *(resp.* $(\bar{\rho}'_1, \bar{\rho}'_2)$ *and* $(\bar{\rho}''_1, \bar{\rho}''_2)$*). We define operations as follows which are used to simulate transitions of UDTPDA1s in the next subsection. Note that except for* **Rotate**, **Map**, *and* **Propogate**, *the* $k + 1$*-pointers* $\bar{\rho}_1$ *is changed corresponding to the operation to ensure that properties of digiwords are still satisfied after operations. Namely when an element is removed, the pointer which points to it is set to empty. And when an element* $(x_i, \mathbf{r})$ *for* $x_i \in \{c\} \cup X$ *is added, the pointer* $\bar{\rho}_1[i]$ *is modified to point to that new element.*

- **Insert$_x$:** *insert$_x(\bar{U}, (x, \mathbf{r}_i))$ for* $x \in X$ *inserts* $(x, \mathbf{r}_i)$ *to* $\bar{U}$ *(may nondeterministically) at*

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

- **Insert$_c$:** *insert$_c(\bar{U}, (c, \mathbf{r}_i))$ for the updatable clock* c *inserts* $(c, \mathbf{r}_i)$ *to* $\bar{U}$ *and updates natural number* num_1 *in one of three following ways:*

$$\begin{cases} (1)\ either\ put\ (c, \mathbf{r}_i)\ into\ Y_j\ for\ j > 0,\ or \\ \quad put\ the\ singleton\ set\ \{(c, \mathbf{r}_i)\}\ at\ any\ place\ after\ Y_0 \\ \quad and\ \ num_1 = 0 & if\ i\ \leq 2n\ and\ i\ is\ odd \\ (2)\ put\ (x, \mathbf{r}_i)\ into\ Y_0\ and\ num_1 = 0 & if\ i \leq 2n\ and\ i\ is\ even \\ (3)\ either\ put\ (c, \mathbf{r}_i)\ into\ Y_j\ for\ j \geq 0,\ or \\ \quad put\ the\ singleton\ set\ \{(c, \mathbf{r}_i)\}\ at\ any\ place\ after\ Y_0 \\ \quad and\ num_1 = d,\ where\ d\ is\ a\ positive\ integer \\ \quad and\ d\ is\ odd\ if\ (c, \mathbf{r}_i)\ is\ put\ into\ Y_0\ otherwise\ even & if\ i = 2n + 1 \end{cases}$$

- **Init:** *For* $\bar{U} = (Y_0 \cdots Y_m, num_1, num_2)$, *init$(\bar{U})$ is obtained by updating* Y_0 *with* $Y_0 \uplus \{(x_i, \mathbf{r}_0) \mid x_i \in \{c\} \cup X\}$ *and* num_1 *with 0.*
- **Delete:** *delete$(\bar{U}, x)$ for* $x \in \{c\} \cup X$ *is obtained from* $\bar{U}$ *by deleting the element* $(x, \mathbf{r})$ *indexed by* x.
- **Increase:** *increase$(\bar{U})$ is obtained from* $\bar{U}$ *by replacing the element* $(c, \mathbf{r}_i)$ *indexed by* c *with element* $(c, \mathbf{r}_{min\{i+2, 2n+1\}})$ *and updating* num_1 *as follows:*

$$\begin{cases} num_1 := num_1 & if\ i < 2n - 1 \\ num_1 := num_1 + 1 & if\ i = 2n - 1 \\ num_1 := num_1 + 2 & otherwise \end{cases}$$

– **Decrease:** $decrease(\bar{U})$ *is obtained from* $\bar{U}$ *by replacing the element* $(c, \mathbf{r}_i)$ *indexed by* c *with element* $(c, \mathbf{r}_j)$ *and updating* num_1 *as follows:*

$$\begin{cases} j = max(i-2, 0) & and \quad num_1 := num_1 & if \ num_1 \leq 1 \\ j = i-1 \ and \ num_1 := 0 & & else \ if \ num_1 = 2 \\ j = i \ and \ num_1 := num_1 - 2 & & otherwise \end{cases}$$

