Title	Well-Structured Pushdown Systems, Part 2: On Reachability of Dense Timed Pushdown Automata
Author(s)	Ogawa, Mizuhito; Cai, Xiaojuan
Citation	Research report (School of Information Science, Japan Advanced Institute of Science and Technology), IS-RR-2013-005: 1-18
Issue Date	2013-08-19
Type	Technical Report
Text version	publisher
URL	http://hdl.handle.net/10119/11446
Rights	
Description	リサーチレポート（北陸先端科学技術大学院大学情報科学研究科）

Well-Structured Pushdown Systems, Part 2: On Reachability of Dense Timed Pushdown Automata.

Mizuhito Ogawa
School of Information Science
Japan Advanced Institute of Science and Technology

Xiaojuan Cai
BASICS Lab, Shanghai Jiotong University

August 19, 2013

IS-RR-2013-005

Well-structured pushdown system, Part 2: On Reachability of Dense Timed Pushdown Automata ⋆

Mizuhito Ogawa[1] and Xiaojuan Cai[2]

[1] Japan Advanced Institute of Science and Technology
mizuhito@jaist.ac.jp
[2] Shanghai Jiao Tong University, China
cxj@sjtu.edu.cn

Abstract. This paper investigates a general framework of a pushdown system with well-quasi-ordered states and stack alphabet to show decidability of reachability. As an instance, an alternative proof of the decidability of the (configuration) reachability for dense-timed pushdown system (in *P.A. Abdulla, M.F. Atig, F. Stenman, Dense-Timed Pushdown Automata, IEEE LICS 2012*) is presented.

1 Introduction

Infinite state transition systems appear in many places still keeping certain decidable properties, e.g., pushdown systems (PDS), timed automata [3], and vector addition systems (VAS, or Petri nets). Well-structured transition systems (WSTSs) [2, 11] are one of successful general frameworks to reason about decidability (except for PDSs). The coverability of VASs, the reachability of communicating finite state machines with lossy channels [11], and the inclusion problem between timed automata with single clocks [15] are beginning of a long list.

A natural extension of WSTS is to associate a stack. It is tempting to apply Higman's lemma on stacks. However this fails immediately, since the monotonicity of transitions with respect to the embedding on stacks hardly holds.

This paper investigates a general framework for PDSs with well-quasi-ordered states and stack alphabet, *well-structured pushdown systems*. Well-quasi-orderings (WQOs) over states and stack alphabet are extended to configurations by the element-wise comparison. Note that this extension will not preserve WQO (nor well founded). By combining classical Pre^*-automaton technique [5, 12, 10], we reduce the argument on stacks to that on stack symbols, and similar to WSTS, finite convergence of antichain techniques during Pre^*-automata saturation is guaranteed by a WQO.

When the set P of states is finite, we have decidability of coverability [6]. When P is infinite (but equipped with WQO), we can state decidability of quasi-coverability only. To compensate, we introduce a well-formed projection $\Downarrow_{\Upsilon}$,

⋆ JAIST Research Report IS-RR-2013-005, August 19th 2013

which extracts a core shape from the stack related to pushdown transitions. If we find $\Downarrow_{\Upsilon}$ such that, for configurations c, c' with $c \hookrightarrow c'$,

- **compatibility**: $\Downarrow_{\Upsilon}(c) \hookrightarrow \Downarrow_{\Upsilon}(c')$, and
- **stability**: $c \in \Upsilon$ if, and only if, $c' \in \Upsilon$, where $\Upsilon = \{c \mid c = \Downarrow_{\Upsilon}(c)\}$,

the quasi-coverability leads the configuration reachability. The compatibility strengthens the quasi-coverability to the coverability, and the stability boosts the coverability to the configuration reachability.

As an instance, we encode a dense-timed pushdown automaton (DTPDA) [1] into a snapshot PDS, inspired by the digitization techniques in [15]. A snapshot PDS has the set of snapshot words as stack alphabet. A snapshot word is essentially a region construction of the dimension equal to its size. Since a snapshot PDS contains non-standard pop rules (i.e., $(p, \gamma\gamma') \to (q, \gamma'')$), by associating a top stack symbol to a state, it is encoded as a PDS with WQO states and stack alphabet. Our general framework shows an alternative decidability proof of the reachability of a DTPDA, which has shown in [1].[3]

Our contribution is not on logically difficult proofs, but clarifying the proof structure behind theorems. Different from [1], our encoding idea into a snapshot PDS is inspired by [15].

Related Work

There are lots of works with context-sensitive infinite state systems. A process rewrite systems combines a PDS and a Petri net, in which vector additions/subtractions between adjacent stack frames during push/pop operations are prohibited [14]. With this restrictions, its reachability becomes decidable. A WQO automaton [7], is a WSTS with auxiliary storage (e.g., stacks and queues). It proves that the coverability is decidable under compatibility of *rank* functions with a WQO, of which an Multiset PDS is an instance. A timed pushdown automaton is a timed extension of a pushdown automaton. It has only global clocks, and the region construction [3] encodes it to a standard PDS [4, 8, 9]. DTPDA [1] firstly introduces local ages, which are stored with stack symbols when pushed, and never reset. DTPDA utilizes them to check whether an age in a stack frame satisfies constraints when pop occurs.

A WSPDS is firstly introduced in [6]. It focuses on WSPDSs with finite control states (and well-quasi-ordered stack alphabet), whereas the paper explores WSPDSs with well-quasi-ordered control states at the cost of weakening the target decidable property from the coverability to the quasi-coverability. The well-formed projection (Section 4), if exists, strengthens it again to the configuration reachability.

[3] In [1], only the state reachability is mentioned, but the proof is applied also for the configuration reachability.

Paper construction

The rest of the paper is constructed as follows. Section 2 briefly reviews a DTPDA [1]. Section 3 introduces P-automaton techniques for reachability of a pushdown system (PDS), which are extended to the coverability and the quasi-coverability. Note that we discuss on their correctness (at the limit), without assuming finite convergence. Section 4 proposes a *well-formed projection*. If we can find it, the quasi-coverability is lifted up to the configuration reachability. Section 5 introduces a Well-Structured Pushdown System (WSPDS) [6] and shows that the backward saturation of P-automaton (with upward ideals, in Section 3.3) finitely converges. Section 6 proposes *snapshot words*, which summarize and discretize the stack content as a (top) stack symbol. Section 7 presents the decidability of the reachability of a DTPDA, by encoding it into a WSPDS and finding a well-formed projection $\Downarrow_{\Upsilon}$ for snapshot words. Finally, Section 8 concludes the paper.

2 Dense-Timed Pushdown Automata

Dense-timed pushdown automaton (DTPDA) extends timed pushdown automaton (TPDA) with *local ages* [1]. A local age in each context is set when a push transition occurs, and restricts a pop transition only when the value of a local age meets the condition. The values of local ages proceed synchronously to global clocks, and they are never reset. Following [1], we omit input alphabet, since our focus is on reachability (regardless of an input word).

As notational convention, Section 2 and 7.2 use I for an interval (obeying to [1]), whereas Section 5 used I for an ideal.

Definition 1. *A DTPDA is a tuple $\langle S, s_{init}, \Gamma, \mathcal{C}, \Delta \rangle$, where*

- *S is a finite set of states with the initial state $s_{init} \in S$,*
- *Γ is a finite stack alphabet,*
- *$\mathcal{C}$ is a finite set of clocks, and*
- *Δ is a finite set of transitions.*

A transition $t \in \Delta$ is a triplet (s, op, s') in which $s, s' \in S$ and op is either of

- **Local** *nop, a state transition in S,*
- **Assignment** *$x \leftarrow I$, assign a clock $x \in \mathcal{C}$ to an arbitrary value in I,*
- **Test** *$x \in I?$, test whether the value of a clock $x \in \mathcal{C}$ is in I,*
- **Push** *$push(\gamma, I)$, push γ on a stack associated with a local age of an arbitrary value in I, and*
- **Pop** *$pop(\gamma, I)$, pop γ on a stack if the associated age a is in I.*

where I is an interval bounded by natural numbers (i.e., $[l, h], (l, h], [l, h), (l, h)$ for $l, h \in \mathbb{N} \cup \{\omega\}$ with $l \leq h$).

If each I in **Push** and **Pop** rules is $[0, \infty)$ (i.e., no conditions on local ages), we say simply a Timed Pushdown Automaton.

Definition 2. *For a DTPDA* $\langle S, s_{init}, \Gamma, \mathcal{C}, \Delta \rangle$*, a configuration is a triplet* (s, ν, w) *with* $s \in S$*, a clock valuation* $\nu : \mathcal{C} \to \mathbb{R}_{\geq 0}$*, and* $w \in (\Gamma \times \mathbb{R}_{\geq 0})^*$*. We refer* s *in a configuration* $c = (s, \nu, w)$ *by* $state(c)$*. For* $t \in \mathbb{R}_{\geq 0}$*, we denote*

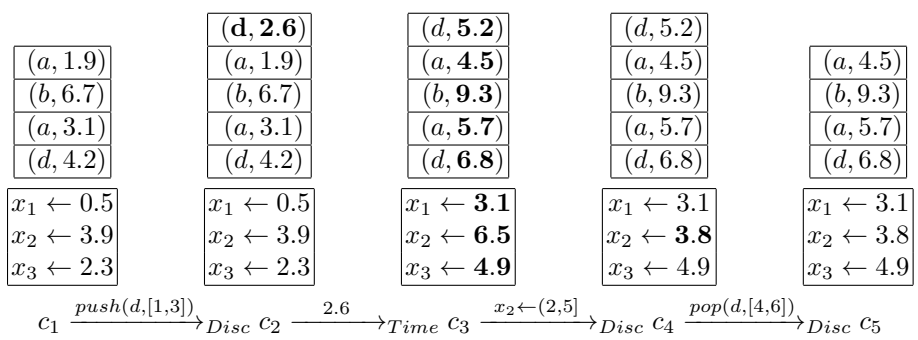
- $\nu_{\mathbf{0}}(x) = 0$ *for* $x \in \mathcal{C}$*,*
- $\nu_{x \leftarrow t}(x) = t$ *and* $\nu_{x \leftarrow t}(y) = \nu(y)$ *if* $y \neq x$*,*
- $(\nu + t)(x) = \nu(x) + t$ *for* $x \in \mathcal{C}$*, and*
- $w + t = (\gamma_1, t_1 + t). \cdots .(\gamma_k, t_k + t)$ *for* $w = (\gamma_1, t_1). \cdots .(\gamma_k, t_k)$*.*

