# Visibly Pushdown Automata: From Language Equivalence to Simulation and Bisimulation

Jiří Srba[*]

**BRICS**[**], Department of Computer Science
Aalborg University, Fredrik Bajersvej 7B, 9220 Aalborg, Denmark
srba@cs.aau.dk

**Abstract.** We investigate the possibility of (bi)simulation-like preorder/equivalence checking on the class of visibly pushdown automata and its natural subclasses visibly BPA (Basic Process Algebra) and visibly one-counter automata. We describe generic methods for proving complexity upper and lower bounds for a number of studied preorders and equivalences like simulation, completed simulation, ready simulation, 2-nested simulation preorders/equivalences and bisimulation equivalence. Our main results are that all the mentioned equivalences and preorders are EXPTIME-complete on visibly pushdown automata, PSPACE-complete on visibly one-counter automata and P-complete on visibly BPA. Our PSPACE lower bound for visibly one-counter automata improves also the previously known DP-hardness results for ordinary one-counter automata and one-counter nets. Finally, we study regularity checking problems for visibly pushdown automata and show that they can be decided in polynomial time.

## 1 Introduction

Visibly pushdown languages were introduced by Alur and Madhusudan in [4] as a subclass of context-free languages suitable for formal program analysis, yet tractable and with nice closure properties like the class of regular languages. Visibly pushdown languages are accepted by visibly pushdown automata whose stack behaviour is determined by the input symbol. If the symbol belongs to the category of *call actions* then the automaton must push, if it belongs to *return actions* then the automaton must pop, otherwise (for the *internal actions*) it cannot change the stack height. In [4] it is shown that the class of visibly pushdown languages is closed under intersection, union, complementation, renaming, concatenation and Kleene star. A number of decision problems like universality, language equivalence and language inclusion, which are undecidable for context-free languages, become EXPTIME-complete for visibly pushdown languages.

Recently, visibly pushdown languages have been intensively studied and applied to e.g. program analysis [2], XML processing [20] and the language theory

---

of this class has been further investigated in [3, 6]. Some recent results show for example the application of a variant of vPDA for proving decidability of contextual equivalence (and other problems) for the third-order fragment of Idealized Algol [18].

In this paper we study visibly pushdown automata from a different perspective. Rather than as language acceptors, we consider visibly pushdown automata as devices that generate infinite-state labelled graphs and we study the questions of decidability of behavioral equivalences and preorders on this class. Our results confirm the tractability of a number of verification problems for visibly pushdown automata.

We prove EXPTIME-completeness of equivalence checking on visibly pushdown automata (vPDA) for practically all preorders and equivalences between simulation preorder and bisimulation equivalence that have been studied in the literature (our focus includes simulation, completed simulation, ready simulation, 2-nested simulation and bisimulation). We then study two natural (and incomparable) subclasses of visibly pushdown automata: visibly basic process algebra (vBPA) and visibly one-counter automata (v1CA). In case of v1CA we demonstrate PSPACE-completeness of the preorder/equivalence checking problems and in case of vBPA even P-completeness. For vBPA we provide also a direct reduction of the studied problems to equivalence checking on finite-state systems, hence the fast algorithms already developed for systems with finitely many reachable states can be directly used. All the mentioned upper bounds are matched by the corresponding lower bounds. The PSPACE-hardness proof for v1CA moreover improves the currently known DP lower bounds [13] for equivalence checking problems on ordinary one-counter automata and one-counter nets and some other problems (see Remark 2). Finally, we consider regularity checking for visibly pushdown automata and show P-completeness for vPDA and vBPA, and NL-completeness for v1CA w.r.t. all equivalences between trace equivalence and isomorphism of labelled transition systems.

*Related work.* The main reason why many problems for visibly pushdown languages become tractable is, as observed in [4], that a pair of visibly pushdown automata can be synchronized in a similar fashion as finite automata. We use this idea to construct, for a given pair of vPDA processes, a single pushdown automaton where we in a particular way encode the behaviour of both input processes so that they can alternate in performing their moves. This is done in such a way that the question of equality of the input processes w.r.t. a given preorder/equivalence can be tested by asking about the validity of particular (and fixed) modal $\mu$-calculus formulae on the single pushdown process. A similar result of reducing weak simulation between a pushdown process and a finite-state process (and vice versa) to the model checking problem appeared in [17]. We generalize these ideas to cover preorders/equivalences between two visibly pushdown processes and provide a generic proof for all the equivalence checking problems. The technical details of our construction are different from [17] and in particular our construction works immediately also for vBPA (as the necessary bookkeeping is stored in the stack alphabet). As a result we thus show how to

handle essentially any so far studied equivalence/preorder between simulation and bisimulation in a uniform way for vPDA, vBPA as well as for v1CA.

In [6] the authors study language regularity problems for visibly pushdown automata. Their line of research is orthogonal to ours because they define a visibly pushdown automaton as regular if it is language equivalent to some visibly one-counter automaton. We study the regularity problems in the context of the standard definitions from the concurrency theory, i.e., whether for a given vPDA process there is a behaviorally equivalent finite-state system. Though, as remarked in more detail in the conclusion, questions of finding an equivalent v1CA and in particular vBPA for a given vPDA could be also interesting to investigate.

*Note: full version of this paper will appear as BRICS technical report.*

## 2    Definitions

A *labelled transition system* (LTS) is a triple $(S, \mathcal{A}ct, \longrightarrow)$ where $S$ is the set of *states* (or *processes*), $\mathcal{A}ct$ is the set of *labels* (or *actions*), and $\longrightarrow \subseteq S \times \mathcal{A}ct \times S$ is the *transition relation*; for each $a \in \mathcal{A}ct$, we view $\stackrel{a}{\longrightarrow}$ as a binary relation on $S$ where $s \stackrel{a}{\longrightarrow} s'$ iff $(s, a, s') \in \longrightarrow$. The notation can be naturally extended to $s \stackrel{w}{\longrightarrow} s'$ for finite sequences of actions $w \in \mathcal{A}ct^*$. For a process $s \in S$ we define the set of its *initial actions* by $I(s) \stackrel{\text{def}}{=} \{a \in \mathcal{A}ct \mid \exists s' \in S.\ s \stackrel{a}{\longrightarrow} s'\}$.

