Reactive Synthesis from Visibly Register Pushdown Automata

Ryoma Senda[1], Yoshiaki Takata[2], and Hiroyuki Seki[1]

[1] Graduate School of Informatics, Nagoya University
Furo-cho, Chikusa, Nagoya 464-8601, Japan
ryoma.private@sqlab.jp, seki@i.nagoya-u.ac.jp
[2] Graduate School of Engineering, Kochi University of Technology
Tosayamada, Kami City, Kochi 782-8502, Japan
takata.yoshiaki@kochi-tech.ac.jp

Abstract. The realizability problem for a given specification $\mathcal{S}$ is to decide whether there exists an implementation satisfying $\mathcal{S}$. Although the problem is important in the field of reactive synthesis of recursive programs, the problem has not been studied yet when specification and implementation are given by pushdown computational models. This paper investigates the realizability problem for the cases that a specification and an implementation are given by a pushdown automaton (PDA) and a pushdown transducer (PDT), and a register pushdown automata (RPDA) and a register pushdown transducer (RPDT).

1 Introduction

Reactive synthesis is a method of synthesizing a system that satisfies a given specification representing the input-output relation of the system. A specification is formally a subset of an infinite alternate sequences of input and output symbols. When an environment or a user gives inputs $i_0, i_1, \ldots$ in this order to a system, the latter is required to emit a sequence of outputs $o_0, o_1, \ldots$ such that $(i_0, o_0)(i_1, o_1) \cdots \in S$. The realizability problem is to decide whether for a given specification S, there exists a reactive system satisfying S, and if exists, to generate such a system (called an implementation of S).

Studies on reactive synthesis has ~~its~~ origin in 1960s and ~~has~~ been one of central topics in formal methods [11]. Among them, Büchi and Landweber [8] showed EXPTIME-completeness of the problem when a specification is given by a finite ω-automaton (a finite automaton on infinite words). Pnueli and Rosner [28] showed 2EXPTIME-completeness of the problem when a specification is given by an LTL formula. We can find an excellent tutorial and survey of the previous studies on the synthesis problem in [2]. The standard approach to the problem is as follows. Assume that, for example, a specification is given as a deterministic ω-automaton $\mathcal{A}$. We convert $\mathcal{A}$ to a tree automaton (or equivalently, a parity game) $\mathcal{B}$ by separating the input and output streams. Then, we test whether $L(\mathcal{B}) \neq \emptyset$ (or equivalently, there is a winning strategy for player I in $\mathcal{B}$). The answer to the problem is affirmative if and only if $L(\mathcal{B}) \neq \emptyset$, and any

$t \in L(\mathcal{B})$ (or any winning strategy for $\mathcal{B}$) is an implementation of the specification.

Classical models such as a finite automaton cannot deal with objects from an infinite set (called data values). However, when we add an ability of manipulating data values to such a classical model, the model easily becomes Turing-complete and basic problems become undecidable. Register automaton (RA) is an extension of finite automaton by adding limited ability of manipulating data values [7, 21, 27, 30]. An RA has a finite number of registers for storing data values taken from an input word and it can compare the contents of its registers with the current input data value to determine the next transition. RA inherits some good properties from finite automaton such as the closure under some language opeations. The membership and emptiness are decidable for RA while the universality is undecidable. With the increase of interest in RA, the realizability problem for register models has been investigated recently [16, 23, 19, 22]. A specification is given by an RA and an implementation is represented by a register transducer (abbreviated as RT).

Pushdown automaton (PDA) is a simple model for recursive programs. Model checking algorithms for pushdown systems (PDA without input) has been extensively studied [20, 36, 6, 17, 18]. Extensions by adding registers are also done for PDA [12, 26, 33], called register pushdown automaton (RPDA). Since many real-world programs are recursive and also manipulate data values, it is important to investigate the realizability problem for PDA/RPDA.

This paper first extends the realizability problem to PDA and pushdown transducer. A pushdown transducer (PDT) is a deterministic PDA with output. PDT serves as a model of a recursive program that emits outputs according to the inputs given from its environment. The main difficulties to solve the realizability problem in this setting come from the facts that the class of languages recognized by nondeterministic PDA (NPDA) (i.e., context-free languages) does not have the closure properties under some set operations and some relevant decision problems such as the universality are undecidable for PDA. To avoid these difficulties, this paper mainly considers deterministic PDA (DPDA). (Note that, PDT is deterministic by definition.) In section 4, we show that the realizability problem for DPDA is decidable while the problem is undecidable for NPDA. The former is proved by using the well-known property that the (two-players zero-sum parity) pushdown game is decidable and a winning strategy of the game can be constructed as a PDT [36].

Then, the paper moves to the realizability problem for RPDA and register PDT. In section 5, we introduce RPDA and register PDT (RPDT). Both RPDA and RPDT read a data word, which is an infinite sequence of a pair (a, d) of a symbol a from a finite alphabet and a data value d from an infinite set. We show that the projection of the language recognized by a nondeterministic RPDA onto the finite alphabet can be recognized by a nondeterministic PDA, when we assume the freshness of input data values [35] for RPDA.

In section 6, we discuss the realizability problem for deterministic RPDA (DRPDA) and RPDT. Our approach is to reduce the problem for DRPDA to

the problem for DPDA. In this reduction, we want to convert a given DRPDA to a DPDA that recognizes the projection onto the finite alphabet of the former language by using the method in section 5. For this purpose, we further assume that a given DPDA has the visibility on the guard condition inherited from the DRPDA as well as the visibility of stack operation (see [1] for the visibility of stack operation). Finally, we show that the realizability problem is decidable for visibly DRPDA and RPDT.

Related work RA is frequently used as computation models for querying structured data such as XML documents and graph databases [25, 24]. LTL with the freeze quantifier (LTL↓) [14, 15] is an extension of LTL by adding the ability of memorizing and comparing data values. LTL↓ has the strong relationship with two-way alternate extension of RA. Two-variable first-order logic on data words [3] is another famous logic for data values, whose expressive power is incomparable with LTL↓. Other extensions of RA are found in [4, 9, 13, 5]. Nominal automaton [4] is an extension of finite automaton by using nominal sets, which are infinite sets having finite orbits of group actions and equivariant functions among them.

When considering the realizability for RA, the difficulty is to identify the number of registers needed for implementing the specification, i.e., the number of registers of RT. If the upperbound of the registers of RT is not given as an input, the realizability problem is undecidable for both of nondeterministic and universal RA [19]. If the upperbound of the registers is *a priori* known, the problem (called the bounded realizability problem) is shown to be decidable in EXPTIME for universal RA [22]. In [19], it is shown that the bounded realizability problem remains undecidable for nondeterministic RA (NRA) and becomes decidable in 2EXPTIME for a subclass called test-free NRA.

Extensions by adding registers are also done for PDA [12, 26, 33] and context-free grammar [10, 31, 32], called register pushdown automaton (RPDA) and register context-free grammar (RCFG), respectively. The expressive powers of RPDA and RCFG are the same. RPDA is a natural model for recursive programs with data values while RCFG has an advantage such that explicit representation of pushdown stack is not needed. For other extensions of PDA and verification of them, see [34, 37, 29].

2 Preliminaries

Let $\mathbb{N} = \{1, 2, \ldots\}$, $\mathbb{N}_0 = \{0\} \cup \mathbb{N}$ and $[n] = \{1, \cdots, n\}$ for $n \in \mathbb{N}$. For a set A, let $\mathscr{P}(A)$ be the power set of A, let A^* and A^ω be the sets of finite and infinite words over A, respectively. We denote $A^+ = A^* \setminus \{\varepsilon\}$ and $A^\infty = A^* \cup A^\omega$. For a word $\alpha \in A^\infty$ over a set A, let $\alpha(i) \in A$ be the i-th element of α ($i \geq 0$), $\alpha(i : j) = \alpha(i)\alpha(i+1)\cdots\alpha(j-1)\alpha(j)$ for $i \geq j$ and $\alpha(i :) = \alpha(i)\cdots$ for $i \geq 0$. Let $\langle u, w\rangle = u(0)w(0)u(1)w(1)\cdots \in A^\infty$ for words $u, w \in A^\infty$ and $\langle B, C\rangle = \{\langle u, w\rangle \mid u \in B, w \in C\}$ for sets $B, C \subseteq A^\infty$. By $|\beta|$, we mean the cardinality of β if β is a set and the length of β if β is a finite sequence. For a function $f : A \to B$ from a set A to a set B, let $f(w) = f(w(0))f(w(1))\ldots$ for

a word $w \in A^\infty$ and let $f(L) = \{f(w) \mid w \in L\}$ for a set $L \subseteq A^\infty$ of words. Let *fst* and *snd* be the functions such that $fst((a,b)) = a$ and $snd((a,b)) = b$ for any pair (a,b). Let *id* be the identity function; i.e., $id(a) = a$ for any a.

2.1 Transition systems

Definition 1. *A transition system (TS) is* $\mathcal{S} = (S, s_0, A, E, \to_{\mathcal{S}}, c)$ *where*

- S *is a (finite or infinite) set of states,*
- $s_0 \in S$ *is the initial state,*
- A, E *are (finite or infinite) alphabets such that* $A \cap E = \emptyset$*,*
- $\to_{\mathcal{S}} \subseteq S \times (A \cup E) \times S$ *is a transition relation, written as* $s \to^a s'$ *if* $(s,a,s') \in \to_{\mathcal{S}}$ *and*
- $c : S \to [n]$ *is a coloring function where* $n \in \mathbb{N}$*.*

An element of A is an observable label and an element of E is an internal label. A *run* of TS $\mathcal{S} = (S, s_0, A, E, \to_{\mathcal{S}}, c)$ is a pair $(\rho, w) \in S^\omega \times (A \cup E)^\omega$ that satisfies $\rho(0) = s_0$ and $\rho(i) \to^{w(i)} \rho(i+1)$ for $i \geq 0$. Let $\min_{\inf} : S^\omega \to [n]$ be the minimal coloring function such that $\min_{\inf}(\rho) = \min\{m \mid$ there exist an infinite number of $i \geq 0$ such that $c(\rho(i)) = m\}$. We call $\mathcal{S}$ deterministic if $s \to^a s_1$ and $s \to^a s_2$ implies $s_1 = s_2$ for all $s, s_1, s_2 \in S$ and $a \in A \cup E$.

For $w \in (A \cup E)^\omega$, let $ef(w) = a_0 a_1 \cdots \in A^\infty$ be the sequence obtained from w by removing all symbols belonging to E. Note that $ef(w)$ is not always an infinite sequence even if w is an infinite sequence. We define the *language* of $\mathcal{S}$ as $L(\mathcal{S}) = \{ef(w) \in A^\omega \mid$ there exists a run (ρ, w) such that $\min_{\inf}(\rho)$ is even$\}$. For $m \in \mathbb{N}_0$, we call $\mathcal{S}$ an m-TS if for every run (ρ, w) of $\mathcal{S}$, w contains no ~~consecutive~~ subsequence $w' \in E^{m+1}$.

Consider two TSs $\mathcal{S}_1 = (S_1, s_{01}, A_1, E_1, \to_{\mathcal{S}_1}, c_1)$ and $\mathcal{S}_2 = (S_2, s_{02}, A_2, E_2, \to_{\mathcal{S}_2}, c_2)$ and a function $\sigma : (A_1 \cup E_1) \to (A_2 \cup E_2)$. We call $R \subseteq S_1 \times S_2$ a *σ-bisimulation relation* from $\mathcal{S}_1$ to $\mathcal{S}_2$ if R satisfies the followings:

(1) $(s_{01}, s_{02}) \in R$.
(2) For any $s_1, s_1' \in S_1$, $s_2 \in S_2$, and $a_1 \in A_1 \cup E_1$, if $s_1 \to_{\mathcal{S}_1}^{a_1} s_1'$ and $(s_1, s_2) \in R$, then $\exists s_2' \in S_2 : s_2 \to_{\mathcal{S}_2}^{\sigma(a_1)} s_2'$ and $(s_1', s_2') \in R$.
(3) For any $s_1 \in S_1$, $s_2, s_2' \in S_2$, and $a_2 \in A_2 \cup E_2$, if $s_2 \to_{\mathcal{S}_2}^{a_2} s_2'$ and $(s_1, s_2) \in R$, then $\exists s_1' \in S_1$, $\exists a_1 \in A_1 \cup E_1 : \sigma(a_1) = a_2$ and $s_1 \to_{\mathcal{S}_1}^{a_1} s_1'$ and $(s_1', s_2') \in R$.
(4) If $(s_1, s_2) \in R$, then $c_1(s_1) = c_2(s_2)$.

We say $\mathcal{S}_1$ is *σ-bisimilar* to $\mathcal{S}_2$ if there exists a σ-bisimulation relation from $\mathcal{S}_1$ to $\mathcal{S}_2$. We call R a bisimulation relation if R is an *id*-bisimulation relation. We say $\mathcal{S}_1$ and $\mathcal{S}_2$ are bisimilar if $\mathcal{S}_1$ is *id*-bisimilar to $\mathcal{S}_2$.

The following lemma can be proved by definition.

Lemma 2. *If* $\mathcal{S}_1 = (S_1, s_{01}, A_1, E_1, \to_{\mathcal{S}_1}, c_1)$ *is σ-bisimilar to* $\mathcal{S}_2 = (S_2, s_{02}, A_2, E_2, \to_{\mathcal{S}_2}, c_2)$ *for a function* $\sigma : (A_1 \cup E_1) \to (A_2 \cup E_2)$ *that satisfies* $a \in A_1 \Leftrightarrow \sigma(a) \in A_2$ *for any* $a \in A_1 \cup E_1$*, then* $\sigma(L(\mathcal{S}_1)) = L(\mathcal{S}_2)$*.*

3 Pushdown Transducers, Automata and Games

We assume that disjoint sets $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ and Γ are given as a (finite) input alphabet, an output alphabet and a stack alphabet, respectively, and $\Sigma = \Sigma_{\mathbb{i}} \cup \Sigma_{\mathbb{o}}$. Let $\mathtt{Com}(\Gamma) = \{\mathtt{pop}, \mathtt{skip}\} \cup \{\mathtt{push}(z) \mid z \in \Gamma\}$ be the set of stack commands over Γ.

3.1 Pushdown transducers

Definition 3. *A pushdown transducer (PDT) over $\Sigma_{\mathbb{i}}$, $\Sigma_{\mathbb{o}}$ and Γ is $\mathcal{T} = (P, p_0, z_0, \Delta)$ where P is a finite set of states, $p_0 \in P$ is the initial state, $z_0 \in \Gamma$ is the initial stack symbol and $\Delta : P \times \Sigma_{\mathbb{i}} \times \Gamma \to P \times \Sigma_{\mathbb{o}} \times \mathtt{Com}(\Gamma)$ is a finite set of deterministic transition rules having one of the following forms:*

- $(p, a, z) \to (q, b, \mathtt{pop})$ *(pop rule)*
- $(p, a, z) \to (q, b, \mathtt{skip})$ *(skip rule)*
- $(p, a, z) \to (q, b, \mathtt{push}(z))$ *(push rule)*

where $p, q \in P$, $a \in \Sigma_{\mathbb{i}}$, $b \in \Sigma_{\mathbb{o}}$ and $z \in \Gamma$.

For a state $p \in P$ and a finite sequence representing stack contents $u \in \Gamma^*$, (p, u) is called a *configuration* or *instantaneous description (abbreviated as ID)* of PDT $\mathcal{T}$. Let $ID_{\mathcal{T}}$ denote the set of all IDs of $\mathcal{T}$. For $u \in \Gamma^+$ and $\mathtt{com} \in \mathtt{Com}(\Gamma)$, let us define $\mathtt{upds}(u, \mathtt{com})$ as $\mathtt{upds}(u, \mathtt{pop}) = u(1 :)$, $\mathtt{upds}(u, \mathtt{skip}) = u$ and $\mathtt{upds}(u, \mathtt{push}(z')) = z'u$.