There are two types of transitions, timed $\xrightarrow{t}_{Time}$ *and* discrete *transitions* $\xrightarrow{op}_{Disc}$*. Semantics of a timed transition is* $(s, \nu, w) \xrightarrow{t}_{Time} (s, \nu{+}t, w{+}t)$*, and a discrete transitions* (s, op, s') *is either*

- **Local** $(s, \nu, w) \xrightarrow{nop}_{Disc} (s', \nu, w)$*,*
- **Assignment** $(s, \nu, w) \xrightarrow{x \leftarrow I}_{Disc} (s', \nu_{x \leftarrow t}, w)$ *for* $t \in I$*,*
- **Test** $(s, \nu, w) \xrightarrow{x \in I?}_{Disc} (s', \nu, w)$ *if* $\nu(x) \in I$*,*
- **Push** $(s, \nu, w) \xrightarrow{push(\gamma, I)}_{Disc} (s', \nu, (\gamma, t).w)$ *for* $t \in I$*, and*
- **Pop** $(s, \nu, (\gamma, t).w) \xrightarrow{pop(\gamma, I)}_{Disc} (s', \nu, w)$ *if* $t \in I$*.*

We assume that the initial configuration is $(s_{init}, \nu_{\mathbf{0}}, \epsilon)$*.*

Example 1. The figure shows transitions between configurations in which $S = \{\bullet\}$ (omitted), $\mathcal{C} = \{x_1, x_2, x_3\}$, and $\Gamma = \{a, b, d\}$. From c_1 to c_2, a discrete transition $push(d, [1, 3])$ pushes $(d, 2.6)$ into the stack. At the timed transition from c_2 to c_3, 2.6 time units have elapsed, and each value grows older by 2.6. From c_3 to c_4, the value of x_2 is assigned to 3.8, which lies in the interval $(2, 5]$, and the last transition pops $(d, 5.2)$ after testing that its local age lies in $[4, 6]$.



3 P-automaton

A textbook standard technique to decide the emptiness of a pushdown automaton is, first converting it to context free grammar (with cubic explosion), and then applying CYK algorithm, which is a well-known dynamic programming technique. A practical alternative (with the same complexity) is a P-automaton [12, 10]. Starting from a regular set C of initial configurations (resp.

target configurations) $Post^*$ (resp. Pre^*) saturation procedure is applied on an initial P-automaton (which accepts C) until convergence. The resulting P-automaton accepts the set of all successors (resp. predecessors) of C. In literature, this technique is applied only for PDSs with finite control states and stack alphabet. We confirm that it works for PDSs without finite restriction (ignoring finite convergence), and extend it to the coverability and the quasi-coverability.

3.1 P-automaton for reachability of pushdown system

In the standard definition, a pushdown system (PDS) has a finite set of states and finite stack alphabet. We will consider a PDS with an infinite set of states and infinite stack alphabet. For (possibly infinitely many) individual transition rules, we introduce a partial function ψ to describe a pattern of transitions. We denote the set of partial functions from X to Y by $\mathcal{P}Fun(X, Y)$.

Definition 3. *A* pushdown system *(PDS) $\mathcal{M} = \langle P, \Gamma, \Delta \rangle$ consists of a finite set $\Delta \subseteq \mathcal{P}Fun(P \times \Gamma, P \times \Gamma^2) \cup \mathcal{P}Fun(P \times \Gamma, P \times \Gamma) \cup \mathcal{P}Fun(P \times \Gamma, P)$ of transition rules. We say that $\psi \in \Delta$ is a push, internal, and pop rule if $\psi \in \mathcal{P}Fun(P \times \Gamma, P \times \Gamma^2)$, $\psi \in \mathcal{P}Fun(P \times \Gamma, P \times \Gamma)$, and $\psi \in \mathcal{P}Fun(P \times \Gamma, P)$, respectively. A* configuration *$\langle p, w \rangle$ consists of $p \in P$ and $w \in \Gamma^*$. For a transition rule $\psi \in \Delta$, a transition is $\langle p, \gamma w \rangle \hookrightarrow \langle p', vw \rangle$ for $(p', v) = \psi(p, \gamma)$*

Remark 1. Often in multi-thread program modelings and in snapshot PDSs (Section 7.2) for discretizing DTPDAs, PDSs are defined with finite control states, but with non-standard pop rules, like $\langle p, \gamma_1\gamma_2 \rangle \hookrightarrow \langle q, \gamma \rangle \in \mathcal{P}Fun(P \times \Gamma^2, P \times \Gamma)$ with $|P| < \infty$. This can be encoded into PDSs in Definition 3 by associating a top stack symbol to a state, like $\langle (p, \gamma_1), \gamma_2 \rangle \hookrightarrow \langle (q, \gamma), \epsilon \rangle \in \mathcal{P}Fun(P' \times \Gamma, P')$ with $P' = P \times \Gamma$, at the cost that the set P' of control states becomes infinite.

We use $c_1, c_2, \cdots$ to range over configurations. $\hookrightarrow^*$ is the reflexive transitive closure of $\hookrightarrow$. There are two kinds of reachability problems.

- **Configuration reachability** : Given configurations $\langle p, w \rangle$, $\langle q, v \rangle$ with $p, q \in P$ and $w, v \in \Gamma^*$, decide whether $\langle p, w \rangle \hookrightarrow^* \langle q, v \rangle$.
- **State reachability** : Given a configuration $\langle p, w \rangle$ and a state q with $p, q \in P$ and $w \in \Gamma^*$, decide whether there exists $v \in \Gamma^*$ with $\langle p, w \rangle \hookrightarrow^* \langle q, v \rangle$.

Given a set of configurations C, we write $pre^*(C)$ (resp. $post^*(C)$) for the set $\{c' \mid c' \hookrightarrow^* c \wedge c \in C\}$ (resp. $\{c' \mid c \hookrightarrow^* c' \wedge c \in C\}$). The reachability problem from $\langle p, w \rangle$ to $\langle q, v \rangle$ is reduced to whether $c \in pre^*(\{c'\})$ (or $c' \in post^*(\{c\})$).

Definition 4. *A Pre^*-automaton $\mathcal{A}$ is a quadruplet (S, Γ, ∇, F) with $F \subseteq S$ and $\nabla \subseteq S \times \Gamma \times S$. A Pre^*-automaton is* initial *if each state in $S \cap P$ has no incoming transitions and S is finite. $\mathcal{A}$ accepts a configuration $\langle p, w \rangle$ with $p \in P$ and $w \in \Gamma^*$, if w is accepted starting from p (as an initial state).*

The set of configurations accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$. When $(p, \gamma, q) \in \nabla$, we denote $p \overset{\gamma}{\mapsto} q$. For $w = \gamma_1 \ldots \gamma_k \in \Gamma^*$, $p \overset{\gamma_1}{\mapsto} \cdots \overset{\gamma_k}{\mapsto} q$ is denoted by $p \overset{w}{\mapsto}{}^* q \in \nabla^*$. If $k = 0$ (i.e., $p \overset{\epsilon}{\mapsto} q$), we assume $p = q$.

Starting from an initial Pre^*-automaton $\mathcal{A}_0$ that accepts C (i.e., $C = L(A_0)$), the repeated (possibly infinite) applications of saturation rules

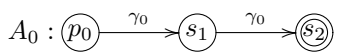
$$\frac{(S, \Gamma, \nabla, F)}{(S \cup \{p'\}, \Gamma, \nabla \cup \{p' \overset{\gamma}{\mapsto} q\}, F)} \quad \text{if } p \overset{w}{\mapsto}{}^* q \in \nabla^* \text{ and } \psi(p', \gamma) = (p, w) \text{ for } \psi \in \Delta$$

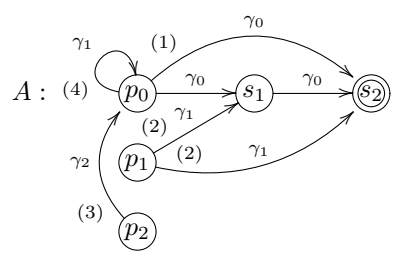
converges to $Pre^*(\mathcal{A}_0)$. Note that saturation rules never eliminate transitions, but monotonically enlarge. When $(p, \gamma, q) \in \nabla$, we denote $p \overset{\gamma}{\mapsto} q$.

Theorem 1. *[12, 10] (Theorem 1 in [6]) For a PDS, $pre^*(C) = L(Pre^*(A_0))$. where $C = L(A_0)$.*

Example 2. Let $\langle \{p_i\}, \{\gamma_i\}, \Delta \rangle$ be a pushdown system with $i = 0, 1, 2$ and Δ given below. The saturation $\mathcal{A}$ of Pre^*-automata started from $\mathcal{A}_0$ accepting $C = \{\langle p_0, \gamma_0\gamma_0 \rangle\}$. $L(\mathcal{A})$ coincides $pre^*(C)$.

(1). $\langle p_0, \gamma_0 \rangle \rightarrow \langle p_1, \gamma_1\gamma_0 \rangle$
(2). $\langle p_1, \gamma_1 \rangle \rightarrow \langle p_2, \gamma_2\gamma_0 \rangle$
(3). $\langle p_2, \gamma_2 \rangle \rightarrow \langle p_0, \gamma_1 \rangle$
(4). $\langle p_0, \gamma_1 \rangle \hookrightarrow \langle p_0, \epsilon \rangle$





Remark 2. Since the saturation procedure monotonically extends Pre^*-automaton, even if a PDS has an infinite set of states / stack alphabet and the initial Pre^*-automaton $\mathcal{A}_0$ has infinite states, it converges (after infinite many saturation steps), and $pre^*(C) = L(Pre^*(A_0))$ holds.