We shall now define the studied equivalences/preorders which are between simulation and bisimilarity. Given an LTS $(S, \mathcal{A}ct, \longrightarrow)$, a binary relation $R \subseteq S \times S$ is a

- *simulation* iff for each $(s, t) \in R$, $a \in \mathcal{A}ct$, and $s'$ such that $s \stackrel{a}{\longrightarrow} s'$ there is $t'$ such that $t \stackrel{a}{\longrightarrow} t'$ and $(s', t') \in R$,
- *completed simulation* iff $R$ is a simulation and moreover for each $(s, t) \in R$ it holds that $I(s) = \emptyset$ if and only if $I(t) = \emptyset$,
- *ready simulation* iff $R$ is a simulation and moreover for each $(s, t) \in R$ it holds that $I(s) = I(t)$,
- *2-nested simulation* iff $R$ is a simulation and moreover $R^{-1} \subseteq R$, and
- *bisimulation* iff $R$ is a simulation and moreover $R^{-1} = R$.

We write $s \sqsubseteq_s t$ if there is a simulation $R$ such that $(s, t) \in R$, $s \sqsubseteq_{cs} t$ if there is a completed simulation $R$ such that $(s, t) \in R$, $s \sqsubseteq_{rs} t$ if there is a ready simulation $R$ such that $(s, t) \in R$, $s \sqsubseteq_{2s} t$ if there is a 2-nested simulation $R$ such that $(s, t) \in R$, $s \sim t$ if there is a bisimulation $R$ such that $(s, t) \in R$. The relations are called the corresponding *preorders* (except for bisimilarity, which is already an equivalence). For a preorder $\sqsubseteq \in \{\sqsubseteq_s, \sqsubseteq_{cs}, \sqsubseteq_{rs}, \sqsubseteq_{2s}\}$ we define the corresponding equivalence by $s = t$ iff $s \sqsubseteq t$ and $t \sqsubseteq s$. We remind the reader of the fact that $\sim \subseteq \sqsubseteq_{2s} \subseteq \sqsubseteq_{rs} \subseteq \sqsubseteq_{cs} \subseteq \sqsubseteq_s$ and $\sim \subseteq =_{2s} \subseteq =_{rs} \subseteq =_{cs} \subseteq =_s$ and all inclusions are strict.

We shall use a standard game-theoretic characterization of (bi)similarity. A *bisimulation game* on a pair of processes $(s_1, t_1)$ is a two-player game between

*Attacker* and *Defender*. The game is played in *rounds* on pairs of states from $S \times S$. In each round the players change the *current pair of states* $(s, t)$ (initially $s = s_1$ and $t = t_1$) according to the following rule:

1. Attacker chooses either $s$ or $t$, $a \in \mathcal{A}ct$ and performs a move $s \xrightarrow{a} s'$ or $t \xrightarrow{a} t'$.
2. Defender responds by choosing the opposite process (either $t$ or $s$) and performs a move $t \xrightarrow{a} t'$ or $s \xrightarrow{a} s'$ under the same action $a$.
3. The pair $(s', t')$ becomes the (new) current pair of states.

A *play* (of the bisimulation game) is a sequence of pairs of processes formed by the players according to the rules mentioned above. A play is finite iff one of the players gets stuck (cannot make a move); the player who got stuck lost the play and the other player is the winner. If the play is infinite then Defender is the winner.

We use the following standard fact.

**Proposition 1.** *It holds that $s \sim t$ iff Defender has a winning strategy in the bisimulation game starting with the pair $(s, t)$, and $s \nsim t$ iff Attacker has a winning strategy in the corresponding game.*

The rules of the bisimulation game can be easily modified in order to capture the other equivalences/preorders.

In the *simulation preorder game*, Attacker is restricted to attack only from the (left-hand side) process $s$. In the *simulation equivalence game*, Attacker can first choose a side (either $s$ or $t$) but after that he is not allowed to change the side any more. *Completed/ready simulation game* has the same rules as the simulation game but Defender is moreover losing in any configuration which brakes the extra condition imposed by the definition (i.e. $s$ and $t$ should have the same set of initial actions in case of ready simulation, and their sets of initial actions should be both empty at the same time in case of completed simulation). Finally, in the *2-nested simulation preorder game*, Attacker starts playing from the left-hand side process $s$ and at most once during the play he is allowed to switch sides (the soundness follows from the characterization provided in [1]). In the *2-nested simulation equivalence game*, Attacker can initially choose any side but he is still restricted that he can change sides at most once during the play.

We shall now define the model of pushdown automata. Let $\mathcal{A}ct$ be a finite set of actions, let $\Gamma$ be a finite set of stack symbols and let $Q$ be a finite set of control states. We assume that the sets $\mathcal{A}ct$, $\Gamma$ and $Q$ are pairwise disjoint. A *pushdown automaton* (PDA) over the set of actions $\mathcal{A}ct$, stack alphabet $\Gamma$ and control states $Q$ is a finite set $\Delta$ of rules of the form $pX \xrightarrow{a} q\alpha$ where $p, q \in Q$, $a \in \mathcal{A}ct$, $X \in \Gamma$ and $\alpha \in \Gamma^*$.

A PDA $\Delta$ determines a labelled transition system $T(\Delta) = (S, \mathcal{A}ct, \longrightarrow)$ where the states are configurations of the form state×stack (i.e. $S = Q \times \Gamma^*$ and configurations like $(p, \alpha)$ are usually written as $p\alpha$ where the top of the stack $\alpha$ is by agreement on the left) and the transition relation is determined by the following prefix rewriting rule.

$$\frac{(pX \xrightarrow{a} q\alpha) \in \Delta, \quad \gamma \in \Gamma^*}{pX\gamma \xrightarrow{a} q\alpha\gamma}$$

A pushdown automaton is called BPA for *Basic Process Algebra* if the set of control states is a singleton set ($|Q| = 1$). In this case we usually omit the control state from the rules and configurations.

