Nested Timed Automata with Invariants

Yuwei Wang[1], Guoqiang Li[1(✉)], and Shoji Yuen[2]

[1] School of Software, Shanghai Jiao Tong University, Shanghai, China
{wangywgg,li.g}@sjtu.edu.cn
[2] Graduate School of Information Science, Nagoya University, Nagoya, Japan
yuen@is.nagoya-u.ac.jp

Abstract. Invariants are usually adopted into timed systems to constrain the time passage within each control location. It is well-known that a timed automaton with invariants can be encoded to an equivalent one without invariants. When recursions are taken into consideration, few results show whether invariants affect expressiveness. This paper investigates the effect of invariants to Nested Timed Automata (NeTAs), a typical real-timed recursive system. In particular, we study the reachability problem for NeTA-Is, which extend NeTAs with invariants. It is shown that the reachability problem is undecidable on NeTA-Is with a single global clock, while it is decidable when no invariants are given. Furthermore, we also show that the reachability is decidable if the NeTA-Is contains no global clocks by showing that a *good* stack content still satisfies well-formed constraints.

1 Introduction

From the past century, many research studies have been carried out on modeling and verification of real time systems. The pioneer work can be traced to *Timed Automata (TAs)* [1,2], which is one of the most successful models among them due to its simplicity, effectiveness and fruitful results. A TA is a finite automaton with a finite set of *clocks* that grow uniformly. Besides the constraints assigned on the transitions of TAs, they can also be assigned to each control location, named *invariants*, to constrain time passages of models. Invariants usually play a crucial role in the application modelling and verification [3], since in reality a system is not allowed to stay in one location for arbitrarily long time. It is well-known that TAs with and without invariants have the same expressive power [3]. However, little research has been conducted in investigating the impact of invariants on the reachability problem of timed systems with recursions.

This paper proposes an extension of *Nested Timed Automata* (NeTAs) [4,5], called NeTA-Is. A NeTA is a pushdown system whose stack contains TAs with global clocks passing information among different contexts. TAs in the stack can either be proceeding, in which clocks proceed as time elapses, or frozen, where clocks remain unchanged. NeTA-Is naturally extend NeTAs with invariants at each control location that must be fulfilled in all valid runs. Studies in [5] have shown that in NeTAs, (i) the reachability with a single global clock is decidable, and (ii) the reachability with multiple global clocks is undecidable. While in this paper, we show that (i) the reachability problem of a NeTA-I is undecidable even

K.G. Larsen et al. (Eds.): SETTA 2017, LNCS 10606, pp. 77–93, 2017.
https://doi.org/10.1007/978-3-319-69483-2_5

with a single global clock by encoding Minsky machines to NeTA-Is, and (ii) it is decidable when the NeTA-I has no global clocks by showing that a *good* stack content still satisfies well-formed constraints [6].

Related Work. *Timed Automata* (TAs) [1,2] are the first model for real-timed systems. TAs are essentially finite automata extended with real-valued variables, called *clocks*. The reachability of TAs is shown to be decidable based on construction of regions and zones. It is also shown that invariants do not affect the decidability and thus only a syntactic sugar. Based on timed automata, lots of extensions are proposed and investigated especially for a recursive structure.

Dense Timed Pushdown Automata (DTPDAs) [7] combine timed automata and pushdown automata, where each stack frame containing not only a stack symbol but also a real-valued clock behaves as a basic unit of push/pop operations. The reachability of a DTPDA is shown to be decidable by encoding it to a PDA using the region technique. Another decidability proof is given in [6] through a general framework, *well-structured pushdown systems*. We adopt this framework in this paper to prove the decidability of reachability of *Constraint DTPDAs*, which extend DTPDAs with clock constraints on each location.

Recursive Timed Automata (RTAs) [8] contain finite components, each of which is a special timed automaton and can recursively invoke other components. Two mechanisms, *pass-by-value* and *pass-by-reference*, can be used to passing clocks among different components. A clock is *global* if it is always passed by reference, whereas it is *local* if it is always passed by value. Although the reachability problem of RTAs is undecidable, it is decidable if all clocks are global or all clocks are local.

Similarly, the reachability problems of both *Timed Recursive State Machines* (TRSMs), which combine *recursive state machines* (RSMs) and TAs, and *Extended Pushdown Timed Automata* (EPTAs), which augment *Pushdown Timed Automata* (PTAs) with an additional stack, are undecidable, while they are decidable in some restricted subclasses [9].

To the best of our knowledge, all these prior formal models focusing on timed systems with recursive structures lacks discussions of the impact of invariants, including DTPDAs, RTAs, TRSMs, EPTAs and NeTAs.

Paper Organization. The remainder of this paper is structured as follows: In Sect. 2 we introduce basic terminologies and notations. Section 3 defines syntax and the semantics of NeTA-Is. Section 4 shows that the reachability problem of NeTA-Is is Turing-complete. Section 5 introduces a model *Constraint DTPDAs* and shows its decidability. Section 6 is devoted to proofs of decidability results of NeTA-Is without global clocks by encoding it to a Constraint DTPDA. Section 7 concludes this paper with summarized results.

2 Preliminaries

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.

Let $\mathbb{R}^{\geq 0}$ and $\mathbb{N}$ denote the sets of non-negative real numbers and natural numbers, respectively. Let ω denote the first limit ordinal. Let $\mathcal{I}$ denote the set of *intervals*. An interval is a set of numbers, written as (a, b'), $[a, b]$, $[a, b')$ or $(a, b]$, where $a, b \in \mathbb{N}$ and $b' \in \mathbb{N} \cup \{\omega\}$. For a number $r \in \mathbb{R}^{\geq 0}$ and an interval $I \in \mathcal{I}$, we use $r \in I$ to denote that r belongs to I.

Let $X = \{x_1, \ldots, x_n\}$ be a finite set of *clocks*. The set of *clock constraints*, $\Phi(X)$, over X is defined by $\phi ::= \top \mid x \in I \mid \phi \wedge \phi$ where $x \in X$ and $I \in \mathcal{I}$. An operation of *extracting constraint* $EC(\phi, x)$ is defined by induction over its argument ϕ.

$$
\begin{array}{lcl}
EC(\top, x) & = & [0, \omega) \\
EC(x \in I, x) & = & I \\
EC(y \in I, x) & = & [0, \omega)\ if\ x \neq y \\
EC(\phi_1 \wedge \phi_2, x) & = & EC(\phi_1, x) \bigcap EC(\phi_2, x)
\end{array}
$$

A *clock valuation* $\nu : X \to \mathbb{R}^{\geq 0}$, assigns a value to each clock $x \in X$. $\nu_{\mathbf{0}}$ denotes the clock valuation assigning each clock in X to 0. For a clock valuation ν and a clock constraint ϕ, we write $\nu \models \phi$ to denote that ν satisfies the constraint ϕ. 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 $\nu[y_1 \leftarrow b_1, \cdots, y_n \leftarrow b_n]$ 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.

2.1 Timed Automata

A timed automaton is a finite automaton augmented with a finite set of clocks [1,2].

Definition 1 (Timed Automata). *A timed automaton (TA) is a tuple* $\mathcal{A} = (Q, q_0, X, \mathbb{I}, \Delta) \in \mathscr{A}$, *where*

- Q *is a finite set of control locations, with the initial location* $q_0 \in Q$,
- X *is a finite set of clocks,*
- $\mathbb{I} : Q \to \Phi(X)$ *is a function assigning each location with a clock constraint on* X, *called* invariants.
- $\Delta \subseteq Q \times \mathcal{O} \times Q$, *where* $\mathcal{O}$ *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,*
  Reset $x \leftarrow 0$ *where* $x \in X$, *and*
  Value passing $x \leftarrow x'$ *where* $x, x' \in X$.

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

We call the four operations **Local**, **Test**, **Reset**, and **Value passing** as *internal* actions which will be used in Definition 3.

Definition 2 (Semantics of TAs). *Given a TA* $(Q, q_0, X, \mathbb{I}, \Delta)$, *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* $t \in \mathbb{R}^{\geq 0}$*,* $\nu \models \mathbb{I}(q)$ *and* $(\nu + t) \models \mathbb{I}(q)$*.*
– Discrete transition*:* $(q_1, \nu_1) \xrightarrow{\phi}_{\mathscr{A}} (q_2, \nu_2)$*, if* $q_1 \xrightarrow{\phi} q_2 \in \Delta$*,* $\nu_1 \models \mathbb{I}(q_1)$*,* $\nu_2 \models \mathbb{I}(q_2)$ *and one of the following holds,*
  - **Local** $\phi = \epsilon$*, then* $\nu_1 = \nu_2$*.*
  - **Test** $\phi = x \in I?$*,* $\nu_1 = \nu_2$ *and* $\nu_2(x) \in I$ *holds. The transition can be performed only if the value of* x *belongs to* I*.*
  - **Reset** $\phi = x \leftarrow 0$*,* $\nu_2 = \nu_1[x \leftarrow 0]$*. This operation resets clock* x *to 0.*
  - **Value passing** $\phi = x \leftarrow x'$*, then* $\nu_2 = \nu_1[x \leftarrow \nu_1(x')]$*. The transition passes value of clock* x' *to clock* x*.*

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

Remark 1. The TA definition in Definition 1 follows the style in [4] and is slightly different from the original definition in [1]. In [1], several test and reset operations could be performed in a single discrete transition. It can be shown that our definition of TA can soundly simulate the time traces in the original definition.

