BRICS RS-06-13 J. Srba: Visibly Pushdown Automata: From Language Equivalence to Simulation and Bisimulation

BRICS

Basic Research in Computer Science

Visibly Pushdown Automata: From Language Equivalence to Simulation and Bisimulation

Jiří Srba

BRICS Report Series RS-06-13

ISSN 0909-0878 July 2006

See back inner page for a list of recent BRICS Report Series publications.
Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
IT-parken, Aabogade 34
DK–8200 Aarhus N
Denmark
Telephone: +45 8942 9300
Telefax: +45 8942 5601
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/06/13/`

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.

* The author is supported in part by Institute for Theoretical Computer Science, project No. 1M0545.
** **B**asic **R**esearch in **C**omputer **S**cience, Centre of the Danish National Research Foundation.

Recently, visibly pushdown languages have been intensively studied and applied to e.g. program analysis [2], XML processing [22] 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 [20].

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 [15] for equivalence checking problems on ordinary one-counter automata and one-counter nets and some other problems (see Remark 4). 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 μ-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 [19]. 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 [19] 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.

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 $\xrightarrow{a}$ as a binary relation on S where $s \xrightarrow{a} s'$ iff $(s, a, s') \in \longrightarrow$. The notation can be naturally extended to $s \xrightarrow{w} s'$ 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 $I(s) \stackrel{\text{def}}{=} \{a \in \mathcal{A}ct \mid \exists s' \in S.\ s \xrightarrow{a} s'\}$.

We shall now define the studied equivalences/preorders which are between simulation and bisimilarity. A complete picture of Glabbeek's linear/branching time hierarchy (spectrum) of behavioral equivalences is available in [29, 30]. Given $(S, \mathcal{A}ct, \longrightarrow)$, 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 $s \xrightarrow{a} s'$ there is t' such that $t \xrightarrow{a} t'$ 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. The hierarchy is depicted in Figure 1.

$$
\begin{tikzcd}
\text{bisimulation} \arrow[d] \\
\text{2-nested simulation} \arrow[d] \\
\text{ready simulation} \arrow[d] \\
\text{completed simulation} \arrow[d] \\
\text{simulation}
\end{tikzcd}
$$

Fig. 1. Hierarchy of simulation-like preorders/equivalences

We shall use a standard game-theoretic characterization of (bi)similarity [28, 27]. 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 $s \not\sim t$ 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 Γ be a finite set of stack symbols and let Q be a finite set of control states. We assume that the sets $\mathcal{A}ct$, Γ and Q are pairwise disjoint. A *pushdown automaton* (PDA) over the set of actions $\mathcal{A}ct$, stack alphabet Γ and control states Q is a finite set Δ 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 Δ determines a labelled transition system $T(\Delta) = (S, \mathcal{A}ct, \longrightarrow)$ where the states are configurations of the form state$\times$stack (i.e. $S = Q \times \Gamma^*$ and configurations like (p, α) are usually written as $p\alpha$ where the top of the stack α 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^nZ 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 α):

- 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.

Example 1. Consider the vPDA rules $pX \xrightarrow{a} pXY$, $pX \xrightarrow{b} p\epsilon$, $pY \xrightarrow{c} p\epsilon$ where $a \in \mathcal{A}ct_c$ and $b, c \in \mathcal{A}ct_r$. The transition system generated by the root pX looks as follows.

$$\begin{tikzcd}
pX \arrow[r, "a"] \arrow[d, "b"] & pXY \arrow[r, "a"] \arrow[d, "b"] & pXYY \arrow[r, "a"] \arrow[d, "b"] & pXYYY \arrow[r, "a"] \arrow[d, "b"] & \cdots \\
p\epsilon & pY \arrow[l, "c"'] & pYY \arrow[l, "c"'] & pYYY \arrow[l, "c"'] & \cdots \arrow[l, "c"']
\end{tikzcd}$$

The vPDA process pX (which is in fact also a vBPA process) generates an infinite-state transition system, which is not trace equivalent (and hence also e.g. not bisimilar) to any finite-state system. Hence the class of visibly pushdown processes strictly contains all finite-state processes. □

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?