– **Rotate:** *Rotate intends to simulate the time progress transition. A rotation* $\bar{U} = (Y_0 \cdots Y_m, num_1, num_2) \Rightarrow \bar{U}' = (Y_0' \cdots Y_{m'}', num_1', num_2')$ *is defined as follows.*

$$\begin{cases} \bar{U}' = (Y_0' \cdots Y_{m+1}', & if \ Y_0 \neq \emptyset, \ Y_0' = \emptyset, \ Y_1' = \{(\gamma, \mathbf{r}_{min\{i+1, 2n+1\}}) \\ \quad num_1', num_2') \mid (\gamma, \mathbf{r}_i) \in Y_0\}, \ Y_j' = Y_{j-1} \ for \ j \in [2..m+1], \\ \quad\quad num_1' = num_1 + 1 \ if \ (c, \mathbf{r}_i) \in Y_0 \ and \ i \geq 2n \\ \quad\quad otherwise \ num_1, \ and \ num_2' = num_2 + 1 \ if \\ \quad\quad \bar{\rho}_1[0] = (\gamma, \mathbf{r}_i) \in Y_0 \ and \ i \geq 2n, \ otherwise \\ \quad\quad num_2 + 1. \\ \bar{U}' = (Y_0' \cdots Y_{m-1}', & otherwise, \ Y_0' = \{(\gamma, \mathbf{r}_{min\{i+1, 2n+1\}}) \mid (\gamma, \mathbf{r}_i) \in Y_m\}, \\ \quad num_1', num_2') \ Y_j' = Y_j \ for \ j \in [1..m-1], \ num_1' = num_1 + 1, \\ \quad\quad if \ (c, \mathbf{r}_i) \in Y_m \ and \ i \geq 2n, \ otherwise \ num_1, \\ \quad\quad and \ num_2' = num_2 + 1 \ if \ \bar{\rho}_1[0] = (\gamma, \mathbf{r}_i) \in Y_m \\ \quad\quad and \ i \geq 2n, \ otherwise \ num_2. \end{cases}$$

$(\bar{\rho}_1, \bar{\rho}_2)$ *are updated to correspond to the permutation accordingly. As convention, we define* $\Rightarrow^*$ *as reflexive transitive closure of* $\Rightarrow$.

– **Map:** *Map intends to simulate the push transition.* $map(\bar{U}, \gamma)$ *for* $\gamma \in \Gamma$ *is obtained from* $\bar{U}$ *by the following operations. First delete all elements pointed by* $\bar{\rho}_2$. *Then replace* $(x, \mathbf{r}_j)$ *pointed by* $\bar{\rho}_1$ *for* $x \in \{c\} \cup X$ *with* $(\gamma, \mathbf{r}_j)$ *and set* $\bar{\rho}_2$ *to point to that. Finally assign value of* num_1 *to* num_2.

– **Propogate:** *Propogate intends to simulate the pop transition.* $propogate(\bar{U}, \bar{U}', \gamma)$ *for* $\gamma \in \Gamma$ *is set to be* $\bar{V}$ *which is obtained by finding a rotation* $\bar{U}' \Rightarrow^* \bar{V}$ *such that* $\bar{\rho}_1''$ *of* $\bar{V}$ *matches the original* $\bar{\rho}_2$ *of* $\bar{U}$. *That is to say, for* $0 \leq i \leq k$, $\bar{\rho}_1''[i] = (x, \mathbf{r}_m)$ *and* $\bar{\rho}_2[i] = (\gamma, \mathbf{r}_n)$, *we have* $m = n$.

Example 4. We begin with the digitized configuration $\bar{U}$ in Example 3, to simulate transitions $\varrho_1 \hookrightarrow^* \varrho_4$ in Example 1.

