Nested timed automata with various clocks

WANG YuWei[a], LI GuoQiang[a,*] & YUEN Shoji[b]

[a] *School of Software, Shanghai Jiao Tong University, Shanghai 200240, China*

[b] *Graduate School of Information Science, Nagoya University, Japan*

Received February 18, 2016; accepted March 16, 2016

Abstract It is a great challenge to analyze a timed system extended with recursions, e. g. interrupt, since two dimensions of infinity, an unbounded stack, dense time recorded by clocks, occur in such a system. Furthermore, when investigating a recursive timed system, various types of clocks, such as global clocks, local clocks, frozen clocks, should be taken into consideration. The mixture of different types of clocks affects the expressiveness of recursive timed models. This paper gives detailed and complete investigation on the decidability results of a recursive timed system with different kinds of clocks, under a model named *nested timed automata*. These results can be naturally extended to other recursive timed systems.

Keywords Nested timed automata; Global clocks; Local clocks; Frozen clocks; Reachability

doi: 10.16262/j.cnki.1005－0841.2016.02.001

1 Introduction

Due to the rapid development of large and complex timed systems, requirements to model and analyze complex real-time frameworks with recursive context switches have been stressed. Difficulty comes from two dimensions of infinity, a stack with the unbounded number of symbols, and clocks recording dense time. The infinity raises difficulty to guarantee decidability of properties, e. g. , safety properties.

There are various formal models that describe the behaviors and expressiveness of timed systems, among which three of them are quite well-known. *Timed automata* (*TAs*) [1] are a finite automaton with a finite set of *clocks* that grow uniformly. A typical timed model with context switches is *timed pushdown automata* (*TPDAs*) [2], equipped with an unbounded stack, where clocks are not updated in the stack. This limitation is found unnatural in analyzing the timed behavior of programs since clock values should be updated in suspension. Recently, a new timed pushdown model, *dense timed pushdown automata* (*DTPDAs*) [3] has been proposed, where each symbol in the stack is equipped with local clocks named "ages", and all ages in the stack are updated uniformly for time passage. Reachability problem of DTPDAs is in EXPTIME [3].

For a component-based recursive timed system, clocks are naturally classified into *global clocks*, which can be updated and observed by all contexts, *local clocks*, which belong to the context of a component and will be stored in the stack when the component is interrupted and may be infinitely many due to recursion, and *frozen clocks*, which are special local clocks such that they are frozen when its context is preempted and restart when resumed.

This paper investigates a detailed and complete decidability results on component-based recursive timed systems with one type or combinations of various types of clocks, under a uniformed model, *Nested Timed Automata* (*NeTAs*) [4—6]. The results can be naturally extended to other similar systems, such as *interrupt timed automata* [7], *recursive timed automata* [8], *timed recursive state machines* [9], etc. The detailed results are shown in Table 1 and Table 2.

* Corresponding author. Email address: li. g@sjtu. edu. cn

URL: http://basics. sjtu. edu. cn/~liguoqiang/ (Guoqiang Li)

Table 1 Decidability results of NeTAs with three types of clocks

	Global	Local	Frozen
Global	DECIDABLE	DECIDABLE	UNDECIDABLE
Local	—	DECIDABLE	DECIDABLE
Frozen	—	—	DECIDABLE

Table 2 Decidability results of NeTAs with three types of clocks

Type of Clocks	Result
local + frozen + 1 global	DECIDABLE
local + frozen + 2 global	UNDECIDABLE

The undecidability results are gained by encoding the *Minsky machine* [10] into the target models. The decidability results are gained by encoding the target models to TPDAs, DTPDAs, or their extensions, preserving the decidability on reachability. The detailed decidability encoding results are summarized in Table 3.

Table 3 Decidability encoding on various NeTAs

Target model	Decidability model
gcNeTAs	TPDAs
fcNeTAs	TPDAs
lcNeTAs	DTPDAs
(gc+lc)NeTAs	DTPDAs
(gc+lc)NeTAs	DTPDA-Fs

"gc" is for global clocks, "lc" is for local clocks, and "fc" is for frozen clocks.

The remainder of this paper is structured as follows: In section 2 we introduce several mathematical models related to our topics. Section 3 defines syntax and the semantics of nested timed automata. Section 4 is devoted to proofs of basic decidability results of NeTAs. Section 5 shows that the general NeTAs are Turing-complete. Section 6 shows some decidable results on NeTAs with mixture of clocks. Section 7 reviews related work, and finally section 8 concludes this paper with summarized results.

2 Mathematical models

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 S be a set, and 2^S denote the power set of S, which is the set of all subsets of S.

Let $X = \{x_1, \cdots, x_n\}$ 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 represents all clocks in X assigned 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. $Val(X)$ is used to denote the set of clock valuation of X.

2.1 Timed automata

A timed automaton is a finite automaton augmented with a finite set of clocks [1, 11]. It can either stay in a control location for arbitrary time with all clocks proceeding accordingly, or switch from one control location to another instantly. At the same time, it may check if a clock belongs to an interval or updates a clock to an arbitrary value in an interval.

Definition 1 (Timed automata). A timed automaton (TA) is a tuple $A = (Q, q_0, F, X, \Delta) \in \mathcal{A}$, 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 control locations,
- X is a finite set of clocks,
- $\Delta \subseteq Q \times O \times Q$, where O is a set of *operations*. A transition $\delta \in \Delta$ is a triplet (q_1, ϕ, q_2), written as $q_1 \xrightarrow{\phi} q_2$, in which ϕ is either of
  - **Local** ϵ, an *empty* operation,
  - **Test** $x \in I$? where $x \in X$ is a clock and $I \in \mathcal{I}$ is an interval, and
  - **Assign** $x \leftarrow I$ where $x \in X$ and $I \in \mathcal{I}$.

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

The semantics of timed automata includes progress transitions, for time elapsing within one control location, and discrete transitions, for transference between two control locations.

Definition 2 (Semantics of TAs). Given a TA $A = (Q, q_0, F, X, \Delta)$, a configuration is a pair (q, ν) of a control location $q \in Q$, and a clock valuation ν on X. The transition relation of the TA is represented as follows:

- *Progress transition*: $(q, \nu) \xrightarrow{t}_{\mathscr{A}} (q, \nu + t)$, where $t \in \mathbb{R}^{\geq 0}$.
- *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 $\nu_1 = \nu_2$. The empty operation does not modify the clock valuations.
  - **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.
  - **Assign** $\phi = x \leftarrow I$, $\nu_2 = \nu_1[x \leftarrow r]$ where $r \in I$. The clock x is assigned an arbitrary value in I.

The initial configuration is (q_0, ν_0).

Example1. Figure 1 illustrates a TA consisting of states $S = \{q_0, q_1, q_2\}$, clocks $X = \{x_1, x_2, x_3\}$. Figure 2 shows one run of it with 4 transitions. Initially it is in location q_0 with both clock x_1 and clock x_2 assigned to zero. From (q_0, ν_0) to (q_0, ν_1), a progress transition elapses \$0.5\$ time units. In configuration (q_0, ν_1), a discrete transition $x_1 \in (0,1]$? tests whether x_1 belongs to the interval $(0,1]$, which is true. Then its location is changed to q_1 with clock valuation remaining the same. That is to say, its configuration is changed to (q_1, ν_2) with $\nu_2 = \nu_1$. Finally, from (q_1, ν_2) to (q_2, ν_3), x_2 is reset to 2.6, which is randomly picked from the interval $(2,3]$.

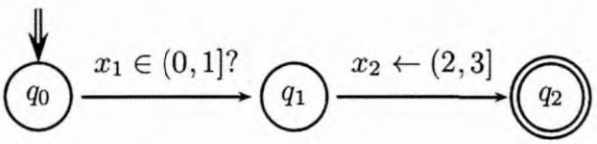

Figure 1 An Example of TAs.

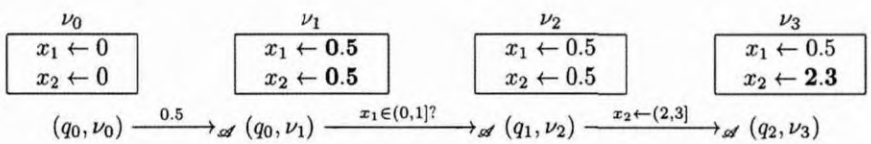

Figure 2 A run of the TA in Figure 1.

Although general verification problems, such as the language inclusion problem, are undecidable on timed automata, the reachability problem for real-time systems [1, 11] is decidable.

Fact 1. The reachability problem of timed automata is decidable [1, 11].

2.2 Timed pushdown automata

Timed pushdown automata (TPDAs) [2] extend TAs with a stack. A TPDA can behave as a TA, push a symbol to the stack or pop a symbol from the stack.

Definition 3 (Timed pushdown automata). A timed pushdown automaton (TPDA) is a quintuple $T = \langle Q, q_0, \Gamma, X, \Delta \rangle \in \mathcal{T}$, where

- Q is a finite set of control states with the initial state $q_0 \in Q$,
- Γ is a finite set of stack alphabets,
- X is a finite set of clocks, and
- $\Delta \subseteq Q \times Action^+ \times Q$ is a finite set of discrete transitions.

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

- **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 $I \in \mathcal{I}$,
- **Value passing** $x \leftarrow x'$ where $x, x' \in X$,
- **Push** $push(\gamma)$, where $\gamma \in \Gamma$ is a stack alphabet and
- **Pop** $pop(\gamma)$, where $\gamma \in \Gamma$ is a stack alphabet.

A transition as a sequence of actions $q_1 \xrightarrow{\varphi_1} \cdots \xrightarrow{\varphi_i} q_{i+1}$ 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.