3.2 P-automata for coverability of OPDS

A quasi-ordering (QO) is a reflexive transitive binary relation. We denote the upward (resp. downward) closure of X by $X^{\uparrow}$ (resp. $X^{\downarrow}$), i.e., $X^{\uparrow} = \{y \mid \exists x \in X.x \leq y\}$ (resp. $X^{\downarrow} = \{y \mid \exists x \in X.y \leq x\}$).

For a PDS $\mathcal{M} = \langle P, \Gamma, \Delta \rangle$, we introduce QOs $(P, \preceq)$ and $(\Gamma, \leq)$ on P and Γ, respectively. We call $\mathcal{M} = \langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$ an ordered PDS (OPDS).

Definition 5. *For $w_1 = \alpha_1\alpha_2 \cdots \alpha_n, w_2 = \beta_1\beta_2 \cdots \beta_m \in \Gamma^*$, let*

- **Element-wise comparison** $w_1 \lll w_2$ *if $m = n$ and $\forall i \in [1..n].\alpha_i \leq \beta_i$.*
- **Embedding** $w_1 \preccurlyeq w_2$ *if there is an order-preserving injection f from $[0..n]$ to $[0..m]$ with $\alpha_i \leq \beta_{f(i)}$ for each $i \in [0..n]$.*

We extend $\lll$ on configurations such that $(p, w) \lll (q, v)$ if $p \preceq q$ and $w \lll v$.

A partial function $\psi \in \mathcal{P}Fun(X, Y)$ is *monotonic* if $\gamma \leq \gamma'$ and $\gamma \in dom(\psi)$ imply $\psi(\gamma) \lll \psi(\gamma')$ and $\gamma' \in dom(\psi)$ for each $\gamma, \gamma' \in \Gamma$. We say that an OPDS $\mathcal{M} = \langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$ is *monotonic* if ψ is monotonic for each $\psi \in \Delta$.

– **Coverability** : Given configurations (p, w), (q, v) with $p, q \in P$ and $w, v \in \Gamma^*$, decide whether there exists $v' \in \Gamma^*$ with $v \lll v'$ and $(p, w) \hookrightarrow^* (q, v')$.

Coverability is reduced to whether $(p, w) \in pre^*(\{(q, v)\}^\uparrow)$. For coverability, we restrict saturation rules of Pre^*-automata.

$$\frac{(S, \Gamma, \nabla, F)}{(S \cup \{p'\}, \Gamma, \nabla \oplus \{p' \overset{\gamma}{\mapsto} q\}, F)} \quad \begin{array}{l} \text{if } p \overset{w}{\mapsto}{}^* q \in \nabla^* \text{ and} \\ \psi(p', \gamma) \in \{(p, w)\}^\uparrow \text{ for } \psi \in \Delta \end{array}$$

where $\nabla \oplus \{p' \overset{\gamma}{\mapsto} q\}$ is

$$\begin{cases} \nabla & \text{if there exists } \{p'' \overset{\gamma'}{\mapsto} q\} \in \nabla \text{ with } p'' \preceq p' \text{ and } \gamma' \leq \gamma \\ \nabla \cup \{p' \overset{\gamma}{\mapsto} q\} & \text{otherwise.} \end{cases}$$

Theorem 2. *(Theorem 3 in [6]) For a monotonic OPDS, $pre^*(C^\uparrow) = L(Pre^*(A_0))^\uparrow$. where $C^\uparrow = L(A_0)$.*

3.3 P-automata for quasi-coverability of OPDS

– **Quasi-coverability**. Given configurations $\langle p, w\rangle$, $\langle q, v\rangle$, decide whether there exist $\langle p', w'\rangle$ and $\langle q', v'\rangle$ such that $\langle p, w\rangle \lll \langle p', w'\rangle$, $\langle q, v\rangle \lll \langle q', v'\rangle$, and $\langle p', w'\rangle \hookrightarrow^* \langle q', v'\rangle$.

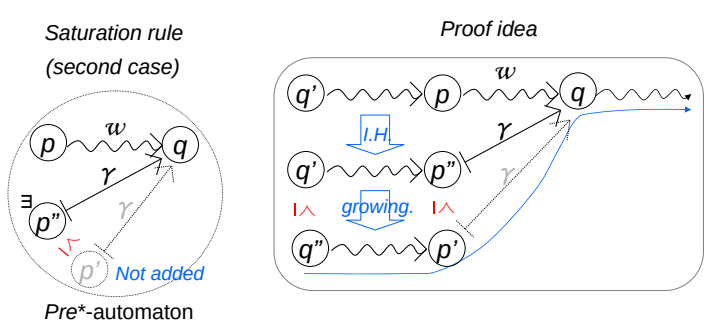
Quasi-coverability is reduced to whether $\langle p, w\rangle \in pre^*(\{(q, v)\}^\uparrow)^\downarrow$. For quasi-coverability, we further restrict saturation rules of Pre^*-automata.

$$\frac{(S, \Gamma, \nabla, F)}{(S \cup \{p'\}, \Gamma, \nabla \oplus \{p' \overset{\gamma}{\mapsto} q\}, F)} \quad \begin{array}{l} \text{if } p \overset{w}{\mapsto}{}^* q \in \nabla^* \text{ and} \\ \psi(p', \gamma) \in \{(p, w)\}^\uparrow \text{ for } \psi \in \Delta \end{array}$$

where $\nabla \oplus \{p' \overset{\gamma}{\mapsto} q\}$ is

$$\begin{cases} \nabla & \text{if there exists } \{p'' \overset{\gamma'}{\mapsto} q\} \in \nabla \text{ with } p'' \preceq p' \text{ and } \gamma' \leq \gamma \\ \nabla \cup \{p'' \overset{\gamma}{\mapsto} q\} & \text{if there exists } p'' \in S \cap P \text{ with } p'' \preceq p' \\ \nabla \cup \{p' \overset{\gamma}{\mapsto} q\} & \text{otherwise.} \end{cases}$$

The second condition (illustrated in the figure below) suppresses adding new states in Pre^*-automata, and the first condition gives a termination condition for adding new edges.



Definition 6. *An OPDS* $\mathcal{M} = \langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$ *is* growing *if, for each* $\psi(p, \gamma) = (q, w)$ *with* $\psi \in \Delta$ *and* $(q', w') \geqq (q, w)$*, there exists* (p', γ') *with* $(p', \gamma') \geqq (p, \gamma)$ *such that* $\psi(p', \gamma') \geqq (q', w')$*.*

Lemma 1 is obtained by induction on steps of Pre^*-automata saturation, of which the proof idea is illustrated in the figure above.

Lemma 1. *Assume* $p \overset{w}{\longmapsto}^* s$ *in* $Pre^*(\mathcal{A}_0)$*. For each* $(p', w') \geqq (p, w)$*,*

- *If* $s \in P$*, there exist* $(p'', w'') \geqq (p', w')$ *and* $q' \succeq s$ *with* $\langle p'', w'' \rangle \hookrightarrow^* \langle q', \epsilon \rangle$*.*
- *If* $s \in S \setminus P$*, there exist* $(p'', w'') \geqq (p', w')$*,* $q \overset{v}{\longmapsto}^* s$ *in* $\mathcal{A}_0$ *with* $q \in P$*, and* $\langle q', v' \rangle \geqq \langle q, v \rangle$ *such that* $\langle p'', w'' \rangle \hookrightarrow^* \langle q', v' \rangle$*.*

For simplicity, we say "c_0 covers c_1" to mean that there exists $c_1' \geqq c_1$ with $c_0 \hookrightarrow^* c_1'$. The next **Claim** is easily proved by induction on the steps of $\hookrightarrow$.

Claim For a monotonic and growing OPDS, if $\langle p, w \rangle \hookrightarrow^* \langle q, v \rangle$, then for any $(q', v') \geqq (q, v)$, there exists $(p', w') \geqq (p, w)$ such that $\langle p', w' \rangle$ covers $\langle q', v' \rangle$.

Proof. By induction on steps of the Pre^* saturation procedure $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \cdots$. For $\mathcal{A}_0$, the statements hold immediately. Assume the statements hold for $\mathcal{A}_i$, and $\mathcal{A}_{i+1}$ is constructed by adding new transition $p_0 \overset{\gamma_0}{\mapsto} q_0$.

$$\frac{(S, \Gamma, \nabla, F)}{(S \cup \{p_0\}, \Gamma, \nabla \oplus \{p_0 \overset{\gamma_0}{\mapsto} q_0\}, F)} \quad \begin{array}{l} \text{if } p_1 \overset{w_1}{\mapsto}^* q_0 \in \nabla^* \text{ and} \\ \psi(p_0, \gamma_0) \in \{(p_1, w_1)\}^{\uparrow} \text{ for } \psi \in \Delta \end{array}$$

We give a proof only for the first statement. The second statement is similarly proved. According to the definition of $\oplus$, there are three cases:

- There exists $\{p_0' \overset{\gamma_0'}{\mapsto} q_0\} \in \nabla$ with $p_0' \preceq p_0$ and $\gamma_0' \leq \gamma_0$. Nothing added.
- There exists p_0' in $S \cap P$ and $p_0' \preceq p_0$. Then, $p_0' \overset{\gamma_0}{\mapsto} q_0$ is added.
- Otherwise. $p_0 \overset{\gamma_0}{\mapsto} q_0$ is added.