For two IDs $(p, u), (q, u') \in ID_{\mathcal{T}}$, $a \in \Sigma_{\mathbb{i}}$ and $b \in \Sigma_{\mathbb{o}}$, $((p, u), ab, (q, u')) \in \Rightarrow_{\mathcal{T}}$, written as $(p, u) \Rightarrow_{\mathcal{T}}^{ab} (q, u')$, if there exist a rule $(p, a, z) \to (q, b, \mathtt{com}) \in \Delta$ such that $z = u(0)$ and $u' = \mathtt{upds}(u, \mathtt{com})$. If $\mathcal{T}$ is clear from the context, we abbreviate $\Rightarrow_{\mathcal{T}}^{ab}$ as $\Rightarrow^{ab}$. We will use similar abbreviations for the other models defined later. Note that there is no transition from an ID with empty stack. We define a run and the language $L(\mathcal{T}) \subseteq (\Sigma_{\mathbb{i}} \cdot \Sigma_{\mathbb{o}})^\omega$ of PDT $\mathcal{T}$ as those of deterministic 0-TS $(ID_{\mathcal{T}}, (q_0, z_0), \Sigma_{\mathbb{i}} \cdot \Sigma_{\mathbb{o}}, \emptyset, \Rightarrow_{\mathcal{T}}, c)$ where $c(s) = 2$ for all $s \in ID_{\mathcal{T}}$. In this paper, we assume that no run of PDT reaches an ID whose stack is empty. Let **PDT** be the class consisting of all PDT.

Example 4. Let us consider PDT $\mathcal{T} = (\{p\}, p, z, \Delta)$ over $\{0, 1\}, \{a, b\}$ and $\{z\}$ where $\Delta = \{(p, 0, z) \to (p, a, \mathtt{skip}), (p, 1, z) \to (p, b, \mathtt{push}(z))\}$. We can see a pair of sequences $(\rho, w) \in ID_{\mathcal{T}}^\omega \times (\{0, 1\} \cdot \{a, b\})^\omega$ where $\rho = (p, z)(p, z)(p, zz)(p, zz)(p, zzz)(p, zzz) \cdots$ and $w = (0a1b)^\omega$ is a run of $\mathcal{T}$. Also, $L(\mathcal{T}) = (\{0a\} \cup \{1b\})^\omega$.

3.2 Pushdown automata

Definition 5. *A nondeterministic pushdown automata (NPDA) over $\Sigma_{\mathbb{i}}$, $\Sigma_{\mathbb{o}}$ and Γ is $\mathcal{A} = (Q, Q_{\mathbb{i}}, Q_{\mathbb{o}}, q_0, z_0, \delta, c)$ where Q, $Q_{\mathbb{i}}, Q_{\mathbb{o}}$ are finite sets of states such that $Q = Q_{\mathbb{i}} \cup Q_{\mathbb{o}}$ and $Q_{\mathbb{i}} \cap Q_{\mathbb{o}} = \emptyset$, $q_0 \in Q_{\mathbb{i}}$ is the initial state, $z_0 \in \Gamma$ is the initial stack symbol, $c : Q \to [n]$ is the coloring function where $n \in \mathbb{N}$ is the number of priorities and $\delta : Q \times \Sigma \times \Gamma \to \mathscr{P}(Q \times \mathtt{Com}(\Gamma))$ is a finite set of transition rules having one of the following forms:*

– $(q_{\natural}, a_{\natural}, z) \to (q_{\overline{\natural}}, \texttt{com})$ *(input/output rules)*
– $(q_{\natural}, \tau, z) \to (q'_{\natural}, \texttt{com})$ *(τ rules)*

where $(\natural, \overline{\natural}) \in \{(\mathbb{i}, \mathbb{o}), (\mathbb{o}, \mathbb{i})\}$, $q_{\natural}, q'_{\natural} \in Q_{\natural}, q_{\overline{\natural}} \in Q_{\overline{\natural}}, a_{\natural} \in \Sigma_{\natural}$, $z \in \Gamma$ *and* $\texttt{com} \in \texttt{Com}(\Gamma)$.

We define $ID_{\mathcal{A}} = Q \times \Gamma^*$ and the transition relation $\vdash_{\mathcal{A}} \subseteq ID_{\mathcal{A}} \times (\Sigma \cup \{\tau\}) \times ID_{\mathcal{A}}$ as $((q,u), a, (q',u')) \in \vdash_{\mathcal{A}}$ iff there exist a rule $(q, a, z) \to (q', \texttt{com}) \in \delta$ and a sequence $u \in \Gamma^*$ such that $z = u(0)$ and $u' = \texttt{upds}(u, \texttt{com})$. We write $(q,u) \vdash^a_{\mathcal{A}} (q',u')$ iff $((q,u), a, (q',u')) \in \vdash_{\mathcal{A}}$. We define a run and the language $L(\mathcal{A})$ of $\mathcal{A}$ as those of TS $\mathcal{S}_{\mathcal{A}} = (ID_{\mathcal{A}}, (q_0, z_0), \Sigma, \{\tau\}, \vdash_{\mathcal{A}}, c')$ where $c'((q,u)) = c(q)$ for every $(q,u) \in ID_{\mathcal{A}}$. We call a PDA $\mathcal{A}$ deterministic if $\mathcal{S}_{\mathcal{A}}$ is deterministic. We call $\mathcal{A}$ an m-NPDA (or m-DPDA when $\mathcal{A}$ is deterministic) if $\mathcal{S}_{\mathcal{A}}$ is an m-TS. We abbreviate 0-NPDA (0-DPDA) as NPDA (DPDA). Let **DPDA** and **NPDA** be the classes of DPDA and NPDA, respectively.

Example 6. Let us consider DPDA $\mathcal{A} = (\{q, p_0, p_1\}, \{q\}, \{p_0, p_1\}, q, z, \delta, c)$ over $\{0,1\}, \{a,b\}$ and $\{z\}$ where $c(q) = c(p_0) = c(p_1) = 2$ and $\delta = \{(q,0,z) \to (p_0, \texttt{skip}), (q,1,z) \to (p_1, \texttt{skip}), (p_0,a,z) \to (q, \texttt{push}(z)), (p_0,b,z) \to (q, \texttt{push}(z)), (p_1,b,z) \to (q, \texttt{push}(z))\}$. We can see a pair of sequences $(\rho, w) \in ID_{\mathcal{T}}^{\omega} \times (\{0,1\} \cdot \{a,b\})^{\omega}$ where $\rho = (q,z)(p_0,z)(q,zz)(p_1,zz)\cdots$ and $w = (0a1b)^{\omega}$ is a run of $\mathcal{A}$. Also, we can check $L(\mathcal{A}) = (\{0a\} \cup \{0b\} \cup \{1b\})^{\omega}$.

The following lemma states that the class of languages recognized by m-DPDA and 0-DPDA are the same for a fixed m.

Lemma 7. *For a given m-DPDA $\mathcal{A}$, we can construct a 0-DPDA $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$*

Proof sketch We define the stack alphabet Γ' of $\mathcal{A}'$ as $\Gamma' = \Gamma^m$. We can simulate m-steps with consecutive push rules (or pop rules) of $\mathcal{A}$ by a single step with a push (or pop) rule of $\mathcal{A}'$.

3.3 Pushdown games

Definition 8. *A pushdown game of DPDA $\mathcal{A} = (Q, Q_{\mathbb{i}}, Q_{\mathbb{o}}, q_0, z_0, \delta, c)$ over $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ and Γ is $\mathcal{G}_{\mathcal{A}} = (V, V_{\mathbb{i}}, V_{\mathbb{o}}, E, C)$ where $V = Q \times \Gamma^*$ is the set of vertices with $V_{\mathbb{i}} = Q_{\mathbb{i}} \times \Gamma^*, V_{\mathbb{o}} = Q_{\mathbb{o}} \times \Gamma^*$, $E \subseteq V \times V$ is the set of edges defined as $E = \{(v,v') \mid v \vdash^a v' \text{for some } a \in \Sigma_{\mathbb{i}} \cup \Sigma_{\mathbb{o}}\}$ and $C : V \to [n]$ is the coloring function such that $C((q,u)) = c(q)$ for all $(q,u) \in V$.*

The game starts with $(q_0, z_0) \in V_{\mathbb{i}}$. When the current vertex is $v \in V_{\mathbb{i}}$, Player II chooses a successor $v' \in V_{\mathbb{o}}$ of v as the next vertex. When the current vertex is $v \in V_{\mathbb{o}}$, Player I chooses a successor $v' \in V_{\mathbb{i}}$ of v. Formally, a finite or infinite sequence $\rho \in V^{\infty}$ is *valid* if $\rho(0) = (q_0, z_0)$ and $(\rho(i-1), \rho(i)) \in E$ for every $i \geq 1$. A *play* of $\mathcal{G}_{\mathcal{A}}$ is an infinite and valid sequence $\rho \in V^{\omega}$. Let PL be the set of plays. A play $\rho \in PL$ is *winning* for Player I iff $\min\{m \in [n] \mid$ there ~~exist~~ an infinite number of $i \geq 0$ such that $c(\rho(i)) = m\}$ is even. Note that by definition, a play ρ is winning for Player I iff (ρ, w) is a run of $\mathcal{A}$ for some w.

Since $\mathcal{A}$ is deterministic, the following lemma holds.

Lemma 9. *Let* $f_1 : PL \to (Q \times \mathtt{Com}(\Gamma))^\omega$ *and* $f_2 : (\Sigma_{\mathbb{i}} \cdot \Sigma_{\mathbb{o}})^\omega \to PL$ *be the functions defined as follows:*

- $f_1(\rho) = (q_0, \mathtt{com}_0)(q_1, \mathtt{com}_1)\cdots \in (Q \times \mathtt{Com})^\omega$ *where* $u_{i+1} = \mathtt{upds}(u_i, \mathtt{com}_i)$ *for all* $i \geq 0$ *and*
- $f_2(w) = \rho$ *where* $\rho = (q_0, u_0)(q_1, u_1)\cdots \in PL$ *and* $\rho(i) \vdash^{w(i)} \rho(i+1)$ *for all* $i \geq 0$.

Then, f_1 *and* f_2 *are well-defined,* f_1 *is an injection and* $f_2(L(\mathcal{A}))$ *is the set of all the winning plays of Player I.*

Theorem 10. *[36] If player I has a winning strategy of* $\mathcal{G}_\mathcal{A}$*, we can construct a PDT* $\mathcal{T}$ *over* $Q_{\mathbb{i}} \times \mathtt{Com}(\Gamma), Q_{\mathbb{o}} \times \mathtt{Com}(\Gamma)$ *and a stack alphabet* Γ' *that gives a winning strategy of* $\mathcal{G}_\mathcal{A}$*. That is,* $\rho \in PL$ *is winning for Player I if* $f_1(\rho) \in L(\mathcal{T})$.

By Lemma 9, a winning strategy can be also given as a subset of sequences $w \in (\Sigma_{\mathbb{i}} \cdot \Sigma_{\mathbb{o}})^\omega$ such that the play $f_2(w)$ is winning for Player I. Thus, we can obtain the following lemma in a similar way to Theorem 10.

Corollary 11. *If player I has a winning strategy of* $\mathcal{G}_\mathcal{A}$*, we can construct a PDT* $\mathcal{T}$ *over* $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ *and* Γ' *that gives a winning strategy of* $\mathcal{G}_\mathcal{A}$*. That is,* $f_2(w) \in PL$ *is winning for Player I if* $w \in L(\mathcal{T})$.

4 Realizability problems for PDA and PDT

For a specification S and an implementation I, we write $I \models S$ if $L(I) \subseteq L(S)$.

Definition 12. *Realizability problem* Real$(\mathcal{S}, \mathcal{I})$ *for a class of specifications* $\mathcal{S}$ *and of implementations* $\mathcal{I}$*: For a specification* $S \in \mathcal{S}$*, is there an implementation* $I \in \mathcal{I}$ *such that* $I \models S$ *?*

Example 13. By Examples 4 and 6, $L(\mathcal{T}) \subseteq L(\mathcal{A})$ holds for PDT $\mathcal{T}$ and DPDA $\mathcal{A}$ defined in the examples. Thus, $\mathcal{T} \models \mathcal{A}$ holds.

Theorem 14. Real(**DPDA**, **PDT**) *is in EXPTIME.*

Proof. Let $\mathcal{A}$ be a given DPDA. By definitions, $w \in L(\mathcal{A})$ iff $f_2(w)$ is a winning play for Player I of $\mathcal{G}_\mathcal{A}$. By Corollary 11, if Player I has a winning strategy, we can construct a PDT $\mathcal{T}$ such that $f_2(w)$ is a winning play of $\mathcal{G}_\mathcal{A}$ if $w \in L(\mathcal{T})$. Hence, $\mathcal{T} \models \mathcal{A}$ holds. If Player I does not have a winning strategy, there is no $\mathcal{T}$ such that $\mathcal{T} \models \mathcal{A}$. Because there is an EXPTIME algorithm for constructing $\mathcal{T}$ (if exists) in [36], Real(**DPDA**, **PDT**) is in EXPTIME.

Theorem 15. Real(**NPDA**, **PDT**) *is undecidable.*

Proof. We prove the theorem by a reduction from the universality problem of NPDA, which is undecidable. For a given NPDA $\mathcal{A} = (Q, Q_{\mathbb{i}}, Q_{\mathbb{o}}, q_0, z_0, \delta, c)$ over $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ and Γ, we can construct an NPDA $\mathcal{A}' = (Q \times [2], Q \times \{1\}, Q \times$

$\{2\}, q_0, z_0, \delta', c')$ over $\Sigma'_{\mathbb{i}}, \Sigma'_{\mathbb{o}}$ and Γ where $\Sigma'_{\mathbb{i}} = \Sigma_{\mathbb{i}} \cup \Sigma_{\mathbb{o}}$, $\Sigma'_{\mathbb{o}}$ is an arbitrary (nonempty) alphabet, $c'((q,1)) = c'((q,2)) = c(q)$ for all $q \in Q$ and $((q,1), a, z) \to ((q',2), \mathtt{com}) \in \delta'$ iff $(q, a, z) \to (q', \mathtt{com}) \in \delta$, and $((q',2), b, z) \to ((q',1), \mathtt{skip}) \in \delta'$ for all $b \in \Sigma'_{\mathbb{o}}$ and $z \in \Gamma$.

We show $L(\mathcal{A}) = (\Sigma'_{\mathbb{i}})^\omega$ iff there exists $\mathcal{T}$ such that $\mathcal{T} \models \mathcal{A}$. By the construction of $\mathcal{A}'$, $L(\mathcal{A}') = \langle L(\mathcal{A}), (\Sigma'_{\mathbb{o}})^\omega \rangle$ holds. If $L(\mathcal{A}) = (\Sigma'_{\mathbb{i}})^\omega$, then $L(\mathcal{A}') = \langle (\Sigma'_{\mathbb{i}})^\omega, (\Sigma'_{\mathbb{o}})^\omega \rangle$ and thus $\mathcal{T} \models \mathcal{A}$ holds for every $\mathcal{T}$. Assume that $L(\mathcal{A}) \neq (\Sigma'_{\mathbb{i}})^\omega$. Then, there exists a word $w \in (\Sigma'_{\mathbb{i}})^\omega$ such that $w \notin L(\mathcal{A})$. For any PDT $\mathcal{T}$ and any $u \in (\Sigma'_{\mathbb{i}})^\omega$, there is $v \in (\Sigma'_{\mathbb{o}})^\omega$ such that $\langle u, v \rangle \in L(\mathcal{A}')$. On the other hand, $\langle w, v \rangle \notin L(\mathcal{A}')$ holds for any $v \in (\Sigma'_{\mathbb{o}})^\omega$. Hence, $\mathcal{T} \not\models \mathcal{A}'$ holds for any PDT $\mathcal{T}$. This completes the reduction and the realizability problem for NPDA and PDT is undecidable.