A pushdown automaton is called 1CA for *one-counter automaton* if the stack alphabet consists of two symbols only, $\Gamma = \{I, Z\}$, and every rule is of the form $pI \xrightarrow{a} q\alpha$ or $pZ \xrightarrow{a} q\alpha Z$, where $\alpha \in \{I\}^*$. This means that every configuration reachable from $pZ$ is of the form $pI^n Z$ where $I^n$ stands for a sequence of $n$ symbols $I$ and $Z$ corresponds to the bottom of the stack (the value zero). We shall simply denote such a configuration by $p(n)$ and say that it represents the counter value $n$.

Assume that $\mathcal{A}ct = \mathcal{A}ct_c \cup \mathcal{A}ct_r \cup \mathcal{A}ct_i$ is partitioned into a disjoint union of finite sets of call, return and internal actions, respectively. A *visibly pushdown automaton* (vPDA) is a PDA which, for every rule $pX \xrightarrow{a} q\alpha$, satisfies additional three requirements (where $|\alpha|$ stands for the length of $\alpha$):

- if $a \in \mathcal{A}ct_c$ then $|\alpha| = 2$ (call),
- if $a \in \mathcal{A}ct_r$ then $|\alpha| = 0$ (return), and
- if $a \in \mathcal{A}ct_i$ then $|\alpha| = 1$ (internal).

Hence in vPDA the type of the input action determines the change in the height of the stack (call by $+1$, return by $-1$, internal by $0$).

The classes of visibly basic process algebra (vBPA) and visibly one-counter automata (v1CA) are defined analogously.

*Remark 1.* For internal actions we allow to modify also the top of the stack. This model (for vPDA) can be easily seen to be equivalent to the standard one (as introduced in [4]) where the top of the stack does not change under internal actions. However, when we consider the subclass vBPA, the possibility of changing the top of the stack under internal actions significantly increases the descriptive power of the formalism. Unlike in [4], we do not allow to perform return actions on the empty stack.

The question we are interested in is: given a vPDA (or vBPA, or v1CA) and two of its initial configurations $pX$ and $qY$, can we algorithmically decide whether $pX$ and $qY$ are equal with respect to a given preorder/equivalence and if yes, what is the complexity?

## 3  Decidability of Preorder/Equivalence Checking

### 3.1  Visibly Pushdown Automata

We shall now study preorder/equivalence checking problems on the class of visibly pushdown automata. We prove the decidability by reducing the problems

to model checking of an ordinary pushdown system against a fixed $\mu$-calculus formula.

Let $\Delta$ be a vPDA over the set of actions $\mathcal{A}ct = \mathcal{A}ct_c \cup \mathcal{A}ct_r \cup \mathcal{A}ct_i$, stack alphabet $\Gamma$ and control states $Q$. We shall construct a PDA $\Delta'$ over the actions $\mathcal{A}ct' \stackrel{\text{def}}{=} \mathcal{A}ct \cup \overline{\mathcal{A}ct} \cup \{\ell, r\}$ where $\overline{\mathcal{A}ct} \stackrel{\text{def}}{=} \{\bar{a} \mid a \in \mathcal{A}ct\}$, stack alphabet $\Gamma' \stackrel{\text{def}}{=} G \times G$ where $G \stackrel{\text{def}}{=} \Gamma \cup (\Gamma \times \Gamma) \cup (\Gamma \times \mathcal{A}ct) \cup \{\epsilon\}$, and control states $Q' \stackrel{\text{def}}{=} Q \times Q$. For notational convenience, elements $(X, a) \in \Gamma \times \mathcal{A}ct$ will be written simply as $X_a$.

The idea is that for a given pair of vPDA processes we shall construct a single PDA process which simulates the behaviour of both vPDA processes by repeatedly performing a move in one of the processes, immediately followed by a move under the same action in the other process. The actions $\ell$ and $r$ make it visible, whether the move is performed on the left-hand side or right-hand side. The assumption that the given processes are vPDA ensures that their stacks are kept synchronized.

We shall define a partial mapping $[\,.\,,\,.\,] : \Gamma^* \times \Gamma^* \to (\Gamma \times \Gamma)^*$ inductively as follows ($X, Y \in \Gamma$ and $\alpha, \beta \in \Gamma^*$ such that $|\alpha| = |\beta|$): $[X\alpha, Y\beta] \stackrel{\text{def}}{=} (X, Y)[\alpha, \beta]$ and $[\epsilon, \epsilon] \stackrel{\text{def}}{=} \epsilon$. The mapping provides the possibility to merge stacks.

Assume a given pair of vPDA processes $pX$ and $qY$. Our aim is to effectively construct a new PDA system $\Delta'$ such that for every $\bowtie \in \{\sqsubseteq_s, =_s, \sqsubseteq_{cs}, =_{cs}, \sqsubseteq_{rs}, =_{rs}, \sqsubseteq_{2s}, =_{2s}, \sim\}$ it is the case that $pX \bowtie qY$ in $\Delta$ if and only if $(p, q)(X, Y) \models \phi_{\bowtie}$ in $\Delta'$ for a fixed $\mu$-calculus formula $\phi_{\bowtie}$. We refer the reader to [16] for the introduction to the modal $\mu$-calculus.

The set of PDA rules $\Delta'$ is defined as follows. Whenever $(pX \stackrel{a}{\longrightarrow} q\alpha) \in \Delta$ then the following rules belong to $\Delta'$:

1. $(p, p')(X, X') \stackrel{\ell}{\longrightarrow} (q, p')(\alpha, X'_a)$ for every $p' \in Q$ and $X' \in \Gamma$,
2. $(p', p)(X', X) \stackrel{r}{\longrightarrow} (p', q)(X'_a, \alpha)$ for every $p' \in Q$ and $X' \in \Gamma$,
3. $(p', p)(\beta, X_a) \stackrel{r}{\longrightarrow} (p', q)[\beta, \alpha]$ for every $p' \in Q$ and $\beta \in \Gamma \cup (\Gamma \times \Gamma) \cup \{\epsilon\}$,
4. $(p, p')(X_a, \beta) \stackrel{\ell}{\longrightarrow} (q, p')[\alpha, \beta]$ for every $p' \in Q$ and $\beta \in \Gamma \cup (\Gamma \times \Gamma) \cup \{\epsilon\}$,
5. $(p, p')(X, X') \stackrel{a}{\longrightarrow} (p, p')(X, X')$ for every $p' \in Q$ and $X' \in \Gamma$, and
6. $(p', p)(X', X) \stackrel{\bar{a}}{\longrightarrow} (p', p)(X', X)$ for every $p' \in Q$ and $X' \in \Gamma$.