Remark 2. Note that the problem of equivalence checking of two configurations belonging to different visibly pushdown automata (under the same partitioning of actions) is also covered by the definition of the problem above. We can always consider only a single vPDA by making a disjoint union of the respective pushdown automata.

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 μ-calculus formula.

Let Δ 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 Γ and control states Q. We shall construct a PDA Δ' over the actions $\mathcal{A}ct' \stackrel{\text{def}}{=} \mathcal{A}ct \cup \overline{\mathcal{A}ct} \cup \{\ell, r\}$ where $\overline{\mathcal{A}ct} \stackrel{\text{def}}{=} \{\overline{a} \mid a \in \mathcal{A}ct\}$, 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 ℓ 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 $|\alpha| = |\beta|$):

$$\begin{aligned} [X\alpha, Y\beta] &\stackrel{\text{def}}{=} (X, Y)[\alpha, \beta] \\ [\epsilon, \epsilon] &\stackrel{\text{def}}{=} \epsilon \ \ . \end{aligned}$$

The mapping provides the possibility to merge stacks.

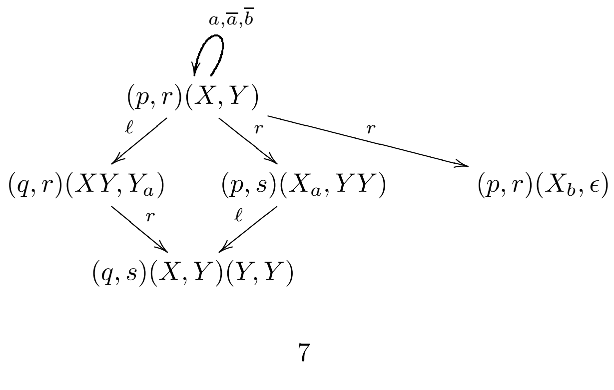
Assume a given pair of vPDA processes pX and qY. Our aim is to effectively construct a new PDA system Δ' such that for every $\bowtie \in \{\sqsubseteq_s, =_s, \sqsubseteq_{cs}, =_{cs}, \sqsubseteq_{rs}, =_{rs}, \sqsubseteq_{2s}, =_{2s}, \sim\}$ it is the case that $pX \bowtie qY$ in Δ if and only if $(p,q)(X,Y) \models \phi_{\bowtie}$ in Δ' for a fixed μ-calculus formula $\phi_{\bowtie}$. We refer the reader to [18] for the introduction to the modal μ-calculus.

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

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

Example 2. Consider the vPDA rules: $pX \xrightarrow{a} qXY$, $rY \xrightarrow{a} sYY$, $rY \xrightarrow{b} r$ where $a \in \mathcal{A}ct_c$ and $b \in \mathcal{A}ct_r$. The transition system generated (in Δ') by the root $(p,r)(X,Y)$ looks as follows.



Note that the configuration $(p, r)(X_b, \epsilon)$ is stuck because there is no b-labelled transition from the state pX. By introducing self-loops we can observe for every stable configuration (where the top of the stack is of the form $\Gamma \times \Gamma$), what actions are currently enabled in the left-hand side process (actions from $\mathcal{A}ct$) and in the right-hand side process (actions from $\overline{\mathcal{A}ct}$). $\square$

Lemma 1. *Let Δ 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 Δ' 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 $(p, q)(X, Y) \models \phi_{\bowtie}$.

Proof. We shall argue only for the case of bisimulation. Proofs for the other cases are similar.