5 Register Pushdown Transducers and Automata

5.1 Data words and registers

We assume a countable set D of *data values*. For finite alphabets $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$, an infinite sequence $(a^{\mathbb{i}}_1, d_1)(a^{\mathbb{o}}, d'_1) \cdots \in ((\Sigma_{\mathbb{i}} \times D) \cdot (\Sigma_{\mathbb{o}} \times D))^\omega$ is called a *data word*. We let $\mathtt{DW}(\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}, D) = ((\Sigma_{\mathbb{i}} \times D) \cdot (\Sigma_{\mathbb{o}} \times D))^\omega$. We define the projection $\mathtt{Lab} : \Sigma \times D \to \Sigma$ as $\mathtt{Lab}((a,d)) = a$ for $(a,d) \in \Sigma \times D$. For $k \in \mathbb{N}_0$, a mapping $\theta : [k] \to D$ is called an *assignment* (of data values to k registers). Let Θ_k denote the collection of assignments to k registers. We assume $\bot \in D$ as the initial data value and let $\theta^k_\bot \in \Theta_k$ be the initial assignment such that $\theta^k_\bot(i) = \bot$ for all $i \in [k]$.

We denote $\mathtt{Tst}_k = \mathscr{P}([k] \cup \{\mathtt{top}\})$ and $\mathtt{Asgn}_k = \mathscr{P}([k])$ where $\mathtt{top} \notin \mathbb{N}$ is a unique symbol that represents a stack top value. $\mathtt{Tst}_k$ is the set of guard conditions. For $\mathtt{tst} \in \mathtt{Tst}_k$, $\theta \in \Theta_k$ and $d, e \in D$, we denote $(\theta, d, e) \models \mathtt{tst}$ if $(\theta(i) = d \Leftrightarrow i \in \mathtt{tst})$ and $(e = d \Leftrightarrow \mathtt{top} \in \mathtt{tst})$ hold. In the definitions of register pushdown transducer and automaton in the next section, the data values d and e correspond to an input data value and a stack top data value, respectively. $\mathtt{Asgn}_k$ is the set of assignment conditions. For $\mathtt{asgn} \in \mathtt{Asgn}_k$, $\theta \in \Theta_k$ and $d \in D$, let $\theta[\mathtt{asgn} \leftarrow d]$ be the assignment $\theta' \in \Theta_k$ such that $\theta'(i) = d$ for $i \in \mathtt{asgn}$ and $\theta'(i) = \theta(i)$ for $i \notin \mathtt{asgn}$.

5.2 Register pushdown transducers

Definition 16. *A k-register pushdown transducer (k-RPDT) over finite alphabets $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ and Γ is $\mathcal{T} = (P, p_0, z_0, \Delta)$ where P is a finite set of states, $p_0 \in P$ is the initial state, $z_0 \in \Gamma$ is the initial stack symbol and $\Delta : P \times \Sigma_{\mathbb{i}} \times \mathtt{Tst}_k \times \Gamma \to P \times \Sigma_{\mathbb{o}} \times \mathtt{Asgn}_k \times [k] \times \mathtt{Com}(\Gamma \times [k])$ is a finite set of deterministic transition rules.*

For $u \in (\Gamma \times D)^+$, $\theta' \in \Theta_k$ and $\mathtt{com} \in \mathtt{Com}(\Gamma \times [k])$, let us define $\mathtt{upds}(u, \theta', \mathtt{com})$ as $\mathtt{upds}(u, \theta', \mathtt{pop}) = u(1:)$, $\mathtt{upds}(u, \theta', \mathtt{skip}) = u$ and $\mathtt{upds}(u, \theta', \mathtt{push}((z, j'))) = (z, \theta'(j'))u$. Let $ID_\mathcal{T} = P \times \Theta_k \times (\Gamma \times D)^*$ and $\Rightarrow_\mathcal{T} \subseteq ID_\mathcal{T} \times ((\Sigma_{\mathbb{i}} \times D) \cdot (\Sigma_{\mathbb{o}} \times D)) \times$

$ID_{\mathcal{T}}$ be the transition relation of $\mathcal{T}$ such that $((p,\theta,u),(a,d^{\mathbb{i}})(b,d^{\mathbb{o}}),(q,\theta',u')) \in \Rightarrow_{\mathcal{T}}$ iff there exists a rule $(p,a,\mathtt{tst},z) \to (q,b,\mathtt{asgn},j,\mathtt{com}) \in \Delta$ that satisfies the following conditions: $(\theta,d^{\mathbb{i}},snd(u(0))) \models \mathtt{tst}$, $\theta' = \theta[\mathtt{asgn} \leftarrow d^{\mathbb{i}}]$, $\theta'(j) = d^{\mathbb{o}}$, $z = fst(u(0))$ and $u' = \mathtt{upds}(u,\theta',\mathtt{com})$, and we write $(p,\theta,u) \Rightarrow_{\mathcal{T}}^{(a,d^{\mathbb{i}})(b,d^{\mathbb{o}})} (q,\theta',u')$.

A run and the language $L(\mathcal{T})$ of $\mathcal{T}$ are those of deterministic 0-TS $(ID_{\mathcal{T}},(q_0,\theta_\perp^k,(z_0,\perp)),(\Sigma_{\mathbb{i}} \times D)\cdot(\Sigma_{\mathbb{o}} \times D),\emptyset,\Rightarrow_{\mathcal{T}},c)$ where $c(s) = 2$ for all $s \in ID_{\mathcal{T}}$. In this paper, we assume that no run of RPDT reaches an ID whose stack is empty. Let $\mathbf{RPDT}[k]$ be the class of k-RPDT and $\mathbf{RPDT}{=} \bigcup_{k\in\mathbb{N}_0}\mathbf{RPDT}[k]$.

5.3 Register pushdown automata

Definition 17. *A nondeterministic k-register pushdown automaton (k-NRPDA) over $\Sigma_{\mathbb{i}},\Sigma_{\mathbb{o}}$ and Γ is $\mathcal{A} = (Q,Q_{\mathbb{i}},Q_{\mathbb{o}},q_0,z_0,\delta,c)$, where Q is a finite set of states, $Q_{\mathbb{i}} \cup Q_{\mathbb{o}} = Q, Q_{\mathbb{i}} \cap Q_{\mathbb{o}} = \emptyset$, $q_0 \in Q$ is the initial state, $z_0 \in \Gamma$ is the initial stack symbol, $c : Q \to [n]$ where $n \in \mathbb{N}$ is the number of priorities and $\delta : Q \times (\Sigma \cup \{\tau\}) \times \mathtt{Tst}_k \times \Gamma \to \mathscr{P}(Q \times \mathtt{Asgn}_k \times \mathtt{Com}(\Gamma \times [k]))$ is a transition function having one of the forms:*

- $(q_{\mathbb{x}},a_{\mathbb{x}},\mathtt{tst},z) \to (q_{\overline{\mathbb{x}}},\mathtt{asgn},\mathtt{com})$ *(input/output rule)*
- $(q_{\mathbb{x}},\tau,\mathtt{tst},z) \to (q'_{\mathbb{x}},\mathtt{asgn},\mathtt{com})$ *(τ rule)*

where $(\mathbb{x},\overline{\mathbb{x}}) \in \{(\mathbb{i},\mathbb{o}),(\mathbb{o},\mathbb{i})\}$, $q_{\mathbb{x}},q'_{\mathbb{x}} \in Q_{\mathbb{x}}, q_{\overline{\mathbb{x}}} \in Q_{\overline{\mathbb{x}}}, a_{\mathbb{x}} \in \Sigma_{\mathbb{x}}$, $\mathtt{tst} \in \mathtt{Tst}_k$, $z \in \Gamma$, $\mathtt{asgn} \in \mathtt{Asgn}_k$ and $\mathtt{com} \in \mathtt{Com}(\Gamma \times [k])$.

Let $ID_{\mathcal{A}} = Q \times \Theta_k \times (\Gamma \times D)^*$. We define the transition relation $\vdash_{\mathcal{A}} \subseteq ID_{\mathcal{A}} \times ((\Sigma \cup \{\tau\}) \times D) \times ID_{\mathcal{A}}$ as $((q,\theta,u),(a,d),(q',\theta',u')) \in \vdash_{\mathcal{A}}$, written as $(q,\theta,u) \vdash_{\mathcal{A}}^{(a,d)} (q',\theta',u')$, iff there exists a rule $(p,a,\mathtt{tst},z) \to (q,\mathtt{asgn},\mathtt{com}) \in \delta$ such that $(\theta,d,snd(u(0))) \models \mathtt{tst}$, $\theta' = \theta[\mathtt{asgn} \leftarrow d]$, $z = fst(u(0))$ and $u' = \mathtt{upds}(u,\theta',\mathtt{com})$. For $s,s' \in ID_{\mathcal{A}}$ and $w \in ((\Sigma_{\mathbb{i}} \times D)\cdot(\Sigma_{\mathbb{o}} \times D))^m$, we write $s \vdash^w s'$ if there exists $\rho \in ID_{\mathcal{A}}^{m+1}$ such that $\rho(0) = s, \rho(m) = s'$, and $\rho(0) \vdash^{w(0)} \cdots \vdash^{w(m-1)} \rho(m)$.

A run and the language $L(\mathcal{A})$ of k-DRPDA $\mathcal{A}$ are those of TS $\mathcal{S}_{\mathcal{A}} = (ID_{\mathcal{A}},(q_0,\theta_\perp^k,(z_0,\perp)),\Sigma \times D,\{\tau\} \times D,\vdash_{\mathcal{A}},c')$ where $c'((q,\theta,u)) = c(q)$ for all $(q,\theta,u) \in ID_{\mathcal{A}}$. We call $\mathcal{A}$ deterministic, or k-DRPDA, if $\mathcal{S}_{\mathcal{A}}$ is deterministic. We call $\mathcal{A}$ an (m,k)-NRPDA (or an (m,k)-DRPDA when $\mathcal{A}$ is deterministic) if $\mathcal{S}_{\mathcal{A}}$ is an m-TS. We abbreviate $(0,k)$-NRPDA ($(0,k)$-DPDA) as k-NRPDA (k-DRPDA). Let **DRPDA** and **NRPDA** be the unions of k-DRPDA and k-NRPDA for all $k \in \mathbb{N}_0$, respectively.

For simplicity, we assume that the set of stack alphabet Γ is the singleton $\{z\}$. We abbreviate k-RPDT $\mathcal{T} = (P,p_0,z_0,\Delta)$ as $\mathcal{T} = (P,p_0,\Delta)$, the set of all IDs of k-RPDT $P \times \Theta_k \times (\Gamma \times D)^+$ as $P \times \Theta_k \times D^+$, the stack command $\mathtt{Com}(\Gamma \times [k])$ as $\mathtt{Com}([k])$, every rule $(p,a,\mathtt{tst},z) \to (q,b,\mathtt{asgn},j,\mathtt{com})$ of $\mathcal{T}$ as $(p,a,\mathtt{tst}) \to (q,b,\mathtt{asgn},j,\mathtt{com}')$ where $\mathtt{com} \in \mathtt{Com}(\Gamma \times [k])$ and $\mathtt{com}' \in \mathtt{Com}([k])$ such that $\mathtt{com}' = \mathtt{com}$ if $\mathtt{com} = \mathtt{pop}$ or $\mathtt{skip}$ and $\mathtt{com}' = \mathtt{push}(j')$ if $\mathtt{com}' = \mathtt{push}(z,j')$ for some $j' \in [k]$. except in the proof of Theorem 24. We apply a similar abbreviation to those of RPDA.

Example 18. Let us consider 1-RPDT $\mathcal{T} = (\{p\}, p, \Delta)$ over $\{a\}, \{b\}$ and Γ $(= \{z\})$ where $\Delta = \{(p, a, \{1, \mathtt{top}\}) \to (p, b, \emptyset, 1, \mathtt{skip}),\ (p, a, \emptyset) \to (p, b, \{1\}, 1, \mathtt{push}(1))\}$ Let $\rho = (p, [\bot], \bot)(p, [d_1], d_1\bot)(p, [d_1], d_1\bot)(p, [d_2], d_2d_1\bot)\cdots$, $[d] \in \Theta_1$ be the assignment such that $[d](1) = d$, and $w = (a, d_1)(b, d_1)(a, d_1)(b, d_1)(a, d_2)(b, d_2)\cdots$, then (ρ, w) is a run of $\mathcal{T}$.

5.4 Visibly RPDA

Let $\mathtt{Com}_v = \{\mathtt{pop}, \mathtt{skip}, \mathtt{push}\}$ and $v : \mathtt{Com}([k]) \to \mathtt{Com}_v$ be the function such that $v(\mathtt{push}(j)) = \mathtt{push}$ for $j \in [k]$ and $v(\mathtt{com}) = \mathtt{com}$ otherwise. We say that a k-DRPDA $\mathcal{A}$ over $\Sigma_{\mathbb{i}}, \Sigma_{\circledcirc}$ and Γ visibly manipulates its stack (or a *stack-visibly* RPDA) if there exists a function $\mathtt{vis} : \Sigma \to \mathtt{Com}_v$ such that every rule $(q, a, \mathtt{tst}) \to (q', \mathtt{asgn}, \mathtt{com})$ of $\mathcal{A}$ satisfies $\mathtt{vis}(a) = v(\mathtt{com})$. We define a stack-visibly PDA in a similar way. Stack-visibility of RPDA will be used in the proof of Lemma 22 in order to take the intersection of the two DRPDA.

Also, we say that $\mathcal{A}$ is a *test-visibly* DRPDA if there exists a function $\mathtt{vis}_t : \Sigma \to \mathtt{Tst}_k$ such that every rule $(q, a, \mathtt{tst}) \to (q', \mathtt{asgn}, \mathtt{com})$ of $\mathcal{A}$ satisfies $\mathtt{vis}_t(a) = \mathtt{tst}$. In the next subsection, we prove that the projection of the language recognized by a NRPDA $\mathcal{A}$ onto the finite alphabets can be recognized by a NPDA $\mathcal{A}'$, and if $\mathcal{A}$ is a test-visibly DRPDA, $\mathcal{A}'$ is deterministic.

If $\mathcal{A}$ is a stack-visibly and test-visibly DRPDA, we call $\mathcal{A}$ a visibly DRPDA. Let **DRPDAv** be the union of visibly k-DRPDA for all $k \in \mathbb{N}_0$, respectively.

5.5 PDA simulating RPDA

In this subsection, we show that we can construct an NPDA $\mathcal{A}'$ from a given k-NRPDA $\mathcal{A}$ over $\Sigma_{\mathbb{i}}$, $\Sigma_{\circledcirc}$, Γ such that $\mathtt{Lab}(L(\mathcal{A})) = L(\mathcal{A}')$.

Let Φ_k be the set of *equivalence relations* over the set of $2k + 1$ symbols $X_k = \{\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_k, \mathtt{x}'_1, \mathtt{x}'_2, \ldots, \mathtt{x}'_k, \mathtt{x}_{\mathtt{top}}\}$. We write $a \equiv_\phi b$ and $a \not\equiv_\phi b$ to mean $(a, b) \in \phi$ and $(a, b) \notin \phi$, respectively, for $a, b \in X_k$ and $\phi \in \Phi_k$. Intuitively, each $\phi \in \Phi_k$ represents the equality and inequality among the data values in the registers and the stack top, as well as the transfer of the values in the registers between two assignments. Two assignments θ, θ' and a value d at the stack top satisfy ϕ, denoted as $\theta, d, \theta' \models \phi$, if and only if for $i, j \in [k]$,

$$\begin{aligned}
&\mathtt{x}_i \equiv_\phi \mathtt{x}_j \Leftrightarrow \theta(i) = \theta(j), && \mathtt{x}_i \equiv_\phi \mathtt{x}_{\mathtt{top}} \Leftrightarrow \theta(i) = d,\\
&\mathtt{x}_i \equiv_\phi \mathtt{x}'_j \Leftrightarrow \theta(i) = \theta'(j), && \mathtt{x}'_j \equiv_\phi \mathtt{x}_{\mathtt{top}} \Leftrightarrow \theta'(j) = d,\\
&\mathtt{x}'_i \equiv_\phi \mathtt{x}'_j \Leftrightarrow \theta'(i) = \theta'(j).
\end{aligned}$$

Let $\phi_\bot \in \Phi_k$ be the equivalence relation satisfying $a \equiv_{\phi_\bot} b$ for any $a, b \in X_k$.