From a configuration $(p, q)[\alpha, \beta]$ the rules of the form 1. and 2. select either the left-hand or right-hand side and perform some transition in the selected process. The next possible transition (by rules 3. and 4.) is only from the opposite side of the configuration than in the previous step. Symbols of the form $X_a$ where $X \in \Gamma$ and $a \in \mathcal{A}ct$ are used to make sure that the transitions in these two steps are due to pushdown rules under the same label $a$. Note that in the rules 3. and 4. it is thus guaranteed that $|\alpha| = |\beta|$. Finally, the rules 5. and 6. introduce a number of self-loops in order to make visible the initial actions of the processes.

**Lemma 1.** *Let $\Delta$ be a vPDA system over the set of actions $\mathcal{A}ct$ and $pX$, $qY$ two of its processes. Let $(p, q)(X, Y)$ be a process in the system $\Delta'$ constructed above. Let*

- $\phi_{\sqsubseteq_s} \equiv \nu Z.[\ell]\langle r \rangle Z,$

- $\phi_{=_s} \equiv \phi_{\sqsubseteq_s} \wedge (\nu Z.[r]\langle \ell \rangle Z)$,
- $\phi_{\sqsubseteq_{cs}} \equiv \nu Z.\big([\ell]\langle r \rangle Z \wedge (\langle \mathcal{A}ct \rangle tt \Leftrightarrow \langle \overline{\mathcal{A}ct} \rangle tt)\big)$,
- $\phi_{=_{cs}} \equiv \phi_{\sqsubseteq_{cs}} \wedge \nu Z.\big([r]\langle \ell \rangle Z \wedge (\langle \mathcal{A}ct \rangle tt \Leftrightarrow \langle \overline{\mathcal{A}ct} \rangle tt)\big)$,
- $\phi_{\sqsubseteq_{rs}} \equiv \nu Z.\big([\ell]\langle r \rangle Z \wedge \bigwedge_{a \in \mathcal{A}ct} (\langle a \rangle tt \Leftrightarrow \langle \overline{a} \rangle tt)\big)$,
- $\phi_{=_{rs}} \equiv \phi_{\sqsubseteq_{rs}} \wedge \nu Z.\big([r]\langle \ell \rangle Z \wedge \bigwedge_{a \in \mathcal{A}ct} (\langle a \rangle tt \Leftrightarrow \langle \overline{a} \rangle tt)\big)$,
- $\phi_{\sqsubseteq_{2s}} \equiv \nu Z.\big([\ell]\langle r \rangle Z \wedge (\nu Z'.[r]\langle \ell \rangle Z')\big)$,
- $\phi_{=_{2s}} \equiv \phi_{\sqsubseteq_{2s}} \wedge \nu Z.\big([r]\langle \ell \rangle Z \wedge (\nu Z'.[\ell]\langle r \rangle Z')\big)$, and
- $\phi_{\sim} \equiv \nu Z.[\ell, r]\langle \ell, r \rangle Z$.

*For every $\bowtie \in \{\sqsubseteq_s, =_s, \sqsubseteq_{cs}, =_{cs}, \sqsubseteq_{rs}, =_{rs}, \sqsubseteq_{2s}, =_{2s}, \sim\}$ it holds that $pX \bowtie qY$ if and only if $(p,q)(X,Y) \models \phi_{\bowtie}$.*

**Theorem 1.** *Simulation, completed simulation, ready simulation and 2-nested simulation preorders and equivalences, as well as bisimulation equivalence are decidable on vPDA and all these problems are EXPTIME-complete.*

*Proof.* EXPTIME-hardness (for all relations between simulation preorder and bisimulation equivalence) follows from [17] as the pushdown automaton constructed in the proof is in fact a vPDA.

For the containment in EXPTIME observe that all our equivalence checking problems are reduced in polynomial time to model checking of a pushdown automaton against a fixed size formula of modal $\mu$-calculus. The complexity of the model checking problem for a pushdown automaton with $m$ states and $k$ stack symbols and a formula of the size $n_1$ and of the alternation depth $n_2$ is $O((k2^{cmn_1n_2})^{n_2}))$ for some constant $c$ [25]. In our case for a given vPDA system with $m$ states and $k$ stack symbols we construct a PDA system with $m^2$ states and with $O(k^3 \cdot |\mathcal{A}ct|)$ stack symbols (used in the transition rules). Hence the overall time complexity of checking whether two vPDA processes $pX$ and $qY$ are equivalent is $(k^3 \cdot |\mathcal{A}ct|)2^{O(m^2)}$. $\qquad\square$

### 3.2 Visibly Basic Process Algebra

We shall now focus on the complexity of preorder/equivalence checking for vBPA, a strict subclass of vPDA.

**Theorem 2.** *Simulation, completed simulation, ready simulation and 2-nested simulation preorders and equivalences, as well as bisimulation equivalence are P-complete on vBPA.*

*Proof.* Recall that a vBPA process is a vPDA processes with a single control state. By using the arguments from the proof of Theorem 1, the complexity of equivalence checking on vBPA is therefore $O(k^3 \cdot |\mathcal{A}ct|)$ where $k$ is the cardinality of the stack alphabet (and where $m = 1$). P-hardness was proved in [21] even for finite-state systems. $\qquad\square$
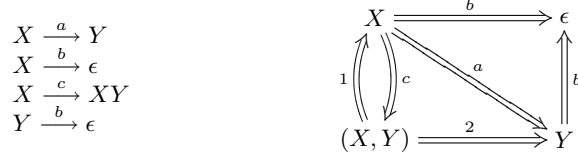
**Fig. 1.** Transformation of a vBPA into a finite-state system

In fact, for vBPA we can introduce even better complexity upper bounds by reducing it to preorder/equivalence checking on finite-state systems.