Definition 4 (Semantics of TPDAs). For a TPDA $\langle Q, q_0, \Gamma, X, \Delta \rangle$, a configuration is a triplet (q, w, ν) with a control state $q \in Q$, a stack $w \in \Gamma^*$, and a clock valuation ν on X. The transition relation of the TPDA is defined as follows:

- Time progress: $(q, w, \nu) \xrightarrow{t}_{\mathcal{T}} (q, w, \nu + t)$, where $t \in \mathbb{R}^{\geq 0}$.
- Discrete transition: $(q_1, w_1, \nu_1) \xrightarrow{\varphi}_{\mathcal{T}} (q_2, w_2, \nu_2)$, 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 $\nu_2(x) \in I$ holds.
  - **Assign** $\varphi = x \leftarrow I$, then $w_1 = w_2$, $\nu_2 = \nu_1[x \leftarrow r]$ where $r \in I$.
  - **Value passing** $\varphi = x \leftarrow x'$, then $w_1 = w_2, \nu_2 = \nu_1[x \leftarrow \nu_1(x')]$.
  - **Push** $\varphi = push(\gamma)$, then $\nu_1 = \nu_2$, and $w_2 = \gamma \cdot w_1$.
  - **Pop** $\varphi = pop(\gamma)$, then $\nu_2 = \nu_1$, and $w_1 = \gamma \cdot w_2$.

The initial configuration $\varrho_0 = (q_0, \epsilon, \nu_0)$. We use $\longrightarrow_{\mathcal{T}}$ to range over these transitions, and $\longrightarrow_{\mathcal{T}}^*$ is the reflexive and transitive closure of $\longrightarrow_{\mathcal{T}}$.

Example 2. Figure 3 shows transitions between configurations of aTPDA consisting of states $S = \{q_i \mid 1 \leq i \leq 4\}$, clocks $X = \{x_1, x_2, x_3\}$ and stack symbols $\Gamma = \{a, b, d\}$. From q_1 to q_2, a discrete transition $push(d)$ pushes d to the stack. At location q_2, a progress transition elapses 2.6 time units, and each value grows older for 2.6. From q_2 to q_3, the value of x_2 is reset to 3.8, which lies in the interval $(2,5]$. The last transition pops symbol d from the stack.

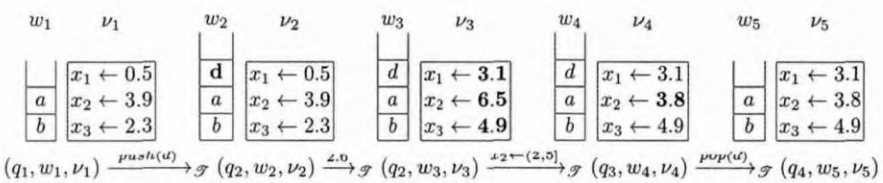

Figure 3 An example of TPDAs.

The reachability problem of TPDAs is gained by regionalizing the dense-time, and the corresponding region TPDA is a pushdown system.

Fact 2. The reachability problem of timed pushdown automata is decidable [2].

2.3 Dense timed pushdown automata

Dense timed pushdown automata [3, 12] extend timed pushdown automata with clocks 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 5 (Dense timed pushdown automata). A dense timed pushdown automaton is a tuple $D = \langle Q, q_0, \Gamma, X, \Delta \rangle \in \mathcal{D}$, where

- Q is a finite set of control states with the initial state $q_0 \in Q$,
- Γ is a finite set of stack alphabets,
- X is a finite set of clocks, and
- $\Delta \subseteq Q \times Action^+ \times Q$ 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 $q_1 \xrightarrow{\varphi_1} \cdots \xrightarrow{\varphi_i} q_{i+1}$, 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 $I \in \mathcal{I}$,
- **Value passing** $x \leftarrow x'$ where $x, x' \in X$.
- **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 6 (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 $q \in Q$, $w \in (\Gamma \times \mathbb{R}^{\geq 0})^*$, and a clock valuation ν on X. Time passage of the stack $w + t = (\gamma_1, t_1 + t). \cdots. (\gamma_n, t_n + t)$ for $w = (\gamma_1, t_1). \cdots. (\gamma_n, t_n)$.

The transition relation of a DTPDA consists of time progress and a discrete transition which is defined by that of actions below.

- Time progress: $(q, w, \nu) \xrightarrow{t}_{\mathcal{D}} (q, w + t, \nu + t)$, where $t \in \mathbb{R}^{\geq 0}$.
- 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 $r \in I$.
  - **Value passing**} $\varphi = x \leftarrow x'$, then $w_1 = w_2$, $\nu_2 = \nu_1[x \leftarrow \nu_1(x')]$.
  - **Push** $\varphi = push(\gamma, x)$, then $\nu_1 = \nu_2$, $w_2 = (\gamma, \nu_1(x)). w_1$.
  - **Pop** $\varphi = pop(\gamma, x)$, then $\nu_2 = \nu_1[x \leftarrow t]$, $w_1 = (\gamma, t). w_2$.

The initial configuration $\varrho_0 = (q_0, \epsilon, \nu_0)$. We use $\longrightarrow_{\mathcal{D}}$ to range over these transitions, and $\longrightarrow_{\mathcal{D}}^*$ is the reflexive and transitive closure of $\longrightarrow_{\mathcal{D}}$.

Remark 1. For simplicity of the later proofs, the definition of DTPDAs is slightly modified from the original [3]. **Value-passing** is introduced; instead $push(\gamma, I)$ and $pop(\gamma, I)$ are dropped, since they are described by $(x \leftarrow I; push(\gamma, x))$ and $(pop(\gamma, x); x \in I?)$, respectively.

Example 3. Figure 4 shows transitions between configurations of aDTPTA consisting of a singleton state set $S = \{ \bullet \}$ (omitted in the figure), clocks $X = \{x_1, x_2, x_3\}$ and stack symbols $\Gamma = \{a, b, d\}$. All transitions are similar to Figure 3. Note that the push transition between κ_1 and κ_2 needs to push symbol d and the clock value x_3 together. From κ_2 to κ_3, except for clocks, ages in the stack grow, too. The last pop transition pops an age in the stack to the clock x_1.

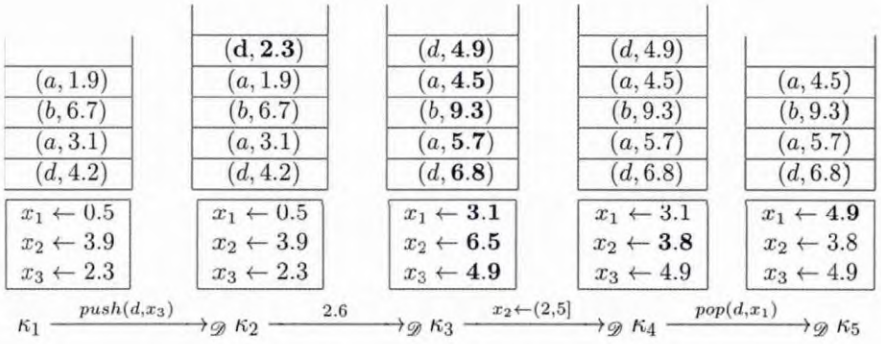

Figure 4 An example of DTPDAs.

Fact 3. The reachability problem of dense timed pushdown automata isdecidable [3, 13].

2.4 Dense timed pushdown automata with frozen ages

For our target proof, dense timed pushdown automata with frozen ages (DTPDA-Fs) [5] are introduced, and extended from DTPDAs, which is different from DTPDA in Definition 5 at:

- clocks are partitioned into the set X of local clocks (of the fixed number k) and the set C of global clocks,
- a tuple of ages (for simplicity, we fix the length of a tuple to be k) is pushed on the stack and/or popped from the stack, and
- each tuple of ages is either *proceeding* (as inDefinition 5) or *frozen*. After pushing the tuple, all local clocks are reset to zero.

Definition 7 (DTPDAs with frozen ages). A DTPDA with frozen ages (DTPDA-F) is a tuple $D = \langle S, s_0, \Gamma, X, C, \Delta \rangle \in \mathcal{D}$, where

- S is a finite set of control states with the initial state $s_0 \in S$,
- Γ is a finite set of stack alphabets,
- X is a finite set of local clocks (with $|X| = k$),
- C is a finite set of global clocks, and
- $\Delta \subseteq S \times Action^+ \times S$ 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 $s_1 \xrightarrow{\varphi_1} \cdots \xrightarrow{\varphi_i} s_{i+1}$, 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 $I \in \mathcal{I}$,
- **Value passing** $x \leftarrow x'$ where $x, x' \in X \cup C$,
- **Push**) $push(\gamma)$, where $\gamma \in \Gamma$ is a stack alphabet,
- **Freeze-Push (F-Push)** $fpush(\gamma)$, where $\gamma \in \Gamma$ is a stack alphabet, and
- **Pop** $pop(\gamma)$, where $\gamma \in \Gamma$ is a stack alphabet.