For $\mathtt{tst} \subseteq [k] \cup \{\mathtt{top}\}$ and $\mathtt{asgn} \subseteq [k]$, define a subset $\Phi_k^{\mathtt{tst},\mathtt{asgn}}$ of Φ_k as:

$$\begin{aligned}
\Phi_k^{\mathtt{tst},\mathtt{asgn}} = \{\phi \in \Phi_k \mid\ & (\forall i \in \mathtt{tst} :\ \forall j \in [k] \cup \{\mathtt{top}\} : j \in \mathtt{tst} \Leftrightarrow \mathtt{x}_i \equiv_\phi \mathtt{x}_j),\\
& (\forall i \in \mathtt{asgn} : \forall j \in [k] \cup \{\mathtt{top}\} : j \in \mathtt{tst} \Leftrightarrow \mathtt{x}_j \equiv_\phi \mathtt{x}'_i),\\
& (\forall i, j \in \mathtt{asgn} : \mathtt{x}'_i \equiv_\phi \mathtt{x}'_j),\ (\forall i \in [k] \setminus \mathtt{asgn} : \mathtt{x}_i \equiv_\phi \mathtt{x}'_i)\}.
\end{aligned}$$

For $j \in [k]$, define $\Phi_k^{=,j} = \{\phi \in \Phi_k \mid \mathtt{x_{top}} \equiv_\phi \mathtt{x}_j,\ \forall i \in [k] : \mathtt{x}_i \equiv_\phi \mathtt{x}'_i\}$. By definition, $\theta, e, \theta' \models \phi$ for $\phi \in \Phi_k^{\mathtt{tst},\mathtt{asgn}}$ iff $(\theta, d, e) \models \mathtt{tst}$ and $\theta' = \theta[\mathtt{asgn} \leftarrow d]$ for some $d \in D$. Similarly, $\theta, e, \theta' \models \phi$ for $\phi \in \Phi_k^{=,j}$ iff $\theta' = \theta$ and $\theta(j) = e$.

Let $\odot$ and $\odot_\mathtt{T}$ be binary predicates over Φ_k defined as:

$$\phi_1 \odot \phi_2 \ :\Leftrightarrow\ \big(\mathtt{x}'_i \equiv_{\phi_1} \mathtt{x}'_j \Leftrightarrow \mathtt{x}_i \equiv_{\phi_2} \mathtt{x}_j \text{ for } i, j \in [k]\big).$$
$$\phi_1 \odot_\mathtt{T} \phi_2 \ :\Leftrightarrow\ \big(\phi_1 \odot \phi_2 \text{ and } (\mathtt{x}'_i \equiv_{\phi_1} \mathtt{x_{top}} \Leftrightarrow \mathtt{x}_i \equiv_{\phi_2} \mathtt{x_{top}} \text{ for } i \in [k])\big).$$

Below we will define the *composition* of two equivalence relations, and $\phi_1 \odot \phi_2$ means that ϕ_1 and ϕ_2 are composable. For $\phi \in \Phi_k$ and $\Phi' \subseteq \Phi_k$, let $\phi \odot \Phi' = \{\phi' \in \Phi' \mid \phi \odot \phi'\}$ and $\phi \odot_\mathtt{T} \Phi' = \{\phi' \in \Phi' \mid \phi \odot_\mathtt{T} \phi'\}$. By definition, $\phi \odot_\mathtt{T} \Phi_k^{\mathtt{tst},\mathtt{asgn}}$ consists of at most one equivalence relation for any $\phi \in \Phi_k$, $\mathtt{tst} \subseteq [k] \cup \{\mathtt{top}\}$, and $\mathtt{asgn} \subseteq [k]$. Similarly, $\phi \odot \Phi_k^{=,j}$ consists of exactly one equivalence relation for any $\phi \in \Phi_k$ and $j \in [k]$.

For $\phi_1, \phi_2 \in \Phi_k$ with $\phi_1 \odot \phi_2$, the *composition* $\phi_1 \circ \phi_2$ of them is the equivalence relation in Φ_k that satisfies the followings:

$$\begin{aligned}
\mathtt{x}_i \equiv_{\phi_1} \mathtt{x}_j \Leftrightarrow \mathtt{x}_i \equiv_{\phi_1 \circ \phi_2} \mathtt{x}_j \quad & \text{for } i, j \in [k] \cup \{\mathtt{top}\},\\
\mathtt{x}'_i \equiv_{\phi_2} \mathtt{x}'_j \Leftrightarrow \mathtt{x}'_i \equiv_{\phi_1 \circ \phi_2} \mathtt{x}'_j \quad & \text{for } i, j \in [k],\\
(\exists l \in [k] : \mathtt{x}_i \equiv_{\phi_1} \mathtt{x}'_l \wedge \mathtt{x}_l \equiv_{\phi_2} \mathtt{x}'_j) \Leftrightarrow \mathtt{x}_i \equiv_{\phi_1 \circ \phi_2} \mathtt{x}'_j \quad & \text{for } i \in [k] \cup \{\mathtt{top}\},\ j \in [k].
\end{aligned}$$

By definition, $\circ$ is associative. We say that $\theta_1, d_1, \theta_2, \theta_3$ satisfy the *freshness* property if for every $i, j \in [k]$, $\big(\theta_1(i) \neq \theta_2(l)$ for all $l \in [k]$ implies $\theta_1(i) \neq \theta_3(j)\big)$ and $\big(d_1 \neq \theta_2(l)$ for all $l \in [k]$ implies $d_1 \neq \theta_3(j)\big)$. By definition, if $\theta_1, d_1, \theta_2 \models \phi_1$ and $\theta_2, d_2, \theta_3 \models \phi_2$ and $\theta_1, d_1, \theta_2, \theta_3$ satisfy the freshness property, then $\theta_1\, d_1, \theta_3 \models \phi_1 \circ \phi_2$. We extend the freshness property to a sequence $\theta_0, d_0, \theta_1, d_1, \ldots, d_{n-1}, \theta_n$. This sequence satisfies the freshness property if $\theta_i, d_i, \theta_l, \theta_j$ satisfy the property for every i, l, j such that $0 \leq i < l < j \leq n$.

Let $\mathcal{A} = (Q, Q_\mathbb{i}, Q_\mathbb{o}, q_0, \delta, c)$ be a k-NRPDA over $\Sigma_\mathbb{i}$, $\Sigma_\mathbb{o}$, and Γ. As mentioned in Section 5.3, we assume that Γ is a singleton and use the simplified definition of δ for readability. Note that we can extend the following results to arbitrary Γ by replacing Φ_k used as the stack alphabet of the constructed PDA with $\Gamma \times \Phi_k$. From $\mathcal{A}$, we construct a PDA $\mathcal{A}' = (Q', Q'_\mathbb{i}, Q'_\mathbb{o}, q'_0, \phi_\bot, \delta', c')$ over $\Sigma_\mathbb{i}$, $\Sigma_\mathbb{o}$, and Φ_k, where $Q' = Q \times \Phi_k$, $Q'_\mathbb{i} = Q_\mathbb{i} \times \Phi_k$, $Q'_\mathbb{o} = Q_\mathbb{o} \times \Phi_k$, $q'_0 = (q_0, \phi_\bot)$, $c'((q, \phi)) = c(q)$ for any $q \in Q$ and $\phi \in \Phi_k$, and for any $(q, \phi_2) \in Q'$, $a \in \Sigma \cup \{\tau\}$, and $\phi_1 \in \Phi_k$, $\delta'((q, \phi_2), a, \phi_1)$ is the smallest set satisfying the following inference rules:

$$\frac{\delta(q, a, \mathtt{tst}) \ni (q', \mathtt{asgn}, \mathtt{skip}),\ \phi_1 \odot \phi_2,\ \phi_3 \in \phi_2 \odot_\mathtt{T} \Phi_k^{\mathtt{tst},\mathtt{asgn}}}{\delta'((q, \phi_2), a, \phi_1) \ni ((q', \phi_2 \circ \phi_3), \mathtt{skip})} \tag{1}$$

$$\frac{\delta(q, a, \mathtt{tst}) \ni (q', \mathtt{asgn}, \mathtt{pop}),\ \phi_1 \odot \phi_2,\ \phi_3 \in \phi_2 \odot_\mathtt{T} \Phi_k^{\mathtt{tst},\mathtt{asgn}}}{\delta'((q, \phi_2), a, \phi_1) \ni ((q', \phi_1 \circ \phi_2 \circ \phi_3), \mathtt{pop})} \tag{2}$$

$$\frac{\delta(q, a, \mathtt{tst}) \ni (q', \mathtt{asgn}, \mathtt{push}(j)),\ \phi_1 \odot \phi_2,\ \phi_3 \in \phi_2 \odot_\mathtt{T} \Phi_k^{\mathtt{tst},\mathtt{asgn}},\ \phi_4 \in \phi_3 \odot \Phi_k^{=,j}}{\delta'((q, \phi_2), a, \phi_1) \ni ((q', \phi_4), \mathtt{push}(\phi_2 \circ \phi_3))} \tag{3}$$

Note that if $\mathcal{A}$ is a test-visibly DRPDA, $\mathcal{A}'$ is deterministic. The number of equivalence relations in Φ_k equals the $(2k+1)$th Bell number and is $2^{O(k \log k)}$. When constructing δ', we choose arbitrary ϕ_1 and ϕ_2 for each transition rule of $\mathcal{A}$ in general, and thus the size of $\mathcal{A}'$ is exponential to the one of $\mathcal{A}$.

The main idea of this construction is as follows: $\mathcal{A}'$ simulates $\mathcal{A}$ without keeping data values in the stack. When `pop` is performed, $\mathcal{A}'$ must know whether or not the data value in the new stack top of $\mathcal{A}$ equals the *current* value of each register. For this purpose, $\mathcal{A}'$ keeps an abstract "history" of the register assignments in the stack, which tells whether each of the data values in the stack of $\mathcal{A}$ equals the current value of each register. The precise meanings of the stack of $\mathcal{A}'$ will become clear by considering the `Lab`-bisimulation relation shown in the proof of Lemma 19 below.

Let $\mathcal{S}_{\mathcal{A}'} = (ID_{\mathcal{A}'}, (q'_0, \phi_\perp), \Sigma, \{\tau\}, \vdash_{\mathcal{A}'}, c_{\mathcal{A}'})$ be the TS that represents the semantics of $\mathcal{A}'$. We define a TS $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}} = (ID^{\mathrm{aug}}_{\mathcal{A}}, (q_0, \theta_\perp, (\perp, \theta_\perp)), \Sigma \times D, \{\tau\} \times D, \vdash_{\mathcal{A}^{\mathrm{aug}}}, c_{\mathcal{A}})$ where $ID^{\mathrm{aug}}_{\mathcal{A}} = Q \times \Theta_k \times (D \times \Theta_k)^*$, $c_{\mathcal{A}}((q, \theta, v)) = c(q)$ for any $(q, \theta, v) \in ID^{\mathrm{aug}}_{\mathcal{A}}$, and $\vdash_{\mathcal{A}^{\mathrm{aug}}}$ is defined as follows: $(q, \theta, v) \vdash^{(a,d)}_{\mathcal{A}^{\mathrm{aug}}} (q', \theta', v')$ iff $\delta(q, a, \mathtt{tst}) \ni (q', \mathtt{asgn}, \mathtt{com})$, $(\theta, d, \mathit{fst}(v(0))) \models \mathtt{tst}$, $\theta' = \theta[\mathtt{asgn} \leftarrow d]$, and $v' = v(1{:})$, v, or $(\theta'(j'), \theta')v$ if $\mathtt{com} = \mathtt{pop}$, `skip`, or $\mathtt{push}(j')$, respectively. $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ is essentially the same as the TS $\mathcal{S}_{\mathcal{A}}$ for $\mathcal{A}$, but $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ "saves" the current register assignment in the stack when performing `push`. The saved assignments do not take part in transitions and thus the behavior of $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ is the same as $\mathcal{S}_{\mathcal{A}}$. Additionally, we define the freshness property of $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ as follows: A transition $(q, \theta, v) \vdash^{(a,d)}_{\mathcal{A}^{\mathrm{aug}}} (q', \theta', v')$ of $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ by a rule $\delta(q, a, \mathtt{tst}) \ni (q', \mathtt{asgn}, \mathtt{com})$ with $\mathtt{tst} = \emptyset$ and $\mathtt{asgn} \neq \emptyset$ is allowed only when d does not appear in any saved assignment in v. Intuitively, this property means that when `tst` designates a data value not in the registers or the stack top, the RPDA chooses a *fresh* data value that has never been used before. We assume that $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ satisfies the freshness property. This assumption guarantees that for every $(q, \theta_n, (d_{n-1}, \theta_{n-1}) \dots (d_1, \theta_1)(d_0, \theta_0)) \in ID^{\mathrm{aug}}_{\mathcal{A}}$ reachable from the initial state of $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$, the sequence $\theta_0, d_0, \theta_1, d_1, \dots, d_{n-1}, \theta_n$ satisfies the freshness property.

Each equivalence relation in the stack of $\mathcal{A}'$ represents the relation among each data value in the stack and two adjacent saved assignments of $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$, which yields `Lab`-bisimilarity from $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ to $\mathcal{S}_{\mathcal{A}'}$, as shown in the following lemma.

Lemma 19. *If $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ satisfies the freshness property, then $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ is* `Lab`*-bisimilar to $\mathcal{S}_{\mathcal{A}'}$.*

Proof sketch. Let $R \subseteq ID^{\mathrm{aug}}_{\mathcal{A}} \times ID_{\mathcal{A}'}$ be the smallest relation that satisfies for every $q \in Q$, $\theta_0, \dots, \theta_n \in \Theta_k$, $d_0, \dots, d_{n-1} \in D$, and $\phi_0, \dots, \phi_n \in \Phi_k$, $((q, \theta_n, (d_{n-1}, \theta_{n-1}) \dots (d_1, \theta_1)(d_0, \theta_0)), ((q, \phi_n), \phi_{n-1} \dots \phi_1 \phi_0)) \in R$ if $\forall i \in [n] : \theta_{i-1}, d_{i-1}, \theta_i \models \phi_i$ and $\theta_\perp, \perp, \theta_0 \models \phi_0$. If $((q, \theta_n, v), ((q, \phi_n), u)) \in R$ and $(q, \theta_n, v) \vdash^{(a,d)}_{\mathcal{A}^{\mathrm{aug}}} (q', \theta', v')$, then by the definition of δ', $((q, \phi_n), u) \vdash^{a}_{\mathcal{A}'} ((q', \phi''), u')$ such that $((q', \theta', v'), ((q', \phi''), u')) \in R$. Conversely, if $((q, \theta_n, v), ((q, \phi_n), u)) \in R$ and $((q, \phi_n), u) \vdash^{a}_{\mathcal{A}'} ((q', \phi''), u')$, then by the definition of δ', there must be a transition of $\mathcal{A}$ that enables $(q, \theta_n, v) \vdash^{(a,d)}_{\mathcal{A}^{\mathrm{aug}}} (q', \theta', v')$ such that

$((q', \theta', v'), ((q', \phi''), u')) \in R$. Therefore, R is a $\mathtt{Lab}$-bisimulation relation from $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ to $\mathcal{S}_{\mathcal{A}'}$.

By Lemmas 2 and 19, we obtain the following theorem.

Theorem 20. *For a given (m,k)-NRPDA (resp. test-visibly (m,k)-DRPDA) $\mathcal{A}$, we can construct an m-NPDA (resp. m-DPDA) $\mathcal{A}'$ such that $\mathtt{Lab}(L(\mathcal{A})) = L(\mathcal{A}')$, if we assume the freshness property on the semantics of $\mathcal{A}$.*

6 Realizability problems for RPDA and RPDT

6.1 Finite actions

In [19], the abstraction of the behavior of k-register transducer (k-RT), called finite actions, was introduced to reduce the realizability problem for register automata (RA) and RT to the problem on finite alphabets. We extend the idea of [19] and define the finite actions of k-RPDT.