– $push(d)$ is simulated by $\bar{U}_1 = init(map(\bar{U}, d))$.
  $\bar{U}_1 = (\{\overline{(c, \mathbf{r}_0)}^0, \overline{(x_1, \mathbf{r}_0)}^1, \overline{(x_2, \mathbf{r}_0)}^2\}\{\underline{(d, \mathbf{r}_5)}_0\}\{\underline{(d, \mathbf{r}_1)}_1\}\{\underline{(d, \mathbf{r}_7)}_2\}, 0, 0)$
– Time elapse of 2.6 time units is simulated by $\bar{U}_1 \Rightarrow^* \bar{U}_2$
  $\bar{U}_2 = (\{\}\{\underline{(d, \mathbf{r}_7)}_1\}\{\underline{(d, \mathbf{r}_{13})}_2\}\{\overline{(c, \mathbf{r}_5)}^0, \overline{(x_1, \mathbf{r}_5)}^1, \overline{(x_2, \mathbf{r}_5)}^2\}\{\underline{(d, \mathbf{r}_9)}_0\}, 0, 0)$
– $c := c + 1$ is simulated by $\bar{U}_3 = increase(\bar{U}_2)$.
  $\bar{U}_3 = (\{\}\{\underline{(d, \mathbf{r}_7)}_1\}\{\underline{(d, \mathbf{r}_{13})}_2\}\{\overline{(c, \mathbf{r}_7)}^0, \overline{(x_1, \mathbf{r}_5)}^1, \overline{(x_2, \mathbf{r}_5)}^2\}\{\underline{(d, \mathbf{r}_9)}_0\}, 0, 0)$

4.3   Snapshot Vector Pushdown System

A *snapshot vector pushdown system* (snapshot VPS) keeps the digitization of clock valuation and ages in the top stack frame and a pair of natural numbers that record roughly how much the current updatable clock pointed by $\bar{\rho}_1[0]$ and the age of element pointed by $\bar{\rho}_2[0]$ exceed the maximum integer n in the top stack frame, as a *digitized configuration*.

  We show that a UDTPDA1 is encoded into its digitization, called a *snapshot VPS*. The keys of the encoding are, when a pop occurs, the time progress recorded at the top stack symbol is propagated to the next stack symbol after finding a series of rotations by matching between $k + 1$-pointers $\bar{\rho}_2$ and $\bar{\rho}_1'$. Using digitized configuration and its operations defined in the last subsection, the encoding is quite natural.

Definition 14. *Let $\pi : \varrho_0 = (s_0, \epsilon, \nu_0) \hookrightarrow^* \varrho = (s, w, \nu)$ be a transition sequence of a UDTPDA1 from the initial configuration. If π is not empty, we refer the last step as $\lambda : \varrho' \hookrightarrow \varrho$, and the preceding sequence by $\pi' : \varrho_0 \hookrightarrow^* \varrho'$. Let $w = v_m \cdots v_1$. A snapshot is a digitized configuration $snap(\pi) = (\bar{Y}, num_1, num_2)$, where $num_1 = 2 \times (floor(\nu(c)) - n) + ceiling(frac(\nu(c)))$, $num_2 = 2 \times (floor(t_0) - n) + ceiling(frac(t_0))$ if w is not empty and $v_m = (\gamma, t_0, \cdots, t_k)$, otherwise $num_2 = 0$, and $\bar{Y} = \mathtt{digi}(dist(v_m) \uplus time(\nu))$.*

  In $snap(\pi)$, we define the $k + 1$-pointer $\bar{\rho}_1[i] = (x_i, \nu(x_i))$ for $0 \le i \le k$. We also define $\bar{\rho}_2[i] = (\gamma, proj(t_i))$ for $(\gamma, t_i) \in dist(v_m)$ if w is not empty, otherwise $\bar{\rho}_2$ is left undefined. A snapshot configuration $Snap(\pi)$ is inductively defined from $Snap(\pi')$.

$$\begin{cases} (s_0, snap(\epsilon)) & if \ \pi = \epsilon. \\ (s', snap(\pi) \ tail(Snap(\pi'))) & if \ \lambda \ is \ \mathbf{Timeprogress, Local, Test,} \\ & \mathbf{Assign, \ Increment} \ and \ \mathbf{Decrement.} \\ (s', snap(\pi) \ Snap(\pi')) & if \ \lambda \ is \ \mathbf{Push.} \\ (s', snap(\pi) \ tail(tail(Snap(\pi')))) & if \ \lambda \ is \ \mathbf{Pop.} \end{cases}$$