The second case is the most complex, and we focus on it. Assume that a path $p \overset{w}{\longmapsto}^* q$ contains $p_0' \overset{\gamma_0}{\mapsto} q_0$ k-times. We apply (nested) induction on k, and we focus on its leftmost occurrence. Let $w = w_l \gamma_0 w_r$ and $p \overset{w_l}{\longmapsto}^* p_0' \overset{\gamma_0}{\mapsto} q_0 \overset{w_r}{\longmapsto}^* q$. For each $p' \succeq p, w_l' \geqq w_l, w_r' \geqq w_r$ and $\gamma_0' \geq \gamma_0$:

1. By induction hypothesis on $p \overset{w_l}{\longmapsto}^* p_0'$, there exists $(p'', w_l'') \geqq (p', w_l')$ such that $\langle p'', w_l'' \rangle$ covers $\langle p_0', \epsilon \rangle$.
2. By the definition of saturation rules, there exist $p_1' \succeq p_1$ and $w_1' \geqq w_1$ such that $\langle p_0, \gamma_0 \rangle \hookrightarrow \langle p_1', w_1' \rangle$.
3. By induction hypothesis on $p_1 \overset{w_1 w_r}{\longmapsto}^* q$, there exist $p_1'' \succeq p_1'$ and $w_1'' w_r'' \geqq w_1' w_r'$ such that $\langle p_1'', w_1'' w_r'' \rangle$ covers $\langle q, \epsilon \rangle$.
4. By the growing property, there exist $p_0'' \succeq p_0 \succeq p_0'$ and $\gamma_0'' \geq \gamma_0'$ such that $\langle p_0'', \gamma_0'' \rangle$ covers $\langle p_1'', w_1'' \rangle$.

By **Claim** and 1., there exists $(p''', w_l''') \gg (p'', w_l'') \gg (p', w_l')$ such that $\langle p''', w_l''' \rangle$ covers $\langle p_0'', \epsilon \rangle$. Put all these together, for each $(p', w_l'\gamma_0'w_r') \gg (p, w_l\gamma_0 w_r)$, there exists $(p''', w_l'''\gamma_0''w_r'') \gg (p', w_l'\gamma_0'w_r')$. Therefore, each of $\langle p''', w_l'''\gamma_0''w_r'' \rangle$, $\langle p_0'', \gamma_0''w_r'' \rangle$, $\langle p_1'', w_1''w_r'' \rangle$, and $\langle q, \epsilon \rangle$ covers the next. □

From Lemma 1, Theorem 3 is immediate.

Theorem 3. *For a monotonic and growing OPDS, $pre^*(C^\uparrow)^\downarrow = (L(Pre^*(A_0))^\uparrow)^\downarrow$. where $C^\uparrow = L(A_0)$.*

4 Well-formed projection and well-formed constraint

Definition 7. *For an OPDS M, a pair $(\Upsilon, \Downarrow_\Upsilon)$ of a set $\Upsilon \subseteq P \times \Gamma^*$ and a projection function $\Downarrow_\Upsilon$: $P \times \Gamma^* \to (P \times \Gamma^*) \cup \{\#\}$ is a* well-formed projection *if, for configurations c, c' with $c \hookrightarrow c'$,*

- *$c \in \Upsilon$ if, and only if $c' \in \Upsilon$,*
- *$\Downarrow_\Upsilon (c) \hookrightarrow \Downarrow_\Upsilon (c')$,*
- *$\Downarrow_\Upsilon (c) \ll c$, and*
- *$c_1 \ll c_2$ implies either $\Downarrow_\Upsilon (c_1) = \Downarrow_\Upsilon (c_2)$ or $\Downarrow_\Upsilon (c_1) = \#$,*

where $\#$ is added to $P \times \Gamma^*$ as the least element (wrt $\ll$) and $\Upsilon = \{c \in P \times \Gamma^* \mid c = \Downarrow_\Upsilon (c)\}$. Υ is called a well-formed constraint*. ($\#$ represents failures of $\Downarrow_\Upsilon$.)*

Lemma 2. *For a monotonic OPDS M with a well-formed projection $\Downarrow_\Upsilon$, assume $C \subseteq \Upsilon$. Then, $pre^*(C) = pre^*(C^\uparrow)^\downarrow \cap \Upsilon$.*

Proof. From $C \subseteq \Upsilon$, $pre^*(C) \subseteq pre^*(C^\uparrow)^\downarrow \cap \Upsilon$ is obvious, For the opposite direction, we first show $\Downarrow_\Upsilon (pre^*(C^\uparrow)) \subseteq pre^*(C)$. Since $c \in pre^*(C^\uparrow)$ is equivalent to $\exists c' \in C^\uparrow . c \hookrightarrow^* c'$, we have $\Downarrow_\Upsilon (c) \hookrightarrow^* \Downarrow_\Upsilon (c') \in C$. Since $C \subseteq \Upsilon$ implies $\Downarrow_\Upsilon (c') \in C$, $\Downarrow_\Upsilon (c) \in pre^*(C)$ is obtained. For $pre^*(C) \supseteq pre^*(C^\uparrow)^\downarrow \cap \Upsilon$,

$$pre^*(C^\uparrow)^\downarrow \cap \Upsilon = \Downarrow_\Upsilon (pre^*(C^\uparrow)^\downarrow \cap \Upsilon) \subseteq \Downarrow_\Upsilon (pre^*(C^\uparrow)^\downarrow) = \Downarrow_\Upsilon (pre^*(C^\uparrow)) \cup \{\#\}.$$

From $\Downarrow_\Upsilon (pre^*(C^\uparrow)) \subseteq pre^*(C)$, $\Downarrow_\Upsilon (pre^*(C^\uparrow)) \cup \{\#\} \subseteq pre^*(C) \cup \{\#\}$. Thus, $pre^*(C^\uparrow)^\downarrow \cap \Upsilon \subseteq (pre^*(C) \cup \{\#\}) \cap \Upsilon = pre^*(C)$. ■

From Theorem 3 and Lemma 2, Theorem 4 is immediate, which strengthens the quasi-coverability to the configuration reachability, and the decidability is reduced to finite convergence of $L(Pre^*(\mathcal{A}_0))$.

Theorem 4. *Let C be a regular set of configurations with a P-automaton $\mathcal{A}_0$) with $C^\uparrow = L(A_0)$. For a monotonic and growing OPDS and a well-formed constraint Υ, $pre^*(C) = L(Pre^*(A_0))^\downarrow \cap \Upsilon$.*

Example 3. In Example 4, let Υ be

$$\left\{ \begin{array}{l} \langle p_0, (n,n) \cdots (0,0) \rangle, \langle p_2, (n,n) \cdots (0,0) \rangle \\ \langle p_1, (n, n-2)(n-1, n-1) \cdots (0,0) \rangle, \end{array} \mid n \geq m \geq 0 \right\}$$

It is easy to see that Υ is compatible. Since both $\langle p_0, (0,0) \rangle$ and $\langle p_2, (0,0) \rangle$ are in Υ and $\{\langle p, (0,0) \rangle\}^\uparrow \cap \Upsilon = \{\langle p, (0,0) \rangle\}$, we conclude that $\langle p_0, (0,0) \rangle \hookrightarrow^* \langle p_2, (0,0) \rangle$ by Theorem 4.

5 Finite convergence of Pre^*-automata

Definition 8. *A QO $\leq$ is a* well-quasi-ordering *(WQO) if, for each infinite sequence $a_1, a_2, \cdots$, there exist i, j with $i < j$ and $a_i \leq a_j$.*

A QO $\leq$ is a WQO, if, and only if each upward closed set $X^{\uparrow}$ has finite basis (i.e., minimal elements). Note that $\leqslant$ may be no longer a WQO (nor well founded), while the embedding $(\Gamma^*, \preccurlyeq)$ stays a WQO by *Higman's lemma.*

Lemma 3. *Let $(D, \leq)$ and $(D', \leq')$ be WQOs.*

- **(Dickson's lemma)** *$(D \times D', \leq \times \leq')$ is a WQO.*
- **(Higman's lemma)** *$(D^*, \preccurlyeq)$ is a WQO, where $\preccurlyeq$ is the embedding.*

For a monotonic OPDS, if $(P, \preceq), (\Gamma, \leq)$ are WQOs, we call it *a Well-Structured PDS* (WSPDS). For a WSPDS $((P, \preceq), (\Gamma, \leq), \Delta)$, $\psi^{-1}(\{(p, w)\}^{\uparrow})$ is upward-closed and has finite basis (i.e., finitely many minimal elements). In the Pre^* saturation rule of Section 3.3, its side condition contains $\psi(p', \gamma) \in \{(p, w)\}^{\uparrow}$ for $\psi \in \Delta$, which allows arbitrary choices of (p', γ). For a WSPDS, we focus only on finite basis of upward-closed sets $(p', \gamma) \in Min(\psi^{-1}(\{(p, w)\}^{\uparrow}))$.

We assume that such finite basis are computable for each $\psi \in \Delta$, and the initial Pre^*-automaton has finitely many states S_0.

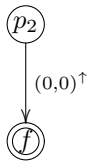
Theorem 5. *For a WSPDS $((P, \preceq), (\Gamma, \leq), \Delta)$, if (i) $(P, \preceq), (\Gamma, \leq)$ are computable WQOs, and (ii) a finite basis of $\psi^{-1}(\{(p, w)\}^{\uparrow})$ is computable for each $\psi \in \Delta$ and $\langle p, w\rangle \in P \times \Gamma^{\leq 2}$, $Pre^*(\mathcal{A}_0)$ finitely converges.*

Proof. (*Sketch*) Starting from a WQO over S such that $\preceq$ over $S_0 \cap P$ and $=$ on $S_0 \setminus P$, the set S of states of the Pre^*-automaton make a bad sequence, since saturation rules in Section 3.3 do not add larger states. For each pair (p, q) of states, they also do not add larger stack symbols as labels of Pre^* automaton transitions $p \overset{\gamma}{\mapsto} q$. Thus, during the saturation procedure, a sequence of added edges $p_1 \overset{\gamma_1}{\mapsto} q_1, p_2 \overset{\gamma_2}{\mapsto} q_2, \cdots$ is bad. Thus, it finitely terminates. Since Δ has finitely many transition rules (i.e., partial functions), dependency during generation of Pre^* automaton transitions is finitely branching. Thus, by König's lemma, $Pre^*(\mathcal{A}_0)$ finitely converges. $\square$

Example 4. Let $M = \langle \{p_i\}, \mathbb{N}^2, \Delta\rangle$ be a monotonic OPDS with vectors in $\mathbb{N}^2$ as a stack alphabet and Δ consists of four rules given in the figure. The figure illustrates a Pre^*-automaton construction starting from initial $\mathcal{A}_0$ that accepts $C = \langle p_2, (0, 0)^{\uparrow}\rangle$. For $v \in \mathbb{N}^2$, we abbreviate $\{v\}^{\uparrow}$ by $v^{\uparrow}$. Note that $\mathbb{N}^2$ is WQO by the element-wise comparison. $\mathcal{A}$ is the saturation of the Pre^*-automaton.