For $k \in \mathbb{N}_0$, we define the set of finite input actions as $A^{\mathbb{i}}_k = \Sigma_{\mathbb{i}} \times \mathtt{Tst}_k$ and the set of finite output actions as $A^{\mathbb{o}}_k = \Sigma_{\mathbb{o}} \times \mathtt{Asgn}_k \times [k] \times \mathtt{Com}([k])$. Note that $\mathtt{Com}([k])$ appearing in the definition of $A^{\mathbb{o}}_k$ is not the abbreviation of $\mathtt{Com}(\Gamma \times [k])$. Finite actions have no information on finite stack alphabet Γ even if Γ is not a singleton. A sequence $w = (a^{\mathbb{i}}_0, d^{\mathbb{i}}_0)(a^{\mathbb{o}}_0, d^{\mathbb{o}}_0) \cdots \in \mathtt{DW}(\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}, D)$ is *compatible* with a sequence $\overline{a} = (a^{\mathbb{i}}_0, \mathtt{tst}_0)(a^{\mathbb{o}}_0, \mathtt{asgn}_0, j_0, \mathtt{com}_0) \cdots \in (A^{\mathbb{i}}_k \cdot A^{\mathbb{o}}_k)^\omega$ iff there exists a sequence $(\theta_0, u_0)(\theta_1, u_1) \cdots \in (\Theta_k \times D^*)^\omega$, called a *witness*, such that $\theta_0 = \theta^k_\perp$, $u_0 = \perp$, $(\theta_i, d^{\mathbb{i}}_i, u_i(0)) \models \mathtt{tst}_i, \theta_{i+1} = \theta_i[\mathtt{asgn}_i \leftarrow d^{\mathbb{i}}_i], \theta_{i+1}(j_i) = d^{\mathbb{o}}_i$ and $u_{i+1} = \mathtt{upds}(u_i, \theta_{i+1}, \mathtt{com}_i)$. Let $\mathtt{Comp}(\overline{a}) = \{w \in \mathtt{DW}(\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}, D) \mid w \text{ is compatible with } \overline{a}\}$. For a specification $S \subseteq \mathtt{DW}(\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}, D)$, we define $W_{S,k} = \{\overline{a} \mid \mathtt{Comp}(\overline{a}) \subseteq S\}$.

For a data word $w \in \mathtt{DW}(\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}, D)$ and a sequence $\overline{a} \in (A^{\mathbb{i}}_k \cdot A^{\mathbb{o}}_k)^\omega$ such that for each $i \geq 0$, there exists $a \in \Sigma$ and we can write $w(i) = (a, d)$ and $\overline{a}(i) = (a, \mathtt{tst})$ if i is even and $\overline{a}(i) = (a, \mathtt{asgn}, j, \mathtt{com})$ if i is odd, we define $w \otimes \overline{a} \in \mathtt{DW}(A^{\mathbb{i}}_k, A^{\mathbb{o}}_k, D)$ as $w \otimes \overline{a}(i) = (\overline{a}(i), d)$ where $w(i) = (a, d)$.

6.2 Decidability and undecidability of realizability problems

Lemma 21. *$L_k = \{w \otimes \overline{a} \mid w \in \mathtt{Comp}(\overline{a})\}$ is definable as the language of a $(2, k+2)$-DRPDA.*

Proof sketch. Let $(2,k+2)$-DRPDA $\mathcal{A}_1 = (Q_1, Q^{\mathbb{i}}_1, Q^{\mathbb{o}}_1, p, \delta_1, c_1)$ over $A^{\mathbb{i}}_k, A^{\mathbb{o}}_k$ and Γ where $Q_1 = \{p, q\} \cup (\mathtt{Asgn}_k \times [k] \times \mathtt{Com}([k])) \cup [k]$, $Q^{\mathbb{i}}_1 = \{p\}$, $Q^{\mathbb{o}}_1 = Q_1 \setminus Q^{\mathbb{i}}_1$, $c_1(s) = 2$ for every $s \in Q$ and δ_1 consists of all the rules of the form

$$(p, (a_{\mathbb{i}}, \mathtt{tst}), \mathtt{tst} \cup \mathtt{tst}') \to (q, \{k+1\}, \mathtt{skip}) \tag{4}$$

$$(q, (a_{\mathbb{o}}, \mathtt{asgn}, j, \mathtt{com}), \mathtt{tst}'') \to ((\mathtt{asgn}, j, \mathtt{com}), \{k+2\}, \mathtt{skip}) \tag{5}$$

$$((\mathtt{asgn}, j, \mathtt{com}), \tau, \{k+1\} \cup \mathtt{tst}'') \to (j, \mathtt{asgn}, \mathtt{com}) \tag{6}$$

$$(j, \tau, \{j, k+2\} \cup \mathtt{tst}'') \to (p, \emptyset, \mathtt{skip}) \tag{7}$$

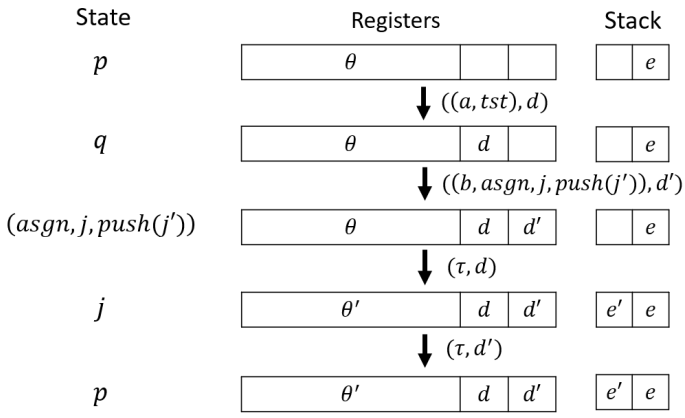

Fig. 1. An example of transitions of $\mathcal{A}_k$.

for $(a_{\mathbb{i}}, \mathtt{tst}) \in A^{\mathbb{i}}_k$, $(a_{\mathbb{o}}, \mathtt{asgn}, j, \mathtt{com}) \in A^{\mathbb{o}}_k$, $\mathtt{tst}' \subseteq \{k+1, k+2\}$ and $\mathtt{tst}'' \in \mathtt{Tst}_{k+2}$. As in Fig. 1, $\mathcal{A}_1$ checks whether an input sequence satisfies the conditions of compatibility by nondeterministically generating a candidate of a witness of the compatibility step by step.

Lemma 22. *For a specification $\mathcal{S}$ defined by some visibly k'-DRPDA, $L_{\overline{S},k} = \{w \otimes \overline{a} \mid w \in \mathtt{Comp}(\overline{a}) \cap \overline{S}\}$ is definable as the language of a $(4, k+k'+4)$-DRPDA.*

Proof sketch. Let $L_{\overline{S}} = \{w \otimes \overline{a} \mid w \in \overline{S}, \overline{a} \in (A^{\mathbb{i}}_k \cdot A^{\mathbb{o}}_k)^\omega\}$. Because the class of languages defined by visibly DRPDA is closed under the complement, we can construct a visibly k'-DRPDA $\mathcal{A}_2 = (Q_2, Q^{\mathbb{i}}_2, Q^{\mathbb{o}}_2, q^0_2, \delta_2, c_2)$ over $A^{\mathbb{i}}_k, A^{\mathbb{o}}_k$ and Γ such that $L(\mathcal{A}_2) = L_{\overline{S}}$. Let $\mathcal{A}_1$ be the $(2, k+2)$-DRPDA such that $L(\mathcal{A}_1) = L_k$, which is given in Lemma 21. Because $L_{\overline{S},k} = L_{\overline{S}} \cap L_k$, we will construct a $(4, k+k'+4)$-DRPDA $\mathcal{A}$ over $A^{\mathbb{i}}_k, A^{\mathbb{o}}_k$ and Γ such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$. We use the properties of $\mathcal{A}_1$ that $c_1(q)$ is even for every $q \in Q_1$ and δ_1 consists of several groups of three consecutive rules having the following forms:

$$(q_1, a, \mathtt{tst}_1) \to (q_2, \mathtt{asgn}_1, \mathtt{skip}) \tag{5'}$$
$$(q_2, \tau, \mathtt{tst}_2) \to (q_3, \mathtt{asgn}_2, \mathtt{com}_1) \tag{6'}$$
$$(q_3, \tau, \mathtt{tst}_3) \to (q_4, \mathtt{asgn}_3, \mathtt{skip}). \tag{7'}$$

Note that $\mathtt{vis}(a) = v(\mathtt{com}_1)$ always holds for such three consecutive rules. $(5'), (6')$ and $(7')$ correspond to $(5), (6)$ and (7), respectively, and (4) is also converted to three consecutive rules like $(5')$-$(7')$ by adding dummy τ rules.

We let $k_1 = k+2$ and $k_2 = k'$. We construct $(4, k_1+k_2+2)$-DRPDA $\mathcal{A} = (Q_{\mathbb{i}} \cup Q_{\mathbb{o}} \cup \{q_0\}, Q_{\mathbb{i}} \cup \{q_0\}, Q_{\mathbb{o}}, q_0, \delta, c)$ where $Q_{\mathbb{i}} = Q^{\mathbb{i}}_1 \times Q^{\mathbb{i}}_2 \times [5]$, $Q_{\mathbb{o}} = Q^{\mathbb{o}}_1 \times Q^{\mathbb{o}}_2 \times [5]$. c is defined as $c(q_0) = 1$ and $c((q_1, q_2, i)) = c_2(q_2)$ for all $(q_1, q_2, i) \in Q$. δ has a τ rule $(q_0, \tau, [k] \cup \{\mathtt{top}\}) \to ((p, q^2_0, 1), \mathtt{push}(1))$. For all rules $(5')$, $(6')$, $(7')$ in δ_1 and $(q, a, \mathtt{tst}) \to (q', \mathtt{asgn}, \mathtt{com}) \in \delta_2$ (8) such that $v(\mathtt{com}_1) = v(\mathtt{com})$ $(= \mathtt{vis}(a))$

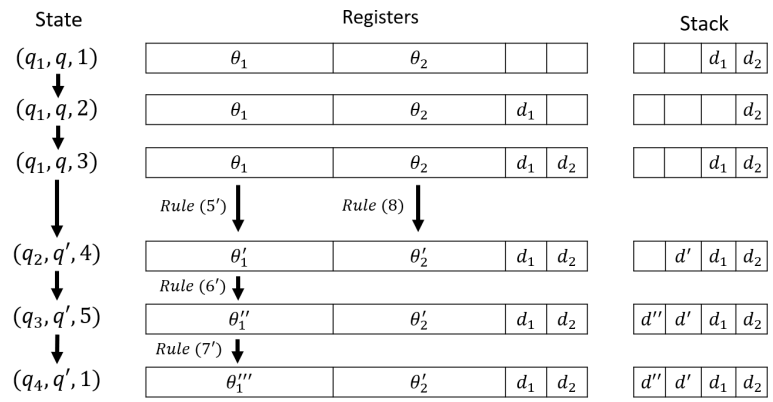

Fig. 2. An example of transitions of $\mathcal{A}$ with $\mathtt{vis}(a) = \mathtt{push}$.

for $a \in A_k^{\overset{\circ}{\mathrm{i}}} \cup A_k^{\circledcirc}$, we construct the rules in δ that can do the transitions as in Fig. 2. The figure illustrates an example of transitions of $\mathcal{A}$ from $(q_1, q, 1)$ to $(q_4, q', 1)$ with updating contents of its registers and stack. The first to k_1-th registers simulate the registers of $\mathcal{A}_1$, (k_1+1)-th to (k_1+k_2)-th registers simulate the registers of $\mathcal{A}_2$ and $(k_1 + k_2 + 1)$-th and $(k_1 + k_2 + 2)$-th registers are for keeping the first and second stack top contents, respectively. The stack contents of $\mathcal{A}$ simulates those of $\mathcal{A}_1$ and $\mathcal{A}_2$ by restoring the contents of stacks of $\mathcal{A}_1$ and $\mathcal{A}_2$ alternately. The transition rules from $(q_1, q, 1)$ to $(q_1, q, 3)$ are for moving the two data values at the stack top to $(k_1 + k_2 + 1)$-th and $(k_1 + k_2 + 2)$-th registers. The transition rule from $(q_1, q, 3)$ to $(q_2, q', 4)$ is for updating states, registers and stacks by simulating the rules (5') and (8). The transition rules from $(q_2, q', 4)$ to $(q_4, q', 1)$ simulate the rules (6') and (7'), respectively. We can show $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ by checking the simulation in Fig. 2 is correct.

Lemma 23. $W_{S,k} = \overline{\mathtt{Lab}(L_{\overline{S},k})}$.

Proof. For every $\overline{a} \in (A_k^{\overset{\circ}{\mathrm{i}}} A_k^{\circledcirc})^\omega$, $\overline{a} \notin W_{S,k} \Leftrightarrow \mathtt{Comp}(\overline{a}) \not\subseteq S \Leftrightarrow \exists w.w \in \mathtt{Comp}(\overline{a}) \cap \overline{S} \Leftrightarrow \exists w.w \otimes \overline{a} \in L_{\overline{S},k} \Leftrightarrow \overline{a} \in \mathtt{Lab}(L_{\overline{S},k})$. Thus, $W_{S,k} = \overline{\mathtt{Lab}(L_{\overline{S},k})}$ holds.

Theorem 24. *For all $k \geq 0$,* Real(**DRPDAv**, **RPDT**$[k]$) *is in 2EXPTIME.*

Proof. By Lemma 22 and Theorem 20, $W_{S,k}$ is definable by a 4-DPDA $\mathcal{A}_f$. By Lemma 7, we can construct a 0-DPDA $\mathcal{A}'_f$ where $L(\mathcal{A}'_f) = L(\mathcal{A}_f)$. By the construction of $\mathcal{A}$ in Lemma 22, the stack height of the current ID increases by two during the transitions from $(q_1, q, 1)$ to $(q_4, q', 1)$ in Fig. 2 if $\mathtt{vis}(a) = \mathtt{push}$, does not change if $\mathtt{vis}(a) = \mathtt{skip}$ and decreases by two if $\mathtt{vis}(a) = \mathtt{pop}$. Thus, $\mathcal{A}'_f$ is a stack-visibly DPDA, that is, every transition rule $(p, a, z) \to (q, \mathtt{com})$ of $\mathcal{A}'_f$ satisfies $\mathtt{vis}(a) = v(\mathtt{com})$. We show the following two conditions are equivalent.

- There exists a k-RPDT $\mathcal{T}$ such that $L(\mathcal{T}) \subseteq S$.
- There exists a PDT $\mathcal{T}'$ such that $L(\mathcal{T}') \subseteq W_{S,k}$.

Assume that a k-RPDT $\mathcal{T}$ over $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ and Γ satisfies $L(\mathcal{T}) \subseteq S$. Then, consider the PDT $\mathcal{T}'$ over $A^{\mathbb{i}}_k, A^{\mathbb{o}}_k$ and Γ such that $(q, (a, \mathtt{tst}), z) \to (q', (b, \mathtt{asgn}, j, \mathtt{com}), \mathtt{com}')$ is a rule of $\mathcal{T}'$ iff $(q, a, \mathtt{tst}, z) \to (q', b, \mathtt{asgn}, j, \mathtt{com})$ is a rule of $\mathcal{T}$ where $\mathtt{com}' = \mathtt{com}$ if $v(\mathtt{com}) = \mathtt{pop}$ or $\mathtt{skip}$ and $\mathtt{com}' = \mathtt{push}(z')$ if $\mathtt{com} = \mathtt{push}(z', j')$ for some $j' \in [k]$. For $\overline{a} \in L(\mathcal{T}')$, every $w \in \mathtt{Comp}(\overline{a})$ has a witness (see Section 6.1) and thus $w \in L(\mathcal{T})$ holds. By the assumption $L(\mathcal{T}) \subseteq S$, $\mathtt{Comp}(\overline{a}) \subseteq S$ holds and thus $\overline{a} \in W_{S,k}$. Hence, we obtain $L(\mathcal{T}') \subseteq W_{S,k}$.