3 Nested Timed Automata with Invariants

A *nested timed automaton with invariants (NeTA-I)* extended from NeTAs[1] [5] is a pushdown system whose stack alphabet is timed automata. It can either behave like a TA (internal operations), push or fpush the current working TA to the stack, pop a TA from the stack or reference global clocks. Global clocks can be used to constrain the global behavior or passing value of local clocks among different TAs. The invariants can be classified into *global invariants*, which are constraints on global clocks, and *local invariants*, which are constraints on local clocks. In the executions of a NeTA-I, all invariants must be satisfied at all reachable configurations, including global invariants and local invariants. Note that because the stack contains only information belonging to TAs and does not contain the global clock valuation, there is no need to check global invariants in the stack.

Definition 3 (Nested Timed Automata with Invariants). *A nested timed automaton with invariants (NeTA-I) is a tuple* $\mathcal{N} = (T, \mathcal{A}_0, X, C, \mathbb{I}, \Delta)$*, where*

– T *is a finite set of TAs* $\{\mathcal{A}_0, \mathcal{A}_1, \cdots, \mathcal{A}_n\}$*, with the initial TA* $\mathcal{A}_0 \in T$*. We assume the sets of control locations of* $\mathcal{A}_i$*, denoted by* $Q(\mathcal{A}_i)$*, are mutually disjoint, i.e.,* $Q(\mathcal{A}_i) \cap Q(\mathcal{A}_j) = \emptyset$ *for* $i \neq j$*. For simplicity, we assume that each* $\mathcal{A}_i$ *in* T *shares the same set of local clocks* X*.*
– C *is a finite set of global clocks, and* X *is the finite set of* k *local clocks.*
– $\mathbb{I} : Q \to \Phi(C)$ *is a function that assigns to each control location an invariant on global clocks. For clarity,* $\mathbb{I}(q)$ *denotes the global invariant in* q*, and* $\mathbb{I}(\mathcal{A}_i)(q)$ *denotes the local invariant in* q *where* $q \in \mathcal{A}_i$*.*
– $\Delta \subseteq Q \times (Q \cup \{\varepsilon\}) \times Actions^+ \times Q \times (Q \cup \{\varepsilon\})$ *describes transition rules below, where* $Q = \cup_{\mathcal{A}_i \in T} Q(\mathcal{A}_i)$*.*

[1] The NeTAs here are called "NeTA-Fs" in [5].

A transition rule is described by a sequence of Actions $= \{internal, push, fpush, pop, c \in I, c \leftarrow 0, x \leftarrow c, c \leftarrow x\}$ *where* $c \in C$ *and* $x \in X$.

Internal $(q, \varepsilon, internal, q', \varepsilon)$, *which describes an internal transition in the working TA with* $q, q' \in Q(\mathcal{A}_i)$.

Push $(q, \varepsilon, push, q_0(\mathcal{A}_{i'}), q)$, *which interrupts the currently working TA* $\mathcal{A}_i$ *at* $q \in Q(\mathcal{A}_i)$ *and pushes it to the stack with all local clocks of* $\mathcal{A}_i$. *The local clocks in the stack generated by* **Push** *operation are proceeding, i.e., still evolve as time elapses. Then, a TA* $\mathcal{A}_{i'}$ *newly starts.*

Freeze-Push (F-Push) $(q, \varepsilon, fpush, q_0(\mathcal{A}_{i'}), q)$, *which is similar to* **Push** *except that all local clocks in the stack generated by* **F-Push** *are frozen (i.e. stay the same as time elapses).*

Pop $(q, q', pop, q', \varepsilon)$, *which restarts* $\mathcal{A}_{i'}$ *in the stack from* $q' \in Q(\mathcal{A}_{i'})$ *after* $\mathcal{A}_i$ *has finished at* $q \in Q(\mathcal{A}_i)$ *and all local clocks restart with values in the top stack frame.*

Global-test $(q, \varepsilon, c \in I?, q', \varepsilon)$, *which tests whether the value of a global clock* c *is in* I *with* $q, q' \in Q(\mathcal{A}_i)$.

Global-reset $(q, \varepsilon, c \leftarrow 0, q', \varepsilon)$ *with* $c \in C$, *which resets the global clock* c *to 0 with* $q, q' \in Q(\mathcal{A}_i)$.

Global-load $(q, \varepsilon, x \leftarrow c, q', \varepsilon)$, *which assigns the value of a global clock* c *to a local clock* $x \in X$ *in the working TA with* $q, q' \in Q(\mathcal{A}_i)$.

Global-store $(q, \varepsilon, c \leftarrow x, q', \varepsilon)$, *which assigns the value of a local clock* $x \in X$ *of the working TA to a global clock* c *with* $q, q' \in Q(\mathcal{A}_i)$.

Definition 4 (Semantics of NeTA-Is). *Given a NeTA-I* $(T, \mathcal{A}_0, X, C, \mathbb{I}, \Delta)$, *let* $\mathcal{V}al_X = \{\nu : X \to \mathbb{R}^{\geq 0}\}$ *and* $\mathcal{V}al_C = \{\mu : C \to \mathbb{R}^{\geq 0}\}$. *A configuration of a NeTA-I is an element* $(\langle q, \nu, \mu\rangle, v)$ *with a control location* $q \in Q$, *a local clock valuation* $\nu \in \mathcal{V}al_X$, *a global clock valuation* $\mu \in \mathcal{V}al_C$ *and a stack* $v \in (Q \times \{0,1\} \times \mathcal{V}al_X)^*$. *We say a stack* v *is* good, *written as* $v^{\Uparrow}$, *if all local invariants are satisfied in* v, *i.e., for each content* $\langle q_i, flag_i, \nu_i\rangle$ *in* v *with* $q_i \in Q(\mathcal{A}_j)$, $\nu_i \models \mathbb{I}(\mathcal{A}_j)(q_i)$ *holds. We also denote* $v + t$ *by setting* $\nu_i := progress(\nu_i, t, flag_i)$ *of each* $\langle q_i, flag_i, \nu_i\rangle$ *in the stack where* $progress(\nu, t, flag) = \begin{cases} \nu + t & \text{if } flag = 1 \\ \nu & \text{if } flag = 0 \end{cases}$

– Progress transition: $(\langle q, \nu, \mu\rangle, v) \xrightarrow{t} (\langle q, \nu + t, \mu + t\rangle, v + t)$ *for* $t \in \mathbb{R}^{\geq 0}$, *where* $q \in Q(\mathcal{A}_i)$, $\nu \models \mathbb{I}(\mathcal{A}_i)(q)$, $\mu \models \mathbb{I}(q)$, $(\nu + t) \models \mathbb{I}(\mathcal{A}_i)(q)$, $(\mu + t) \models \mathbb{I}(q)$, $v^{\Uparrow}$ *and* $(v + t)^{\Uparrow}$.
– Discrete transition: $(\langle q, \nu, \mu\rangle, v) \xrightarrow{\varphi} (\langle q', \nu', \mu'\rangle, v')$, *where* $q \in Q(\mathcal{A}_i)$, $q' \in Q(\mathcal{A}'_i)$, $\nu \models \mathbb{I}(\mathcal{A}_i)(q)$, $\mu \models \mathbb{I}(q)$, $\nu' \models \mathbb{I}(\mathcal{A}'_i)(q')$, $\mu' \models \mathbb{I}(q')$, $v^{\Uparrow}$, $v'^{\Uparrow}$, *and one of the following holds.*
  - **Internal** $(\langle q, \nu, \mu\rangle, v) \xrightarrow{\varphi} (\langle q', \nu', \mu\rangle, v)$, *if* $(q, \varepsilon, internal, q', \varepsilon) \in \Delta$ *and* $\langle q, \nu\rangle \xrightarrow{\varphi} \langle q', \nu'\rangle$ *is in Definition 2.*
  - **Push** $(\langle q, \nu, \mu\rangle, v) \xrightarrow{push} (\langle q_0(\mathcal{A}_{i'}), \nu_{\mathbf{0}}, \mu\rangle, \langle q, 1, \nu\rangle.v)$, *if* $(q, \varepsilon, push, q_0(\mathcal{A}_{i'}), q) \in \Delta$.
  - **F-Push** $(\langle q, \nu, \mu\rangle, v) \xrightarrow{f\text{-}push} (\langle q_0(\mathcal{A}_{i'}), \nu_{\mathbf{0}}, \mu\rangle, \langle q, 0, \nu\rangle.v)$, *if* $(q, \varepsilon, fpush, q_0(\mathcal{A}_{i'}), q) \in \Delta$.
  - **Pop** $(\langle q, \nu, \mu\rangle, \langle q', flag, \nu'\rangle.w) \xrightarrow{pop} (\langle q', \nu', \mu\rangle, w)$, *if* $(q, q', pop, q', \varepsilon) \in \Delta$.
  - **Global-test** $(\langle q, \nu, \mu\rangle, v) \xrightarrow{c \in I?} (\langle q', \nu, \mu\rangle, v)$, *if* $(q, \varepsilon, c \in I?, q', \varepsilon) \in \Delta$ *and* $\mu(c) \in I$.