Definition 15. *For a UDTPDA1 $\langle S, s_0, \Gamma, X, c, \Delta \rangle$, we define the corresponding encoded snapshot VPS $\langle S, s_0, Digi, \mathbb{N}^2, \Delta_d \rangle$ with the initial configuration $\langle s_0, snap(\epsilon) \rangle$. Then Δ_d consists of:*

Time progress $\langle s, \bar{U} \rangle \hookrightarrow_{\mathcal{S}} \langle s, \bar{U}' \rangle$ *for* $\bar{U} \Rightarrow^* \bar{U}'$.
Local *($s \xrightarrow{\epsilon} s' \in \Delta$)* $\langle s, \bar{U} \rangle \hookrightarrow_{\mathcal{S}} \langle s', \bar{U} \rangle$.
Test *($s \xrightarrow{x \in I?} s' \in \Delta$ with $x \in X \cup \{c\}$) If $\mathtt{r}_i \subseteq I$ and $(x, \mathtt{r}_i) \in \bar{Y}$, where $\bar{U} = (\bar{Y}, num_1, num_2)$,* $\langle s, \bar{U} \rangle \hookrightarrow_{\mathcal{S}} \langle s', \bar{U} \rangle$.
Assign *($s \xrightarrow{x \leftarrow I} s' \in \Delta$ with $x \in X$) For $\mathtt{r}_i \subseteq I$,*
  $\langle s, \bar{U} \rangle \hookrightarrow_{\mathcal{S}} \langle s', (insert_x(delete(\bar{U}, x), (x, \mathtt{r}_i))) \rangle$.
Assign *($s \xrightarrow{c \leftarrow I} s' \in \Delta$) For $\mathtt{r}_i \subseteq I$,*
  $\langle s, \bar{U} \rangle \hookrightarrow_{\mathcal{S}} \langle s', (insert_c(delete(\bar{U}, c), (c, \mathtt{r}_i))) \rangle$.
Increment *($s \xrightarrow{c := c+1} s' \in \Delta$)*
  $\langle s, \bar{U} \rangle \hookrightarrow_{\mathcal{S}} \langle s', increase(\bar{U}) \rangle$.

Decrement $(s \xrightarrow{c := c - 1} s' \in \Delta)$
  $\langle s, \bar{U} \rangle \hookrightarrow_{\mathcal{S}} \langle s', decrease(\bar{U}) \rangle$.
Push $(s \xrightarrow{push(\gamma)} s' \in \Delta)$
  $\langle s, \bar{U} \rangle \hookrightarrow_{\mathcal{S}} \langle s', (init(map(\bar{U}, \gamma)))\bar{U} \rangle$.
Pop $(s \xrightarrow{pop(\gamma)} s' \in \Delta)$
  $\langle s, \bar{U}\bar{U}' \rangle \hookrightarrow_{\mathcal{S}} \langle s', propagate(\bar{U}, \bar{U}', \gamma) \rangle$.

By induction on the number of steps of transitions, the encoding relation between a UDTPDA1 with a single updatable clock and a snapshot VPS is observed.

Lemma 1. *Let us denote ϱ_0 and ϱ (resp. $\langle s_0, \tilde{w}_0 \rangle$ and $\langle s, \tilde{w} \rangle$) for the initial configuration and a configuration of a UDTPDA1 (resp. its snapshot VPS $\mathcal{S}$).*

(Preservation) *If $\pi : \varrho_0 \hookrightarrow^* \varrho$, there exists $\langle s, \tilde{w} \rangle$ such that $\langle s_0, \tilde{w}_0 \rangle \hookrightarrow_{\mathcal{S}}^* \langle s, \tilde{w} \rangle$ and $Snap(\pi) = \langle s, \tilde{w} \rangle$.*
(Reflection) *If $\langle s_0, \tilde{w}_0 \rangle \hookrightarrow_{\mathcal{S}}^* \langle s, \tilde{w} \rangle$, there exists $\pi : \varrho_0 \hookrightarrow^* \varrho$ with $Snap(\pi) = \langle s, \tilde{w} \rangle$.*

5   Nested Updatable Timed Automata

5.1   Nested Updatable Timed Automata

Nested Updatable Timed Automata(NeUTAs) extend NeTAs [1,2] by replacing every TA in NeTAs to a UTA1. A NeUTA has internal transitions, in which it will behave as a individual UTA1 having local, test, assign, increment and decrement transitions, and push and pop transitions. The stack of a NeUTA contains a pile of UTA1s which have been pushed.