Conversely, assume there exists a PDT $\mathcal{T}'$ over $A^{\mathbb{i}}_k, A^{\mathbb{o}}_k$ and Γ that satisfies $L(\mathcal{T}') \subseteq W_{S,k}$. We can in particular construct a PDT $\mathcal{T}''$ such that $L(\mathcal{T}'') \subseteq W_{S,k}$ and every rule $(q, (a, \mathtt{tst}), z) \to (q', (b, \mathtt{asgn}, j, \mathtt{com}), \mathtt{com}')$ satisfies $\mathtt{vis}(b) = v(\mathtt{com}')$ (note that $\mathtt{vis}(a) = \mathtt{skip}$ always holds) by the construction algorithm in [36]. In the rule, $v(\mathtt{com}) = v(\mathtt{com}')$ holds because $\mathcal{A}'_f$ is a stack-visibly PDA and thus $\mathtt{vis}(b) = v(\mathtt{com})$ holds. Consider the k-RPDT $\mathcal{T}$ over $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ and Γ such that $(q, a, \mathtt{tst}, z) \to (q', b, \mathtt{asgn}, j, \mathtt{com}'')$ is a rule of $\mathcal{T}$ iff $(q, (a, \mathtt{tst}), z) \to (q', (b, \mathtt{asgn}, j, \mathtt{com}), \mathtt{com}')$ is a rule of $\mathcal{T}''$ where $\mathtt{com}'' = \mathtt{com}'$ if $\mathtt{com}' = \mathtt{pop}$ or $\mathtt{skip}$ and $\mathtt{com}'' = \mathtt{push}(z', j')$ if $\mathtt{com}' = \mathtt{push}(z')$ and $\mathtt{com} = \mathtt{push}(j')$. Further, assume $w \in L(\mathcal{T})$, and let $\overline{a} \in (A^{\mathbb{i}}_k \cdot A^{\mathbb{o}}_k)^\omega$ be the sequence with which w is compatible. Then, by the definition of $\mathcal{T}$, $\overline{a} \in L(\mathcal{T}'')$. By the assumption $L(\mathcal{T}'') \subseteq W_{S,k}$, every $\overline{a} \in L(\mathcal{T}'')$ satisfies $\mathtt{Comp}(\overline{a}) \subseteq S$, and thus $w \in S$ holds. Hence, we obtain $L(\mathcal{T}) \subseteq S$.

By the equivalence, we can check Real(**DPDA**, **PDT**) for $\mathcal{A}'_f$, which is shown to be EXPTIME in Theorem 14, instead of checking Real(**DRPDAv**, **RPDT**$[k]$). Because the size of $\mathcal{A}'_f$ is exponential to $k + k'$, Real(**DRPDAv**, **RPDT**$[k]$) is in 2EXPTIME.

Theorem 25. *For all $k \geq 0$,* Real(**NRPDA**, **RPDT**$[k]$) *is undecidable.*

Proof. We can easily reduce the Real(**NPDA**, **PDT**), whose undecidability is proved in Theorem 15, to this problem.

7 Conclusion

We have discussed the realizability problem whose specification and implementation are DPDA (NPDA) and PDT in Section 4. By using the result in [36], we show Real(**DPDA**,**PDT**) is in EXPTIME. We also show the undecidability of Real(**NPDA**,**PDT**) by a reduction from the universality problem of NPDA. In Section 5, we have defined RPDT, RPDA and ~~shown~~ a way to recognize the label of the language of RPDA by PDA. We have introduced the notions of stack-visibly [1] and test-visibly to discuss the decidability of realizability for RPDA and RPDT. We show that the behavior of registers and stack of RPDT can be simulated by the finite ~~alphabels~~ defined as finite actions, and prove that Real(**DRPDAv**,**RPDT**$[k]$) can be reduced to Real(**DPDA**,**PDT**) and is in 2EXPTIME.

It is still unknown whether the realizability problem is decidable in several cases such as a specification is given by a universal PDA, DRPDA has no restriction on visibility and the number of registers of RPDT is not given. Investigating these cases are future work.

References

[1] R. Alur and P. Madhusudan, Visibly pushdown languages, 36th ACM Symp. Theory of Computing (STOC 2004).

[2] R. Bloem, K. Chatterjee and B. Jobstmann, Graph Games and Reactive Synthesis, E. M. Clarke et al. (eds.), Handbook of Model Checking, Chapter 27, 921–962, Springer, 2018.

[3] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick and L. Segoufin, Two-variable logic on data words, ACM Trans. Computational Logic 12(4), 2011.

[4] M. Bojańczyk, B. Klin, S. Lasota, Automata theory in nominal sets, Logical Methods in Computer Science, 10(3:4), 1–44, 2014.

[5] B. Bollig, A. Cyriac, P. Gastin and K. N. Kumar, Model checking languages of data words, 15th Int. Conf. Foundations of Software Science and Computation Structures (FoSSaCS 2012), 391–405.

[6] A. Bouajjani, J. Esparza and O. Maler, Reachability analysis of pushdown automata: Application to model-checking, 8th Int. Conf. on Concurrency Theory (CONCUR 1997), 135–150.

[7] P. Bouyer, A logical characterization of data languages, Inform. Process. Lett. 84(2), 75–85, 2002.

[8] J. R. Büchi and L. H. Landweber, Solving sequential conditions by finite-state strategies, Trans. American Mathematical Society 138, 295–311, 1969.

[9] Y-F. Chen, O. Lengál, T. Tan and Z. Wu, Register automata with linear arithmetic, 32nd Annual ACM/IEEE Symp. on Logic in Computer Science (LICS 2017).

[10] E. Y. C. Cheng and M. Kaminski, Context-free languages over infinite alphabets, Acta Informatica 35, 245–267, 1998.

[11] E. M. Clarke, O. Grumberg and D. A. Peled, Model checking. MIT Press, 2001.

[12] L. Clemente and S Lasota, Reachability analysis of first-order definable pushdown systems, 24th EACSL Annual Conf. on Computer Science Logic (CSL 2015), 244–259.

[13] L. D'Antoni, T. Ferreira, M. Sammartino and A. Silva, Symbolic register automata, 31th Int. Conf. on Computer Aided Verification (CAV 2019).

[14] S. Demri and R. Lazić, LTL with freeze quantifier and register automata, ACM Trans. on Computational Logic 10(3), 2009.

[15] S. Demri, R. Lazić and D. Nowak, On the freeze quantifier in constraint LTL: Decidability and complexity, Inform. Comput. 205(1), 2–24, 2007.

[16] R. Ehlers, S. A. Seshia and H. Kress-Gazit, Synthesis with identifiers, 15th Int. Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI 2014), 415–433.

[17] J. Esparza, D. Hansel, P. Rossmanith and S. Schwoon, Efficient algorithms for model checking pushdown systems, 12th Int. Conf. on Computer Aided Verification (CAV 2000), 232–247.

[18] J. Esparza, A. Kučera, S. Schwoon, Model checking LTL with regular valuations for pushdown systems, Inform. and Comput, 186(2), 355–376, 2003.

[19] L. Exibard, E. Filiot and P.-A. Reynier, Synthesis of data word transducers, 30th Int. Conf. on Concurrency Theory (CONCUR 2019).

[20] S. A. Greibach, A note on pushdown store automata and regular systems, Proc. the American Mathematical Society 18, 263–268, 1967.

[21] M. Kaminski and N. Francez, Finite-memory automata, Theoret. Comput. Sci. 134, 322–363, 1994.

[22] A. Khalimov and O. Kupferman, Register-bounded synthesis, 30th Int. Conf. on Concurrency Theory (CONCUR 2019).
[23] A. Khalimov, B. Maderbacher and R. Bloem, Bounded synthesis of register transducers, 16th Int Symp on Automated Technology for Verification and Analysis (ATVA 2018), 494–510.
[24] L. Libkin, T. Tan and D. Vrgoč, Regular expressions for data words, J. Computer and System Sciences 81(7), 1278–1297, 2015.
[25] L. Libkin and D. Vrgoč, Regular path queries on graphs with data, 15th Int. Conf. on Database Theory (ICDT 2012), 74–85.
[26] A. S. Murawski, S. J. Ramsay and N. Tzevelekos: Reachability in pushdown register automata, J. Computer and System Sciences 87, 58–83, 2017.
[27] F. Neven, T. Schwentick and V. Vianu, Finite state machines for strings over infinite alphabets, ACM Trans. on Computational Logic 5(3), 403–435, 2004.
[28] A. Pnueli and R. Rosner, On the synthesis of a reactive module, 16th ACM Symp. on Principles of Programming Languages (POPL 1989), 179–190.
[29] J. Rot, F. de Boer and M. Bonsangue, Pushdown system representation for unbounded object creation, Tech. Rep. KIT-13, Karlsruhe Institute of Technology, 38–52, 2010.
[30] L. Segoufin, Automata and logics for words and trees over an infinite alphabet, 15th EACSL Annual Conf. on Computer Science Logic (CSL 2006), 41–57.
[31] R. Senda, Y. Takata and H. Seki, Complexity results on register context-free grammars and register tree automata, Int. Colloquium on Theoretical Aspects of Computing (ICTAC 2018), 415–434.
[32] R. Senda, Y. Takata and H. Seki, Generalized register context-free grammars, 13th Int. Conf. Language and Automata Theory and Applications (LATA 2019), 259–271, revised version: IEICE Trans. Inf. & Syst., E103-D(3), 540–548, 2020.
[33] R. Senda, Y. Takata and H. Seki, Forward regularity preservation property of register pushdown systems, IEICE Trans. Inf. & Syst., E104-D(3), 370–380, 2021.
[34] F. Song and T. Touili, Pushdown model checking for malware detection, TACAS 2012, extended version: Int. J. Softw. Tools. Tehchnol. Transfer 16, 147–173, 2014.
[35] N. Tzevelekos, Fresh-register automata, 36th ACM Annual Symp. on Principles of Programming Languages (POPL 2009), 295–306.
[36] I. Walukiewicz, Pushdown processes: Games and model-checking, Pushdown processes: Games and model checking, 8th Int. Conf. on Computer Aided Verification (CAV 1996), 62–74, revised version: Inform. Comput. 164, 234–263, 2001.
[37] C. Xiaojuan and M. Ogawa, Well-structured extensions of pushdown systems, 24th Int. Conf. on Concurrency Theory (CONCUR 2013), 121–136.

A Appendix

A.1 A proof of Lemma 7

Lemma 7. *For a given m-DPDA $\mathcal{A}$, we can construct a 0-DPDA $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$*

Proof. For a given m-DPDA $\mathcal{A}$, we can construct ~~an~~ $2m$-DPDA $\mathcal{A}'$ such that $L(\mathcal{A}) = L(\mathcal{A}')$ and $\mathcal{A}'$ has no skip rule by replacing every skip rule $(q, a, z) \to (q', \mathtt{skip})$ of $\mathcal{A}$ to a pair of push and pop rules $(q, a, z) \to (q'', \mathtt{push}(z')), (q, \tau, z') \to (q', \mathtt{pop})$ of $\mathcal{A}'$ for $a \in \Sigma \cup \{\tau\}$. Thus, we show the lemma for m-DPDA $\mathcal{A}$ that has no skip rule by the induction on m. The case $m = 0$ is obvious. For an arbitrary m, m-DPDA $\mathcal{A} = (Q, Q_{\mathbb{i}}, Q_{\mathbb{o}}, q_0, z_0, \delta, c)$ over $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ and Γ can be converted to an $(m-1)$-DPDA $\mathcal{A}'$ over $\Sigma_{\mathbb{i}}, \Sigma_{\mathbb{o}}$ and Γ^2 such that $L(\mathcal{A}) = L(\mathcal{A}')$. Let $\mathcal{A}' = (Q \cup (Q \times \Gamma), Q_{\mathbb{i}} \cup (Q_{\mathbb{i}} \times \Gamma), Q_{\mathbb{o}} \cup (Q_{\mathbb{o}} \times \Gamma), (q_0, z_0), (z_0, z_0), \delta', c')$ such that $c'(q) = c(q), c'((q, a)) = c(q)$ for all $q \in Q, a \in \Sigma$ and

- $(q, a, z_1) \to (q', \mathtt{pop}) \in \delta$ iff $(q, a, (z_1, z_2)) \to ((q', z_2), \mathtt{pop})$, $((q, z_1), a, (z_c, z_c')) \to (q', \mathtt{skip}) \in \delta'$ for all $z_c, z_c' \in \Gamma$.
- $(q, a, z_1) \to (q', \mathtt{skip}) \in \delta$ iff $(q, a, (z_1, z_2)) \to (q', \mathtt{skip})$, $((q, z_1), a, (z_c, z_c')) \to ((q', z_1), \mathtt{skip}) \in \delta'$ for all $z_c, z_c' \in \Gamma$.
- $(q, a, z_1) \to (q', \mathtt{push}(z')) \in \delta$ iff $(q, a, (z_1, z_2)) \to ((q', z'), \mathtt{skip})$, $((q, z_1), a, (z_c, z_c')) \to (q', \mathtt{push}((z', z_1))) \in \delta'$ for all $z_c, z_c' \in \Gamma$.

for $a \in \Sigma$, and

- $(q, a, z_1) \to (q', \mathtt{pop}), (q', b, z_2) \to (q'', \mathtt{pop}) \in \delta$ iff $(q, x, (z_1, z_2)) \to (q'', \mathtt{pop})$, $((q, z_1), x, (z_2, z_c)) \to ((q'', z_c), \mathtt{pop}) \in \delta'$ for all $z_c \in \Gamma$.
- $(q, a, z_1) \to (q', \mathtt{push}(z')) \in \delta', (q', b, z') \to (q'', \mathtt{pop}) \in \delta$ iff $(q, x, (z_1, z_c)) \to (q'', \mathtt{skip})$, $((q, z_1), x, (z_c, z_c')) \to ((q'', z_1), \mathtt{skip}) \in \delta'$ for all $z_c, z_c' \in \Gamma$.
- $(q, a, z_1) \to (q', \mathtt{push}(z')), (q', b, z') \to (q'', \mathtt{push}(z'')) \in \delta$ iff $(q, x, (z_1, z_c)) \to (q'', \mathtt{push}((z'', z'')))$, $((q, z_1), x, (z_c, z_c')) \to ((q'', z''), \mathtt{push}(z', z_1)) \in \delta'$ for all $z_c, z_c' \in \Gamma$.

where $a, b \in \Sigma \cup \{\tau\}$, $x = a$ if $a \in \Sigma$ and $x = b$ otherwise. As the definition of $\mathcal{A}'$, the ID $(q, z_1 z_2 z_3 \cdots z_n)$ of $\mathcal{A}$ corresponds to an ID $(q, (z_1, z_2) \cdots (z_{n-1}, z_n))$ of $\mathcal{A}'$ if n is even and an ID $((q, z_1), (z_2, z_3) \cdots (z_{n-1}, z_n))$ if n is odd. We can check $L(\mathcal{A}) = L(\mathcal{A}')$ by the induction on the length of a sequence $w \in L(\mathcal{A})$.

A.2 A full proof of Lemma 19

Lemma 19. *If $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ satisfies the freshness property, then $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ is* Lab*-bisimilar to $\mathcal{S}_{\mathcal{A}'}$.*

Proof. Let $R \subseteq ID^{\mathrm{aug}}_{\mathcal{A}} \times ID_{\mathcal{A}'}$ be the smallest relation that satisfies for every $q \in Q$, $\theta_0, \ldots, \theta_n \in \Theta_k$, $d_0, \ldots, d_{n-1} \in D$, and $\phi_0, \ldots, \phi_n \in \Phi_k$, $((q, \theta_n, (d_{n-1}, \theta_{n-1}) \ldots (d_1, \theta_1)(d_0, \theta_0)), ((q, \phi_n), \phi_{n-1} \ldots \phi_1 \phi_0)) \in R$ if $\forall i \in [n] : \theta_{i-1}, d_{i-1}, \theta_i \models \phi_i$

and $\theta_\perp, \perp, \theta_0 \models \phi_0$. We can show that R is a $\mathtt{Lab}$-bisimulation relation from $\mathcal{S}^{\mathrm{aug}}_{\mathcal{A}}$ to $\mathcal{S}_{\mathcal{A}'}$.