Definition 8 (Semantics of DTPDA-Fs). For a DTPDA-F $\langle S, s_0, \Gamma, X, C, \Delta \rangle$, a configuration is a triplet (s, w, ν) with a control location $s \in S$, a stack $w \in (\Gamma \times (\mathbb{R}^{\geq 0})^k \times \{0,1\})^*$, and a clock valuation ν on $X \cup C$. For a stack $w = (\gamma_1, \overline{t_1}, flag_1), \cdots, (\gamma_n, \overline{t_n}, flag_n)$, t-time passage on the stack, written as $w + t$, is $(\gamma_1, progress(\overline{t_1}, t, flag_1), flag_1), \cdots, (\gamma_n, progress(\overline{t_n}, t, flag_n), flag_n)$ where

$$progress(\overline{t}, t, flag) = \begin{cases} (t_1 + t, \cdots, t_k + t) & \text{if flag} = 1 \text{ and } \overline{t} = (t_1, \cdots, t_k) \\ \overline{t} & \text{if flag} = 0 \end{cases}$$

The transition relation of the DTPDA-F is defined as follows:

- Time progress: $(s, w, \nu) \xrightarrow{t}_{\mathcal{D}} (s, w + t, \nu + t)$, where $t \in \mathbb{R}^{\geq 0}$.
- Discrete transition: $(s_1, w_1, \nu_1) \xrightarrow{\varphi}_{\mathcal{D}} (s_2, w_2, \nu_2)$, 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 $r \in I$.
— **Value passing** $\varphi = x \leftarrow x'$, then $w_1 = w_2$, $\nu_2 = \nu_1[x \leftarrow \nu_1(x')]$.
— **Push** $\varphi = push(\gamma)$, then $\nu_2 = \nu_0$, $w_2 = (\gamma, (\nu_1(x_1), \cdots, \nu_1(x_k)), 1) \cdot w_1$ for $X = \{x_1, \cdots, x_k\}$.
— **F-Push** $\varphi = fpush(\gamma)$, then $\nu_2 = \nu_0$, $w_2 = (\gamma, (\nu_1(x_1), \cdots, \nu_1(x_k)), 0) \cdot w_1$ for $X = \{x_1, \cdots, x_k\}$.
— **Pop** $\varphi = pop(\gamma)$, then $\nu_2 = \nu_1[\overline{x} \leftarrow (t_1, \cdots, t_k)]$, $w_1 = (\gamma, (t_1, \cdots, t_k), flag) \cdot w_2$.

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

Example 4. Figure 5 shows transitions $\varrho_1 \longrightarrow_g \varrho_2 \longrightarrow_g \varrho_3 \longrightarrow_g \varrho_4$ of a DTPDA-F with $S = \{\bullet\}$ (omitted in the figure) $X = \{x_1, x_2\}$, $C = \{c_1\}$, and $\Gamma = \{a, b, d\}$. At $\varrho_1 \longrightarrow_g \varrho_2$, the values of x_1 and x_2 (0.5 and 3.9) are pushed with d , and frozen. After pushing, the value of x_1 and x_2 will be reset to zero. Then, x_2 is set a value in $(1, 2]$, say 1.7. At $\varrho_2 \longrightarrow_g \varrho_3$, time elapses but frozen ages in the top and third stack frames do not change. The rest (in **bold**) proceed. At $\varrho_3 \longrightarrow_g \varrho_4$, test whether the value of x_2 is in $(4, 6)$. Yes, then pop the stack and x_1 , x_2 are set to the popped ages. Last, the value of x_1 is set to c_1 .

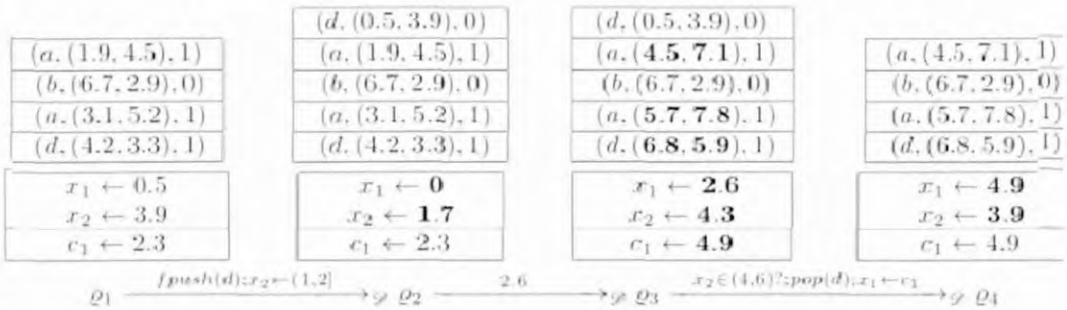

Figure 5 An example of DTPDA-Fs.

DTPDA-Fs are more expressive than DTPDAs. We have the following decidability result.

Fact 4. The reachability problem of dense timed pushdown automata with frozen ages is decidable, if $|C| = 1$[5].

3 Nested timed automata

Definition 9 (Nested Timed Automata). A nested timed automaton (NeTA) [4, 5] is a tuple $N = (T, A_0, X, C, \Delta) \in \mathcal{N}$, where

- T is a finite set of TAs $\{A_0, A_1, \cdots, A_n\}$, with the initial TA $A_0 \in T$. We assume the sets of control states of A_i , denoted by $Q(A_i)$, are mutually disjoint, i. e. , $Q(A_i) \cap Q(A_j) = \varnothing$ for $i \neq j$. We denote the initial state of A_i by $q_0(A_i)$.
- C is a finite set of global clocks, and X is the finite set of k local clocks.
- $\Delta \subseteq Q \times (Q \cup \{\epsilon\}) \times Actions^+ \times Q \times (Q \cup \{\epsilon\})$ describes transition rules below, where $Q = \bigcup_{A_i \in T} Q(A_i)$.

A transition rule is described by a sequence of $Actions = \{internal, push, fpush, pop, c \in I, c \leftarrow I, x \leftarrow c, c \leftarrow x\}$ where $c \in C$, $x \in X$, and $I \in \mathcal{I}$. The internal actions are **Local**, **Test**, **Assign**, and **Value passing** in Definition 3.

- **Internal** $(q, \epsilon, internal, q', \epsilon)$, which describes an internal transition in the working TA (placed at a control location) with $q, q' \in A_i$.
- **Push** $(q, \epsilon, push, q_0(A_{i'}), q)$, which interrupts the currently working TA A_i at $q \in Q(A_i)$. Then, a TA $A_{i'}$ newly starts. Note that all local clocks of A_i pushed onto the stack simultaneously proceed to global clocks.
- **F-Push** $(q, \epsilon, fpush, q_0(A_{i'}), q)$, which is the same as **Push** except that all local clocks of A_i are fro-

zen.

- **Pop** $(q,q',pop,q',\varepsilon)$, which restarts $A_{i'}$ in the stack from $q' \in Q(A_{i'})$ after A_i has finished at $q \in S(A_i)$ and all local clocks restart with the values of the ages.
- **Global-test** $(q,\varepsilon,c \in I?,q',\varepsilon)$, which tests whether the value of a global clock c is in I.
- **Global-assign** $(q,\varepsilon,c \leftarrow I,q',\varepsilon)$, which assigns a value in $r \in I$ to a global clock c.
- **Global-load** $(q,\varepsilon,x \leftarrow c,q',\varepsilon)$, which assign the value of a global clock c to a local clock $x \in X$ in the working TA.
- **Global-store** $(q,\varepsilon,c \leftarrow x,q',\varepsilon)$, which assign the value of a local clock $x \in X$ of the working TA to a global clock c.

Definition 10 (Semantics of NeTAs). Given a NeTA (T,A_0,X,C,Δ), the current control state is referred by q. Let $Val_X = \{\nu : X \to \mathbb{R}^{\geq 0}\}$ and $Val_C = \{\mu : C \to \mathbb{R}^{\geq 0}\}$. A configuration of a NeTA is an element in $(Q \times Val_X \times Val_C, (Q \times \{0,1\} \times Val_X)^*)$.

- Time progress transitions: $(\langle q,\nu,\mu\rangle,v) \xrightarrow{t} (\langle q,\nu+t,\mu+t\rangle,v+t)$ for $t \in \mathbb{R}^{\geq 0}$, where $v+t$ set $\nu' := progress(\nu',t,flag)$ of each $\langle q',flag,\nu'\rangle$ in the stack.
- Discrete transitions: are defined as follows.
  - **Internal** $(\langle q,\nu,\mu\rangle,v) \xrightarrow{\varphi} (\langle q',\nu',\mu\rangle,v)$, if $\langle q,\nu\rangle \xrightarrow{\varphi} \langle q',\nu'\rangle$ is in Definition 4, except for **push** or **pop**.
  - **Push** $(\langle q,\nu,\mu\rangle,v) \xrightarrow{push} (\langle q_0(A_{i'}),\nu_0,\mu\rangle.\langle q,1,\nu\rangle.v)$.
  - **F-Push** $(\langle q,\nu,\mu\rangle,v) \xrightarrow{f\text{-}push} (\langle q_0(A_{i'}),\nu_0,\mu\rangle.\langle q,0,\nu\rangle.v)$.
  - **Pop** $(\langle q,\nu,\mu\rangle.\langle q',flag,\nu'\rangle.w) \xrightarrow{pop} (\langle q',\nu',\mu\rangle,w)$.
  - **Global-test** $(\langle q,\nu,\mu\rangle,v) \xrightarrow{c \in I?} (\langle q',\nu,\mu\rangle,v)$, if $\mu(c) \in I$.
  - **Global-assign** $(\langle q,\nu,\mu\rangle,v) \xrightarrow{c \leftarrow I} (\langle q',\nu,\mu[c \leftarrow r]\rangle,v)$ for $r \in I$.
  - **Global-load** $(\langle q,\nu,\mu\rangle,v) \xrightarrow{x \leftarrow c} (\langle q',\nu[x \leftarrow \mu(c)],\mu\rangle,v)$.
  - **Global-store** $(\langle q,\nu,\mu\rangle,v) \xrightarrow{c \leftarrow x} (\langle q',\nu,\mu[c \leftarrow \nu(x)]\rangle,v)$.

The initial configuration of a NeTA is $\kappa_0 = (\langle q_0(A_0),\nu_0,\mu_0\rangle,\varepsilon)$, where $\nu_0(x) = 0$ for $x \in X$ and $\mu_0(c) = 0$ for $c \in C$. We use $\longrightarrow$ to range over these transitions, and $\longrightarrow^*$ is the reflexive and transitive closure of $\longrightarrow$.

Semantically, three kinds of clocks exist in NeTAs: global clocks, frozen clocks and local clocks. Global clocks are clocks belonging to the set C and only one copy of valuation Val_C is in the configuration. Frozen clocks are clocks in the stack which have been fpushed, and may have multiple copies of valuations in the stack. Local clocks are similar to frozen clocks, except that they have been pushed instead of fpushed.

Now we introduce three subclasses of NeTAs naturally, obtained by constraining which kinds of clocks they can have. The specific definition and semantics are omitted.