Definition 16 (Nested Updatable Timed Automata). *A nested updatable timed automaton (NeUTA) is a quadruplet $\mathcal{N} = (T, \mathcal{A}_0, X, c, \Delta)$, where*

 – *T is a finite set $\{\mathcal{A}_0, \mathcal{A}_1, \cdots, \mathcal{A}_m\}$ of UTA1s, with the initial UTA1 $\mathcal{A}_0 \in T$. We assume the sets of states of $\mathcal{A}_i$, denoted by $S(\mathcal{A}_i)$, are mutually disjoint, i.e., $S(\mathcal{A}_i) \cap S(\mathcal{A}_j) = \emptyset$ for $i \neq j$. We denote the initial state of $\mathcal{A}_i$ by $q_0(\mathcal{A}_i)$.*
 – *X is the finite set of k local clocks and c is the updatable clock.*
 – *$\Delta \subseteq Q \times (Q \cup \{\varepsilon\}) \times Actions \times Q \times (Q \cup \{\varepsilon\})$ describes transition rules below, where $Q = \cup_{\mathcal{A}_i \in T} S(\mathcal{A}_i)$.*

Internal *$(q, \varepsilon, internal, q', \varepsilon)$, which describes an internal transition in the working UTA1 (placed at a control location) with $q, q' \in \mathcal{A}_i$.*
Push *$(q, \varepsilon, push, q_0(\mathcal{A}_{i'}), q)$, which interrupts the currently working UTA1 $\mathcal{A}_i$ at $q \in S(\mathcal{A}_i)$. Then, a UTA1 $\mathcal{A}_{i'}$ newly starts.*
Pop *$(q, q', pop, q', \varepsilon)$, which restarts $\mathcal{A}_{i'}$ in the stack from $q' \in S(\mathcal{A}_{i'})$ after $\mathcal{A}_i$ has finished at $q \in S(\mathcal{A}_i)$.*

Definition 17 (Semantics of NeUTAs). *Given a NeUTA $(T, \mathcal{A}_0, X, c, \Delta)$, the current control state is referred by q. Let $Val_X = \{\nu : X \cup \{c\} \to \mathbb{R}^{\geq 0}\}$. A configuration of a NeUTA is an element in $(Q \times Val_X, (Q \times Val_X)^*)$.*

- *Time progress transitions: $(\langle q, \nu \rangle, v) \xrightarrow{t} (\langle q, \nu + t \rangle, v + t)$ for $t \in \mathbb{R}^{\geq 0}$, where $v + t$ set $\nu' := \nu' + t$ of each $\langle q', \nu' \rangle$ in the stack.*
- *Discrete transitions: $\kappa \xrightarrow{\varphi} \kappa'$ is defined as follows.*
    - **Internal** *$(\langle q, \nu \rangle, v) \xrightarrow{\varphi} (\langle q', \nu' \rangle, v)$, if $\langle q, \nu \rangle \xrightarrow{\varphi} \langle q', \nu' \rangle$ is in Definition 2.*
    - **Push** *$(\langle q, \nu \rangle, v) \xrightarrow{push} (\langle q_0(\mathcal{A}_{i'}), \nu_0 \rangle, \langle q, \nu \rangle.v)$. The current working UTA1 (including its control location and clock valuation) is pushed to the stack.*
    - **Pop** *$(\langle q, \nu \rangle, \langle q', \nu' \rangle.w) \xrightarrow{pop} (\langle q', \nu' \rangle, w)$. The current working UTA1 is replaced with the topmost UTA1 in the stack and the topmost stack frame is removed.*

The initial configuration of NeUTA is $(\langle q_0(\mathcal{A}_0), \nu_0 \rangle, \varepsilon)$, where $\nu_0(x) = 0$ for $x \in X \cup \{c\}$. We use $\longrightarrow$ to range over these transitions, and $\longrightarrow^*$ is the reflexive and transitive closure of $\longrightarrow$.