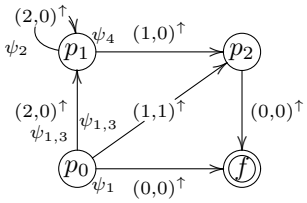
For instance, when $m = 2$, $p_0 \overset{(2,2)^{\uparrow}}{\mapsto} p_1$ in $\mathcal{A}$ is generated from $p_1 \overset{(2,0)^{\uparrow}}{\mapsto} p_1$ by ψ_3. By repeating application of ψ_1 twice to $p_0 \overset{(2,2)^{\uparrow}}{\mapsto} p_1 \overset{(2,0)^{\uparrow}}{\mapsto} p_1$, we obtain $p_0 \overset{(2,0)^{\uparrow}}{\mapsto} p_1$. Then, applying ψ_1 to $p_0 \overset{(2,0)^{\uparrow}}{\mapsto} p_1 \overset{(1,0)^{\uparrow}}{\mapsto} p_2$, we obtain $p_0 \overset{(1,0)^{\uparrow}}{\mapsto} p_2$. $p_0 \overset{(1,2)^{\uparrow}}{\mapsto} p_2$ is also generated from $p_1 \overset{(1,0)^{\uparrow}}{\mapsto} p_2$ by ψ_3 (since $\psi_3^{-1}(\{(1, 0)\}^{\uparrow}) = \{(1, 2)\}^{\uparrow}$), but it will not affect.

$\mathcal{A}_0$:

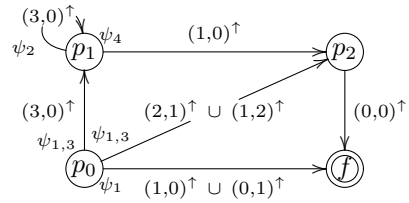


$\psi_1 : \langle p_0, v\rangle \to \langle p_0, (v+(1,1))v\rangle$		
$\psi_2 : \langle p_1, v\rangle \to \langle p_1, \epsilon\rangle$	if $v \geq (m,0)$	
$\psi_3 : \langle p_0, v\rangle \to \langle p_1, v-(0,2)\rangle$	if $v \geq (0,2)$	
$\psi_4 : \langle p_1, v\rangle \to \langle p_2, \epsilon\rangle$	if $v \geq (1,0)$	

$\mathcal{A}$:
$(m=2)$



$\mathcal{A}'$:
$(m=3)$



By Theorem 2, we obtain

$$pre^*(C) = \{\langle p_2, (0,0)^\uparrow\rangle, \langle p_1, ((2,0)^\uparrow)^*(1,0)^\uparrow(0,0)^\uparrow\rangle, \\ \langle p_0, (0,0)^\uparrow\rangle, \langle p_0, (1,1)^\uparrow(0,0)^\uparrow\rangle, \langle p_0, ((2,0)^\uparrow)^+(1,0)^\uparrow(0,0)^\uparrow\rangle\}$$

Thus, $\langle p_0, (0,0)\rangle$ covers $\langle p_2, (0,0)\rangle$. Actually,

$$\langle p_0, (0,0))\rangle \hookrightarrow \langle p_0, (1,1)(0,0)\rangle \hookrightarrow \langle p_0, (2,2)(1,1)(0,0)\rangle \hookrightarrow \langle p_1, (2,0)(1,1)(0,0)\rangle \\ \hookrightarrow \langle p_1, (1,1)(0,0)\rangle \hookrightarrow \langle p_2, (0,0)\rangle$$

Note that if we change the condition of ψ_2 from $v \geq (2,0)$ to $v \geq (3,0)$, the saturated Pre^*-automaton becomes $\mathcal{A}'$, and $\langle p_0, (0,0)\rangle$ no more covers $\langle p_2, (0,0)\rangle$, though $\langle p_0, (0,0)\rangle$ is reachable to p_2. Actually,

$$\langle p_0, (0,0))\rangle \hookrightarrow \langle p_0, (1,1)(0,0)\rangle \hookrightarrow \langle p_0, (2,2)(1,1)(0,0)\rangle \hookrightarrow \langle p_0, (3,3)(2,2)(1,1)(0,0)\rangle \\ \hookrightarrow \langle p_1, (3,1)(2,2)(1,1)(0,0)\rangle \hookrightarrow \langle p_1, (2,2)(1,1)(0,0)\rangle \hookrightarrow \langle p_2, (1,1)(0,0)\rangle$$

To detect the state reachability, instead of $\mathcal{A}_0$, we can start with an initial automaton $\mathcal{A}'_0$ that accepts $p_2 \times \Gamma^* = \{\langle p_2, ((0,0)^\uparrow)^*\}$.

6 Snapshot Word

In a DTPDA, the stack content is a sequence of pairs of stack symbols and local ages. When a DTPDA is encoded into a discrete WSPDS, it can operate only the top stack symbol. Such a target WSPDS is a *snapshot PDS* (Section 7.2), of which stack symbols are snapshot words. A *snapshot word* summarizes the ordering of fractions of all local ages and global clocks in the stack, after applying the digitization technique in [15], whereas the encoding in [1] summarizes global clocks and an age in the top stack frame and copies of global clocks in the next stack frame. Then, a snapshot PDS handles all timed behavior at the top stack symbol, and left untouched inside the stack. When a pop occurs, time progress recorded at the top stack symbol is propagated to the next stack symbol after finding a permutation (of time progress) by matching via markings ρ_1 and ρ_2.

6.1 Snapshot word

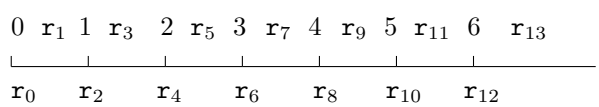
As notational convention, let $\mathcal{MP}(D)$ be the set of finite multisets over D. We regard a finite set as a multiset in which the multiplicity of each element is 1. For a finite word $w = a_1 a_2 \cdots a_k$, we denote $w(j) = a_j$

Let $\langle S, s_{init}, \Gamma, \mathcal{C}, \Delta \rangle$ be a DTPDA, and let n be the largest integer (except for ∞) that appears in Δ. For $v \in \mathbb{R}_{\geq 0}$, $proj(v) = \mathtt{r}_i$ if $v \in \mathtt{r}_i \in Intv(n)$ and

$$Intv(n) = \begin{cases} \mathtt{r}_{2i} & = [i, i] \quad \text{if } 0 \leq i \leq n \\ \mathtt{r}_{2i+1} & = (i, i+1) \text{ if } 0 \leq i < n \\ \mathtt{r}_{2n+1} & = (n, \infty) \end{cases}$$

Definition 9. *Let $frac(x,t) = t - floor(t)$ for $(x,t) \in (\mathcal{C} \cup \Gamma) \times \mathbb{R}_{\geq 0}$. A* digitization $\mathtt{digi} : \mathcal{MP}((\mathcal{C} \cup \Gamma) \times \mathbb{R}_{\geq 0}) \to (\mathcal{MP}((\mathcal{C} \cup \Gamma) \times Intv(n)))^*$ *is as follows. For $\mathcal{X} \in \mathcal{MP}((\mathcal{C} \cup \Gamma) \times \mathbb{R}_{\geq 0})$, let $X_1, \cdots, X_k$ be multisets that collect $(x, proj(t))$'s in $\mathcal{X}$ having the same $frac(x,t)$. We assume that X_i's are sorted by the increasing order of $frac(x,t)$ (i.e., , $frac(x,t) < frac(x',t')$ for $(x, proj(t)) \in X_i$ and $(x', proj(t')) \in X_{i+1}$). Then, $\mathtt{digi}(\mathcal{X})$ is a word $X_1 \cdots X_k$.*

Example 5. In Example 1, $n = 6$ and we have 13 intervals illustrated below.



From the configuration c_1 in Example 1, the clock information is extracted from the stack content of c_1 as a multiset

$$\mathcal{X} = \{(a, 1.9), (b, 6.7), (a, 3.1), (d, 4.2), (x_1, 0.5), (x_2, 3.9), (x_3, 2.3)\}$$

and $\mathtt{digi}(\mathcal{X}) = \{(a, \mathtt{r}_7)\}\{(d, \mathtt{r}_9)\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{(b, \mathtt{r}_{13})\}\{(x_2, \mathtt{r}_7), (a, \mathtt{r}_3)\}$. For instance, The value of the clock x_2 and the age of the top stack frame $(a, 1.9)$ have the same fraction 0.9, thus they are packed into the same multiset $\{(x_2, \mathtt{r}_7), (a, \mathtt{r}_3)\}$, and placed at the last since their fraction is the largest.