- **gcNeTAs** global clock NeTAs, where $X = \varnothing$.
- **fcNeTAs** frozen clock NeTAs, where $C = \varnothing$, and has no push rules.
- **lcNeTAs** local clock NeTAs, where $C = \varnothing$, and has no fpush rules.

Example 5. We take a simple example to show the usage of NeTAs. Assume that two processes access a shared buffer. One is to read from the buffer periodically each 4 time units. It accomplishes after it reads one or more data. The other is to write to the buffer periodically. The execution time is between 3 and 5time units. It will return after writing one or more data. The writing process may overtake the reading process which initially starts running. The NeTA is shown in Figure 6, with two TAs. A_1 and A_2 are for reading and writing processes, respectively. We have transition rules: $q_i^1 \xrightarrow{push} q_0^2$ and $q_w^2 \xrightarrow{pop} q_i^1$ for all $q_i^1 \in A_1$. Push transitions from q_1^1 to q_0^2 and from q_r^1 to q_0^2 are omitted in the figure. We use dash-line frames to represent the border of TAs in the NeTA, double-line arrows to indicate the initial location/TA, and double-line circles to represent the final locations of TAs. Obviously this example is a lcNeTA.

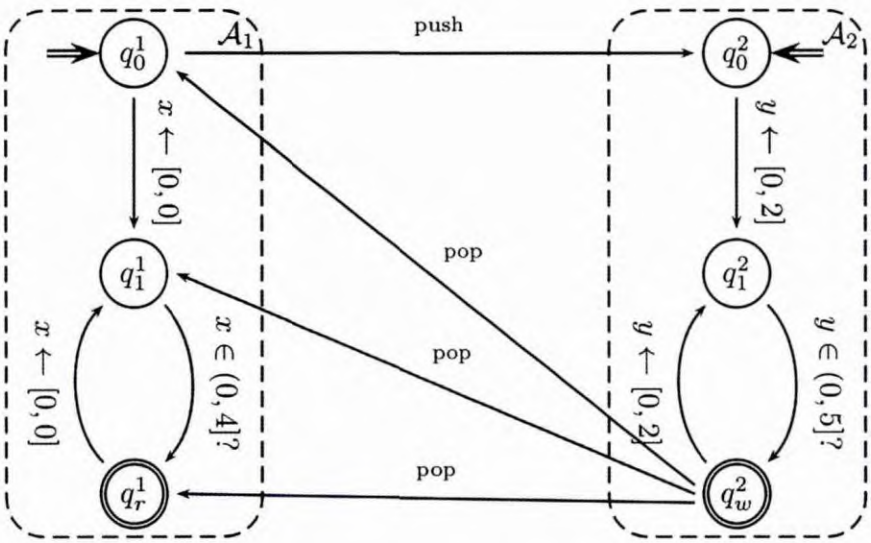

Figure 6 An Example of NeTAs.

Definition 11 (Reachability problem). A reachability problem of NeTAs is defined as the state reachability problem: Given a NeTA $N = (T, A_0, X, C, \Delta)$, and a control location $p_f \in Q(A)$ for some $A \in T$, decide whether there exists a stack v and clock valuations ν and μ, such that $\kappa_0 \longrightarrow^* (\langle q_f, \nu, \mu \rangle, v)$.

4 Basic decidable results

In this section we show that the reachability problem of a NeTA is decidable if it contains only one kind of clocks (global clocks, frozen clocks or local clocks).

4.1 gcNeTAs

Given a gcNeTA $N = (T, A_0, X, C, \Delta)$ with $X = \varnothing$, we define $\mathcal{E}(N) = \langle Q, q_0, \Gamma, X', \nabla \rangle$, as the target of TPDA encoding of N, such that

- $Q = \Gamma = \bigcup_{A_i \in T} Q(A_i)$ is the set of all control locations of TAs in T.
- $q_0 = q_0(A_0)$ is the initial control location of the initial TA A_0.
- $X' = C$ is the set of global clocks.
- ∇ is the union $\bigcup_{A_i \in T} \Delta(A_i) \cup \mathcal{G}(N) \cup \mathcal{H}(N)$ where

$$\begin{cases} \Delta(A_i) & = \{\textbf{Local}\}, \\ \mathcal{G}(N) & = \{\textbf{Global-test}, \textbf{Global-assign}\}, \\ \mathcal{H}(N) & \text{consists of rules below.} \end{cases}$$

Push $q \xrightarrow{push(q)} q_0(A_{i'})$ if $(q, \varepsilon, \varphi, q_0(A_{i'}), q) \in \Delta(N)$ where φ can be either push or fpush

Pop $q \xrightarrow{pop(q')} q'$ if $(q, q', pop, q', \varepsilon)) \in \Delta(N)$

Definition 12. Let N be a gcNeTA (T, A_0, X, C, Δ) with $X = \varnothing$ and let $\mathcal{E}(N)$ be a TPDA $\langle Q, q_0, \Gamma, X', \nabla \rangle$. For a configuration $\kappa = (\langle q, \nu, \mu \rangle, v)$ of N such that $v = (q_1, flag_1, \varepsilon) \cdots (q_n, flag_n, \varepsilon)$, $[\![\kappa]\!]$ denotes a configuration $(q, \bar{w}(\kappa), \mu)$ of $\mathcal{E}(N)$ where $q_i \in Q(A_i)$ and $\bar{w}(\kappa) = q_1 \cdots q_n$.

A gcNeTA has no **Test**, **Assign**, **Value passing**, **Global-load** and **Global-store** rules. Moreover, in a time progress transition, the stack of the gcNeTA will remain the same since the stack contains no clock valuations. We can prove that transitions are preserved and reflected by the encoding.

Lemma 1. For a gcNeTA N, its encoded TPDA $\mathcal{E}(N)$, and configurations κ, κ' of N

- (**Preservation**) if $\kappa \longrightarrow \kappa'$, then $[\![\kappa]\!] \longrightarrow_{\mathcal{T}}^* [\![\kappa']\!]$, and
- (**Reflection**) if $[\![\kappa]\!] \longrightarrow_{\mathcal{T}}^* \varrho$, there exists κ' with $\varrho \longrightarrow_{\mathcal{T}}^* [\![\kappa']\!]$ and $\kappa \longrightarrow^* \kappa'$.

By this encoding, we have the following theorem.

Theorem 1. The reachability of a gcNeTA is decidable.

4.2 fcNeTAs

In this subsection we prove a fcNeTA is decidable by encoding it to a TPDA using the *digitization* technique.

We introduce the notation in our digitization first. Let $N=(T,A_0,X,C,\Delta)$ be a fcNeTA with $C=\varnothing$ and no push rules, and let n be the largest integer (except for ω) appearing in Δ. For $v\in\mathbb{R}^{\geq 0}$, $proj(v)=r_i$ if $v\in r_i\in Intv(n)$, where

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

The idea of the next digitization is inspired by [14—16].

Definition 13. Let $frac(t)=t-floor(t)$ for $t\in\mathbb{R}^{\geq 0}$, and let $k=|X|$ be the number of local clocks. A *digitization* $digi:(X\to\mathbb{R}^{\geq 0})\to(2^{X\times Intv(n)})^{\leq k+1}$ is defined as follows.

For a clock valuation $\nu:(X\to\mathbb{R}^{\geq 0})$, let $Y_0,Y_1,\cdots,Y_m$ be a finite set that collect $(x,proj(t))$'s having the same $frac(t)$ for $t=\nu(x)$. Among them, Y_0 (which is possibly empty) is reserved for the collection of $(x,proj(t))$ with $frac(t)=0$ and $t\leq n$ (i.e., $proj(t)=r_{2i}$ for $0\leq i\leq n$). We assume that Y_i except for Y_0 is non-empty (i.e., $Y_i=\varnothing$ 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 $(x,proj(t))\in Y_m$ and $(x',proj(t'))\in Y_{i+1}$). We have $digi(\nu)=Y_0,Y_1,\cdots,Y_m$. Note that $m\leq k$ since each clock in X appears exactly once in $Y_0,Y_1,\cdots,Y_m$.

A word in $(2^{X\times Intv(n)})^{\leq k+1}$ is called a *digiword*. For simplicity, we define $\overline{Y}_0=digi(\nu_0)=\{(x,r_0)\mid x\in X\}$ is the initial digiword which corresponds to all clocks with the initial value 0.

Definition 14. Let $\overline{Y}=Y_0\cdots Y_m$, $\overline{Y}'=Y'_0\cdots Y'_{m'}\in(2^{X\times Intv(n)})^{\leq k+1}$ be digiwords. We define digiword operations as follows.

- **Inser** t_I Let $Z\in 2^{X\times Intv(n)}$ with $(x,r_i)\in Z$ for $x\in X$. $insert_I(\overline{Y},Z)$ inserts Z to $\overline{Y}$ such that
$$\begin{cases}\text{either take the union of Z and } Y_j \text{ for } j>0 \text{, or put Z at any place after } Y_0 & \\ & \text{if i is odd}\\ \text{take the union of Z and } Y_0 & \text{if i is even}\end{cases}$$
- **Delete** $delete(\overline{Y},x)$ for $x\in X$ is obtained from $\overline{Y}$ by deleting the element (x,r) indexed by x.
- **Permutation** A one-step permutation $\overline{Y}\Rightarrow\overline{Y}'$ is given by $\Rightarrow=\Rightarrow_s\cup\Rightarrow_c$, defined below. We denote $inc(Y_j)$ for Y_j in which each r_i is updated to r_{i+1} for $i<2k+1$.
  - $(\Rightarrow_s)$ $\overline{Y}\Rightarrow_s\overline{Y}'=\varnothing inc(Y_0)Y_1\cdots Y_m$.
  - $(\Rightarrow_c)$ $\overline{Y}\Rightarrow_c\overline{Y}'=inc(Y_m)inc(Y_0)Y_1\cdots Y_{m-1}$.