5.2   Termination and Boundedness of NeUTAs

In this subsection we present a trivial encoding from NeUTAs to UDTPDA1s and so the termination and boundedness of NeUTAs are decidable.

Let $\mathcal{N} = (T, \mathcal{A}_0, X, c, \Delta)$ be a NeUTA. We define a corresponding UDTPDA1 $\mathcal{E}(\mathcal{N}) = \langle S, s_0, \Gamma, X, c, \nabla \rangle$, such that

- $S = \Gamma = \bigcup_{\mathcal{A}_i \in T} S(\mathcal{A}_i)$ is the set of all locations of UTA1s in T, with
- $s_0 = q_0(\mathcal{A}_0)$ is the initial location of the initial UTA $\mathcal{A}_0$ of $\mathcal{N}$.
- $X = \{x_1, \ldots, x_k\}$ is the set of k local clocks, and c is the singleton updatable clock.
- ∇ is the union $\bigcup_{\mathcal{A}_i \in T} \Delta(\mathcal{A}_i) \bigcup \mathcal{H}(\mathcal{N})$ where

$$\begin{cases} \Delta(\mathcal{A}_i) = \{\textbf{Local}, \textbf{Test}, \textbf{Assign}, \textbf{Increment}, \textbf{Decrement}\}, \\ \mathcal{H}(\mathcal{N}) \text{ consists of rules below.} \end{cases}$$

$$\begin{array}{ll} \textbf{Push } q \xrightarrow{push(q)} q_0(\mathcal{A}_{i'}) & \text{if } (q, \varepsilon, push, q_0(\mathcal{A}_{i'}), q) \in \Delta(\mathcal{N}) \\ \textbf{Pop } \quad q \xrightarrow{pop(q')} q' & \text{if } (q, q', pop, q', \varepsilon)) \in \Delta(\mathcal{N}) \end{array}$$

Definition 18. *Let $\mathcal{N}$ be a NeUTA $(T, \mathcal{A}_0, X, c, \Delta)$ and let $\mathcal{E}(\mathcal{N})$ be a UDT-PDA1 $\langle S, s_0, \Gamma, X, c, \nabla \rangle$. For a configuration $\kappa = (\langle \mathcal{A}, q, \nu \rangle, v)$ of $\mathcal{N}$ such that $v = (\mathcal{A}_1, q_1, \nu_1) \ldots (\mathcal{A}_n, q_n, \nu_n)$, $[\![\kappa]\!]$ denotes a configuration $(q, \overline{w}(\kappa), \nu)$ of $\mathcal{E}(\mathcal{N})$ where $\overline{w}(\kappa) = w_1 \cdots w_n$ with $w_i = (q_i, \nu_i)$.*

Lemma 2. *For a NeUTA $\mathcal{N}$, a UDTPDA1 $\mathcal{E}(\mathcal{N})$, and configurations κ, κ' of $\mathcal{N}$,*

(Preservation) *if $\kappa \longrightarrow_{\mathcal{N}} \kappa'$, then $[\![\kappa]\!] \hookrightarrow^*_{\mathcal{E}(\mathcal{N})} [\![\kappa']\!]$, and*

(Reflection) *if $[\![\kappa]\!] \hookrightarrow^*_{\mathcal{N}} \varrho$, there exists κ' with $\varrho \hookrightarrow^*_{\mathcal{E}(\mathcal{N})} [\![\kappa']\!]$ and $\kappa \longrightarrow^*_{\mathcal{N}} \kappa'$.*

By this encoding, we have our main result in Theorem 3.