Definition 10. *A word $\bar{\gamma} \in (\mathcal{MP}((\mathcal{C} \cup \Gamma) \times Intv(n)))^*$ is a* snapshot word *if it has two pointers ρ_1, ρ_2 such that $\rho_1(\bar{\gamma}), \rho_2(\bar{\gamma})$ point to different elements of $\Gamma \times Intv(n)$ appearing in $\bar{\gamma}$. We denote the set of snapshot word by $sw(\mathcal{C}, \Gamma, n)$, and $\bar{\gamma}|_\Gamma$ is obtained by removing all elements in $\mathcal{C} \times Intv(n)$ from $\bar{\gamma}$.*

Example 6. From $\mathtt{digi}(\mathcal{X})$ in Example 5, by adding ρ_1 and ρ_2 (marked with *overline* and *underline*), which point to $(a, \mathtt{r}_3)$ and $(b, \mathtt{r}_{13})$, respectively, we have

$$\{(a, \mathtt{r}_7)\}\{(d, \mathtt{r}_9)\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{\underline{(b, \mathtt{r}_{13})}\}\{(x_2, \mathtt{r}_7), \overline{(a, \mathtt{r}_3)}\}$$

and $\mathtt{digi}(\mathcal{X})|_\Gamma = \{(a, \mathtt{r}_7)\}\{(d, \mathtt{r}_9)\}\{\underline{(b, \mathtt{r}_{13})}\}\{\overline{(a, \mathtt{r}_3)}\}$.

Definition 11. *For snapshot words $\bar{\gamma} = X_1 \cdots X_m$ and $\bar{\gamma}' = Y_1 \cdots Y_n$ with $X_i, Y_j \in \mathcal{MP}((\mathcal{C} \cup \Gamma) \times Intv(n))$, we define the embedding $\bar{\gamma} \sqsubseteq \bar{\gamma}'$, if there exists a monotonic injection $f : [1..m] \to [1..n]$ such that*

– $X_k \subseteq Y_{f(k)}$ *for each* $k \in [1..m]$,
– $\rho_i(\bar{\gamma}) \in X_j$ *implies* $\rho_i(\bar{\gamma}') \in Y_{f(j)}$ *for* $i = 1, 2$ *and* $j \in [1..m]$, *and*
– $\rho_i(\bar{\gamma}) = \rho_i(\bar{\gamma}')$ *for* $i = 1, 2$.

Since Γ and $\mathcal{C}$ are finite, $\sqsubseteq$ is a WQO over $sw(\mathcal{C}, \Gamma, n)$ by Higman's lemma.

Definition 12. *Let* $c = (s, \nu, w)$ *be a configuration of a DTPDA with* $s \in S$, $w \in (\Gamma \times \mathbb{R}_{\geq 0})^*$, *and* $\nu : \mathcal{C} \to \mathbb{R}_{\geq 0}$, *and let* $\mathtt{mp}(w, \nu) = w \cup \{(x, \nu(x)) \mid x \in C\}$ *by regarding* w *as a multiset (i.e., ignore the ordering).* $snap(c)$ *is a snapshot word obtained by adding* ρ_1, ρ_2 *to* $\mathtt{digi}(\mathtt{mp}(w, \nu))$ *as:*

$$\begin{cases} \rho_1,\ \rho_2 \text{ are left undefined} & \text{if } w = \epsilon \\ \rho_1(snap(c)) = (\gamma, proj(t)),\ \rho_2 \text{ is left undefined} & \text{if } w = (\gamma, t) \\ \rho_1(snap(c)) = (\gamma, proj(t)),\ \rho_2(snap(c)) = \rho_1(snap((s, \nu, w'))) & \text{if } w = (\gamma, t)w' \end{cases}$$

Example 7. For c_2 in Example 1, $snap(c_1)$ is $\mathtt{digi}(\mathcal{X})$ (with ρ_1 and ρ_2) in Example 6. ρ_1 and ρ_2 point to the top and second stack frames $(a, 1.9), (b, 6.7)$.

Definition 13. *For a configuration* $c = (s, \nu, w)$ *of a DTPDA, a snapshot configuration* $Snap(c) = (s, \tilde{w})$ *with stack alphabet* $sw(\mathcal{C}, \Gamma, n)^*$ *is with*

$$\tilde{w} = snap(s, \nu, w[m])\ snap(s, \nu, w[m-1])\ \cdots\ snap(s, \nu, w[1])\ snap(s, \nu, \epsilon)$$

where $w = (a_m, t_m) \cdots (a_1, t_1) \in (\Gamma \times \mathbb{R}_{\geq 0})^*$ *and* $w[i] = (a_i, t_i) \cdots (a_1, t_1)$.

Example 8. For c_1 in Example 1 (with $\nu(x_1) = 0.5, \nu(x_2) = 3.9, \nu(x_3) = 2.3$), $Snap(c_1)$ is shown below. The top snapshot word in the stack summarizes a current time sequence of values of all clocks and ages.

$(a, 1.9)$		$\{(a, \mathtt{r}_7)\}\{(d, \mathtt{r}_9)\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{\underline{(b, \mathtt{r}_{13})}\}\{(x_2, \mathtt{r}_7), \overline{(a, \mathtt{r}_3)}\}$
$(b, 6.7)$		$\{\underline{(a, \mathtt{r}_7)}\}\{(d, \mathtt{r}_9)\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{\overline{(b, \mathtt{r}_{13})}\}\{(x_2, \mathtt{r}_7)\}$
$(a, 3.1)$	$\Rightarrow$	$\{\overline{(a, \mathtt{r}_7)}\}\{\underline{(d, \mathtt{r}_9)}\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{(x_2, \mathtt{r}_7)\}$
$(d, 4.2)$		$\{\overline{(d, \mathtt{r}_9)}\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{(x_2, \mathtt{r}_7)\}$
$\perp$		$\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{(x_2, \mathtt{r}_7)\}$
Stack of c_1		Stack of $Snap(c_1)$

6.2 Operations on snapshot words

Definition 14. *Let* $\bar{\gamma} = X_1 \cdots X_m \in (\mathcal{MP}((\mathcal{C} \cup \Gamma) \times Intv(n)))^*$ *be a snapshot word and let* $\gamma \in \Gamma \cup \mathcal{C}$. *We define operations as follows.*

– **Insert** $\bar{\gamma}' = insert(\bar{\gamma}, (\delta, \mathtt{r}_k))$ *is obtained from* $\bar{\gamma}$ *by inserting* $(\delta, \mathtt{r}_k)$

$$\begin{cases} \text{either into } X_j, \text{ or between } X_j \text{ and } X_{j+1} \text{ for some } j \in [0..m] & \text{if } k \text{ is odd} \\ \text{into } X_1, \text{ if each } \mathtt{r}_i \text{ in } X_1 \text{ has an even index; before } X_1, \text{ o.w.} & \text{if } k \text{ is even} \end{cases}$$

and setting $\rho_1(\bar{\gamma}') = (\delta, \mathtt{r}_k)$ *and* $\rho_2(\bar{\gamma}') = \rho_1(\bar{\gamma})$.

- **Delete**$_\Gamma$ $\bar{\gamma}' = delete_\Gamma(\bar{\gamma})$ *is obtained from* $\bar{\gamma}$ *by deleting* $\rho_1(\bar{\gamma})$ *and setting* $\rho_1(\bar{\gamma}') = \rho_2(\bar{\gamma})$ *and* $\rho_2(\bar{\gamma}')$ *left* undefined.
- **Delete**$_\mathcal{C}$ *For* $x \in \mathcal{C}$, $delete_\mathcal{C}(\bar{\gamma}, x)$ *is obtained from* $\bar{\gamma}$ *by deleting* $(x, \mathtt{r})$ *(and* ρ_1, ρ_2 *are kept unchanged).*
- **Assignment** *For* $x \in \mathcal{C}$, $\mathtt{r} \in Intv(n)$, $assign(\bar{\gamma}, x, \mathtt{r}) = insert(delete_C(\bar{\gamma}, x), (x, \mathtt{r}))$.
- **Permutation** *Let* $i \in [1..m]$ *and* $0 \leq k \leq n$. *Permutation* $\sigma(\bar{\gamma})$ *is either* $\dot{\sigma}_{i,k}(\bar{\gamma})$ *or* $\ddot{\sigma}_{i,k}(\bar{\gamma})$, *defined by*

$$\begin{cases} \dot{\sigma}_{i,k}(\bar{\gamma}) = (X_i \dot{+} 2k+2)(X_{i+1} \dot{+} 2k+2) \cdots (X_m \dot{+} 2k+2)(X_1 \dot{+} 2k) \cdots (X_{i-1} \dot{+} 2k) \\ \ddot{\sigma}_{i,k}(\bar{\gamma}) = (X_i \ddot{+} 2k+2)(X_{i+1} \dot{+} 2k+2) \cdots (X_m \dot{+} 2k+2)(X_1 \dot{+} 2k) \cdots (X_{i-1} \dot{+} 2k) \end{cases}$$

  where, for $y \in \mathcal{C} \cup \Gamma$, $X_i \dot{+} j$ *updates each* $(y, \mathtt{r}_l) \in X_i$ *with* $(y, \mathtt{r}_{min(l+j,2n+1)})$ *if* l *is odd, and* $(y, \mathtt{r}_{min(l+j+1,2n+1)})$ *if* l *is even.* $X_i \ddot{+} j$ *updates each* $(y, \mathtt{r}_l) \in X_i$ *with* $(y, \mathtt{r}_{min(l+j,2n+1)})$ *if* $i = 1$ *and* l *is even; with* $(y, \mathtt{r}_{min(l+j-1,2n+1)})$, *otherwise.*
- **Propagate** $propagate(\bar{\gamma}, \bar{\gamma}')$ *is obtained from* $delete_\Gamma(\bar{\gamma})$ *by assigning* $\sigma(\rho_2(\bar{\gamma}'))$ *to* $\rho_2(delete_\Gamma(\bar{\gamma}))$ *for a permutation* σ *with* $\bar{\gamma}|_\Gamma = \sigma(\bar{\gamma}')|_\Gamma$.