The following is the encoding from a fcNeTA to a TPDA. Given a fcNeTA $N=(T,A_0,X,C,\Delta)$ with $C=\varnothing$ and no push rules, $\mathcal{E}(N)=\langle Q,q_0,\Gamma,X',\nabla\rangle$, as the target of TPDA encoding of N, such that

- $Q=\Gamma=\bigcup_{A_i\in T}Q(A_i)\times(2^{X\times Intv(n)})^{\leq k+1}$. The first part is the set of all control locations of TAs in T. The second part is the set of digiwords. Note that both Q and Γ are finite.
- $q_0=\langle q_0(A_0),\overline{Y}_0\rangle$. $q_0(A_0)$ is the initial control location of the initial TA A_0. The following digiword corresponds to all clocks with the initial value 0.
- $X'=\varnothing$ is the empty set.
- ∇ consists of rules below.
  - **Local** $\langle q,\overline{Y}\rangle\xrightarrow{\epsilon}\langle q,\overline{Y}'\rangle$ for $\overline{Y}\Rightarrow^*\overline{Y}'$.
  - **Local** $\langle q,\overline{Y}\rangle\xrightarrow{\epsilon}\langle q',\overline{Y}\rangle$, if $(q,\varepsilon,\epsilon,q',\varepsilon))\in\Delta(N)$.
  - **Local** $\langle q,\overline{Y}\rangle\xrightarrow{\epsilon}\langle q',\overline{Y}\rangle$, if $(q,\varepsilon,x\in I?,q',\varepsilon))\in\Delta(N)$, $r_i\subseteq I$ and $(x,r_i)\in\overline{Y}$.
  - **Local** $\langle q,\overline{Y}\rangle\xrightarrow{\epsilon}\langle q',insert_I(delete(\overline{Y},x),\{(x,r_i)\})\rangle$, if $(q,\varepsilon,x\leftarrow I,q',\varepsilon))\in\Delta(N)$ and $r_i\subseteq I$.

— **Push** $\langle q, \overline{Y} \rangle \xrightarrow{push(\langle q, \overline{Y} \rangle)} \langle q_0(A_i), \overline{Y}_0 \rangle$, if $(q, \varepsilon, fpush, q_0(A_i), q) \in \Delta(N)$.

— **Pop** $\langle q, \overline{Y} \rangle \xrightarrow{pop(\langle q', \overline{Y}' \rangle)} \langle q', \overline{Y}' \rangle$, if $(q, q', pop, q', \varepsilon)) \in \Delta(N)$.

The initial configuration of the encoded TPDA is $\langle q_0, \epsilon \rangle$.

Definition 15. Let N be a fcNeTA (T, A_0, X, C, Δ) with $C = \varnothing$ and no push rules, and let $\mathcal{E}(N)$ be a TPDA $\langle Q, q_0, \Gamma, X', \nabla \rangle$. For a configuration $\kappa = (\langle q, \nu, \mu \rangle, v)$ of N such that $v = (q_1, 0, \nu_1) \cdots (q_n, 0, \nu_n)$, $[\![\kappa]\!]$ denotes a configuration $(\langle q, digi(\nu) \rangle, \overline{w}(\kappa), \mu)$ of $\mathcal{E}(N)$, where $q_i \in Q(A_i)$ and $\overline{w}(\kappa) = w_1 \cdots w_n$ with $w_i = \langle q_i, digi(\nu_i) \rangle$.

A fcNeTA has no **Push**, **Global-test**, **Global-assign**, **Global-load** and **Global-store** rules. In a time progress transition, the stack of the fcNeTA will stay the same since clocks are f-pushed. We can prove that transitions are preserved and reflected by the encoding.

Lemma 2. For a fcNeTA N , its encoded TPDA $\mathcal{E}(N)$, and configurations κ , κ' of N

- (**Preservation**) if $\kappa \longrightarrow \kappa'$, then $[\![\kappa]\!] \longrightarrow_T^* [\![\kappa']\!]$, and
- (**Reflection**) if $[\![\kappa]\!] \longrightarrow_{\mathcal{T}}^* \varrho$, there exists κ' with $\varrho \longrightarrow_{\mathcal{T}}^* [\![\kappa']\!]$ and $\kappa \longrightarrow^* \kappa'$.

By this encoding, we have the following theorem.

Theorem 2. The reachability of a fcNeTA is decidable.

4.3 lcNeTAs

Given a lcNeTA $N = (T, A_0, X, C, \Delta)$ with $C = \varnothing$ and no fpush rules, we define $\mathcal{E}(N) = \langle S, s_0, \Gamma, X', \nabla \rangle$, as the target of DTPDA encoding of N , such that

- $S = \Gamma = \bigcup_{A_i \in T} Q(A_i)$ is the set of all control states of TAs in T .
- $s_0 = q_0(A_0)$ is the initial control state of the initial TA A_0 .
- $X' = X \cup \{d\}$ is the union of set of local clocks and global clocks and a special dummy clock d to fulfill the field of push and pop rules, whose value does not matter. $X = \{x_1, \ldots, x_k\}$ is the set of k local clocks.
- ∇ is the union $\bigcup_{A_i \in T} \Delta(A_i) \cup H(N)$ where

$\begin{cases} \Delta(A_i) & = \{\textbf{Local}, \textbf{Test}, \textbf{Assign}, \textbf{Value-passing}\}, \\ \mathcal{H}(N) & \text{consists of rules below.} \end{cases}$

— **Push** $q \xrightarrow{push(q,d)} \xrightarrow{push(x_1,x_1)} \cdots \xrightarrow{push(x_k,x_k)} \xrightarrow{x_1 \leftarrow [0,0]} \cdots \xrightarrow{x_k \leftarrow [0,0]} q_0(A_{i'})$, if $(q, \varepsilon, push, q_0(A_{i'}), q) \in \Delta(N)$.

— **Pop** $q \xrightarrow{pop(x_k,x_k)} \cdots \xrightarrow{pop(x_1,x_1)} \xrightarrow{pop(q',d)} q'$, if $(q, q', pop, q', \varepsilon)) \in \Delta(N)$.

Example 6. The lcNeTA in Figure 6 is encoded into a DTPDA in Figure 7. The encoded push transitions from q_1^1 to q_0^2 and from q_r^1 to q_0^2 are omitted.

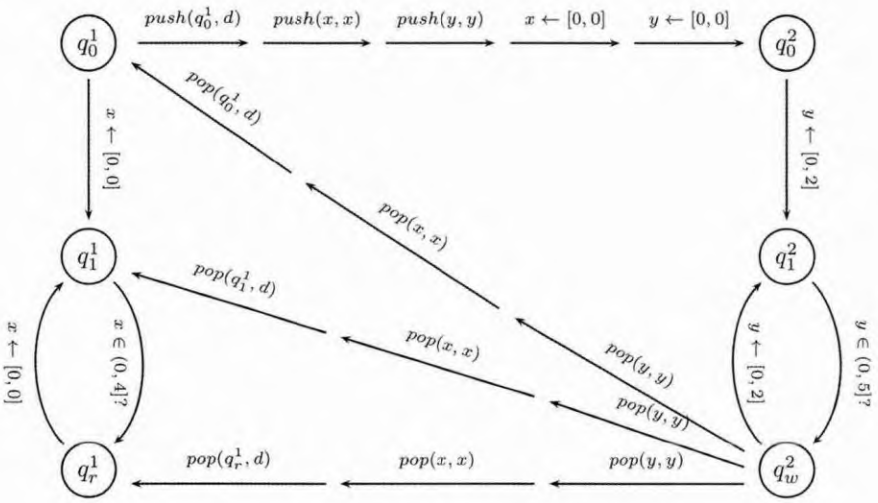

Figure 7 Encoding a lcNeTA to a DTPDA.

Definition 16. Let N be a lcNeTA (T, A_0, X, C, Δ) with $C = \varnothing$ and no fpush rules and let $\mathcal{E}(N)$ be a DTPDA $\langle S, s_0, \Gamma, X', \nabla \rangle$. For a configuration $\kappa = (\langle q, \nu, \mu \rangle, v)$ of N such that $v = (q_1, 1, \nu_1) \cdots (q_n, 1, \nu_n)$, $[\![\kappa]\!]$ denotes a configuration $(q, \overline{w}(\kappa), \mu)$ of $\mathcal{E}(N)$, where $\overline{w}(\kappa) = w_1 \cdots w_n$ with $w_i = (x_k, \nu_i(x_k)) \cdots (x_1, \nu_1(x_k))(q_i, 0)$.

A lcNeTA has no **F-Push**, **Global-test**, **Global-assign**, **Global-load** and **Global-store** rules. We can prove that transitions are preserved and reflected by the encoding.

Lemma 3. For a lcNeTA N, its encoded DTPDA $\mathcal{E}(N)$, and configurations κ, κ' of N

- (**Preservation**) if $\kappa \longrightarrow \kappa'$, then $[\![\kappa]\!] \longrightarrow_{\mathcal{G}}^{*} [\![\kappa']\!]$, and
- (**Reflection**) if $[\![\kappa]\!] \longrightarrow_{\mathcal{G}}^{*} \varrho$, there exists κ' with $\varrho \longrightarrow_{\mathcal{G}}^{*} [\![\kappa']\!]$ and $\kappa \longrightarrow^{*} \kappa'$.

Theorem 3. The reachability of a lcNeTA is decidable.

5 General undecidable results

For showing the undecidability, we encode the halting problem of Minsky machines [10] into the reachability of a NeTA.

Definition 17 (Minsky machine). 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, and
- 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 $(ct := ct - 1$, goto $l_k)$ else goto l_m,

  where $ct \in C$, $d_i \in D$ and $l_k, l_m \in L$.