- **Global-reset** $(\langle q, \nu, \mu \rangle, v) \xrightarrow{c \leftarrow 0} (\langle q', \nu, \mu[c \leftarrow 0] \rangle, v)$, *if* $(q, \varepsilon, c \leftarrow 0, q', \varepsilon) \in \Delta$.
- **Global-load** $(\langle q, \nu, \mu \rangle, v) \xrightarrow{x \leftarrow c} (\langle q', \nu[x \leftarrow \mu(c)], \mu \rangle, v)$, *if* $(q, \varepsilon, x \leftarrow c, q', \varepsilon) \in \Delta$.
- **Global-store**$(\langle q, \nu, \mu \rangle, v) \xrightarrow{c \leftarrow x} (\langle q', \nu, \mu[c \leftarrow \nu(x)] \rangle, v)$, *if* $(q, \varepsilon, c \leftarrow x, q', \varepsilon) \in \Delta$.

The initial configuration of a NeTA-I is $(\langle q_0(\mathcal{A}_0), \nu_\mathbf{0}, \mu_\mathbf{0} \rangle, \varepsilon)$, *where* $\nu_\mathbf{0}(x) = 0$ *for* $x \in X$ *and* $\mu_\mathbf{0}(c) = 0$ *for* $c \in C$. *We use* $\longrightarrow$ *to range over these transitions, and* $\longrightarrow^*$ *is the reflexive and transitive closure of* $\longrightarrow$.

Intuitively, in a stack $v = (q_1, flag_1, \nu_1) \ldots (q_n, flag_n, \nu_n)$, q_i is the control location of the pushed/fpushed TA, $flag_i \in \{0, 1\}$ is a flag for whether the TA is pushed ($flag_i = 1$) or fpushed ($flag_i = 0$) and ν_i is a clock valuation for the local clocks of the pushed/fpushed TA.

4 Undecidability Results of NeTA-Is

In this section, we prove undecidability of NeTA-Is by encoding the halting problem of Minsky machines [10] to NeTA-Is with a single global clock.

Definition 5 (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** $d = inc(l, ct_i, l_k)$*: start from* l, $ct_i := ct_i + 1$, *goto* l_k,
  - **test-and-decrement counter** $d = dec(l, ct_i, l_k, l_m)$*: start from* l, *if* $(ct_i > 0)$ *then* $(ct_i := ct_i - 1$, *goto* $l_k)$ *else goto* l_m,

  where $ct_i \in C$, $d \in D$ *and* $l, l_k, l_m \in L$.

In this encoding, we use three TAs, $\mathcal{A}_0, \mathcal{A}_1$ and $\mathcal{A}_2$. Each TA has three local clocks x_0, x_1 and x_2. $\mathcal{A}_0$ is a special TA, as two local clocks of $\mathcal{A}_0$, x_1 and x_2 encode values of two counters as $x_i = 2^{-ct_i}$ for $i = 1, 2$. Decrementing and incrementing the counter ct_i are simulated by doubling and halving of the value of the local clock x_i in $\mathcal{A}_0$, respectively. In all TAs, x_0 is used to prevent time progress. In $\mathcal{A}_1$ and $\mathcal{A}_2$, x_1 and x_2 are used for temporarily storing value. We use only one global clock c to pass value among different TAs.

There are two types of locations in the encoding, q-locations and e-locations. All q-locations are assigned with invariants $x_0 \in [0, 0]$. These invariants ensures that in all reachable configurations at q-locations, the value of x_0 must be 0. So time does not elapse at q-locations.

The idea of doubling or halving of x_i in $\mathcal{A}_0$ is as follows. First the value of x_i is stored to the global clock c. Then the current TA $\mathcal{A}_0$ is fpushed to the stack and through transitions in $\mathcal{A}_1$ and $\mathcal{A}_2$, the global clock c is doubled or halved. Later $\mathcal{A}_0$ is popped back and the value of c is loaded to x_i. Since all locations are q-locations in $\mathcal{A}_0$, time does not elapse in $\mathcal{A}_0$. This ensures that while doubling or halving a local clock, the other one is left unchanged.

The encoding is shown formally as follows.

A Minsky machine $\mathcal{M} = (L, C, D)$ can be encoded into a NeTA-I $\mathcal{N} = (T, \mathcal{A}_0, X, C', \mathbb{I}, \Delta)$, with $T = \{\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2\}$ where

– $Q(\mathcal{A}_0) = \{q_l \mid l \in L\} \bigcup \{q_1^{inc,i,l_k} \mid inc(ct_i, l, l_k) \in D\}$
  $\bigcup \{q_j^{dec,i,l_k} \mid dec(ct_i, l, l_k, l_m) \in D, 1 \le j \le 2\}$
  $Q(\mathcal{A}_1) = \{q_j^{inc,i,l_k} \mid inc(ct_i, l, l_k) \in D, 2 \le j \le 8\}$
  $\bigcup \{e_j^{inc,i,l_k} \mid inc(ct_i, l, l_k) \in D, j = 1, 2 \text{ or } 4\}$
  $\bigcup \{q_j^{dec,i,l_k} \mid dec(ct_i, l, l_k, l_m) \in D, 3 \le j \le 7\}$
  $\bigcup \{e_2^{dec,i,l_k} \mid dec(ct_i, l, l_k, l_m) \in D\}$
  $Q(\mathcal{A}_2) = \{q_j^{inc,i,l_k} \mid inc(ct_i, l, l_k) \in D, 9 \le j \le 11\}$
  $\bigcup \{e_3^{inc,i,l_k} \mid inc(ct_i, l, l_k) \in D\}$
  $\bigcup \{q_j^{dec,i,l_k} \mid dec(ct_i, l, l_k, l_m) \in D, 8 \le j \le 10\}$
  $\bigcup \{e_1^{dec,i,l_k} \mid dec(ct_i, l, l_k, l_m) \in D\}$
– $X = \{x_0, x_1, x_2\}$ and $C' = \{c\}$.
– $\mathbb{I}(A_i)(q_\_) = x_0 \in [0, 0]$ and $\mathbb{I}(A_i)(e_\_) = \top$ where $0 \le i \le 2$ and $\_$ denotes any valid symbol. Here $q_\_$ denotes the q-location, which is labeled with q, and $e_\_$ denotes the e-location, which is labeled with e.
– Δ is shown implicitly in the following simulations due to limited space.
  - **increment counter** simulate $inc(l, ct_i, l_k)$. Initially $\nu(x_i) = d$ with $0 < d \le 1$. In q_{l_k}, x_i will be halved. The value of x_i is stored to the global clock c and context is changed to $\mathcal{A}_1$. Then the value of c is halved. Although the timed elapsed in state e_2^{inc,i,l_k} and e_3^{inc,i,l_k} are nondeterministic, to reach the location q_{l_k}, the value of x_1 and c must coincide (i.e., they reach 1 together) at state e_4^{inc,i,l_k}. The readers can check that timed elapsed in e_1^{inc,i,l_k} must be $1 - d$, in e_2^{inc,i,l_k} and e_4^{inc,i,l_k} must be $d/2$, and in e_3^{inc,i,l_k} must be $1 - d/2$
    $q_l \xrightarrow{c \leftarrow x_i} q_1^{inc,i,l_k} \xrightarrow{fpush} e_1^{inc,i,l_k} \xrightarrow{x_0 \leftarrow 0} q_2^{inc,i,l_k} \xrightarrow{c \in [1,1]?} q_3^{inc,i,l_k} \xrightarrow{c \leftarrow 0} e_2^{inc,i,l_k} \xrightarrow{x_0 \leftarrow 0} q_4^{inc,i,l_k} \xrightarrow{fpush} e_3^{inc,i,l_k} \xrightarrow{x_0 \leftarrow 0} q_9^{inc,i,l_k} \xrightarrow{c \in [1,1]?} q_{10}^{inc,i,l_k} \xrightarrow{c \leftarrow x_1} q_{11}^{inc,i,l_k} \xrightarrow{pop} q_4^{inc,i,l_k} \xrightarrow{x_2 \leftarrow 0} e_4^{inc,i,l_k} \xrightarrow{x_0 \leftarrow 0} q_5^{inc,i,l_k} \xrightarrow{c \in [1,1]?} q_6^{inc,i,l_k} \xrightarrow{x_1 \in [1,1]?} q_7^{inc,i,l_k} \xrightarrow{c \leftarrow x_2} q_8^{inc,i,l_k} \xrightarrow{pop} q_1^{inc,i,l_k} \xrightarrow{x_i \leftarrow c} q_{l_k}$
  - **test-and-decrement counter** simulate $dec(l, ct_i, l_k, l_m)$. Initially $\nu(x_i) = d$ with $0 < d \le 1$. At the beginning of the simulation, $x_i = 1$ is tested, which encodes the zero test of ct_i. In q_{l_k}, x_i will be doubled. The readers can also check that to reach the location q_{l_k}, timed elapsed in e_1^{dec,i,l_k} must be $1 - d$, and in e_2^{dec,i,l_k} must be d.
    $q_l \xrightarrow{x_i \in [1,1]?} q_{l_m}$ and
    $q_l \xrightarrow{x_i \in (0,1)?} q_1^{dec,i,l_k} \xrightarrow{c \leftarrow x_i} q_2^{dec,i,l_k} \xrightarrow{fpush} q_3^{dec,i,l_k} \xrightarrow{x_1 \leftarrow c} q_4^{dec,i,l_k} \xrightarrow{fpush} e_1^{dec,i,l_k} \xrightarrow{x_0 \leftarrow 0} q_8^{dec,i,l_k} \xrightarrow{c \in [1,1]?} q_9^{dec,i,l_k} \xrightarrow{c \leftarrow x_1} q_{10}^{dec,i,l_k} \xrightarrow{pop} q_4^{dec,i,l_k} \xrightarrow{\epsilon} e_2^{dec,i,l_k} \xrightarrow{x_0 \leftarrow 0} q_5^{dec,i,l_k} \xrightarrow{c \in [1,1]?} q_6^{dec,i,l_k} \xrightarrow{c \leftarrow x_1} q_7^{dec,i,l_k} \xrightarrow{pop} q_2^{dec,i,l_k} \xrightarrow{x_i \leftarrow c} q_{l_k}$