Assume that $((q, \theta_n, v), ((q, \phi_n), u)) \in R$ for $v = (d_{n-1}, \theta_{n-1}) \dots (d_0, \theta_0)$ and $u = \phi_{n-1} \dots \phi_0$. Because $\forall i \in [n] : \theta_{i-1}, d_{i-1}, \theta_i \models \phi_i$ and $\theta_\perp, \perp, \theta_0 \models \phi_0$, $\forall i \in [n] : \phi_{i-1} \odot \phi_i$. If $(q, \theta_n, v) \vdash^{(a,d)}_{\mathcal{A}^{\mathrm{aug}}} (q', \theta', v')$, then there exist $\mathtt{tst}$, $\mathtt{asgn}$, $\mathtt{com}$, and d such that $\delta(q, a, \mathtt{tst}) \ni (q', \mathtt{asgn}, \mathtt{com})$, $(\theta_n, d, d_{n-1}) \models \mathtt{tst}$, $\theta' = \theta_n[\mathtt{asgn} \leftarrow d]$, and $v' = v(1{:})$, v, or $(\theta'(j'), \theta')v$ if $\mathtt{com} = \mathtt{pop}$, $\mathtt{skip}$, or $\mathtt{push}(j')$, respectively. There exists $\phi' \in \phi_n \odot_{\mathtt{T}} \Phi_k^{\mathtt{tst},\mathtt{asgn}}$ because $\theta_{n-1}, d_{n-1}, \theta_n \models \phi_n$ and $(\theta_n, d, d_{n-1}) \models \mathtt{tst}$. Thus, by definition, $\delta'((q, \phi_n), a, \phi_{n-1}) \ni ((q', \phi''), \mathtt{com}')$ where $\mathtt{com}' = \mathtt{pop}$ and $\phi'' = \phi_{n-1} \circ \phi_n \circ \phi'$ if $\mathtt{com} = \mathtt{pop}$, $\mathtt{com}' = \mathtt{skip}$ and $\phi'' = \phi_n \circ \phi'$ if $\mathtt{com} = \mathtt{skip}$, and $\mathtt{com}' = \mathtt{push}(\phi_n \circ \phi')$ and $\phi'' \in \phi' \odot \Phi_k^{=,j'}$ if $\mathtt{com} = \mathtt{push}(j')$. Therefore, $((q, \phi_n), u) \vdash^{a}_{\mathcal{A}'} ((q', \phi''), u')$ where $u' = \mathtt{upds}(u, \mathtt{com}')$. Because $\theta_{n-1}, d_{n-1}, \theta_n \models \phi_n$ and $\theta_n, d_{n-1}, \theta' \models \phi'$, $\theta_{n-1}, d_{n-1}, \theta' \models \phi_n \circ \phi'$. In the case of $\mathtt{com} = \mathtt{pop}$, $\theta_{n-2}, d_{n-2}, \theta' \models \phi_{n-1} \circ \phi_n \circ \phi'$ because $\theta_{n-2}, d_{n-2}, \theta_{n-1} \models \phi_{n-1}$ and $\theta_{n-1}, d_{n-1}, \theta' \models \phi_n \circ \phi'$. In the case of $\mathtt{com} = \mathtt{push}(j')$, $\theta', \theta'(j'), \theta' \models \phi''$ because $\phi'' \in \Phi_k^{=,j'}$. Hence, $((q', \theta', v'), ((q', \phi''), u')) \in R$ in any case.

On the other hand, if $((q, \phi_n), u) \vdash^{a}_{\mathcal{A}'} ((q', \phi''), u')$, then $\delta'((q, \phi_n), a, \phi_{n-1}) \ni ((q', \phi''), \mathtt{com}')$ where $u' = \mathtt{upds}(u, \mathtt{com}')$ holds. By definition, there exist $\mathtt{tst}$, $\mathtt{asgn}$, $\mathtt{com}$, and $\phi' \in \phi_n \circ \Phi_k^{\mathtt{tst},\mathtt{asgn}}$ such that $\delta(q, a, \mathtt{tst}) \ni (q', \mathtt{asgn}, \mathtt{com})$, and $\mathtt{com} = \mathtt{pop}$ and $\phi'' = \phi_{n-1} \circ \phi_n \circ \phi'$ if $\mathtt{com}' = \mathtt{pop}$, $\mathtt{com} = \mathtt{skip}$ and $\phi'' = \phi_n \circ \phi'$ if $\mathtt{com}' = \mathtt{skip}$, or $\mathtt{com} = \mathtt{push}(j')$, $\phi'' \in \phi' \odot \Phi_j^{=,j'}$ and $\phi''' = \phi_n \circ \phi'$ if $\mathtt{com}' = \mathtt{push}(\phi''')$. Because $\theta_{n-1}, d_{n-1}, \theta_n \models \phi_n$ and $\phi' \in \phi_n \odot_{\mathtt{T}} \Phi_k^{\mathtt{tst},\mathtt{asgn}}$, there exists $d \in D$ satisfying $(\theta_n, d, d_{n-1}) \models \mathtt{tst}$ and $\theta_n, d_{n-1}, \theta' \models \phi'$ for $\theta' = \theta_n[\mathtt{asgn} \leftarrow d]$. Therefore, $(q, \theta_n, v) \vdash^{(a,d)}_{\mathcal{A}^{\mathrm{aug}}} (q', \theta', v')$ where $v' = v(1{:})$, v, or $(\theta'(j'), \theta')v$ if $\mathtt{com} = \mathtt{pop}$, $\mathtt{skip}$, or $\mathtt{push}(j')$, respectively. We can show that $((q', \theta', v'), ((q', \phi''), u')) \in R$ in the same way as the last paragraph.

A.3 A full proof of Lemma 21

Lemma 21. *$L_k = \{w \otimes \overline{a} \mid w \in \mathtt{Comp}(\overline{a})\}$ is definable as the language of a $(2, k+2)$-DRPDA.*

Proof. Let $(2, k+2)$-DRPDA $\mathcal{A}_1 = (Q_1, Q_1^{\mathbb{i}}, Q_1^{\mathbb{o}}, q_1^0, \delta_1, c_1)$ over $A_k^{\mathbb{i}}, A_k^{\mathbb{o}}$ and D where $Q_1 = \{p, q\} \cup (\mathtt{Asgn}_k \times [k] \times \mathtt{Com}([k])) \cup [k]$, $Q_1^{\mathbb{i}} = \{p\}$, $Q_1^{\mathbb{o}} = Q_1 \setminus Q_1^{\mathbb{i}}$, $q_1^0 = p$, $c_1(s) = 2$ for every $s \in Q$ and δ_1 consists of rules of the form

$$(p, (a_{\mathbb{i}}, \mathtt{tst}), \mathtt{tst} \cup \mathtt{tst}') \to (q, \{k+1\}, \mathtt{skip}) \tag{4}$$

$$(q, (a_{\mathbb{o}}, \mathtt{asgn}, j, \mathtt{com}), \mathtt{tst}'') \to ((\mathtt{asgn}, j, \mathtt{com}), \{k+2\}, \mathtt{skip}) \tag{5}$$

$$((\mathtt{asgn}, j, \mathtt{com}), \tau, \{k+1\} \cup \mathtt{tst}'') \to (j, \mathtt{asgn}, \mathtt{com}) \tag{6}$$

$$(j, \tau, \{j, k+2\} \cup \mathtt{tst}'') \to (p, \emptyset, \mathtt{skip}) \tag{7}$$

for all $(a_{\mathbb{i}}, \mathtt{tst}) \in A_k^{\mathbb{i}}$, $(a_{\mathbb{o}}, \mathtt{asgn}, j, \mathtt{com}) \in A_k^{\mathbb{o}}$, $\mathtt{tst}' \subseteq \{k+1, k+2\}$ and $\mathtt{tst}'' \in \mathtt{Tst}_{k+2}$.

We show $L(\mathcal{A}_k) = L_k$. For this proof, we redefine compatibility for finite sequences $w \in ((\Sigma_{\mathbb{i}} \times D) \cdot (\Sigma_{\mathbb{o}} \times D))^*$ and $\overline{a} \in (A_k^{\mathbb{i}} \cdot A_k^{\mathbb{o}})^*$. We show the following claim.

Claim. Let $n \in \mathbb{N}_0$ and $w \otimes \overline{a} = ((a_0^{\mathbb{i}}, \mathtt{skip}), d_0^{\mathbb{i}})((a_0^{\mathbb{o}}, \mathtt{asgn}_0, j_0, \mathtt{com}_0), d_0^{\mathbb{o}}) \cdots \in ((A_k^{\mathbb{i}} \times D) \cdot (A_k^{\mathbb{o}} \times D))^*$ whose length is $2n$ and $\rho = (\theta_0, u_0)(\theta_1, u_1) \cdots \in (\Theta_k \times D^*)^*$ whose length is $n+1$ and $(\theta_0, u_0) = (\theta_\perp^k, \perp)$. Then, ρ is a witness of the compatibility between w and $\overline{a}$ iff $(p, \theta'_0, u_0) \vdash^{w \otimes \overline{a}(0:1)(\tau, d_0^{\mathbb{i}})(\tau, d_0^{\mathbb{o}})} (\theta'_1, u_1) \vdash^{w \otimes \overline{a}(2:3)(\tau, d_1^{\mathbb{i}})(\tau, d_1^{\mathbb{o}})} \dots \vdash^{w \otimes \overline{a}(2n-2:2n-1)(\tau, d_{n-1}^{\mathbb{i}})(\tau, d_{n-1}^{\mathbb{o}})} (p, \theta'_n, u_n)$ where $\theta'_i \in \Theta_{k+2}$ $(i \in [n])$ satisfies $\theta'_i(j) = \theta_i(j)$ for $j \in [k]$.

(Proof of the claim) We show the claim by induction on n. The case of $n = 0$ is obvious. We show the claim for arbitrary $n > 0$ with the induction hypothesis.

We first show left to right. By the induction hypothesis, $(p, \theta'_0, u_0) \vdash^{w \otimes \overline{a}(0:1)(\tau, d_0^{\mathbb{i}})(\tau, d_0^{\mathbb{o}}) \cdots w \otimes \overline{a}(2n-4:2n-3)(\tau, d_{n-2}^{\mathbb{i}})(\tau, d_{n-2}^{\mathbb{o}})} (p, \theta'_{n-1}, u_{n-1})$ holds. By the assumption, because ρ is the witness, (a) $\theta_{n-1}, d_{n-1}^{\mathbb{i}}, u_{n-1}(0) \models \mathtt{tst}_{n-1}$, (b) $\theta_n = \theta_{n-1}[\mathtt{asgn}_{n-1} \leftarrow d_{n-1}^{\mathbb{i}}]$, (c) $\theta_n(j_{n-1}) = d_{n-1}^{\mathbb{o}}$ and (d) $u_n = \mathtt{upds}(u_{n-1}, \theta_n, \mathtt{com}_{n-1})$ hold. By the condition (a), $\mathcal{A}_k$ can do a transition $(p, \theta'_{n-1}, u_{n-1}) \vdash^{w \otimes \overline{a}(2n-2)} (q, \theta^1_{n-1}, u_{n-1})$ for unique $\theta^1_{n-1} \in \Theta_{k+2}$ by the rule $(p, (a_{n-1}^{\mathbb{i}}, \mathtt{tst}_{n-1}), \mathtt{tst}_{n-1} \cup \mathtt{tst}') \to (q, \{k+1\}, \mathtt{skip})$ of the form (4). We can also say $(q, \theta^1_{n-1}, u_{n-1}) \vdash^{w \otimes \overline{a}(2n-1)} ((\mathtt{asgn}_{n-1}, j_{n-1}, \mathtt{com}_{n-1}), \theta^2_{n-1}, u_{n-1})$ by the rule of the form (5). Note that $\theta^2_{n-1}(j) = \theta_{n-1}(j)$ if $j \in [k]$, $\theta^2_{n-1}(k+1) = d_{n-1}^{\mathbb{i}}$ and $\theta^2_{n-1}(k+2) = d_{n-1}^{\mathbb{o}}$. $((\mathtt{asgn}_{n-1}, j_{n-1}, \mathtt{com}_{n-1}), \theta^2_{n-1}, u_{n-1}) \vdash^{(\tau, d_{n-1}^{\mathbb{i}})} (j_{n-1}, \theta^3_{n-1}, u_n)$ is also valid transition of $\mathcal{A}_k$ of the form (6) by the conditions (b) and (d) where $\theta_3^{n-1}(j) = \theta_n(j)$ for $j \in [k]$ and $\theta^3_{n-1}(k+2) = d_{n-1}^{\mathbb{o}}$. By the condition (c), $\theta_3^{n-1}(j_{n-1}) = \theta_3^{n-1}(k+1) = d_{n-1}^{\mathbb{o}}$ holds. Thus, a transition $(j_{n-1}, \theta^3_{n-1}, u_n) \vdash^{(\tau, d_{n-1}^{\mathbb{o}})} (p, \theta'_n, u_n)$ is valid with the rule of the form (7). In conclusion, $(p, \theta'_{n-1}, u_{n-1}) \vdash^{w \otimes \overline{a}(2n-2:2n-1)(\tau, d_{n-1}^{\mathbb{i}})(\tau, d_{n-1}^{\mathbb{o}})} (p, \theta'_n, u_n)$ holds, and with the induction hypothesis, we obtain the left to right of the claim.

Next, we prove right to left. By the assumption, $(p, \theta'_{n-1}, u_{n-1}) \vdash^{w \otimes \overline{a}(2n-2:2n-1)(\tau, d_{n-1}^{\mathbb{i}})(\tau, d_{n-1}^{\mathbb{o}})} (p, \theta'_n, u_n)$ holds. By checking four transition rules that realize the above transition relation, we can obtain that $\rho(n-1), \rho(n), w \otimes \overline{a}(2n-2)$ and $w \otimes \overline{a}(2n-1)$ satisfies the conditions (a) to (d) described in the previous paragraph. Thus, by the induction hypothesis, we obtain ρ is a witness of the compatibility between w and $\overline{a}$.

(end of the proof of the claim)

By the claim, $w \otimes \overline{a} \in L_k \Leftrightarrow$ there exists a witness $(\theta_0, u_0)(\theta_1, u_1) \cdots \in (\Theta_k \times D^*)^\omega$ of w and $\overline{a} \Leftrightarrow$ there exists a run $(p, \theta'_0, u_0) \vdash^{w \otimes \overline{a}(0:1)(\tau, d_0^{\mathbb{i}})(\tau, d_0^{\mathbb{o}})} (\theta'_1, u_1) \vdash^{w \otimes \overline{a}(2:3)(\tau, d_1^{\mathbb{i}})(\tau, d_1^{\mathbb{o}})} \cdots$ of $\mathcal{A} \Leftrightarrow w \otimes \overline{a} \in L(\mathcal{A}_k)$ holds for all $w \otimes \overline{a} \in \mathtt{DW}(A_k^{\mathbb{i}}, A_k^{\mathbb{o}}, D)$.

A.4 A full proof of Lemma 22

Lemma 22. *For a specification $\mathcal{S}$ defined by some visibly k'-DRPDA, $L_{\overline{S},k} = \{w \otimes \overline{a} \mid w \in \mathtt{Comp}(\overline{a}) \cap \overline{S}\}$ is definable as the language of a $(4, k+k'+4)$-DRPDA.*

Proof. Let $L_{\overline{S}} = \{w \otimes \overline{a} \mid w \in \overline{S}, \overline{a} \in (A_k^{\mathbb{i}} \cdot A_k^{\mathbb{o}})^\omega\}$. We can construct a visibly k'-DRPDA $\mathcal{A}_2 = (Q_2, Q_2^{\mathbb{i}}, Q_2^{\mathbb{o}}, q_2^0, \delta_2, c_2)$ over $A_k^{\mathbb{i}}, A_k^{\mathbb{o}}$ and D such that $L(\mathcal{A}_2) = L_{\overline{S}}$.