By the N-wrapping technique [17], a Minsky machine can be encoded into a NeTA $N = (T, A_0, X, C, \Delta)$, with $T = \{A_0, A_1, A_2\}$ where

- $S(A_0) = \{q_i \mid 0 \leq i \leq 6\}$, $X(A_0) = \{x_f, x_p\}$, $S(A_1) = \{q_i \mid 7 \leq i \leq 12\}$, $X(A_1) = \{x_1, dum_1\}$, $S(A_2) = \{q_i \mid 13 \leq i \leq 15\}$, and $X(A_2) = \{dum_2, dum_3\}$, where the dummy clocks dum_i's are prepared for fulfilling $k = 2$.
- $|X| = 2$. We introduce different names of 2 local clocks to clarify the context.
- $C = \{c_{sys}, c_v\}$ where c_{sys} is a system clock that will be reset to zero when its value becomes N; c_v encodes values of two counters as $\mu(c_v) = 2^{-ct_1} \cdot 3^{-ct_2}$.

Decrementing and incrementing the counter ct_1 are simulated by doubling and halving of the value of the clock c_v, respectively, while those for ct_2 are simulated by tripling and thirding the value of clock c_v. Zero-test of ct_1 is simulated by (1) multiplying the value of c_v by a power of 3, and (2) comparing it with 3. Similar for ct_2. These operations are formally described below.

Doubling: Initially $\nu(c_{sys}) = 0$ and $\nu(c_v) = d$ with $0 < d < 1$. Then the doubling the value of c_v is obtained at the end, as $\nu(c_{sys}) = 0$ and $\nu(c_v) = 2d$.

$$q_0 \xrightarrow{c_v \in [N,N]?} \xrightarrow{c_v \leftarrow [0,0]} q_1 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} \xrightarrow{x_f \leftarrow c_v} q_2 \xrightarrow{f-push}$$

$$q_7 q_2 \xrightarrow{c_{sys} \in [0,0]?} q_8 q_2 \xrightarrow{c_v \in [N,N]?} q_9 q_2 \xrightarrow{pop} q_2 \xrightarrow{c_v \in [N,N]?} \xrightarrow{c_v \leftarrow x_f}$$

$$q_3 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} q_4$$

Halving: During halving the value of c_v, it will be nondeterministically stored to x_f in a frozen TA. When c_{sys} is reset to zero, x_f will be popped to restart. Only if the values of x_f and c_v coincide (i.e., they reach N together), the value of c_v becomes $d/2$ when c_{sys} is wrapped twice.

$$q_0 \xrightarrow{c_v \in [N,N]?} \xrightarrow{c_v \leftarrow [0,0]} q_1 \xrightarrow{x_f \leftarrow c_v} \xrightarrow{c_v \leftarrow [0,0]} q_2 \xrightarrow{f-push} q_7 q_2 \xrightarrow{c_v \in [0,0]?}$$

$$q_8 q_2 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} q_9 q_2 \xrightarrow{pop} q_2 \xrightarrow{c_{sys} \in [0,0]?}$$

$$q_3 \xrightarrow{c_v \in [N,N]?} \xrightarrow{x_f \in [N,N]?} \xrightarrow{c_v \leftarrow [0,0]} q_4 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} q_5$$

Tripling: Tripling requires an extra local clock x_p in A_0.

$$q_0 \xrightarrow{c_v \in [N,N]?} \xrightarrow{c_v \leftarrow [0,0]} q_1 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} \xrightarrow{x_f \leftarrow c_v} q_2 \xrightarrow{f-push}$$

$$q_7 q_2 \xrightarrow{c_{sys} \in [0,0]?} q_8 q_2 \xrightarrow{c_v \in [N,N]?} q_9 q_2 \xrightarrow{pop} q_2 \xrightarrow{c_v \in [N,N]?} \xrightarrow{c_v \leftarrow x_f} \xrightarrow{x_p \leftarrow [0,0]}$$

$$q_3 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} \xrightarrow{x_f \leftarrow c_v} \xrightarrow{c_v \leftarrow x_p} q_4 \xrightarrow{f-push} q_{10} q_4 \xrightarrow{c_{sys} \in [0,0]?}$$

$$q_{11} q_4 \xrightarrow{c_v \in [N,N]?} q_{12} q_4 \xrightarrow{pop} q_4 \xrightarrow{c_v \in [N,N]?} \xrightarrow{c_v \leftarrow x_f} q_5 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} q_6$$

Thirding: Thirding requires an extra TA A_2 with a local clock x_1.

$$q_0 \xrightarrow{c_v \in [N,N]?} \xrightarrow{c_v \leftarrow [0,0]} q_1 \xrightarrow{x_f \leftarrow c_v} \xrightarrow{c_v \leftarrow [0,0]} q_2 \xrightarrow{f-push} q_7 q_2 \xrightarrow{c_v \in [0,0]?}$$

$$q_8 q_2 \xrightarrow{x_1 \leftarrow c_v} \xrightarrow{c_v \leftarrow [0,0]} q_9 q_2 \xrightarrow{f-push} q_{13} q_9 q_2 \xrightarrow{c_v \in [0,0]?}$$

$$q_{14} q_9 q_2 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} q_{15} q_9 q_2 \xrightarrow{pop} q_9 q_2 \xrightarrow{c_{sys} \in [0,0]?}$$

$$q_{10} q_2 \xrightarrow{c_v \in [N,N]?} \xrightarrow{x_1 \in [N,N]?} \xrightarrow{c_v \leftarrow [0,0]} q_{11} q_2 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} q_{12} q_2 \xrightarrow{pop}$$

$$q_2 \xrightarrow{c_{sys} \in [0,0]?} q_3 \xrightarrow{c_v \in [N,N]?} \xrightarrow{x_f \in [N,N]?} \xrightarrow{c_v \leftarrow [0,0]} q_4 \xrightarrow{c_{sys} \in [N,N]?} \xrightarrow{c_{sys} \leftarrow [0,0]} q_5$$

6 Advanced decidable results

In this section we show two decidable cases of NeTAs which contain more than one kinds of clocks.

6.1 (gc+lc)NeTAs

A (gc+lc)NeTA is a NeTA having no fpush rules. In this subsection we prove a (gc+lc)NeTA is decidable by encoding it to a DTPDA. Note that this encoding is similar to the lcNeTA case except that a (gc+lc)NeTA have global transitions which need to be encoded.

Given a (gc+lc)NeTA $N = (T, A_0, X, C, \Delta)$ with no fpush rules, we define $\mathcal{E}(N) = \langle S, s_0, \Gamma, X', \nabla \rangle$, as the target of DTPDA encoding of N, such that

- $S = \Gamma = \bigcup_{A_i \in T} Q(A_i)$ is the set of all control states of TAs in T.
- $s_0 = q_0(A_0)$ is the initial control state of the initial TA A_0.
- $X' = X \cup C \cup \{d\}$ is the union of sets of local clocks and global clocks and a special dummy clock d to fulfill the field of push and pop rules, whose value does not matter. $X = \{x_1, \ldots, x_k\}$ is the set of k local clocks.
- ∇ is the union $\bigcup_{A_i \in T} \Delta(A_i) \cup G(N) \cup H(N)$ where

$$\begin{cases} \Delta(A_i) & = \{\textbf{Local}, \textbf{Test}, \textbf{Assign}, \textbf{Value-passing}\}, \\ G(N) & = \{\textbf{Global-test}, \textbf{Global-assign}, \textbf{Global-load}, \textbf{Global-store}\}, \\ H(N) & \text{consists of rules below.} \end{cases}$$

— **Push** $q \xrightarrow{push(q,d)} \xrightarrow{push(x_1,x_1)} \cdots \xrightarrow{push(x_k,x_k)} \xrightarrow{x_1 \leftarrow [0,0]} \cdots \xrightarrow{x_k \leftarrow [0,0]} q_0(A_i)$, if $(q, \varepsilon, push, q_0(A_i), q) \in \Delta(N)$.

— **Pop** $q \xrightarrow{pop(x_k,x_k)} \cdots \xrightarrow{pop(x_1,x_1)} \xrightarrow{pop(q',d)} q'$, if $(q, q', pop, q', \varepsilon)) \in \Delta(N)$.

Definition 18. Let N be a (gc+lc)NeTA (T, A_0, X, C, Δ) with no fpush rules and let $\mathcal{E}(N)$ be a DTPDA $\langle S, s_0, \Gamma, X', \nabla \rangle$. For a configuration $\kappa = (\langle q, \nu, \mu \rangle, v)$ of N such that $v = (q_1, 1, \nu_1) \cdots (q_n, 1, \nu_n)$, $[\![\kappa]\!]$ denotes a configuration $(q, \overline{w}(\kappa), \mu)$ of $\mathcal{E}(N)$ where $\overline{w}(\kappa) = w_1 \cdots w_n$ with $w_i = (x_k, \nu_i(x_k)) \cdots (x_1, \nu_1(x_k))(q_i, 0)$.

We can prove that transitions are preserved and reflected by the encoding.

Lemma 4. For a (gc+lc)NeTA N, its encoded DTPDA $\mathcal{E}(N)$, and configurations κ, κ' of N

- (**Preservation**) if $\kappa \longrightarrow \kappa'$, then $[\![\kappa]\!] \longrightarrow_{\mathcal{D}}^{*} [\![\kappa']\!]$, and
- (**Reflection**) if $[\![\kappa]\!] \longrightarrow_{\mathcal{D}}^{*} \varrho$, there exists κ' with $\varrho \longrightarrow_{\mathcal{D}}^{*} [\![\kappa']\!]$ and $\kappa \longrightarrow^{*} \kappa'$.

Theorem 4. The reachability of a (gc+lc)NeTA is decidable.

6.2 (1gc+1c+fc)NeTAs

In this subsection, we show that even having all three types of clocks, a NeTA is decidable iff it contains only one global clock, by encoding the model to DTPDA-F with one global clock.