Theorem 1. *The reachability of a NeTA-I with a single global clock is undecidable.*

Remark 2. The invariants here are used to prevent time progress, and it can not be simulated by the traditional approach if pop rules are allowed, i.e., simply resetting x_0 to 0 first, and then using a test transition $x_0 \in [0,0]?$ at the tail. For example, in the pop rule $q_{11}^{inc,i,l_k} \xrightarrow{pop} q_4^{inc,i,l_k}$, the state q_{11}^{inc,i,l_k} is the final state of $\mathcal{A}_2$, and there is no way using only test transition $x_0 \in [0,0]?$ to promise time not elapsing in q_{11}^{inc,i,l_k}. Because after popping, we can not check values of the local clocks in the original TA $\mathcal{A}_2$, which has been already popped from the stack. Of course, if we introduce a fresh global clock, say c_0, the test transition $c_0 \in [0,0]?$ can prevent time progress. Then it is actually an encoding from a Minsky machine to a NeTA with two global clocks and without invariants, which is consistent with results in [5].

5 Constraint Dense Timed Pushdown Automata

In this section, we first present syntax and semantics of Constraint Dense Timed Pushdown Automata. Later, we introduce digiwords and operations which are used for encoding from a Constraint Dense Timed Pushdown Automaton to a snapshot pushdown system. Finally, the decidability of reachability of a snapshot pushdown system is shown by observing that it is a growing WSPDS with a well-formed constraint [6].

Definition 6 (Constraint Dense Timed Pushdown Automata). *A constraint dense timed pushdown automaton (Constraint DTPDA) is a tuple* $\mathcal{D} = \langle S, s_0, \Gamma, X, \mathbb{I}, \Delta\rangle \in \mathscr{D}$, *where*

- S *is a finite set of states with the initial state* $s_0 \in S$,
- Γ *is a finite stack alphabet,*
- X *is a finite set of clocks (with* $|X| = k$*),*
- $\mathbb{I} : S \to \Phi(X)$ *is a function that assigns to each state an invariant, and*
- $\Delta \subseteq S \times Action^{+} \times S$ *is a finite set of transitions.*

A (discrete) transition $\delta \in \Delta$ *is a sequence of actions* $(s_1, o_1, s_2), \cdots, (s_i, o_i, s_{i+1})$ *written as* $s_1 \xrightarrow{o_1;\cdots;o_i} s_{i+1}$, *in which* o_j *(for* $1 \leq j \leq i$*) is one of the followings,*

- **Local** ϵ, *an empty operation,*
- **Test** ϕ, *where* $\phi \in \Phi(X)$ *is a clock constraint,*
- **Reset** $x \leftarrow 0$ *where* $x \in X$,
- **Value passing** $x \leftarrow x'$ *where* $x, x' \in X$,
- **Push** $push(\gamma)$, *where* $\gamma \in \Gamma$ *is a stack symbol,*
- **F-Push** $fpush(\gamma)$, *where* $\gamma \in \Gamma$ *is a stack symbol, and*
- **Pop** $pop(\gamma)$, *where* $\gamma \in \Gamma$ *is a stack symbol.*

Definition 7 (Semantics of Constraint DTPDAs). *For a Constraint DTPDA* $\langle S, s_0, \Gamma, X, \mathbb{I}, \Delta\rangle$, *a configuration is a triplet* (s, w, ν) *with a state* $s \in S$, *a stack* $w \in (\Gamma \times (\mathbb{R}^{\geq 0})^k \times \{0,1\} \times \Phi(X))^*$, *and a clock valuation* ν *on* X. *Similarly, a stack* w good, *written as* $w^{\Uparrow}$, *if for each content* $(\gamma_i, \bar{t}_i, flag_i, \phi_i)$ *in* w, *we have* $\nu[x_1 \leftarrow t_1, \cdots, x_k \leftarrow t_k] \models \phi_i$ *where*

$\bar{t}_i = (t_1, \cdots, t_k)$. *For* $w = (\gamma_1, \bar{t}_1, flag_1, \phi_1). \cdots .(\gamma_n, \bar{t}_n, flag_n, \phi_n)$, *a t-time passage on the stack, written as* $w + t$, *is* $(\gamma_1, progress'(\bar{t}_1, t, flag_1), flag_1, \phi_1). \cdots .(\gamma_n, progress'(\bar{t}_n, t, flag_n), flag_n, \phi_n)$ *where*

$$progress'(\bar{t}, t, flag) = \begin{cases} (t_1 + t, \cdots, t_k + t) & if\ flag = 1\ and\ \bar{t} = (t_1, \cdots, t_k) \\ \bar{t} & if\ flag = 0 \end{cases}$$

The transition relation of the Constraint DTPDA is defined as follows:

- Progress transition*:* $(s, w, \nu) \xrightarrow{t}_{\mathscr{D}} (s, w + t, \nu + t)$, *where* $t \in \mathbb{R}^{\geq 0}$, $w^{\Uparrow}$, $\nu \models \mathbb{I}(s)$, $(w + t)^{\Uparrow}$ *and* $(\nu + t) \models \mathbb{I}(s)$.
- Discrete transition*:* $(s_1, w_1, \nu_1) \xrightarrow{o}_{\mathscr{D}} (s_2, w_2, \nu_2)$, *if* $s_1 \xrightarrow{o} s_2$, $w_1^{\Uparrow}, \nu_1 \models \mathbb{I}(s_1)$, $w_2^{\Uparrow}, \nu_2 \models \mathbb{I}(s_2)$ *and one of the following holds,*
  - **Local** $o = \epsilon$, *then* $w_1 = w_2$, *and* $\nu_1 = \nu_2$.
  - **Test** $o = \phi$, *then* $w_1 = w_2$, $\nu_1 = \nu_2$ *and* $\nu_1 \models \phi$.
  - **Reset** $o = x \leftarrow 0$, *then* $w_1 = w_2$, $\nu_2 = \nu_1[x \leftarrow 0]$.
  - **Value passing** $o = x \leftarrow x'$, *then* $w_1 = w_2, \nu_2 = \nu_1[x \leftarrow \nu_1(x')]$.
  - **Push** $o = push(\gamma)$, *then* $\nu_2 = \nu_{\mathbf{0}}$, $w_2 = (\gamma, (\nu_1(x_1), \cdots, \nu_1(x_k)), 1, \mathbb{I}(s_1)).w_1$ *for* $X = \{x_1, \cdots, x_k\}$.
  - **F-Push**
    $o = fpush(\gamma)$, *then* $\nu_2 = \nu_{\mathbf{0}}$, $w_2 = (\gamma, (\nu_1(x_1), \cdots, \nu_1(x_k)), 0, \mathbb{I}(s_1)).w_1$ *for* $X = \{x_1, \cdots, x_k\}$.
  - **Pop** $o = pop(\gamma)$, *then* $\nu_2 = \nu_1[x_1 \leftarrow t_1, \cdots, x_k \leftarrow t_k]$, $w_1 = (\gamma, (t_1, \cdots, t_k), flag, \phi).w_2$.

The initial configuration $\kappa_0 = (s_0, \epsilon, \nu_{\mathbf{0}})$. *We use* $\longrightarrow_{\mathscr{D}}$ *to range over these transitions, and* $\longrightarrow^*_{\mathscr{D}}$ *is the transitive closure of* $\longrightarrow_{\mathscr{D}}$.

Intuitively, in a stack $w = (\gamma_1, \bar{t}_1, flag_1, \phi_1). \cdots .(\gamma_n, \bar{t}_n, flag_n, \phi_n)$, γ_i is a stack symbol, $\bar{t}_i$ is k-tuple of clocks values of $x_1, \cdots, x_k$ respectively, $flag_i = 1$ if the stack frame is pushed and $flag_i = 0$ if fpushed and ϕ_i is a clock constraint.

Example 1. Figure 1 shows transitions between configurations of a Constraint DTPDA with $S = \{s_1, s_2, s_3, \cdots\}$, $X = \{x_1, x_2\}$, $\Gamma = \{a, b, d\}$ and $\mathbb{I} = \{\mathbb{I}(s_1) = x_1 \in [0, 1) \wedge x_2 \in [3, 4), \mathbb{I}(s_2) = x_1 \in [0, 3), \mathbb{I}(s_3) = \top, \cdots\}$. Values changed from the last configuration are in bold. For simplicity, we omit some transitions and start from s_1. From s_1 to s_2, a discrete transition $fpush(d)$ pushes d to the stack with the values of x_1 and x_2, frozen. After pushing, value of x_1 and x_2 will be reset to zero. Then, at state s_2, a progress transition elapses 2.6 time units, and each value grows older for 2.6 except for frozen clocks in the top. From s_2 to s_3, the batched transition first pops symbol d from the stack and clock values are recovered from the poped clocks. Then, the value of x_1 is reset to 0. Note that the invariants are always satisfied in these reachable configurations.

In the following subsections, 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. Let $frac(t) = t - floor(t)$ for $t \in \mathbb{R}^{\geq 0}$.