**Theorem 3.** *Simulation, completed simulation, ready simulation and 2-nested simulation preorders and equivalences, as well as bisimulation equivalence on vBPA is reducible to checking the same preorder/equivalence on finite-state systems. For any vBPA process $\Delta$ (with the natural requirement that every stack symbol appears at least in one rule from $\Delta$), the reduction is computable in time $O(|\Delta|)$ and outputs a finite-state system with $O(|\Delta|)$ states and $O(|\Delta|)$ transitions.*

*Proof.* Let $\mathcal{A}ct = \mathcal{A}ct_c \cup \mathcal{A}ct_r \cup \mathcal{A}ct_i$ be the set of actions and let $\Gamma$ be the stack alphabet of a given vBPA system $\Delta$ (we shall omit writing the control states as this is a singleton set). Let $S \stackrel{\text{def}}{=} \{(Y, Z) \in \Gamma \times \Gamma \mid \exists (X \stackrel{a}{\longrightarrow} YZ) \in \Delta$ for some $X \in \Gamma$ and $a \in \mathcal{A}ct_c \}$. We construct a finite-state transition system $T = (\Gamma \cup \{\epsilon\} \cup S, \mathcal{A}ct \cup \{1, 2\}, \Longrightarrow)$ for fresh actions 1 and 2 as follows. For every vBPA rule $(X \stackrel{a}{\longrightarrow} \alpha) \in \Delta$, we add the transitions:

- $X \stackrel{a}{\Longrightarrow} \epsilon$ if $a \in \mathcal{A}ct_r$ (and $\alpha = \epsilon$),
- $X \stackrel{a}{\Longrightarrow} Y$ if $a \in \mathcal{A}ct_i$ and $\alpha = Y$,
- $X \stackrel{a}{\Longrightarrow} (Y, Z)$ if $a \in \mathcal{A}ct_c$ and $\alpha = YZ$,
- $(Y, Z) \stackrel{1}{\Longrightarrow} Y$ if $a \in \mathcal{A}ct_c$ and $\alpha = YZ$, and
- $(Y, Z) \stackrel{2}{\Longrightarrow} Z$ if $a \in \mathcal{A}ct_c$ and $\alpha = YZ$ such that $Y \longrightarrow^* \epsilon$.

Note that the set $\{Y \in \Gamma \mid Y \longrightarrow^* \epsilon\}$ can be (by standard techniques) computed in time $O(|\Delta|)$. Moreover, the finite-state system $T$ has $O(|\Delta|)$ states and $O(|\Delta|)$ transitions. See Figure 1 for an example of the transformation.

Let us now observe that in vBPA systems we have the following decomposition property. It is the case that $X\alpha \sim X'\alpha'$ in $\Delta$ (where $X, X' \in \Gamma$ and $\alpha, \alpha' \in \Gamma^*$) if and only if in $\Delta$ the following two conditions hold: (i) $X \sim X'$ and (ii) if ($X \longrightarrow^* \epsilon$ or $X' \longrightarrow^* \epsilon$) then $\alpha \sim \alpha'$. Hence for any $X, Y \in \Gamma$ we have that $X \sim Y$ in $\Delta$ iff $X \sim Y$ in $T$. It is easy to check that the fact above holds also for any other preorder/equivalence as stated by the theorem. □

This means that for preorder/equivalence checking on vBPA we can use the efficient algorithms already developed for finite-state systems. For example, for finite-state transition systems with $k$ states and $t$ transitions, bisimilarity can be decided in time $O(t \log k)$ [19]. Hence bisimilarity on a vBPA system $\Delta$ is decidable in time $O(|\Delta| \cdot \log |\Delta|)$.

### 3.3 Visibly One-Counter Automata

We will now continue with studying preorder/equivalence checking problems on v1CA, a strict subclass of vPDA and an incomparable class with vBPA (w.r.t. bisimilarity). We start by showing PSPACE-hardness of the problems. The proof is by reduction from a PSPACE-complete problem of emptiness of one-way alternating finite automata over one-letter alphabet [11].

A *one-way alternating finite automaton over one-letter alphabet* is a 5-tuple $A = (Q_\exists, Q_\forall, q_0, \delta, F)$ where $Q_\exists$ and $Q_\forall$ are finite and disjoint sets of existential, resp. universal control states, $q_0 \in Q_\exists \cup Q_\forall$ is the initial state, $F \subseteq Q_\exists \cup Q_\forall$ is the set of final states and $\delta : Q_\exists \cup Q_\forall \to 2^{Q_\exists \cup Q_\forall}$ is the transition function.

A *computational tree* for an input word of the form $I^n$ (where $n$ is a natural number and $I$ is the only letter in the input alphabet) is a tree where every branch has exactly $n+1$ nodes labelled by control states from $Q_\exists \cup Q_\forall$ such that the root is labelled with $q_0$ and every non-leaf node that is already labelled by some $q \in Q_\exists \cup Q_\forall$ such that $\delta(q) = \{q_1, \ldots, q_k\}$ has either

- one child labelled by $q_i$ for some $i$, $1 \le i \le k$, if $q \in Q_\exists$, or
- $k$ children labelled by $q_1, \ldots, q_k$, if $q \in Q_\forall$.

A computational tree is *accepting* if the labels of all its leaves are final (i.e. belong to $F$). The language of $A$ is defined by $L(A) \overset{\text{def}}{=} \{I^n \mid I^n$ has some accepting computational tree $\}$.

The emptiness problem for one-way alternating finite automata over one-letter alphabet (denoted as EMPTY) is to decide whether $L(A) = \emptyset$ for a given automaton $A$. The problem EMPTY is known to be PSPACE-complete due to Holzer [11].

In what follows we shall demonstrate a polynomial time reduction from EMPTY to equivalence/preorder checking on visibly one-counter automata. We shall moreover show the reduction for any (arbitrary) relation between simulation preorder and bisimulation equivalence. This in particular covers all preorders/equivalences introduced in this paper.