Example 9. Consider $snap(c_i)$ in Example 7 for c_1 in Example 1.

$$\{(a, \mathtt{r}_7)\}\{(d, \mathtt{r}_9)\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{\underline{(b, \mathtt{r}_{13})}\}\{(x_2, \mathtt{r}_7), \overline{(a, \mathtt{r}_3)}\}$$

- $insert(snap(c_1), (d, \mathtt{r}_5))$ has lots of choices, e.g.,
  $\{(a, \mathtt{r}_7)\}\{(d, \mathtt{r}_9)\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1), \overline{(d, \mathtt{r}_5)}\}\{(b, \mathtt{r}_{13})\}\{(x_2, \mathtt{r}_7), \underline{(a, \mathtt{r}_3)}\}$,
  $\{(a, \mathtt{r}_7)\}\{(d, \mathtt{r}_9)\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}, \{\overline{(d, \mathtt{r}_5)}\}, \{(b, \mathtt{r}_{13})\}\{(x_2, \mathtt{r}_7), \underline{(a, \mathtt{r}_3)}\}, \cdots$
  The transition from c_1 to c_2 in Example 1 is simulated by pushing the second one (say, $\bar{\gamma}_2$) to $Snap(c_1)$ in Example 8.
- For $c_2 \xrightarrow{2.6}_{Time} c_3$, the permutation $\dot{\sigma}_{4,2}(\bar{\gamma}_2)$ results in $\bar{\gamma}_3$ below.
  $\{(x_1, \mathtt{r}_7)\}, \{\overline{(d, \mathtt{r}_{11})}\}, \{(b, \mathtt{r}_{19})\}\{(x_2, \mathtt{r}_{13}), \underline{(a, \mathtt{r}_9)}\}\{(a, \mathtt{r}_{11})\}\{(d, \mathtt{r}_{13})\}\{(x_3, \mathtt{r}_9)\}$.
  If a timed transition is $c_2 \xrightarrow{2.5}_{Time} c_3$ (in time elapses 2.5 such that the fraction of $\nu(x_1)$ becomes 0), $\ddot{\sigma}_{4,2}(\bar{\gamma}_2)$ simulates it as
  $\{(x_1, \mathtt{r}_6)\}, \{\overline{(d, \mathtt{r}_{11})}\}, \{(b, \mathtt{r}_{19})\}\{(x_2, \mathtt{r}_{13}), \underline{(a, \mathtt{r}_9)}\}\{(a, \mathtt{r}_{11})\}\{(d, \mathtt{r}_{13})\}\{(x_3, \mathtt{r}_9)\}$.

Propagate is used with $delete_\Gamma$ to simulate a pop transition. Since time progress is recorded only at the top stack frame (including updates on clock values), after $delete_\Gamma$ is applied to the top stack frame, the second stack frame is replaced with the top. Lacking information is a pointer ρ_2, which is recovered from the second stack frame. This will be illustrated in Example 11.

7 Decidability of reachability of DTPDA

7.1 Well-formed projection on snapshot configurations

Let $\langle s, \bar{\gamma}_k \cdots \bar{\gamma}_2 \bar{\gamma}_1 \rangle$ be a snapshot configuration for $s \in S$ and $\bar{\gamma}_i \in (\mathcal{MP}((\mathcal{C} \cup \Gamma) \times Intv(n)))^*$ (regarding $\bar{\gamma}_k$ as a top stack symbol). A marking completion marks elements in $\Gamma \times Intv(n)$ that relate to pushdown transitions.

Definition 15. *For $\bar{\gamma}_k \cdots \bar{\gamma}_2\bar{\gamma}_1$ with $\bar{\gamma}_i \in (\mathcal{MP}((\mathcal{C} \cup \Gamma) \times Intv(n)))^*$, the* marking completion $\mathtt{comp}$ *inductively marks elements in $\bar{\gamma}_i|_\Gamma$ for each i.*

$$\begin{cases} \mathtt{comp}(\bar{\gamma}_1) & = \textit{add marking on } \rho_1(\bar{\gamma}_1) \\ \mathtt{comp}(\bar{\gamma}_k \cdots \bar{\gamma}_2\bar{\gamma}_1) & = \bar{\gamma}'_k \cdots \bar{\gamma}'_2\bar{\gamma}'_1 \end{cases}$$

where $\bar{\gamma}'_{k-1} \cdots \bar{\gamma}'_2\bar{\gamma}'_1 = \mathtt{comp}(\bar{\gamma}_{k-1} \cdots \bar{\gamma}_2\bar{\gamma}_1)$ and $\bar{\gamma}'_k$ is obtained from $\bar{\gamma}_k$ by marking

- *$\rho_1(\bar{\gamma}_k)$, and*
- *each element in $delete_\gamma(\bar{\gamma}_k)|_\Gamma$ corresponding to a marked element in $\bar{\gamma}'_{k-1}|_\Gamma$ by a permutation σ satisfying $\sigma(\bar{\gamma}_{k-1})|_\Gamma = delete_\gamma(\bar{\gamma}_k)|_\Gamma$.*

If such σ does not exist, $\mathtt{comp}(\bar{\gamma}_k \cdots \bar{\gamma}_2\bar{\gamma}_1) = \#$.

We define a *well-formed projection* $\Downarrow_\Upsilon (s, \bar{\gamma}_k \cdots \bar{\gamma}_2\bar{\gamma}_1)$ by removing all unmarked elements of $\Gamma \times Intv(n)$ in each $\bar{\gamma}_i$ in $(s, \mathtt{comp}(\bar{\gamma}_k \cdots \bar{\gamma}_2\bar{\gamma}_1))$, and left s as is. A snapshot configuration $(s, \bar{\gamma}_k \cdots \bar{\gamma}_2\bar{\gamma}_1)$ is *well-formed* if $\Downarrow_\Upsilon (s, \bar{\gamma}_k \cdots \bar{\gamma}_2\bar{\gamma}_1) = (s, \bar{\gamma}_k \cdots \bar{\gamma}_2\bar{\gamma}_1)$ (ignoring markings), and Υ is the set of well-formed snapshot configurations.

Example 10. In Example 8, $\bar{\gamma}_5$ is well-formed (i.e., $(a, \mathtt{r}_7), (d, \mathtt{r}_9), (b, \mathtt{r}_{13}), (b, \mathtt{r}_{13})$ are all marked). For instance, a marking on $(a, \mathtt{r}_7)$ succeeds the pointer ρ_1 of $\bar{\gamma}_3$.

7.2 Snapshot PDS

Definition 16. *Let $\langle S, s_{init}, \Gamma, \mathcal{C}, \Delta\rangle$ be a DTPDA and let n be the largest integer in Δ. A* snapshot PDS *is a PDS $\mathcal{S} = \langle S, sw(\mathcal{C}, \Gamma, n), \Delta\rangle$. We assume that its initial configuration is $\langle s_{init}, \{(x, \mathtt{r}_0) \mid x \in \mathcal{C}\}\rangle$.*

Transition rule to simulate timed transitions $\langle s, \bar{\gamma}\rangle \xrightarrow{t}_{\mathcal{S}} \langle s, \sigma(\bar{\gamma})\rangle$, *where σ is either $\dot{\sigma}_{i,m}$ or $\ddot{\sigma}_{i,m}$ with $m = floor(t)$ and $1 \leq i \leq length(\bar{\gamma})$*

Transition rules to simulate discrete transitions (s, op, s')

- **Local** $\langle s,\ \epsilon\rangle \xrightarrow{nop}_{\mathcal{S}} \langle s',\ \epsilon\rangle$,
- **Assignment** $\langle s,\ \bar{\gamma}\rangle \xrightarrow{x \leftarrow I}_{\mathcal{S}} \langle s',\ assign(\bar{\gamma}, x, \mathtt{r})\rangle$ *for* $\mathtt{r} \subseteq I$,
- **Test** $\langle s,\ \bar{\gamma}\rangle \xrightarrow{x \in I?}_{\mathcal{S}} \langle s',\ \bar{\gamma}\rangle$ *if* $\mathtt{r} \subseteq I$ *for* $(x, \mathtt{r})$ *in* $\bar{\gamma}$.
- **Push** $\langle s,\ \bar{\gamma}\rangle \xrightarrow{push(\gamma', I)}_{\mathcal{S}} \langle s',\ insert(\bar{\gamma}, (\gamma', \mathtt{r}))\ \bar{\gamma}\rangle$ *for* $\mathtt{r} \subseteq I$, *and*
- **Pop** $\langle s,\ \bar{\gamma}\ \bar{\gamma}'\rangle \xrightarrow{pop(\gamma', I)}_{\mathcal{S}} \langle s',\ propagate(delete_\Gamma(\bar{\gamma}), \bar{\gamma}')\rangle$.

By induction on the number of steps of transitions, complete and sound simulation between a DTPDA and a snapshot PDS is observed. Note that the initial clock valuation of a DTPDA to be set $\nu_{\mathbf{0}}$ is essential.

Lemma 4. *Let us denote c_0 and c (resp. $\langle s_{init}, \bar{\gamma}_{\mathbf{0}}\rangle$ and $\langle s, \tilde{w}\rangle$) for the initial configuration and a configuration of a DTPDA $\mathcal{T}$ (resp. its snapshot PDS $\mathcal{S}$).*

1. *If $c_0 \hookrightarrow^* c$ then there exists $\langle s, \tilde{w}\rangle$ such that $\langle s_{init}, \bar{\gamma}_{\mathbf{0}}\rangle \underset{\Upsilon}{\hookrightarrow}^*_{\mathcal{S}} \langle s, \tilde{w}\rangle$, $s = state(c)$, and $\tilde{w}$ is well-formed.*

2. *If* $\langle s_{init}, \bar{\gamma}_{\mathbf{0}} \rangle \underset{\Upsilon\ \mathcal{S}}{\hookrightarrow^*} \langle s, \tilde{w} \rangle$ *and* $\tilde{w}$ *is well-formed. there exists* c *such that* $c_0 \hookrightarrow^* c$, $s = state(c)$, *and* $Snap(c) \hookleftarrow \tilde{w}$.