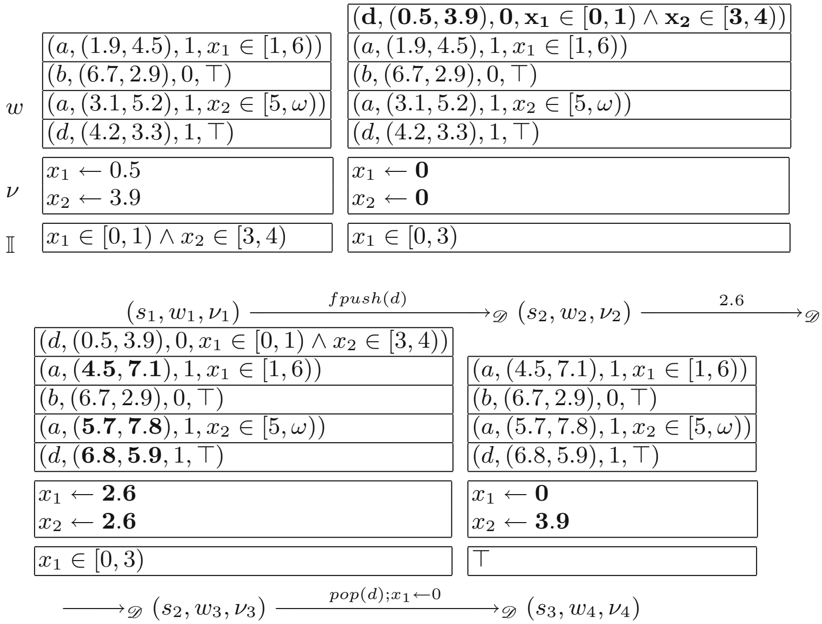


Fig. 1. An example of constraint DTPDAs

5.1 Digiword and Its Operations

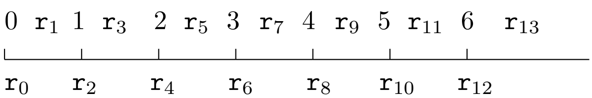
Let $\langle S, s_0, \Gamma, X, \mathbb{I}, \Delta \rangle$ be a Constraint DTPDA, and let n be the largest integer (except for ω) appearing in $\mathbb{I}$ and Δ.

Definition 8 (Two Subsets of Intervals). *Let*

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

Let $\mathcal{I}(n)$ denote a subset of intervals $\mathcal{I}$ such that all integers appearing in $\mathcal{I}(n)$ are less than or equal to n. For $v \in \mathbb{R}^{\ge 0}$, $proj(v) = \mathtt{r}_i$ if $v \in \mathtt{r}_i \in Intv(n)$.

Example 2. In Example 1, $n = 6$ and we have 13 intervals in $Intv(6)$,



$\mathcal{I}(6)$ contains intervals (a, b'), $[a, b]$, $[a, b')$ and $(a, b]$ where $a, b \in \{0, 1, \ldots, 6\}$ and $b' \in \{0, 1, \ldots, 6, \omega\}$.

$Intv(n)$ intend to contain digitizations of clocks, e.g., if a clock has value 1.9, then we say it is in $\mathtt{r}_3$. $\mathcal{I}(n)$ intend to contain intervals in invariants, e.g., an invariant $x \in [1, 2] \wedge y \in (3, 4)$ can be split into two intervals $[1, 2]$ and $(3, 4)$. Both $Intv(n)$ and $\mathcal{I}(n)$ are finite sets.

Definition 9 (Digitization). *A digitization* $\mathtt{digi} : \mathcal{MP}((X \cup \Gamma) \times \mathbb{R}^{\geq 0} \times \{0,1\} \times \mathcal{I}(n)) \to \mathcal{MP}((X \cup \Gamma) \times Intv(n) \times \{0,1\} \times \mathcal{I}(n))^*$ *is defined as follows.*

For $\bar{\mathcal{Y}} \in \mathcal{MP}((X \cup \Gamma) \times \mathbb{R}^{\geq 0} \times \{0,1\} \times \mathcal{I}(n))$, $\mathtt{digi}(\bar{\mathcal{Y}})$ *is a word* $Y_0 Y_1 \cdots Y_m$, *where* $Y_0, Y_1, \cdots, Y_m$ *are multisets that collect* $(x, proj(t), flag, I)$*'s having the same* $frac(t)$ *for* $(x, t, flag, I) \in \bar{\mathcal{Y}}$. *Among them,* Y_0 *(which is possibly empty) is reserved for the collection of* $(x, proj(t), flag, I)$ *with* $frac(t) = 0$ *and* $t \leq n$ *(i.e.,* $proj(t) = \mathtt{r}_{2i}$ *for* $0 \leq i \leq n$*). We assume that* Y_i *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* $(x, proj(t), flag, I) \in Y_i$ *and* $(x', proj(t'), flag', I') \in Y_{i+1}$*).*

For $Y_i \in \mathcal{MP}((X \cup \Gamma) \times Intv(n) \times \{0,1\} \times \mathcal{I}(n))$, we define the projections by $prc(Y_i) = \{(x, proj(t), 1, I) \in Y_i\}$ and $frz(Y_i) = \{(x, proj(t), 0, I) \in Y_i\}$. We overload the projections on $\bar{Y} = Y_0 Y_1 \cdots Y_m \in (\mathcal{MP}((X \cup \Gamma) \times Intv(n) \times \{0,1\} \times \mathcal{I}(n)))^*$ such that $frz(\bar{Y}) = frz(Y_0) frz(Y_1) \cdots frz(Y_m)$ and $prc(\bar{Y}) = prc(Y_0) prc(Y_1) \cdots prc(Y_m)$.

For a stack frame $v = (\gamma, (t_1, \cdots, t_k), flag, \phi)$ of a Constraint DTPDA, we denote a word $(\gamma, t_1, flag, EC(\phi, x_1)) \cdots (\gamma, t_k, flag, EC(\phi, x_k))$ by $dist(v)$. Given a state s and a clock valuation ν, we define a word $time(s, \nu) = (x_1, \nu(x_1), 1, EC(\mathbb{I}(s), x_1)) \ldots (x_k, \nu(x_k), 1, EC(\mathbb{I}(s), x_k))$ where $x_1 \ldots x_k \in X$.

Example 3. For the configuration $\varrho_1 = (s_1, v_4 \cdots v_1, \nu_1)$ in Example 1, let $\bar{\mathcal{Y}} = dist(v_4) \uplus \ldots \uplus dist(v_1) \uplus time(s_1, \nu_1)$, and $\bar{Y} = \mathtt{digi}(\bar{\mathcal{Y}})$, i.e.,

$$\begin{aligned}
\bar{\mathcal{Y}} = \{&(a, 1.9, 1, [1,6)), (a, 4.5, 1, [0,\omega)), (b, 6.7, 0, [0,\omega)), (b, 2.9, 0, [0,\omega)),\\
&(a, 3.1, 1, [0,\omega)), (a, 5.2, 1, [5,\omega)), (d, 4.2, 1, [0,\omega)),\\
&(d, 3.3, 1, [0,\omega)), (x_1, 0.5, 1, [0,1)), (x_2, 3.9, 1, [3,4))\}\\
\bar{Y} = \{&(a, \mathtt{r}_7, 1, [0,\omega))\}\{(a, \mathtt{r}_{11}, 1, [5,\omega)), (d, \mathtt{r}_9, 1, [0,\omega))\}\{(d, \mathtt{r}_7, 1, [0,\omega))\}\\
\{&(x_1, \mathtt{r}_1, 1, [0,1)), (a, \mathtt{r}_9, 1, [0,\omega))\}\{(b, \mathtt{r}_{13}, 0, [0,\omega))\}\{(x_2, \mathtt{r}_7, 1, [3,4)),\\
&(a, \mathtt{r}_3, 1, [1,6)), (b, \mathtt{r}_5, 0, [0,\omega))\}\\
prc(\bar{Y}) = \{&(a, \mathtt{r}_7, 1, [0,\omega))\}\{(a, \mathtt{r}_{11}, 1, [5,\omega)), (d, \mathtt{r}_9, 1, [0,\omega))\}\{(d, \mathtt{r}_7, 1, [0,\omega))\}\\
\{&(x_1, \mathtt{r}_1, 1, [0,1)), (a, \mathtt{r}_9, 1, [0,\omega))\}\{(x_2, \mathtt{r}_7, 1, [3,4)), (a, \mathtt{r}_3, 1, [1,6))\}\\
frz(\bar{Y}) = \{&(b, \mathtt{r}_{13}, 0, [0,\omega))\}(b, \mathtt{r}_5, 0, [0,\omega))\}
\end{aligned}$$

Definition 10 (Digiwords and k-pointers). *A word* $\bar{Y} \in (\mathcal{MP}((X \cup \Gamma) \times Intv(n) \times \{0,1\} \times \mathcal{I}(n)))^*$ *is called a* digiword. *We say a digiword* $\bar{Y}$ *is good, written as* $\bar{Y}^{\Uparrow}$, *if for all* $(x, \mathtt{r}_i, flag, I)$ *in* $\bar{Y}$, $\mathtt{r}_i \subseteq I$. *We denote* $\bar{Y}|_{\Lambda}$ *for* $\Lambda \subseteq \Gamma \cup X$, *by removing* $(x, \mathtt{r}_i, flag, I)$ *with* $x \notin \Lambda$. *A k-*pointer $\bar{\rho}$ *of* $\bar{Y}$ *is a tuple of k pointers to mutually different k elements in* $\bar{Y}|_{\Gamma}$. *We refer to the element pointed by the i-th pointer by* $\bar{\rho}[i]$. *From now on, we assume that a digiword has two pairs of k-pointers* $(\bar{\rho}_1, \bar{\rho}_2)$ *and* $(\bar{\tau}_1, \bar{\tau}_2)$ *that point to only proceeding and frozen clocks, respectively. We call* $(\bar{\rho}_1, \bar{\rho}_2)$ proceeding *k-pointers and* $(\bar{\tau}_1, \bar{\tau}_2)$ frozen *k-pointers. We also assume that they do not overlap each other, i.e., there are no* i, j, *such that* $\bar{\rho}_1[i] = \bar{\rho}_2[j]$ *or* $\bar{\tau}_1[i] = \bar{\tau}_2[j]$.