**Lemma 2.** *All relations between simulation preorder and bisimulation equivalence are PSPACE-hard on v1CA.*

*Proof.* Let $A = (Q_\exists, Q_\forall, q_0, \delta, F)$ be a given instance of EMPTY. We shall construct a visibly one-counter automaton $\Delta$ over the set of actions $\mathcal{A}ct_c \overset{\text{def}}{=} \{i\}$, $\mathcal{A}ct_r \overset{\text{def}}{=} \{d_q \mid q \in Q_\exists \cup Q_\forall\}$, $\mathcal{A}ct_i \overset{\text{def}}{=} \{a, e\}$ and with control states $Q \overset{\text{def}}{=} \{p, p', t\} \cup \{q, q', t_q \mid q \in Q_\exists \cup Q_\forall\}$ such that

- if $L(A) = \emptyset$ then Defender has a winning strategy from $pZ$ and $p'Z$ in the bisimulation game (i.e. $pZ \sim p'Z$), and
- if $L(A) \ne \emptyset$ then Attacker has a winning strategy from $pZ$ and $p'Z$ in the simulation preorder game (i.e. $pZ \not\sqsubseteq_s p'Z$).

The intuition is that Attacker generates some counter value $n$ in both of the processes $pZ$ and $p'Z$ and then switches into a checking phase by changing the configurations to $q_0(n)$ and $q'_0(n)$. Now the players decrease the counter and change the control states according to the function $\delta$. Attacker selects the successor in any existential configuration, while Defender makes the choice of the successor in every universal configuration. Attacker wins if the players reach a pair of configurations $q(0)$ and $q'(0)$ where $q \in F$.

We shall now define the set of rules $\Delta$. The initial rules allow Attacker (by performing repeatedly the action $i$) to set the counter into an arbitrary number, i.e., Attacker generates a candidate word from $L(A)$.

$$
\begin{array}{ll}
pZ \xrightarrow{i} pIZ & p'Z \xrightarrow{i} p'IZ \\
pI \xrightarrow{i} pII & p'I \xrightarrow{i} p'II \\
pZ \xrightarrow{a} q_0 Z & p'Z \xrightarrow{a} q'_0 Z \\
pI \xrightarrow{a} q_0 I & p'I \xrightarrow{a} q'_0 I
\end{array}
$$

Observe that Attacker is at some point forced to perform the action $a$ (an infinite play is winning for Defender) and switch to the checking phase starting from $q_0(n)$ and $q'_0(n)$.

Now for every existential state $q \in Q_\exists$ with $\delta(q) = \{q_1, \ldots, q_k\}$ and for every $i \in \{1, \ldots, k\}$ we add the following rules.

$$
qI \xrightarrow{d_{q_i}} q_i \qquad q'I \xrightarrow{d_{q_i}} q'_i
$$

This means that Attacker can decide on the successor $q_i$ of $q$ and the players in one round move from the pair $q(n)$ and $q'(n)$ into $q_i(n-1)$ and $q'_i(n-1)$.

Next for every universal state $q \in Q_\forall$ with $\delta(q) = \{q_1, \ldots, q_k\}$ and for every $i \in \{1, \ldots, k\}$ we add the rules

$$
\begin{array}{ll}
qI \xrightarrow{a} tI & q'I \xrightarrow{a} t_{q_i} I \\
qI \xrightarrow{a} t_{q_i} I &
\end{array}
$$

and for every $q, r \in Q_\exists \cup Q_\forall$ such that $q \neq r$ we add

$$
\begin{array}{ll}
tI \xrightarrow{d_q} q & t_q I \xrightarrow{d_q} q' \\
& t_q I \xrightarrow{d_r} r \quad .
\end{array}
$$

These rules are more complex and they correspond to a particular implementation of so called *Defender's Choice Technique* (for further examples see e.g. [15]). We shall explain the idea by using Figure 2. Assume that $q \in Q_\forall$ and $\delta(q) = \{q_1, \ldots, q_k\}$. In the first round of the bisimulation game starting from $q(n)$ and $q'(n)$ where $n > 0$, Attacker is forced to take the move $q(n) \xrightarrow{a} t(n)$. On any other move Defender answers by immediately reaching a pair of syntactically equal processes (and thus wins the game). Defender's answer on Attacker's move $q(n) \xrightarrow{a} t(n)$ is to perform $q'(n) \xrightarrow{a} t_{q_i}(n)$ for some $i \in \{1, \ldots, k\}$. The second round thus starts from the pair $t(n)$ and $t_{q_i}(n)$. Should Attacker choose to play the action $d_r$ for some state $r$ such that $r \neq q_i$ (on either side), Defender
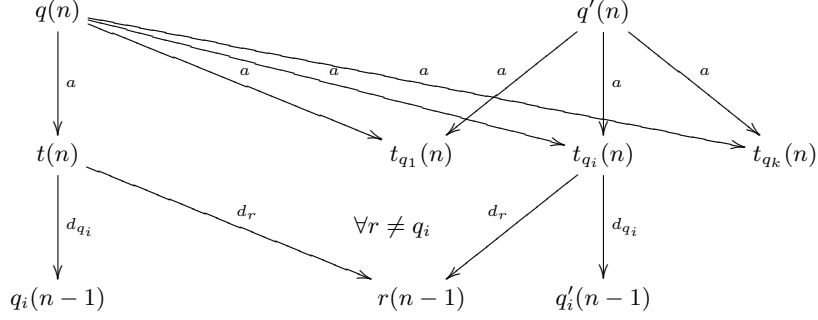
**Fig. 2.** Defender's Choice: $q \in Q_\forall$ and $\delta(q) = \{q_1, \ldots, q_k\}$

can again reach a syntactic equality and wins. Hence Attacker is forced to play the action $d_{q_i}$ on which Defender answers by the same action in the opposite process and the players reach the pair $q_i(n-1)$ and $q_i'(n-1)$. Note that it was Defender who selected the new control state $q_i$.

Finally, for every $q \in F$ we add the rule

$$qZ \xrightarrow{e} qZ \ .$$

It is easy to see that $\Delta$ is a visibly one-counter automaton. Moreover, if $L(A) = \emptyset$ then $pZ \sim p'Z$, and if $L(A) \neq \emptyset$ then $pZ \not\sqsubseteq_s p'Z$. □

*Remark 2.* The reduction above works also for a strict subclass of one-counter automata called one-counter nets (where it is not allowed to test for zero, see e.g. [13]). It is enough to replace the final rule $qZ \xrightarrow{e} qZ$ with two new rules $q \xrightarrow{e} q$ and $q'I \xrightarrow{e} q'I$ for every $q \in F$. Moreover, a slight modification of the system allows to show PSPACE-hardness of simulation preorder checking between one-counter automata and finite-state systems and vice versa. Hence the previously know DP lower bounds [13] for all relations between simulation preorder and bisimulation equivalence on one-counter nets (and one-counter automata) as well as of simulation preorder/equivalence between one-counter automata and finite-state systems, and between finite-state systems and one-counter automata are raised to PSPACE-hardness.