Let $N = (T, A_0, X, C, \Delta)$ be a NeTA with $|C| = 1$. We define a corresponding DTPDA-F $\mathcal{E}(N) = \langle S, s_0, \Gamma, X, C, \nabla \rangle$, such that

- $S = \Gamma = \bigcup_{A_i \in T} S(A_i)$ is the set of all locations of TAs in T, with
- $s_0 = q_0(A_0)$ is the initial location of the initial TA A_0 of N.
- $X = \{x_1, \cdots, x_k\}$ is the set of k local clocks, and C is the singleton set $\{c\}$.
- ∇ is the union $\bigcup_{A_i \in T} \Delta(A_i) \cup G(N) \cup H(N)$ where

$$\begin{cases} \Delta(A_i) & = \{\textbf{Local}, \textbf{Test}, \textbf{Assign}, \textbf{Value-passing}\}, \\ G(N) & = \{\textbf{Global-test}, \textbf{Global-assign}, \textbf{Global-load}, \textbf{Global-store}\}, \\ H(N) & \text{consists of rules below.} \end{cases}$$

— **Push** $q \xrightarrow{push(q)} q_0(A_{i'})$, if $(q, \varepsilon, push, q_0(A_{i'}), q) \in \Delta(N)$.

— **F-Push** $q \xrightarrow{fpush(q)} q_0(A_{i'})$, if $(q, \varepsilon, f\text{-}push, q_0(A_{i'}), q) \in \Delta(N)$.

— **Pop** $q \xrightarrow{pop(q')} q'$, if $(q, q', pop, q', \varepsilon)) \in \Delta(N)$.

Definition 19. Let N be a NeTA (T, A_0, X, C, Δ) and let $\mathcal{E}(N)$ be a DTPDA-F $\langle S, s_0, \Gamma, X, C, \nabla \rangle$. For a configuration $\kappa = (\langle q, \nu, \mu \rangle, v)$ of N such that $v = (q_1, flag_1, \nu_1) \cdots (q_n, flag_n, \nu_n)$, $[\![\kappa]\!]$ denotes a configuration $(q, \overline{w}(v), \nu \cup \mu)$ of $\mathcal{E}(N)$ where $q_i \in S(A_i)$ and $\overline{w}(v) = w_1 \cdots w_n$ with $w_i = (q_i, \nu_i, flag_i)$.

Lemma 5. For a (gc+lc+fc)NeTA N with one global clock, its encoded DTPDA $\mathcal{E}(N)$, and configurations κ, κ' of N

- (**Preservation**) if $\kappa \longrightarrow \kappa'$, then $[\![\kappa]\!] \longrightarrow_{\mathcal{D}}^{*} [\![\kappa']\!]$, and
- (**Reflection**) if $[\![\kappa]\!] \longrightarrow_{\mathcal{D}}^{*} \varrho$, there exists κ' with $\varrho \longrightarrow_{\mathcal{D}}^{*} [\![\kappa']\!]$ and $\kappa \longrightarrow^{*} \kappa'$.

The detailed proof is in Appendix A.

By this encoding, we have the following result.

Theorem 5. The reachability of a NeTA (T, A_0, X, C, Δ) is decidable, if $|C| = 1$.

7 Related work

After TAs [1] had been proposed, lots of researches were intended timed context switches. TPDAs were firstly proposed in [2], which enjoys decidability of reachability problem. Dang proved in [18] the decidability of binary reachability (i. e., the set of all pairs of configurations such that one can reach the other) of TPDAs. All clocks in TPDAs were treated globally, which were not affected when the context switches.

Our model relied heavily on a recent significant result, named *dense timed pushdown automata* (*DTPDAs*) [3, 12]. The difference between DTPDAs and NeTAs was the hierarchical feature. In NeTAs, a finite set of local clocks were pushed into the stack at the same time. When a pop action happens, the values of clocks belonging to popped TA were popped simultaneously and reused. This feature eased much for modelling the behavior of time-aware software. In DTPDAs, local clocks must be dealt within some proper bookkeeping process, which was not essential part of the analysis. In [19], a discrete version of DTPDAs, named *discrete timed pushdown automata* was introduced, where time was incremented in discrete steps and thus the ages of clocks and stack symbols are in the natural numbers. This made the reachability problem much simpler, and easier for efficient implementation.

Based on *recursive state machines* [20], two similar timed extensions, *timed recursive state machines* (*TRSMs*) [9] and *recursive timed automata* (*RTAs*) [8], were given independently. In these models, contexts were explicitly defined. The finite number of clocks was distinguished into two categories, call-by-reference and call-by-value. When entering a fresh context, clock values were stored in the stack. After

popping, the values of call-by-reference clocks were unaltered, while the values of call-by-value ones restored to the previous value from the stack. When either all of clocks or none of them were call-by-reference, the state reachability problem was decidable. The main differences from NeTAs were, the two models had no stack time-update during progress transitions, and the number of clocks was essentially finite. The *hierarchical timed automata* (*HTAs*) [21] kept the similar structure of clocks, where only a bounded number of levels were treated, while NeTAs treated an unbounded number of levels.

Interrupt timed automata (*ITAs*) [7], which are well suited to the description of multi-task systems with interruptions in a single processor environment, is a subclass of hybrid automata. It is shown that in ITA the reachability problem is in 2-EXPSPACE and in PSPACE when the number of clocks is fixed, with a procedure based on a generalized class graph.

The class of *extended timed pushdown automata* (*ETPDAs*) was proposed in [9]. An ETPDA was a pushdown automaton enriched with a set of clocks, with an additional stack used to store/restore clock valuations. Two stacks were independent. ETPDAs have the same expressiveness with TRTMs via weak timed bisimulation. The reachability problem of ETPDAs was undecidable, while the decidability held with a syntactic restriction on the stack.

Controller automata (*CAs*) [22, 23] was proposed to analyze interrupts. In a CA, a TA was assigned to each state. A TA at a state may be preempted by another state by a labeled transition. The number of clocks of CAs was finite, and thus when existing preemption loop, only the newest timed context was kept. Given a strict partial order over states, an ordered controller automaton was able to be faithfully encoded into a TA, and thus the safety property of the restrictive version was preserved.

The *updatable timed automata* (*UTAs*) [24—28] proposed the possibility of updating the clocks in a more elaborate way, where the value of a clock could be reassigned to a basic arithmetic computation result of values of other clocks. The researches gave undecidability and decidability results for several specific cases [26, 27]. The expressiveness of the UTAs was also investigated. UTAs raised up another way to accumulate time when timed context switches, and thus *updatable timed pushdown automata* (*UTPDAs*) could be an interesting extension.

8 Conclusion

We have investigated the decidability of reachability problem of nested timed automata (NeTAs) with different types of clocks. It was shown that the reachability problem of a NeTA is decidable in several special subclasses, while undecidable generally. The former is proven by encoding a NeTA to mathematical models, such as TPDAs, DTPDAs, and DTPDA-Fs; while the latter is proven by encoding a Minsky machine to a general NeTA.

Acknowledgements

This work was supported by the NSFC-JSPS Bilateral Joint Research Project (Grant No. 61511140100), and the National Natural Science Foundation of China (Grant Nos. 61100052 and 61472240).

References

[1] Alur R, Dill D L. A theory of timed automata. Theoretical Computer Science, 1994, 126(2):183—235.

[2] Bouajjani A, Echahed R, Robbana R. On the automatic verification of systems with continuous variables and unbounded discrete data structures. Hybrid Systems II. Springer Berlin Heidelberg, 1994: 64—85.

[3] Abdulla P A, Atig M F, Stenman J. Dense-Timed Pushdown Automata. IEEE Symposium on Logic in Computer Science, 2012:35—44.

[4] Li G, Cai X, Ogawa M, et al. Nested Timed Automata. Proceedings of the 11th International Conference on Formal Modeling and Analysis of Timed Systems. Springer-Verlag, 2013, 8053: 168—182.

[5] Li G, Ogawa M, Yuen S. Nested Timed Automata with Frozen Clocks. Formal Modeling and Analysis of Timed Systems. Springer International Publishing, 2015, 9268:189—205.

[6] Wen Y, Li G, Yuen S. An Over-Approximation Forward Analysis for Nested Timed Automata. Structured Object-Oriented Formal Lan-