$\bar{\rho}_1$ and $\bar{\rho}_2$ intend the store of values of the proceeding clocks at the last and one before the last **Push**, respectively. $\bar{\tau}_1$ and $\bar{\tau}_2$ intend similar for frozen clocks at **F-Push**.

Definition 11 (Embedding over Digiwords). *For digiwords $\bar{Y} = Y_1 \cdots Y_m$ and $\bar{Z} = Z_1 \cdots Z_{m'}$ with pairs of k-pointers $(\bar{\rho}_1, \bar{\rho}_2), (\bar{\tau}_1, \bar{\tau}_2)$, and $(\bar{\rho}'_1, \bar{\rho}'_2), (\bar{\tau}'_1, \bar{\tau}'_2)$, respectively, we define an embedding $\bar{Y} \sqsubseteq \bar{Z}$, if there exists a monotonic injection $f : [1..m] \to [1..m']$ such that $Y_i \subseteq Z_{f(i)}$ for each $i \in [1..m]$, $f \circ \bar{\rho}_i = \bar{\rho}'_i$ and $f \circ \bar{\tau}_i = \bar{\tau}'_i$ for $i = 1, 2$.*

The embedding $\sqsubseteq$ is a well-quasi-ordering which will be exploited in Sect. 5.3.

Definition 12 (Operations on Digiwords). *Let $\bar{Y} = Y_0 \cdots Y_m, \bar{Y}' = Y'_0 \cdots Y'_{m'} \in (\mathcal{MP}((X \cup \Gamma) \times Intv(n) \times \{0,1\} \times \mathcal{I}(n)))^*$ such that $\bar{Y}$ (resp. $\bar{Y}'$) has two pairs of proceeding and frozen k-pointers $(\bar{\rho}_1, \bar{\rho}_2)$ and $(\bar{\tau}_1, \bar{\tau}_2)$ (resp. $(\bar{\rho}'_1, \bar{\rho}'_2)$ and $(\bar{\tau}'_1, \bar{\tau}'_2)$). We define digiword operations as follows.*

- **Decomposition**: *Let $Z \in \mathcal{MP}((X \cup \Gamma) \times Intv(n) \times \{0,1\} \times \mathcal{I}(n))$. If $Z \subseteq Y_j$, $decomp(\bar{Y}, Z) = (Y_0 \cdots Y_{j-1}, Y_j, Y_{j+1} \cdots Y_m)$.*
- **Refresh** *$refresh(\bar{Y}, s)$ for $s \in S$ is obtained by updating all elements $(x, \mathtt{r}_i, 1, I)$ with $(x, \mathtt{r}_i, 1, EC(\mathbb{I}(s), x))$ for $x \in X$.*
- **Init** *$init(\bar{Y})$ is obtained by removing all elements $(x, \mathtt{r}, 1, I)$ from $\bar{Y}$ and inserting $(x, \mathtt{r}_0, 1, [0, w])$ to Y_0 for all $x \in X$.*
- **Insert$_x$** *$insert_x(\bar{Y}, x, y)$ adds $(x, \mathtt{r}_i, 1, I)$ to Y_j for $(y, \mathtt{r}_i, 1, I) \in Y_j$, $x, y \in X$.*
- **Insert$_I$**: *Let $Z \in \mathcal{MP}((X \cup \Gamma) \times Intv(n) \times \{0,1\} \times \mathcal{I}(n))$ with $(x, \mathtt{r}_i, flag, I) \in Z$ for $x \in X \cup \Gamma$. $insert_I(\bar{Y}, Z)$ inserts Z to $\bar{Y}$ such that*

$$\begin{cases} either\, take\, the\, union\, of\, Z\, and\, Y_j\;\; for\, j > 0,\, or\, put\, Z\, at\, any\, place\;\; after\, Y_0 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad if\; i\; is\, odd \\ take\, the\, union\, of\; Z\; and\, Y_0 \qquad if\; i\; is\, even \end{cases}$$

- **Delete.** *$delete(\bar{Y}, x)$ for $x \subseteq X$ is obtained from $\bar{Y}$ by deleting the element $(x, \mathtt{r}, 1, I)$ indexed by x.*
- **Permutation**. *Let $\bar{V} = prc(\bar{Y}) = V_0 V_1 \cdots V_k$ and $\bar{U} = frz(\bar{Y}) = U_0 U_1 \cdots U_{k'}$. A one-step permutation $\bar{Y} \Rightarrow \bar{Y}'$ is given by $\Rightarrow\; = \;\Rightarrow_s \cup \Rightarrow_c$, defined below. We denote $inc(V_j)$ for V_j in which each $\mathtt{r}_i$ is updated to $\mathtt{r}_{i+1}$ for $i < 2n + 1$.*

$(\Rightarrow_s)$ *Let*

$$\begin{cases} decomp(U_0\ .\ inc(V_0)\ .\ tl(\bar{Y}),\ V_k) = (\bar{Y}^k_{\dashv},\ \hat{Y}^k,\ \bar{Y}^k_{\vdash}) \\ decomp(insert_I((\hat{Y}^k \setminus V_k)\ .\ \bar{Y}^k_{\vdash},\ V_k),\ V_k) = (\bar{Z}^k_{\dashv},\ \hat{Z}^k,\ \bar{Z}^k_{\vdash}). \end{cases}$$

For j with $0 \le j < k$, we repeat to set

$$\begin{cases} decomp(\bar{Y}^{j+1}_{\dashv}\ .\ \bar{Z}^{j+1}_{\dashv},\ V_j) = (\bar{Y}^j_{\dashv},\ \hat{Y}^j,\ \bar{Y}^j_{\vdash}) \\ decomp(insert_I((\hat{Y}^j \setminus V_j)\ .\ \bar{Y}^j_{\vdash},\ V_j),\ V_j) = (\bar{Z}^j_{\dashv},\ \hat{Z}^j,\ \bar{Z}^j_{\vdash}). \end{cases}$$

Then, $\bar{Y} \Rightarrow_s \bar{Y}' = \bar{Y}^0_{\dashv}\ \bar{Z}^0_{\dashv}\ \hat{Z}^0\ \bar{Z}^1_{\dashv}\ \hat{Z}^1 \cdots \bar{Z}^k_{\dashv}\ \hat{Z}^k\ \bar{Z}^k_{\vdash}$.

$(\Rightarrow_c)$ *Let $\bar{Y}^k_{\dashv} = U_0 \cup inc(V_k)$ and $\bar{Z}^k_{\dashv} = inc(V_0)\ Y_1 \cdots (Y_{i'} \setminus V_k) \cdots Y_m$.*

For j with $0 \le j < k$, we repeat to set

$$\begin{cases} decomp(\bar{Y}^{j+1}_{\dashv}.\bar{Z}^{j+1}_{\dashv}, V_j) = (\bar{Y}^j_{\dashv}, \hat{Y}^j, \bar{Y}^j_{\vdash}) \\ decomp(insert_I((\hat{Y}^j \setminus V_j).\bar{Y}^j_{\vdash}, V_j), V_j) = (\bar{Z}^j_{\dashv}, \hat{Z}^j, \bar{Z}^j_{\vdash}). \end{cases}$$

Then, $\bar{Y} \Rightarrow_c \bar{Y}' = \bar{Y}^0_{\dashv}\ \bar{Z}^0_{\dashv}\ \hat{Z}^0\ \bar{Z}^1_{\dashv}\ \hat{Z}^1 \cdots \bar{Z}^{k-1}_{\dashv}\ \hat{Z}^{k-1}\ \bar{Z}^{k-1}_{\vdash}$.

$(\bar{\rho}_1, \bar{\rho}_2)$ is updated to correspond to the permutation accordingly, and $(\bar{\tau}_1, \bar{\tau}_2)$ is kept unchanged.