Let $\mathcal{A}_1$ be the $(2, k+2)$-DRPDA such that $L(\mathcal{A}_1) = L_k$, which is given in Lemma 21. Because $L_{\overline{S},k} = L_{\overline{S}} \cap L_k$, it is enough to show that we can construct a $(4, k+k'+4)$-DRPDA $\mathcal{A}$ over $A_k^{\mathbb{i}}, A_k^{\mathbb{o}}$ and D such that $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

To construct $\mathcal{A}$, we use the facts that $c_1(q)$ is even for every $q \in Q_1$ and δ_1 consists of several groups of three consecutive rules having the following forms:

$$(q_1, a, \mathtt{tst}_1) \to (q_2, \mathtt{asgn}_1, \mathtt{skip}) \tag{5'}$$
$$(q_2, \tau, \mathtt{tst}_2) \to (q_3, \mathtt{asgn}_2, \mathtt{com}_1) \tag{6'}$$
$$(q_3, \tau, \mathtt{tst}_3) \to (q_4, \mathtt{asgn}_3, \mathtt{skip}). \tag{7'}$$

Note that $\mathtt{vis}(a) = v(\mathtt{com}_1)$ always holds for every consecutive rules. $(5'), (6')$ and $(7')$ correspond to $(5), (6)$ and (7), respectively, and (4) is also converted to three consecutive rules like $(5')$-$(7')$ by adding dummy τ rules.

We let $k_1 = k+2$ and $k_2 = k'$. We construct $(4, k_1+k_2+2)$-DRPDA $\mathcal{A} = (Q_{\mathbb{i}} \cup Q_{\mathbb{o}} \cup \{q_0\}, Q_{\mathbb{i}} \cup \{q_0\}, Q_{\mathbb{o}}, q_0, \delta, c)$ where $Q_{\mathbb{i}} = Q_1^{\mathbb{i}} \times Q_2^{\mathbb{i}} \times [5]$, $Q_{\mathbb{o}} = Q_1^{\mathbb{o}} \times Q_2^{\mathbb{o}} \times [5]$. c is defined as $c(q_0) = 1$ and $c((q_1, q_2, i)) = c_2(q_2)$ for all $(q_1, q_2, i) \in Q$. δ has first τ rule $(q_0, \tau, [k] \cup \{\mathtt{top}\}) \to ((q_0^1, q_0^2, 1), \mathtt{push}(1))$. For all rules (5'), (6'), (7') in δ_1 and

$$(q, a, \mathtt{tst}) \to (q', \mathtt{asgn}, \mathtt{com}) \in \delta_2 \tag{8}$$

such that $v(\mathtt{com}_1) = v(\mathtt{com})$ $(= \mathtt{vis}(a))$ for $a \in A_k^{\mathbb{i}} \cup A_k^{\mathbb{o}}$, let $\mathtt{tst}^{+k_1} = \{i + k_1 \mid i \in \mathtt{tst}\} \cup \{\mathtt{top} \mid \mathtt{top} \in \mathtt{tst} \setminus [k_1]\}$, $\mathtt{asgn}^{+k_1} = \{i + k_1 \mid i \in \mathtt{asgn}\}$ and $\mathtt{com}^{+k_1} = \mathtt{push}(j + k_1)$ if $\mathtt{com} = \mathtt{push}(j)$ and $\mathtt{com}^{+k_1} = \mathtt{com}$ otherwise, then δ consists of the rules

$$((q_1, q, 1), \tau, T_1^{k_1+k_2+2} \cup \{\mathtt{top}\}) \to ((q_1, q, 2), \{k_1+k_2+1\}, \mathtt{pop}) \tag{9}$$
$$((q_1, q, 2), \tau, T_1^{k_1+k_2+2} \cup \{\mathtt{top}\}) \to ((q_1, q, 3), \{k_1+k_2+2\}, \mathtt{push}(k_1+k_2+1)) \tag{10}$$
$$((q_1, q, 3), a, \mathtt{tst}_1 \cup ((\mathtt{tst}^{+k_1} \setminus \{\mathtt{top}\}) \cup \mathtt{Top})) \to ((q_2, q', 4), \mathtt{asgn}_1 \cup \mathtt{asgn}^{+k_1}, \mathtt{com}^{+k_1}) \tag{11}$$
$$((q_2, q', 4), \tau, ((\mathtt{tst}_2 \setminus \{\mathtt{top}\}) \cup \mathtt{Top}') \cup T_{k_1+1}^{k_1+k_2+1}) \to ((q_3, q', 5), \mathtt{asgn}_2, \mathtt{com}_1) \tag{12}$$
$$((q_3, q', 5), \tau, \mathtt{tst}_3 \cup T_{k_1+1}^{k_1+k_2+2}) \to ((q_4, q', 1), \mathtt{asgn}_3, \mathtt{skip}) \tag{13}$$

for all $T_i^j \in \mathtt{Tst}_j \setminus \mathtt{Tst}_{i-1}$ (we assume $\mathtt{Tst}_0 = \emptyset$), and $\mathtt{Top} = \{k_1+k_2+2\}$ $(\mathtt{Top}' = \{k_1+k_2+1\})$ if $\mathtt{top} \in \mathtt{tst}$ $(\mathtt{top} \in \mathtt{tst}_2)$ and $\mathtt{Top} = \emptyset$ $(\mathtt{Top}' = \emptyset)$ otherwise.

Fig. 2 illustrates an example of transitions of $\mathcal{A}$ from $(q_1, q, 1)$ to $(q_4, q', 1)$ with updating contents of its registers and stack. The first to k_1-th registers simulate the registers of $\mathcal{A}_1$, (k_1+1)-th to (k_1+k_2)-th registers simulate the registers of $\mathcal{A}_2$ and (k_1+k_2+1)-th and (k_1+k_2+2)-th registers are for keeping the first and second stack top contents, respectively. The stack contents of $\mathcal{A}$

simulates those of $\mathcal{A}_1$ and $\mathcal{A}_2$ by restoring the contents of stacks of $\mathcal{A}_1$ and $\mathcal{A}_2$ alternately.

The transition rules (9) and (10) are for moving the two data values at the stack top to (k_1+k_2+1)-th and (k_1+k_2+2)-th registers. The transition rule (11) is for updating states, registers and stacks by simulating the rules (5') and (8). Because the first to k_2-th registers of $\mathcal{A}_2$ are simulated by (k_1+1)-th to (k_1+k_2)-th registers of $\mathcal{A}$, we use $\mathtt{tst}^{+k_1}$, $\mathtt{asgn}^{+k_1}$ and $\mathtt{com}^{+k_1}$ instead of $\mathtt{tst}$, $\mathtt{asgn}$ and $\mathtt{com}$, and replace $\mathtt{top}$ in $\mathtt{tst}^{+k_1}$ to k_1+k_2+2. The transition rules (12) and (13) simulate the rules (6') and (7'), respectively.

For two assignments $\theta_1 \in \Theta_{k_1}$ and $\theta_2 \in \Theta_{k_2}$, let $[\theta_1,\theta_2,d,d'] \in \Theta_{k_1+k_2+2}$ be the assignment such that $[\theta_1,\theta_2,d,d'](i) = \theta_1(i)$ if $i \in [k_1]$, $[\theta_1,\theta_2,d,d'](i) = \theta_2(i)$ if $k_1+1 \leq i \leq k_2$, $[\theta_1,\theta_2,d,d'](k_1+k_2+1) = d$ and $[\theta_1,\theta_2,d,d'](k_1+k_2+2) = d'$. To prove $L(\mathcal{A}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, we show the following claim.

Claim. For all $n \in \mathbb{N}_0$ and $w \in ((A_k^{\mathfrak{i}} \cup A_k^{\mathfrak{o}}) \times D)^n$, there exists sequences of transitions $(q_0^1,\theta_0^1,u_0^1) \vdash_{\mathcal{A}_1}^{w(0)(\tau,d_0)(\tau,d_0')} (q_1^1,\theta_1^1,u_1^1) \vdash_{\mathcal{A}_1}^{w(1)(\tau,d_1)(\tau,d_1')} \cdots \vdash_{\mathcal{A}_1}^{w(n-1)(\tau,d_{n-1})(\tau,d_{n-1}')} (q_n^1,\theta_n^1,u_n^1)$ and $(q_0^2,\theta_0^2,u_0^2) \vdash_{\mathcal{A}_2}^{w(0)} (q_1^2,\theta_1^2,u_1^2) \vdash_{\mathcal{A}_2}^{w(1)} \cdots \vdash_{\mathcal{A}_2}^{w(n-1)} (q_n^2,\theta_n^2,u_n^2)$ iff $(q_0,\theta_\perp^{k_1+k_2+2},\perp) \vdash_{\mathcal{A}}^{(\tau,\perp)} ((q_0^1,q_0^2,1),[\theta_0^1,\theta_0^2,d_0^1,d_0^2],\langle u_0^1,u_0^2\rangle) \vdash_{\mathcal{A}}^{(\tau,u_0^1(0))(\tau,u_0^2(0))w(0)(\tau,d_0)(\tau,d_0')} ((q_1^1,q_1^2,1),[\theta_1^1,\theta_1^2,d_1^1,d_1^2],\langle u_1^1,u_1^2\rangle) \vdash_{\mathcal{A}}^{(\tau,u_1^1(0))(\tau,u_1^2(0))w(1)(\tau,d_1)(\tau,d_1')} \cdots \vdash_{\mathcal{A}}^{(\tau,u_{n-1}^1(0))(\tau,u_{n-1}^2(0))w(n-1)(\tau,d_{n-1})(\tau,d_{n-1}')} ((q_n^1,q_n^2,1),[\theta_n^1,\theta_n^2,d_n^1,d_n^2],\langle u_n^1,u_n^2\rangle)$ holds where $b \in \{1,2\}$, $i \in [n]$, $\theta_0^b = \theta_\perp^{k_b}$, $u_0^b = \perp$, $q_i^b \in Q_b$, $\theta_i^b \in \Theta_{k_b}$, $u_i^b \in D^*$ and $d_{i-1},d_{i-1}' \in D$.

(Proof of the claim) We show the claim by the induction on n. The case $n=0$ is obvious.

We first show left to right. By induction hypothesis, $(q_0,\theta_\perp^{\mathcal{A}},\perp) \vdash_{\mathcal{A}}^{(\tau,\perp)} ((q_0^1,q_0^2,1),[\theta_0^1,\theta_0^2,d_0^1,d_0^2],\langle u_0^1,u_0^2\rangle) \vdash_{\mathcal{A}}^{(\tau,u_0^1(0))(\tau,u_0^2(0))w(0)(\tau,d_0)(\tau,d_0')} \cdots \vdash_{\mathcal{A}}^{(\tau,u_{n-2}^1(0))(\tau,u_{n-2}^2(0))w(n-2)(\tau,d_{n-2})(\tau,d_{n-2}')} ((q_{n-1}^1,q_{n-1}^2,1),[\theta_{n-1}^1,\theta_{n-1}^2,d_{n-1}^1,d_{n-1}^2],\langle u_{n-1}^1,u_{n-1}^2\rangle)$ holds. Also, by the assumption, $(q_{n-1}^1,\theta_{n-1}^1,u_{n-1}^1) \vdash_{\mathcal{A}_1}^{w(n-1)} (q',\theta',u_{n-1}^1) \vdash_{\mathcal{A}_1}^{(\tau,d)} (q'',\theta'',u_n^1) \vdash_{\mathcal{A}_1}^{(\tau,d')} (q_n^1,\theta_n^1,u_n^1)$ and $(q_{n-1}^2,\theta_{n-1}^2,u_{n-1}^2) \vdash_{\mathcal{A}_2}^{w(n-1)} (q_n^2,\theta_n^2,u_n^2)$ for some $q',q'' \in Q_1, \theta',\theta'' \in \Theta_{k_1}$ and $d,d' \in D$, and let the following be the rules used in these transitions.

$$(q_{n-1},(a,v(\mathtt{com}_1)),\mathtt{tst}_1) \to (q',\mathtt{asgn}_1,\mathtt{skip}) \in \delta_1 \tag{5"}$$

$$(q',\tau,\mathtt{tst}_2) \to (q'',\mathtt{asgn}_2,\mathtt{com}_1) \in \delta_1 \tag{6"}$$

$$(q'',\tau,\mathtt{tst}_3) \to (q_n,\mathtt{asgn}_3,\mathtt{skip}) \in \delta_1 \tag{7"}$$

$$(q_{n-1},(a,v(\mathtt{com})),\mathtt{tst}) \to (q_n,\mathtt{asgn},\mathtt{com}) \in \delta_2 \tag{8'}$$

We can check the follows.

$$((q^1_{n-1}, q^2_{n-1}, 1), [\theta^1_{n-1}, \theta^2_{n-1}, d^1_{n-1}, d^2_{n-1}], \langle u^1_{n-1}, u^2_{n-1}\rangle) \vdash_{\mathcal{A}}^{(\tau, u^1_{n-1}(0))(\tau, u^2_{n-1}(0))}$$
$$((q^1_{n-1}, q^2_{n-1}, 3), [\theta^1_{n-1}, \theta^2_{n-1}, u^1_{n-1}(0), u^1_{n-1}(0)], \langle u^1_{n-1}, u^2_{n-1}\rangle) \vdash_{\mathcal{A}}^{w(n-1)}$$
$$((q^1_{n-1}, q^2_{n-1}, 4), [\theta^1_{n}, \theta', u^1_{n-1}(0), u^1_{n-1}(0)], \langle u^1_{n-1}, u^2_{n}\rangle) \vdash_{\mathcal{A}}^{(\tau, d_{n-1})}$$
$$((q^1_{n-1}, q^2_{n-1}, 5), [\theta^1_{n}, \theta'', u^1_{n-1}(0), u^1_{n-1}(0)], \langle u^1_{n}, u^2_{n}\rangle) \vdash_{\mathcal{A}}^{(\tau, d'_{n-1})}$$
$$((q^1_{n}, q^2_{n}, 1), [\theta^1_{n}, \theta^2_{n}, u^1_{n-1}(0), u^1_{n-1}(0)], \langle u^1_{n}, u^2_{n}\rangle)$$

Thus, the right side of the claim holds. In a similar way, we can also show right to left.

(end of the proof of the claim)

By the claim, $w \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \Leftrightarrow$ there exists runs $(q^1_0, \theta^1_0, u^1_0) \vdash_{\mathcal{A}_1}^{w(0)(\tau,d_0)(\tau,d'_0)} (q^1_1, \theta^1_1, u^1_1) \vdash_{\mathcal{A}_1}^{w(1)(\tau,d_1)(\tau,d'_1)} \cdots$ and $(q^2_0, \theta^2_0, u^2_0) \vdash_{\mathcal{A}_2}^{w(0)} (q^2_1, \theta^2_1, u^2_1) \vdash_{\mathcal{A}_2}^{w(1)} \cdots$ that satisfies the minimum number appearing in the sequence $q^1_0, q^1_1, \cdots$ infinitely is even. $\Leftrightarrow$ there exists a run $(q_0, \theta^{\mathcal{A}}_{\perp}, \perp) \vdash_{\mathcal{A}}^{(\tau,\perp)} ((q^1_0, q^2_0, 1), [\theta^1_0, \theta^2_0, d^1_0, d^2_0], \langle u^1_0, u^2_0\rangle) \vdash_{\mathcal{A}}^{(\tau,u^1_0(0))(\tau,u^2_0(0))w(0)(\tau,d_0)(\tau,d'_0)} ((q^1_1, q^2_1, 1), [\theta^1_1, \theta^2_1, d^1_1, d^2_1], \langle u^1_1, u^2_1\rangle) \vdash_{\mathcal{A}}^{(\tau,u^1_1(0))(\tau,u^2_1(0))w(1)(\tau,d_1)(\tau,d'_1)} \cdots$ that satisfies the minimum number appearing in the sequence $(q^1_0, q^2_0, 1), (q^1_0, q^2_0, 2) \cdots, (q^1_1, q^2_1, 1), \cdots$ infinitely is even. $\Leftrightarrow w \in L(\mathcal{A})$ holds.