guage and Method. Springer International Publishing, 2014, 8979:62—80.
[7] Bérard B, Haddad S, Sassolas M. Real time properties for interrupt timed automata. Temporal Representation and Reasoning (TIME), 2010 17th International Symposium on. IEEE, 2010: 69—76.
[8] Trivedi A, Wojtczak D. Recursive timed automata. Automated Technology for Verification and Analysis. Springer Berlin Heidelberg, 2010, 6252: 306—324.
[9] Benerecetti M, Minopoli S, Peron A. Analysis of timed recursive state machines. Temporal Representation and Reasoning (TIME), 2010 17th International Symposium on. IEEE, 2010: 61—68.
[10] M. L Minsky. Computation: Finite and Infinite Machines. Prentice-Hall, 1967
[11] Henzinger T A, Nicollin X, Sifakis J, et al. Symbolic model checking for real-time systems. Information and computation, 1994, 111(2): 193—244.
[12] Clemente L, Lasota S. Timed pushdown automata revisited. Logic in Computer Science (LICS), 2015 30th Annual ACM/IEEE Symposium on. IEEE, 2015: 738—749.
[13] Cai X, Ogawa M. Well-structured pushdown system: case of dense timed pushdown automata. Functional and Logic Programming. Springer International Publishing, 2014, 8475: 336—352.
[14] Ouaknine J, Worrell J. On the language inclusion problem for timed automata: Closing a decidability gap. Logic in Computer Science, 2004. Proceedings of the 19th Annual IEEE Symposium on. IEEE, 2004: 54—63.
[15] Abdulla P A, Jonsson B. Verifying networks of timed processes. Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, 1998, 1384: 298—312.
[16] Abdulla P A, Jonsson B. Model checking of systems with many identical timed processes. Theoretical Computer Science, 2003, 290(1): 241—264.
[17] Henzinger T A, Kopke P W, Puri A, et al. What's Decidable about Hybrid Automata ?Journal of Computer & System Sciences, 2010, 57(1):94—124.
[18] Dang Z. Pushdown timed automata: a binary reachability characterization and safety verification. Theoretical Computer Science, 2003, 302(1):93—121.
[19] Abdulla P A, Atig M F, Stenman J. The minimal cost reachability problem in priced timed pushdown systems. Language and Automata Theory and Applications. Springer Berlin Heidelberg, 2012: 58—69.
[20] Alur R, Benedikt M, Etessami K, et al. Analysis of recursive state machines. ACM Transactions on Programming Languages and Systems (TOPLAS), 2005, 27(4): 786—818.
[21] David A, Oller M O M. From HUPPAAL to UPPAAL — A Translation from Hierarchical Timed Automata to Flat Timed Automata. Department of Computerence University of Aarhus, 2001:1—11.
[22] Li G, Cai X, Yuen S. Modeling and analysis of real-time systems with mutex components. International Journal of Foundations of Computer Science, 2012, 23(4): 831—851.
[23] Li G, Yuen S, Adachi M. Environmental simulation of real-time systems with nested interrupts. Theoretical Aspects of Software Engineering, 2009. TASE 2009. Third IEEE International Symposium on. IEEE, 2009: 21—28.
[24] Bouyer P, Dufourd C, Fleury E, et al. Are timed automata updatable ?Computer Aided Verification. Springer Berlin Heidelberg, 2000: 464—479.
[25] Bouyer P, Dufourd C, Fleury E, et al. Expressiveness of updatable timed automata. Mathematical Foundations of Computer Science 2000. Springer Berlin Heidelberg, 2000: 232—242. [26] Fleury E, Petit A, Bouyer P, et al. Updatable timed automata. Theoretical Computer Science, 2000, 321(23):291—345.
[27] Wen Y, Li G, Yuen S. On Reachability Analysis of Updatable Timed Automata with One Updatable Clock. Structured Object-Oriented Formal Language and Method. Springer International Publishing, 2015: 147—161.
[28] Fang B, Li G, Fang L, et al. A Refined Algorithm for Reachability Analysis of Updatable Timed Automata. Software Quality, Reliability and Security-Companion (QRS-C), 2015 IEEE.

Appendix A. A Proof of the Lemma 5

Lemma 5. For a (gc+lc+fc)NeTA N with one global clock, its encoded DTPDA $\mathcal{E}(N)$, and configurations κ, κ' of N

- (**Preservation**) if $\kappa \longrightarrow \kappa'$, then $[\![\kappa]\!] \longrightarrow_{\mathcal{D}}^{*} [\![\kappa']\!]$, and
- (**Reflection**) if $[\![\kappa]\!] \longrightarrow_{\mathcal{D}}^{*} \varrho$, there exists κ' with $\varrho \longrightarrow_{\mathcal{D}}^{*} [\![\kappa']\!]$ and $\kappa \longrightarrow^{*} \kappa'$.

Proof: Let $\kappa = (\langle q, \nu, \mu\rangle, v)$ such that $v = (q_1, flag_1, \nu_1)\cdots(q_n, flag_n, \nu_n)$, and $[\![\kappa]\!] = (q, \bar{w}(v), \nu \cup \mu)$ with $\bar{w}(v) = (q_1, \nu_1, flag_1)\cdots(q_n, \nu_n, flag_n)$.

The *preservation* part is proved by case analysis of $\kappa \longrightarrow \kappa'$. We omit some similar cases.

- **Progress transition**: $\kappa \xrightarrow{t} \kappa' = (\langle q, \nu + t, \mu + t\rangle, v + t)$. We have $[\![\kappa]\!] = (q, \bar{w}(v), \nu \cup \mu) \xrightarrow{t}_{\mathcal{D}} (q, \bar{w}(v) + t, (\nu \cup \mu) + t) = (q, \bar{w}(v + t), (\nu + t) \cup (\mu + t)) = [\![\kappa']\!]$.
- **Assign**: $\kappa \xrightarrow{x \leftarrow I} \kappa' = (\langle q', \nu[x \leftarrow I], \mu\rangle, v)$. Then $[\![\kappa]\!] \xrightarrow{x \leftarrow I}_{\mathcal{D}} (q', \bar{w}(v), (\nu \cup \mu)[x \leftarrow I]) = (q, \bar{w}(v), \nu[x \leftarrow I] \cup \mu) = [\![\kappa']\!]$.
- **Global-load**: $\kappa \xrightarrow{x \leftarrow c} \kappa' = (\langle q', \nu[x \leftarrow \mu[c]], \mu\rangle, v)$. Then $[\![\kappa]\!] \xrightarrow{x \leftarrow c}_{\mathcal{D}} (q', \bar{w}(v), (\nu \cup \mu)[x \leftarrow (\nu \cup \mu)(c)]) = (q, \bar{w}(v), \nu[x \leftarrow \mu(c)] \cup \mu) = [\![\kappa']\!]$.
- **Push**: $\kappa \xrightarrow{push} \kappa' = (\langle q_0(A_{i'}), \nu_0, \mu\rangle, \langle q, 1, \nu\rangle . v)$. Then $[\![\kappa]\!] \xrightarrow{push(q)}_{\mathcal{D}} (q_0(A_{i'}), (q, \nu, 1) . \bar{w}(v), (\nu \cup \mu)_0) = (q_0(A_{i'}), \bar{w}((q, \nu, 1) . v), \nu_0 \cup \mu) = [\![\kappa']\!]$.
- **Pop**: $\kappa = (\langle q, \nu, \mu\rangle, \langle q', flag, \nu'\rangle . v') \xrightarrow{pop} \kappa' = (\langle q', \nu', \mu\rangle, v')$. Then $[\![\kappa]\!] = (q, (q', \nu', flag) . \bar{w}(v'), \nu \cup \mu) \xrightarrow{pop(q')}_{\mathcal{D}} (q', \bar{w}(v'), (\nu \cup \mu)[\bar{x} \leftarrow \nu']) = (q', \bar{w}(v'), \nu' \cup \mu) = [\![\kappa']\!]$. .

For the *reflection* part, we prove a stronger one: if $[\![\kappa]\!] \longrightarrow_{\mathcal{D}} \varrho$, there exists κ' with $\varrho = [\![\kappa']\!]$. and $\kappa \longrightarrow \kappa'$. The *reflection* part is obviously true by induction if this one holds. We analyze $[\![\kappa]\!] \longrightarrow_{\mathcal{D}} \varrho$ case by case and omit some similar cases.

- **Progress transition**: $[\![\kappa]\!] \xrightarrow{t}_{\mathcal{D}} \varrho = (q, \bar{w}(v) + t, (\nu \cup \mu) + t) = (q, \bar{w}(v + t), (\nu + t) \cup (\mu + t))$. Then there exists $\kappa' = (\langle q, \nu + t, \mu + t\rangle, v + t)$ with $\varrho = [\![\kappa']\!]$ and $\kappa \xrightarrow{t} \kappa'$.
- **Assign**: $[\![\kappa]\!] \xrightarrow{x \leftarrow I}_{\mathcal{D}} \varrho = (q', \bar{w}(v), (\nu \cup \mu)[x \leftarrow I])$. Suppose $x \in X$, then there must be a corresponding assign transition in the original NeTA $(q, \varepsilon, x \leftarrow I, q', \varepsilon)$ with $x \in X$. We have $\kappa' = (\langle q', \nu[x \leftarrow I], \mu\rangle, v)$ with $\varrho = (q', \bar{w}(v), \nu[x \leftarrow I] \cup \mu) = [\![\kappa']\!]$ and $\kappa \xrightarrow{x \leftarrow I} \kappa'$. The other case $x \in C$ is similar with the global-assign transition in the original NeTA.
- **Push**: $[\![\kappa]\!] \xrightarrow{push(q)}_{\mathcal{D}} \varrho = (q', (q, \nu, 1) . \bar{w}(v), (\nu \cup \mu)_0) = (q', (q, \nu, 1) . \bar{w}(v), \nu_0 \cup \mu)$. Then in the NeTA, a push rule $(q, \varepsilon, push, q_0(A_{i'}), q)$ exists with $q' = q_0(A_{i'})$. Hence there exists $\kappa' = (\langle q_0(A_{i'}), \nu_0, \mu\rangle, \langle q, 1, \nu\rangle . v)$ with $\varrho = [\![\kappa']\!]$ and $\kappa \xrightarrow{push} \kappa'$.
- **Pop**: $[\![\kappa]\!] = (q, (q', \nu', flag) . \bar{w}(v'), \nu \cup \mu) \xrightarrow{pop(q')}_{\mathcal{D}} \varrho = (q', \bar{w}(v'), (\nu \cup \mu)[\bar{x} \leftarrow \nu']) = (q', \bar{w}(v'), \nu' \cup \mu)$. Then in the NeTA, a pop rule $(q, q', pop, q', \varepsilon)$ exists. Hence there exists $\kappa' = (\langle q', \nu', \mu\rangle, v')$ with $\varrho = [\![\kappa']\!]$ and $\kappa = (\langle q, \nu, \mu\rangle, \langle q', flag, \nu'\rangle . v') \xrightarrow{pop} \kappa'$.

Li Guoqiang

Dr. Li received his B. S., M. S., and Ph. D. degrees from Taiyuan University of Technology, Shanghai Jiao Tong University, and Japan Advanced Institute of Science and Technology in 2001, 2005, and 2008, respectively. He worked as a postdoctoral research fellow in the Graduate School of Information Science, Nagoya University, Japan, during 2008—2009, as an assistant professor in the School of Software, Shanghai Jiao Tong University, during 2009—2013, and as an academic visitor at the Department of Computer Science, University of Oxford during 2015—2016. He is now an associate professor in the School of Software, Shanghai Jiao Tong University. His research interests include formal verification, programming language theory and computational learning theory. He has published more than 30 research papers in the international journals and mainstream conferences, including *IJFCS*, *Science China Information Sciences*, *FORMATS*, *ATVA*, etc..