We are now ready to state the precise complexity of (bi)simulation-like pre-orders/equivalences on visibly one-counter automata.

**Theorem 4.** *Simulation, completed simulation, ready simulation and 2-nested simulation preorders and equivalences, as well as bisimulation equivalence are PSPACE-complete on v1CA.*

*Proof.* PSPACE-hardness follows from Lemma 2. Containment in PSPACE is due to Lemma 1 and due to [23] where it was very recently showed that model checking modal $\mu$-calculus on one-counter automata is decidable in PSPACE.

The only slight complication is that the system used in Lemma 1 is not necessarily a one-counter automaton. All stack symbols are of the form $(I, I)$ or $(Z, Z)$ which is fine, except for the very top of the stack where more stack symbols are used. Nevertheless, by standard techniques, the top of the stack can be remembered in the control states in order to apply the result from [23]. □

## 4 Decidability of Regularity Checking

In this section we ask the question whether a given vPDA process is equivalent to some finite-state system. Should this be the case, we call the given process *regular* (w.r.t. the considered equivalence). The main result of this section is a semantical characterization of regular vPDA processes via the property of unbounded popping and a polynomial time decision algorithm to test whether a given process satisfies this property.

Let $\mathcal{Act} = \mathcal{Act}_c \cup \mathcal{Act}_r \cup \mathcal{Act}_i$ be the set of actions of a given vPDA. We define a function $h : \mathcal{Act} \to \{-1, 0, +1\}$ by $h(a) = +1$ for all $a \in \mathcal{Act}_c$, $h(a) = -1$ for all $a \in \mathcal{Act}_r$, and $h(a) = 0$ for all $a \in \mathcal{Act}_i$. The function $h$ can be naturally extended to sequences of actions by $h(a_1 \ldots a_n) = \sum_{i \in \{1, \ldots, n\}} h(a_i)$. Observe now that for any computation $p\alpha \xrightarrow{w} q\beta$ we have $|\beta| = |\alpha| + h(w)$.

**Definition 1.** *Let $pX$ be a vPDA configuration. We say that $pX$ provides* unbounded popping *if for every natural number $d$ there is a configuration $q\beta$ and a word $w \in \mathcal{Act}^*$ such that $h(w) \le -d$ and $pX \longrightarrow^* q\beta \xrightarrow{w}$ .*

**Lemma 3.** *Let $pX$ be a vPDA configuration which provides unbounded popping. Then $pX$ is not regular w.r.t. trace equivalence.*

*Proof (Sketch).* By contradiction. Let $pX$ be trace equivalent to some finite-state system $A$ with $n$ states. Let us consider a trace $w_1 w_2$ such that $pX \xrightarrow{w_1} q\beta \xrightarrow{w_2}$ for some $q\beta$ and $h(w_2) \le -n$. Such a trace must exist because $pX$ provides unbounded popping. The trace $w_1 w_2$ must be executable also in $A$. However, because $A$ has $n$ states, during the computation on $w_2$, it must necessarily enter twice the same state such that it forms a loop on some substring $w'$ of $w_2$. We can moreover assume that $h(w') < 0$. This means that by taking the loop sufficiently many times $A$ can achieve a trace $w$ with $h(w) < 1$. However, this trace is not possible from $pX$ (any word $w$ such that $pX \xrightarrow{w}$ satisfies that $h(w) \ge -1$). This is a contradiction. □

**Lemma 4.** *Let $pX$ be a vPDA configuration which does not provide unbounded popping. Then $pX$ is regular w.r.t. isomorphism of labelled transition systems.*

*Proof.* Assume that $pX$ does not provide unbounded popping. In other words, there is a constant $d_{max}$ such that for every process $q\beta$ reachable from $pX$ it is the case that for any computation starting from $q\beta$, the stack height $|\beta|$ cannot be decreased by more than $d_{max}$ symbols. This means that in any reachable configuration it is necessary to remember only $d_{max}$ top-most stack symbols and hence the system can be up to isomorphism described as a finite-state system (in general of exponential size). □

**Theorem 5.** *Let pX be a vPDA configuration. Then, for any equivalence relation between trace equivalence and isomorphism of labelled transition systems, pX provides unbounded popping if and only if pX is not regular.*

*Proof.* Directly from Lemma 3 and Lemma 4. □

**Theorem 6.** *Regularity checking of vPDA w.r.t. any equivalence between trace equivalence and isomorphism of labelled transition systems (in particular also w.r.t. any equivalence considered in this paper) is decidable in deterministic polynomial time. The problems are P-complete for vPDA and vBPA and NL-complete for v1CA.*

*Proof (Sketch).* We can check for every $q \in Q$ and $Y \in \Gamma$ whether the regular set $post^*(qY) \cap pre^*(\{r\epsilon \mid r \in Q\})$ is infinite. If yes, this means that $qY$ has infinitely many different successors (with higher and higher stacks) such that all of them can be completely emptied. To see whether a given vPDA process $pX$ provides unbounded popping, it is now enough to test whether $pX \in pre^*(qY\Gamma^*)$ for some $qY$ satisfying the condition above. The test can be done in polynomial time because the sets $pre^*$ and $post^*$ are regular and computable in polynomial time as showed e.g. in [7]. The proofs of P-completeness and NL-completeness are in the full version of the paper. □

## 5 Conclusion

In the following table we provide a comparison of bisimulation, simulation and regularity (w.r.t. bisimilarity) checking on PDA, 1CA, BPA and their subclasses vPDA, v1CA, vBPA. Results achieved in this paper are in bold.