- **Rotate:** *For proceeding k-pointers $(\bar{\rho}_1, \bar{\rho}_2)$ of $\bar{Y}$ and $\bar{\rho}$ of $\bar{Z}$, let $\bar{Y}|_\Gamma \Rightarrow^* \bar{Z}|_\Gamma$ such that the permutation makes $\bar{\rho}_1$ match with $\bar{\rho}$. Then, $rotate_{\bar{\rho}_1 \mapsto \bar{\rho}}(\bar{\rho}_2)$ is the corresponding k-pointer of $\bar{Z}$ to $\bar{\rho}_2$.*
- **Map**$_{\rightarrow}^{flag}$ *$map_{\rightarrow}^{fl}(\bar{Y}, \gamma)$ for $\gamma \in \Gamma$ is obtained from $\bar{Y}$ by, for each $x_i \in X$, replacing $(x_i, \mathtt{r}_j, 1, I)$ with $(\gamma, \mathtt{r}_j, fl, I)$. Accordingly, if $fl = 1$, $\bar{\rho}_1[i]$ is updated to point to $(\gamma, \mathtt{r}_j, 1, I)$, and $\bar{\rho}_2$ is set to the original $\bar{\rho}_1$. If $fl = 0$, $\bar{\tau}_1[i]$ is updated to point to $(\gamma, \mathtt{r}_j, 0, I)$, and $\bar{\tau}_2$ is set to the original $\bar{\tau}_1$.*
- **Map**$_{\leftarrow}^{flag}$ *$map_{\leftarrow}^{fl}(\bar{Y}, \bar{Y}', \gamma)$ for $\gamma \in \Gamma$ is obtained,*
  (if $fl = 1$**)** *by replacing each $\bar{\rho}_1[i] = (\gamma, \mathtt{r}_j, 1, I)$ in $\bar{Y}|_\Gamma$ with $(x_i, \mathtt{r}_j, 1, I)$ for $x_i \in X$. Accordingly, new $\bar{\rho}_1$ is set to the original $\bar{\rho}_2$, and new $\bar{\rho}_2$ is set to $rotate_{\bar{\rho}'_1 \mapsto \bar{\rho}_2}(\bar{\rho}'_2)$. $\bar{\tau}_1$ and $\bar{\tau}_2$ are kept unchanged.*
  (if $fl = 0$**)** *by replacing each $\bar{\tau}_1[i] = (\gamma, \mathtt{r}_j, 0, I)$ in $\bar{Y}|_\Gamma$ with $(x_i, \mathtt{r}_j, 1, I)$ for $x_i \in X$. Accordingly, new $\bar{\tau}_1$ is set to the original $\bar{\tau}_2$, and new $\bar{\tau}_2$ is set to $\bar{\tau}'_2$. $\bar{\rho}_1$ and $\bar{\rho}_2$ are kept unchanged.*

We will use these operations on digiwords for encoding in the next subsection.

5.2 Snapshot Pushdown System

In this subsection, we show that a Constraint DTPDA is encoded into its digitization, called a *snapshot pushdown system* (snapshot PDS), which keeps the digitization of all clocks in the top stack frame, as a *digiword*. The keys of the encoding are, (1) when a pop occurs, the time progress recorded at the top stack symbol is propagated to the next stack symbol after finding a permutation by matching between proceeding k-pointers $\bar{\rho}_2$ and $\bar{\rho}'_1$, and (2) only invariants in the top stack frame need to be checked. Before showing the encoding, we first define the encoded configuration, called *snapshot configuration*.

Definition 13 (Snapshot Configuration). *Let $\pi : \varrho_0 = (s_0, \epsilon, \nu_\mathbf{0}) \longrightarrow^*_{\mathscr{D}} \varrho = (s, w, \nu)$ be a transition sequence of a Constraint DTPDA from the initial configuration. If π is not empty, we refer the last step as $\lambda : \varrho' \longrightarrow_{\mathscr{D}} \varrho$, and the preceding sequence by $\pi' : \varrho_0 \longrightarrow^*_{\mathscr{D}} \varrho'$. Let $w = v_m \cdots v_1$. A* snapshot *is $snap(\pi) = (\bar{Y}, flag(v_m))$, where $\bar{Y} = \mathtt{digi}(\uplus_i dist(v_i) \uplus time(s, \nu))$. Let a k-pointer $\bar{\xi}(\pi)$ be $\bar{\xi}(\pi)[i] = (\gamma, proj(t_i), flag(v_m), I)$ for $(\gamma, t_i, flag(v_m), I) \in dist(v_m)$. A* snapshot configuration *$Snap(\pi)$ is inductively defined from $Snap(\pi')$.*

$$\begin{cases} (s_0, snap(\epsilon)) \quad\quad if\, \pi = \epsilon. (\bar{\rho}_1, \bar{\rho}_2)\, and\, (\bar{\tau}_1, \bar{\tau}_2)\, are\, \text{undefined}. \\ (s', snap(\pi)\ tail(Snap(\pi'))) \quad\quad if\, \lambda\, is\, \textbf{Timeprogress}\, with\, \bar{Y}' \Rightarrow^* \bar{Y}. \\ \quad\quad Then, the permutation \bar{Y}' \Rightarrow^* \bar{Y}\, updates\, (\bar{\rho}'_1, \bar{\rho}'_2)\, to\, (\bar{\rho}_1, \bar{\rho}_2). \\ (s', snap(\pi)\ tail(Snap(\pi'))) \quad\quad if\, \lambda\, is\, \textbf{Local}, \textbf{Test}, \textbf{Reset}, \textbf{Value}-\textbf{passing}. \\ (s, snap(\pi)\ Snap(\pi')) \quad\quad if\, \lambda\, is\, \textbf{Push}. Then,\, (\bar{\rho}_1, \bar{\rho}_2) = (\bar{\xi}(\pi), \bar{\rho}'_1). \\ (s, snap(\pi)\ Snap(\pi')) \quad\quad if\, \lambda\, is\, \textbf{F}-\textbf{Push}. Then,\, (\bar{\tau}_1, \bar{\tau}_2) = (\bar{\xi}(\pi), \bar{\tau}'_1). \\ (s, snap(\pi)\ tail(tail(Snap(\pi')))) \quad if\, \lambda\, is\, \textbf{Pop}. \\ \quad If\, flag = 1, (\bar{\rho}_1, \bar{\rho}_2) = (\bar{\rho}'_2, rotate_{\bar{\rho}''_1 \mapsto \bar{\rho}'_2}(\bar{\rho}''_2)); otherwise,\, (\bar{\tau}_1, \bar{\tau}_2) = (\bar{\tau}'_2, \bar{\tau}''_2). \end{cases}$$

We refer $head(Snap(\pi'))$ by $\bar{Y}'$, $head(tail(Snap(\pi'))$ by $\bar{Y}''$. Pairs of pointers of $\bar{Y}$, $\bar{Y}'$, and $\bar{Y}''$ are denoted by $(\bar{\rho}_1, \bar{\rho}_2)$, $(\bar{\rho}'_1, \bar{\rho}'_2)$, and $(\bar{\rho}''_1, \bar{\rho}''_2)$, respectively. If not mentioned, pointers are kept as is.

Definition 14 (Snapshot PDS). *For a Constraint DTPDA* $\langle S, s_0, \Gamma, X, \mathbb{I}, \nabla\rangle$, *a* snapshot PDS $\mathcal{S}$ *is a PDS (with possibly infinite stack alphabet)*

$$\langle S \cup \{s_{err}\}, s_0, (\mathcal{MP}((X \cup \Gamma) \times Intv(n) \times \{0,1\} \times \mathcal{I}(n)))^* \times \{0,1\}, \Delta_d\rangle.$$

with the initial configuration $\langle s_0, (\{(x, \mathtt{r}_0, 1, EC(\mathbb{I}(s_0), x)) \mid x \in X\}, 1)\rangle$. *For simplicity, we define* $s'' = \begin{cases} s' & \text{if } \bar{Y}'^{\Uparrow}, \\ s_{err} & \text{otherwise} \end{cases}$ *where* s_{err} *is a special error state that is used to indicate invariants are violated. Then* Δ_d *consists of:*

Progress $\langle s, (\bar{Y}, flag)\rangle \hookrightarrow_{\mathcal{S}} \langle s'', (\bar{Y}', flag)\rangle$ *for* $\bar{Y} \Rightarrow^* \bar{Y}'$, *where* $s' = s$.

Local *(*$s \xrightarrow{\epsilon} s' \in \Delta$*)* $\langle s, (\bar{Y}, flag)\rangle \hookrightarrow_{\mathcal{S}} \langle s'', (\bar{Y}', flag)\rangle$, *where* $\bar{Y}' = refresh(\bar{Y}, s')$.

Test *(*$s \xrightarrow{\phi} s' \in \Delta$*)* $\langle s, (\bar{Y}, flag)\rangle \hookrightarrow_{\mathcal{S}} \langle s'', (\bar{Y}', flag)\rangle$, *where* $\bar{Y}' = refresh(\bar{Y}, s')$, *if for every* $(x, \mathtt{r}_i, flag, I) \in \bar{Y}$ *with* $x \in X$, $\mathtt{r}_i \subseteq EC(\phi, x)$ *holds,*

Reset *(*$s \xrightarrow{x \leftarrow 0} s' \in \Delta$ *with* $\lambda \subseteq X$*)* $\langle s, (\bar{Y}, flag)\rangle \hookrightarrow_{\mathcal{S}} \langle s'', (\bar{Y}', flag)\rangle$, *where* $\bar{Y}' = refresh(insert_I(delete(\bar{Y}, x), (x, \mathtt{r}_0, 1, [0, w))), s')$.

Value-passing *(*$s \xrightarrow{x \leftarrow y} s' \in \Delta$ *with* $x, y \in X$*)* $\langle s, (\bar{Y}, flag)\rangle \hookrightarrow_{\mathcal{S}} \langle s'', (\bar{Y}', flag)\rangle$,
where $\bar{Y}' = refresh(insert_x(delete(\bar{Y}, x), x, y), s')\rangle$.