"$\Rightarrow$": We show that $p\alpha \sim q\beta$ implies that $(p, q)[\alpha, \beta] \models \nu Z.[\ell, r]\langle \ell, r\rangle Z$. We prove that the set $F \overset{\text{def}}{=} \{(p, q)[\alpha, \beta] \mid p\alpha \sim q\beta \wedge |\alpha| = |\beta|\}$ is a fixed point of the function corresponding to our formula. This amounts to checking (without loss of generality as the other case is symmetric) that for every $(p, q)[\alpha, \beta] \in F$ and for every ℓ-move $(p, q)[\alpha, \beta] \xrightarrow{\ell} c$ for some configuration c of Δ', there is an r-move $c \xrightarrow{r} (p', q')[\alpha', \beta']$ such that $(p', q')[\alpha', \beta'] \in F$. As $\nu Z.[\ell, r]\langle \ell, r\rangle Z$ is the greatest fixed point, this will establish our claim. We remind the reader of the fact that there are no ℓ-transitions enabled from the configuration c.

Let $(p, q)[\alpha, \beta] \in F$ and let $(p, q)[\alpha, \beta] \xrightarrow{\ell} (p', q)(\alpha', Y_a)[\gamma, \delta]$ due to some rule $(pX \xrightarrow{a} p'\alpha') \in \Delta$ where $\alpha = X\gamma$ and $\beta = Y\delta$. This means that $p\alpha = pX\gamma \xrightarrow{a} p'\alpha'\gamma$ and because $p\alpha \sim q\beta$ we have that $q\beta = qY\delta \xrightarrow{a} q'\beta'\delta$ due to some rule $(qY \xrightarrow{a} q'\beta') \in \Delta$ such that $p'\alpha'\gamma \sim q'\beta'\delta$. Hence $(p', q)(\alpha', Y_a)[\gamma, \delta] \xrightarrow{r} (p', q')[\alpha', \beta'][\gamma, \delta] = (p', q')[\alpha'\gamma, \beta'\delta]$ and $(p', q')[\alpha'\gamma, \beta'\delta] \in F$ as required.

"$\Leftarrow$": We prove that $(p, q)[\alpha, \beta] \models \nu Z.[\ell, r]\langle \ell, r\rangle Z$ implies that $p\alpha \sim q\beta$. To do so we define a binary relation $R \overset{\text{def}}{=} \{(p\alpha, q\beta) \mid (p, q)[\alpha, \beta] \models \nu Z.[\ell, r]\langle \ell, r\rangle Z \wedge |\alpha| = |\beta|\}$ and show that R is a bisimulation. Let $(p\alpha, q\beta) \in R$ and let us without loss of generality (the other case is completely symmetric) assume that $p\alpha \xrightarrow{a} p'\alpha'$ due to some rule $(pX \xrightarrow{a} p'\alpha'') \in \Delta$. Hence $\alpha = X\gamma$ and $\alpha' = \alpha''\gamma$ for some $\gamma \in \Gamma^*$. We aim to show that also $q\beta \xrightarrow{a} q'\beta'$ such that $(p', q')[\alpha', \beta'] \models$

$\nu Z.[\ell, r]\langle \ell, r\rangle Z$ and thus $(p'\alpha', q'\beta') \in R$. The fact that $p\alpha \xrightarrow{a} p'\alpha'$ means that in Δ' we have a transition $(p,q)[\alpha,\beta] = (p,q)[X\gamma,\beta] \xrightarrow{\ell} (p',q)(\alpha'', Y_a)[\gamma,\beta']$ where $\beta = Y\beta'$. As $(p,q)[\alpha,\beta]$ satisfies the formula $\nu Z.[\ell, r]\langle \ell, r\rangle Z$ (and due to the unfolding law also the formula $[\ell, r]\langle \ell, r\rangle(\nu Z.[\ell, r]\langle \ell, r\rangle Z))$, we have that $(p',q)(\alpha'', Y_a)[\gamma,\beta'] \xrightarrow{r} (p',q')[\alpha'',\beta''][\gamma,\beta']$ due to some rule $(qY \xrightarrow{a} q'\beta'') \in \Delta$ such that $(p',q')[\alpha'',\beta''][\gamma,\beta'] \models \nu Z.[\ell, r]\langle \ell, r\rangle Z$. Note that from $(p',q)(\alpha'', Y_a)[\gamma,\beta']$ no r-action is enabled on the "left-hand side" and there is no ℓ-action available, so this is indeed the only possibility. This, however, means that $(p',q')[\alpha'',\beta''][\gamma,\beta'] = (p',q')[\alpha''\gamma,\beta''\beta'] = (p',q')[\alpha',\beta''\beta']$, and $q\beta \xrightarrow{a} q'\beta''\beta'$, which implies that $(p'\alpha', q'\beta''\beta') \in R$. $\square$