| | $\sim$ | $\sqsubseteq_s$ and $=_s$ | $\sim$-regularity |
|---|---|---|---|
| PDA | decidable [22] <br> EXPTIME-hard [17] | undecidable [10] | ? <br> EXPTIME-hard [17, 24] |
| vPDA | **in EXPTIME** <br> EXPTIME-hard [17] | **in EXPTIME** <br> EXPTIME-hard [17] | **P-compl.** |
| 1CA | decidable [12] <br> **PSPACE-hard** | undecidable [14] | decidable [12] <br> P-hard [5, 24] |
| v1CA | **PSPACE-compl.** | **PSPACE-compl.** | **NL-compl.** |
| BPA | in 2-EXPTIME [8] <br> PSPACE-hard [24] | undecidable [10] | in 2-EXPTIME [9, 8] <br> PSPACE-hard [24] |
| vBPA | **in P** <br> P-hard [5] | **in P** <br> P-hard [21] | **P-compl.** |

In fact, our results about EXPTIME-completeness for vPDA, PSPACE-completeness for v1CA and P-completeness for vBPA hold for all preorders and equivalences between simulation preorder and bisimulation equivalence studied in the literature (like completed simulation, ready simulation and 2-nested simulation). The results confirm a general trend seen in the classical language theory

of pushdown automata: a relatively minor restriction (from the practical point of view) of being able to distinguish call, return and internal actions often significantly improves the complexity of the studied problems and sometimes even changes undecidable problems into decidable ones, moreover with reasonable complexity upper bounds.

All the upper bounds proved in this paper are matched by the corresponding lower bounds. Here the most interesting result is PSPACE-hardness of preorder/equivalence checking on v1CA for all relations between simulation preorder and bisimulation equivalence. As noted in Remark 2, this proof improves also a number of other complexity lower bounds for problems on standard one-counter nets and one-counter automata, which were previously known to be only DP-hard (DP-hardness is, most likely, a slightly stronger result than NP and co-NP hardness).

Finally, we have proved that for all the studied equivalences, the regularity problem is decidable in polynomial time. Checking whether an infinite-state process is equivalent to some regular one is a relevant question because many problems about such a process can be answered by verifying the equivalent finite-state system and for finite-state systems many efficient algorithms have been developed. A rather interesting observation is that preorder/equivalence checking on vBPA for preorders/equivalences between simulation and bisimilarity can be polynomially translated to verification problems on finite-state systems. On the other hand, the class of vBPA processes is significantly larger than the class of finite-state processes and hence the questions, whether for a given vPDA (or v1CA) process there is some equivalent vBPA process, are of a particular interest. We shall investigate these questions in the future research, as well as a generalization of the unbounded popping property for visibly pushdown automata that enable to perform return actions also on the empty stack.

# References

1. L. Aceto, W. Fokkink, and A. Ingólfsdóttir. 2-nested simulation is not finitely equationally axiomatizable. In *Proc. of STACS'01*, vol. 2010 of *LNCS*, p. 39–50. Springer, 2001.
2. R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Proc. of TACAS'04*, vol. 2988 of *LNCS*, p. 467–481. Springer, 2004.
3. R. Alur, V. Kumar, P. Madhusudan, and M. Viswanathan. Congruences for visibly pushdown languages. In *Proc. of ICALP'05*, vol. 3580 of *LNCS*, p. 1102–1114. Springer, 2005.
4. R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proc. of STOC'04*, p. 202–211. ACM Press, 2004.

5. J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4(6A):638–648, 1992.

6. V. Bárány, Ch. Löding, and O. Serre. Regularity problems for visibly pushdown languages. In *Proc. of STACS'06*, vol. 3884 of *LNCS*, p. 420–431. Springer, 2006.

7. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proc. of CONCUR'97*, vol. 1243 of *LNCS*, p. 135–150. Springer, 1997.

8. O. Burkart, D. Caucal, and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proc. of MFCS'95*, vol. 969 of *LNCS*, p. 423–433. Springer, 1995.

9. O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proc. of CONCUR'96*, vol. 1119 of *LNCS*, p. 247–262. Springer, 1996.

10. J.F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 115(2):353–371, 1994.

11. M. Holzer. On emptiness and counting for alternating finite automata. In *Proc. of DLT'95*, pages 88–97. World Scientific, 1996.

12. P. Jančar. Decidability of bisimilarity for one-counter processes. *Information and Computation*, 158(1):1–17, 2000.

13. P. Jančar, A. Kučera, F. Moller, and Z. Sawa. DP lower bounds for equivalence-checking and model-checking of one-counter automata. *Information and Computation*, 188(1):1–19, 2004.

14. P. Jančar, F. Moller, and Z. Sawa. Simulation problems for one-counter machines. In *Proc. of SOFSEM'99*, vol. 1725 of *LNCS*, p. 404–413. Springer, 1999.

15. P. Jančar and J. Srba. Highly undecidable questions for process algebras. In *Proc. of TCS'04*, p. 507–520. Kluwer Academic Publishers, 2004.

16. D. Kozen. Results on the propositional $\mu$-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

17. A. Kučera and R. Mayr. On the complexity of semantic equivalences for pushdown automata and BPA. In *Proc. of MFCS'02*, vol. 2420 of *LNCS*, p. 433–445. Springer, 2002.

18. A. Murawski and I. Walukiewicz. Third-order idealized algol with iteration is decidable. In *Proc. of FOSSACS'05*, vol. 3441 of *LNCS*, p. 202–218, 2005.

19. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, December 1987.

20. C. Pitcher. Visibly pushdown expression effects for XML stream processing. In *Proc. of PLAN-X*, pages 5–19, 2005.

21. Z. Sawa and P. Jančar. P-hardness of equivalence testing on finite-state processes. In *Proc. of SOFSEM'01*, vol. 2234 of *LNCS*, p. 326–345. Springer, 2001.

22. G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proc. of FOCS'98*, p. 120–129. IEEE Computer Society, 1998.

23. O. Serre. Parity games played on transition graphs of one-counter processes. In *Proc. of FOSSACS'06*, vol. 3921 of *LNCS*, p. 337–351. Springer, 2006.

24. J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proc. of ICALP'02*, vol. 2380 of *LNCS*, p. 716–727. Springer, 2002.

25. I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.