Push *(*$s \xrightarrow{push(\gamma)} s' \in \Delta$; $fl = 1$*) and* **F-Push** *(*$s \xrightarrow{fpush(\gamma)} s' \in \Delta$; $fl = 0$*)* $\langle s, (\bar{Y}, flag)\rangle \hookrightarrow_{\mathcal{S}} \langle s'', (\bar{Y}', fl)(\bar{Y}, flag)\rangle$,
where $\bar{Y}' = refresh(init(map^{fl}_{\rightarrow}(\bar{Y}, \gamma), s')$.

Pop *(*$s \xrightarrow{pop(\gamma)} s' \in \Delta$*)* $\langle s, (\bar{Y}, flag)(\bar{Y}'', flag')\rangle \hookrightarrow_{\mathcal{S}} \langle s'', (\bar{Y}', flag')\rangle$,
where $\bar{Y}' = refresh(map^{flag}_{\leftarrow}(\bar{Y}, \bar{Y}'', \gamma), s')$.

By induction on the number of steps of transitions, the encoding relation between a Constraint DTPDA and a snapshot PDS 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 Constraint DTPDA (resp. its encoded snapshot PDS* $\mathcal{S}$*).*

(Preservation) *If* $\pi : \varrho_0 \longrightarrow^*_{\mathscr{D}} \varrho$, *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* $\langle s_0, \tilde{w}_0\rangle \hookrightarrow^*_{\mathcal{S}} \langle s, \tilde{w}\rangle$,
$s = s_{err}$ *is an error state, or*
$s \neq s_{err}$ *and there exists* $\pi : \varrho_0 \longrightarrow^*_{\mathscr{D}} \varrho$ *with* $Snap(\pi) = \langle s, \tilde{w}\rangle$.

5.3 Well-Formed Constraint

A snapshot PDS is *a growing WSPDS* (Definition 6 in [6]) and $\Downarrow_\Upsilon$ gives a *well-formed constraint* (Definition 8 in [6]). Let us recall the definitions.

Let P be a set of control locations and let Γ be a stack alphabet. Different from an ordinary definition of PDSs, we do not assume that P and Γ are finite, but associated with well-quasi-orderings (WQOs) $\preceq$ and $\leq$, respectively. Note that the embedding $\sqsubseteq$ over digiwords is a WQO by Higman's lemma.

For $w = \alpha_1\alpha_2\cdots\alpha_n, v = \beta_1\beta_2\cdots\beta_m \in \Gamma^*$, let $w \lll v$ if $m = n$ and $\forall i \in [1..n].\alpha_i \leq \beta_i$. We extend $\lll$ on configurations such that $(p, w) \lll (q, v)$ if $p \preceq q$ and $w \lll v$ for $p, q \in P$ and $w, v \in \Gamma^*$. A partial function $\psi \in \mathcal{P}Fun(X, Y)$ is *monotonic* if $\gamma \leq \gamma'$ with $\gamma \in dom(\psi)$ implies $\psi(\gamma) \lll \psi(\gamma')$ and $\gamma' \in dom(\psi)$.

A *a well-structured PDS* (WSPDS) is a triplet $\langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$ of a set $(P, \preceq)$ of WQO states, a WQO stack alphabet $(\Gamma, \leq)$, and a finite set $\Delta \subseteq \mathcal{P}Fun(P \times \Gamma, P \times \Gamma^{\leq 2})$ of monotonic partial functions. A WSPDS is *growing* if, for each $\psi(p, \gamma) = (q, w)$ with $\psi \in \Delta$ and $(q', w') \ggg (q, w)$, there exists (p', γ') with $(p', \gamma') \ggg (p, \gamma)$ such that $\psi(p', \gamma') \ggg (q', w')$.

A well-formed constraint describes a syntactical feature that is preserved under transitions. Theorem 5 in [6] ensures the reachability of a growing WSPDS when it has a well-formed constraint.

Definition 15 (Well-formed constraint). *Let a configuration $(s, \tilde{w})$ of a snapshot PDS $\mathcal{S}$. An element in a stack frame of $\tilde{w}$ has a* parent *if it has a corresponding element in the next stack frame. The transitive closure of the parent relation is* an ancestor. *An element in $\tilde{w}$ is* marked, *if its ancestor is pointed by a pointer in some stack frame. We define a* projection $\Downarrow_\Upsilon(\tilde{w})$ *by removing unmarked elements in $\tilde{w}$. We say that $\tilde{w}$ is* well-formed *if $\Downarrow_\Upsilon(\tilde{w}) = \tilde{w}$.*

The idea of $\Downarrow_\Upsilon$ is to remove unnecessary elements (i.e., elements not related to previous actions) from the stack content. Note that a configuration reachable from the initial configuration by $\hookrightarrow^*_{\mathcal{S}}$ is always well-formed. Since a snapshot PDS is a growing WSPDS with $\Downarrow_\Upsilon$, we conclude Theorem 2 from Lemma 1.

Theorem 2. *The reachability of a Constraint DTPDA is decidable.*

6 Decidability Results of NeTA-Is

In this section, we encode NeTA-I with no global clocks to constraint DTPDAs and thus show the decidability of the former model.

Given a NeTA-I $\mathcal{N} = (T, \mathcal{A}_0, X, C, \mathbb{I}, \Delta)$ with no global clocks ($C = \emptyset$), we define the target Constraint DTPDA $\mathcal{E}(\mathcal{N}) = \langle S, s_0, \Gamma, X, \mathbb{I}', \nabla \rangle$ such that

- $S = \Gamma = \bigcup_{A_i \in T} Q(A_i)$ is the set of all control locations of TAs in T.
- $s_0 = q_0(\mathcal{A}_0)$ is the initial control location of the initial TA $\mathcal{A}_0$.
- $X = \{x_1, ..., x_k\}$is the set of k local clocks.
- $\mathbb{I}' : S \to \Phi(X)$ is a function such that $\mathbb{I}'(s) = \mathbb{I}(\mathcal{A}_i)(s)$ where $s \in Q(\mathcal{A}_i)$.
- ∇ is the union $\bigcup_{\mathcal{A}_i \in T} \Delta(\mathcal{A}_i) \bigcup \mathcal{H}(\mathcal{N})$ where
  $\begin{cases} \Delta(\mathcal{A}_i) = \{\textbf{Local}, \textbf{Test}, \textbf{Reset}, \textbf{Value-passing}\}, \\ \mathcal{H}(\mathcal{N}) \text{ consists of rules below.} \end{cases}$

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

Definition 16. *Let $\mathcal{N}$ be a NeTA-I $(T, \mathcal{A}_0, X, C, \mathbb{I}, \Delta)$ with no global clocks and let $\mathcal{E}(\mathcal{N})$ be the encoded constraint DTPDA $\langle S, s_0, \Gamma, X, \mathbb{I}', \nabla\rangle$. For a configuration $\kappa = (\langle q, \nu, \mu\rangle, v)$ of $\mathcal{N}$ such that $v = (q_1, flag_1, \nu_1)\ldots(q_n, flag_n, \nu_n)$, $[\![\kappa]\!]$ denotes a configuration $(q, \overline{w}(\kappa), \nu)$ of $\mathcal{E}(\mathcal{N})$ where $\overline{w}(\kappa) = w_1 \cdots w_n$ with $w_i = (q_i, \nu_i, flag_i, \mathbb{I}(q_i))$.*

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

Lemma 2. *For a NeTA-I $\mathcal{N}$ with no global clocks, its encoded Constraint DTPDA $\mathcal{E}(\mathcal{N})$, and configurations κ, κ' of $\mathcal{N}$,*

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

Theorem 3. *The reachability of a NeTA-I with no global clocks is decidable.*

7 Conclusion

This paper proposes a model NeTA-Is by extending NeTAs with invariants assigned to each control location. We have shown that the reachability problem of a NeTA-I with a single global clock is undecidable, while that of a NeTA-I without global clocks is decidable. Compared to the different result of NeTA [5], it is revealed that unlike that of timed automata, invariants affect the expressiveness of timed recursive systems. Hence, when adopting timed recursive systems to model and verify complex real-time systems, one should carefully consider the introduction of invariants.

Acknowledgements. This work is supported by National Natural Science Foundation of China with grant Nos. 61472240, 61672340, 61472238, and the NSFC-JSPS bilateral joint research project with grant No. 61511140100.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoret. Comput. Sci. **126**, 183–235 (1994)
2. Henzinger, T.A., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. Inf. Comput. **111**, 193–244 (1994)
3. Bengtsson, J., Yi, W.: Timed automata: semantics, algorithms and tools. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 87–124. Springer, Heidelberg (2004). doi:10.1007/978-3-540-27755-2_3
4. 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
5. 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
6. Cai, X., Ogawa, M.: Well-structured pushdown system: case of dense timed pushdown automata. In: Codish, M., Sumii, E. (eds.) FLOPS 2014. LNCS, vol. 8475, pp. 336–352. Springer, Cham (2014). doi:10.1007/978-3-319-07151-0_21

7. Abdulla, P.A., Atig, M.F., Stenman, J.: Dense-timed pushdown automata. In: Proceedings of the LICS 2012, pp. 35–44. IEEE Computer Society (2012)
8. Trivedi, A., Wojtczak, D.: Recursive timed automata. In: Bouajjani, A., Chin, W.-N. (eds.) ATVA 2010. LNCS, vol. 6252, pp. 306–324. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15643-4_23
9. Benerecetti, M., Minopoli, S., Peron, A.: Analysis of timed recursive state machines. In: Proceedings of the TIME 2010, pp. 61–68. IEEE Computer Society (2010)
10. Minsky, M.: Computation: Finite and Infinite Machines. Prentice-Hall, Upper Saddle River (1967)