Theorem 3. *The termination and boundedness of a NeUTA $(T, \mathcal{A}_0, X, c, \Delta)$ are decidable.*

6   Conclusion

This paper investigates termination and boundedness of NeUTAs, which extend NeTAs by replacing TAs with UTA1s. The proof of decidability can be seen as two phases of encoding, first an encoding NeUTAs to UDTPDA1s, then the one from UDTPDA1s to snapshot vector pushdown systems which extends the idea of digitization. Finally, the decidability of termination and boundedness of vector pushdown systems is obtained by the reduced reachability tree technique. The future work includes consider more verification problems of NeUTAs, as well as vector pushdown systems, such as coverability, reachability, and temporal logic model checking [18–21] et al..

References

1. 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). doi:10.1007/978-3-642-40229-6_12
2. 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, Cham (2015). doi:10.1007/978-3-319-22975-1_13
3. Bouyer, P., Dufourd, C., Fleury, E., Petit, A.: Updatable timed automata. Theoret. Comput. Sci. **321**(2–3), 291–345 (2004)
4. Wen, Y., Li, G., Yuen, S.: On reachability analysis of updatable timed automata with one updatable clock. In: Liu, S., Duan, Z. (eds.) SOFL+MSVL 2015. LNCS, vol. 9559, pp. 147–161. Springer, Cham (2016). doi:10.1007/978-3-319-31220-0_11
5. Leroux, J., Praveen, M., Sutre, G.: Hyper-ackermannian bounds for pushdown vector addition systems. In: Proceeding of the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2014), pp. 63:1–63:10. ACM (2014)
6. Alur, R., Dill, D.L.: A theory of timed automata. Theoret. Comput. Sci. **126**(2), 183–235 (1994)
7. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.-H.: Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) HS 1991-1992. LNCS, vol. 736, pp. 209–229. Springer, Heidelberg (1993). doi:10.1007/3-540-57318-6_30
8. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? J. Comput. Syst. Sci. **57**, 94–124 (1998)
9. Henzinger, T.A.: The theory of hybrid automata. In: Inan, M.K., Kurshan, R.P. (eds.) Verification of Digital and Hybrid Systems. NATO ASI Series, vol. 170, pp. 265–292. Springer, Heidelberg (2000). doi:10.1007/978-3-642-59615-5_13
10. Abdulla, P.A., Atig, M.F., Stenman, J.: Dense-timed pushdown automata. In: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science (LICS 2012), pp. 35–44. IEEE Computer Society (2012)
11. Bérard, B., Haddad, S., Sassolas, M.: Interrupt timed automata: verification and expressiveness. Formal Methods Syst. Des. **40**(1), 41–87 (2012)

12. Bérard, B., Haddad, S.: Interrupt timed automata. In: Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 197–211. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00596-1_15

13. 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, pp. 464–479. Springer, Heidelberg (2000). doi:10.1007/10722167_35

14. Bouyer, P., Dufourd, C., Fleury, E., Petit, A.: Expressiveness of updatable timed automata. In: Nielsen, M., Rovan, B. (eds.) MFCS 2000. LNCS, vol. 1893, pp. 232–242. Springer, Heidelberg (2000). doi:10.1007/3-540-44612-5_19

15. 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 2004), pp. 54–63. IEEE Computer Society (2004)

16. Abdulla, P.A., Jonsson, B.: Verifying networks of timed processes. In: Steffen, B. (ed.) TACAS 1998. LNCS, vol. 1384, pp. 298–312. Springer, Heidelberg (1998). doi:10.1007/BFb0054179

17. Abdulla, P., Jonsson, B.: Model checking of systems with many identical time processes. Theoret. Comput. Sci. **290**(1), 241–264 (2003)

18. Duan, Z.: Temporal Logic and Temporal Logic Programming. Science Press, Beijing (2005)

19. Duan, Z., Tian, C., Zhang, L.: A decision procedure for propositional projection temporal logic with infinite models. Acta Informatica **45**(1), 43–78 (2008)

20. Duan, Z., Yang, X., Koutny, M.: Framed temporal logic programming. Sci. Comput. Program. **70**(1), 31–61 (2008)

21. Tian, C., Duan, Z., Zhang, N.: An efficient approach for abstraction-refinement in model checking. Theoret. Comput. Sci. **461**, 76–85 (2012)