Example 11. We show how a snapshot PDS simulates a DTPDA in Example 1, as continuation to Example 9 (which shows transitions from c_1 to c_3).

- $c_3 \xrightarrow{x_2 \leftarrow (2,5]}_{Disc} c_4$ is simulated by $assign(delete_{\mathcal{C}}(snap(c_3), x_2), x_2, \mathtt{r}_7)$ at the top stack frame, since $\nu(x_2) = 3.8 \in \mathtt{r}_7$. There are several choices of $assign(delete_{\mathcal{C}}(snap(c_3), x_2), x_2, \mathtt{r}_7)$. Among them, $\{(x_1, \mathtt{r}_7)\}, \{\overline{(d, \mathtt{r}_{11})}\}, \{(b, \mathtt{r}_{19})\}\{\underline{(a, \mathtt{r}_9)}\}\{(a, \mathtt{r}_{11})\}\{(x_2, \mathtt{r}_7), (d, \mathtt{r}_{13})\}\{(x_3, \mathtt{r}_9)\}$. corresponds to 3.8. A different value, e.g., $\nu(x_2) = 3.3$, corresponds to $\{(x_1, \mathtt{r}_7)\}, \{\overline{(d, \mathtt{r}_{11})}\}, \{(x_2, \mathtt{r}_7), (b, \mathtt{r}_{19})\}\{\underline{(a, \mathtt{r}_9)}\}\{(a, \mathtt{r}_{11})\}\{(d, \mathtt{r}_{13})\}\{(x_3, \mathtt{r}_9)\}$.
- $c_4 \xrightarrow{pop(d,[4,6])}_{Disc} c_5$ is simulated by $propagate(delete_{\Gamma}(snap(c_4)), snap(c_1))$. Note that a snapshot PDS does not change anything except for the top stack frame. Thus, the second stack frame is kept unchanged from $snap(c_1)$. First, $delete_{\Gamma}$ removes the element pointed by ρ_1, which results in
$$\{(x_1, \mathtt{r}_7)\}, \{(b, \mathtt{r}_{19})\}\{\underline{(a, \mathtt{r}_9)}\}\{(a, \mathtt{r}_{11})\}\{(x_2, \mathtt{r}_7), (d, \mathtt{r}_{13})\}\{(x_3, \mathtt{r}_9)\}.$$
$snap(c_1) = \{(a, \mathtt{r}_7)\}\{(d, \mathtt{r}_9)\}\{(x_3, \mathtt{r}_5)\}\{(x_1, \mathtt{r}_1)\}\{\underline{(b, \mathtt{r}_{13})}\}\{(x_2, \mathtt{r}_7), \overline{(a, \mathtt{r}_3)}\}$ and, by pattern matching between ρ_2 in the former and ρ_1 in the latter, $\dot{\sigma}_{4,2}$ (which is used in the timed transition from c_2 to c_3 in Example 9) is found. Then ρ_1 is updated with the current ρ_2 and ρ_2 is recovered by σ as
$$\{(x_1, \mathtt{r}_7)\}, \{\underline{(b, \mathtt{r}_{19})}\}\{\overline{(a, \mathtt{r}_9)}\}\{(a, \mathtt{r}_{11})\}\{(x_2, \mathtt{r}_7), (d, \mathtt{r}_{13})\}\{(x_3, \mathtt{r}_9)\}.$$

It is not difficult to see that $\Downarrow_{\Upsilon}$ satisfies Definition 7. A snapshot PDS has finite states and WQO stack alphabet. By applying the encoding in Remark 1, we obtain our main result from Theorem 3, 5, Lemma 2, and 4.

Corollary 1. *The (configuration) reachability of a DTPDA is decidable.*

7.3 Comparison among discretizations

In [13], we apply slight extensions of a DTPDA to make it able to set the value of an age to that of a clock when a push occurs, and set the value of a clock to that of an age when a pop occurs. They are easily encoded into snapshot words.

- **Push-set** $push(\gamma, x)$, push γ on a stack associated with a local age of the value of a clock $x \in \mathcal{C}$, and
- **Pop-set** $pop(\gamma, x)$, pop γ on a stack and set the value of a clock $x \in \mathcal{C}$ to the value of the associated age a.

When we consider extensions of DTPDA [1] with such operations, we see the difference between the original discretization [1] and ours as a snapshot PDS. Note that our snapshot word encoding summarizes the ordering of fractions of all local ages and global clocks in the stack, whereas the encoding in [1] summarizes boundedly many information, i.e., global clocks and an age in the top stack frame and copies of global clocks in the next stack frame.

Example 12. The encoding in [1] does not contain $x^\bullet$ for $x \in \mathcal{C}$, which represents the position of the value of a clock x in the previous stack frame. Our encoding of a DTPDA as a snapshot word PDS is quite equivalent to an extension of that in [1] with $x^\bullet$ for $c \in \mathcal{C}$. With and without $x^\bullet$ are different when we consider an extension of DTPDA, e.g., that with

- **Compare** $compare_\sim(x)$ for a clock $x \in \mathcal{C}$ and $\sim \in \{\geq, >, \leq, <, =\}$, which compares values between an age in the top stack frame and a clock x by $\sim$.

  $compare_\sim(x)$ is a quite strong operator. It enables us to define

- **Push-set** $push(\gamma, x)$, push γ on a stack associated with a local age of the value of a clock $c \in \mathcal{C}$, and
- **Push-set**$^+$ $push^+(\gamma, x)$, push γ on a stack associated with a local age whose value is between the value of c and its ceiling value.
- **Push-set**$^-$ $push^+(\gamma, x)$, push γ on a stack associated with a local age whose value is between the value of c and its floor value.

Similar for **Pop-set**. With a fresh clock y prepared as a stop watch, we can encode these operations with $compare_\sim(x)$ as follows.

- **Push-set** $push(\gamma, x)$ is encoded as
  $y \leftarrow [0,0]; push(\gamma, [0,\infty)); compare_=(x); y \in [0,0]?;$.
- **Push-set**$^+$ $push^+(\gamma, x)$ is encoded as
  $y \leftarrow [0,0]; x \in (j, j+1)?; push(\gamma, (j, j+1)); compare_>(x); y \in [0,0]?;$.
- **Push-set**$^-$ $push^+(\gamma, x)$ is encoded as
  $y \leftarrow [0,0]; x \in (j, j+1)?; push(\gamma, (j, j+1)); compare_<(x); y \in [0,0]?;$.

Note that these operations enables us to prepare an operation that compares values of ages in different stack frames. For instance, a sequence

$$push^+(\gamma, x); (\mathbf{push})^*; push^-(\gamma', x); (x \leftarrow [0,\infty)); pop(\gamma', y); (\mathbf{pop})^*; compare_<(y);$$

compares ages in the different stack frames containing γ and γ'.

Note that the original encoding in [1] cannot decide $compare_<(y)$. With $x^\bullet$, it can correctly decide that $compare_<(y)$ interrupts transitions.

8 Conclusion

This paper investigated a general framework of pushdown systems with well-quasi-ordered control states and stack alphabet, *well-structured pushdown systems*, to show decidability of the reachability. This extends the decidability results on a pushdown system with finite control states and well-quasi-ordered stack alphabet [6]. The ideas behind are,

- combining WSTS [2, 11] and classical Pre^*-automaton technique [5, 12, 10], which enables us to reduce arguments on stacks to on stack symbols, and
- introduction of a well-formed projection $\Downarrow_\Upsilon$, which extracts the shape of reachable configurations.

As an instance, an alternative decidability proof of the reachability for dense-timed pushdown system [1] was shown. Note that the original encoding [1] cannot handle the extension with $compare_{\sim}(x)$ (which compares values of a local age in the top stack frame and a clock x). The encoding is inspired by the digitization techniques in [15].

Acknowledgements

The authors would like to thank Shoji Yuen, Yasuhiko Minamide, Tachio Terauchi, and Guoqiang Li for valuable comments and discussions. This work is supported by the NSFC-JSPS bilateral joint research project (61011140074), NSFC projects (61003013,61100052,61033002), NSFC-ANR joint project (61261130589), and JSPS KAKENHI Grant-in-Aid for Scientific Research(B) (23300008).

References

1. P.A. Abdulla, M.F. Atig, and F. Stenman. Dense-Timed Pushdown Automata. *IEEE LICS 2012*, pages 35–44, 2012.
2. P.A. Abdulla, K. Cerans, C. Jonsson, and T. Yih-Kuen. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1–2):109–127, 2000.
3. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. A. Bouajjani, R. Echahed, and R. Robbana On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures. *Hybrid Systems II*, *LNCS 999*, pages 64–85, 1995.
5. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. *CONCUR 1997*, *LNCS 1243*, pages 135–150, 1997.
6. X. Cai and M. Ogawa. Well-Structured Pushdown Systems. CONCUR 2013, LNCS 8052 (2013), 121–136. Long version: JAIST Research Report IS-RR-2013-001.
7. R. Chadha and M. Viswanathan. Decidability results for well-structured transition systems with auxiliary storage. *CONCUR 2007*, *LNCS 4703*, pages 136–150, 2007.
8. Z. Dang. Pushdown timed automata:a binary reachability characterization and safety verification. *Theoretical Computer Science*, 302:93–121, 2003.
9. M. Emmi and R. Majumdar. Decision Problems for the Verification of Real-Time Software. *HSCC'06*, *LNCS 3927*, pages 200–211, 2006.
10. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. *CAV 2000*, *LNCS 1855*, pages 232–247, 2000.
11. A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1–2):63–92, 2001.
12. A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems (extended abstract). *INFINITY'97*, *ENTCS 9*. 1997.
13. G. Li, X. Cai, M. Ogawa, and S. Yuen. Nested Timed Automata. *FORMATS 2013*, *LNCS 8503*, pages 168–182, 2013.
14. R. Mayr. Process rewrite systems. *Information and Computation*, 156:264–286, 1999.
15. J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. *IEEE LICS 2004*, pages 54–63, 2004.