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 [19] 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 μ-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 [31]. 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 $(k^3 \cdot |\mathcal{A}ct|)2^{O(m^2)}$. $\square$

Remark 3. A straightforward optimization of the presented construction can reduce the number of control states in Δ'. The idea is to move the information about the pair of current control states in Δ' into the stack alphabet. The only place where states have to be used is when a pop action is performed as the set of the next control states has to be down-propagated in the control state unit. For a given vPDA Δ we can hence define a parameter called the *number of return points* as the cardinality of the set $\{q \in Q \mid (pX \xrightarrow{a} q\epsilon) \in \Delta\}$. The complexity of equivalence checking is then exponential only in the number of return points.

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.*

$$X \xrightarrow{a} Y$$
$$X \xrightarrow{b} \epsilon$$
$$X \xrightarrow{c} XY$$
$$Y \xrightarrow{b} \epsilon$$

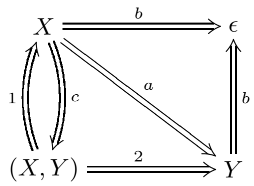


Fig. 2. Transformation of a vBPA into a finite-state system

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 [23] even for finite-state systems. $\square$

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 Δ (with the natural requirement that every stack symbol appears at least in one rule from Δ), 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 Γ be the stack alphabet of a given vBPA system Δ (we shall omit writing the control states as this is a singleton set). Let $S \stackrel{\text{def}}{=} \{(Y, Z) \in \Gamma \times \Gamma \mid \exists (X \xrightarrow{a} YZ) \in \Delta$ 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 $(X \xrightarrow{a} \alpha) \in \Delta$, 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 2 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 Δ (where $X, X' \in \Gamma$ and $\alpha, \alpha' \in \Gamma^*$) if and only if in Δ 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 Δ 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. $\square$

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 $O(t \log k)$ [21]. Hence bisimilarity on a vBPA system Δ 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 [13].

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, $1 \leq i \leq k$, 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 $L(A) \stackrel{\text{def}}{=} \{I^n \mid I^n \text{ has some accepting computational tree}\}$.

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 [13].

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 Δ over the set of actions $\mathcal{A}ct_c \stackrel{\text{def}}{=} \{i\}$, $\mathcal{A}ct_r \stackrel{\text{def}}{=} \{d_q \mid q \in Q_\exists \cup Q_\forall\}$, $\mathcal{A}ct_i \stackrel{\text{def}}{=} \{a, e\}$ and with control states $Q \stackrel{\text{def}}{=} \{p, p', t\} \cup \{q, q', t_q \mid q \in Q_\exists \cup Q_\forall\}$ 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 $L(A) \neq \emptyset$ 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 δ. 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 Δ. 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 $L(A)$.

$$\begin{array}{ll} pZ \xrightarrow{i} pIZ & p'Z \xrightarrow{i} p'IZ \\ pI \xrightarrow{i} pII & p'I \xrightarrow{i} p'II \\ pZ \xrightarrow{a} q_0Z & p'Z \xrightarrow{a} q_0'Z \\ pI \xrightarrow{a} q_0I & p'I \xrightarrow{a} q_0'I \end{array}$$

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

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

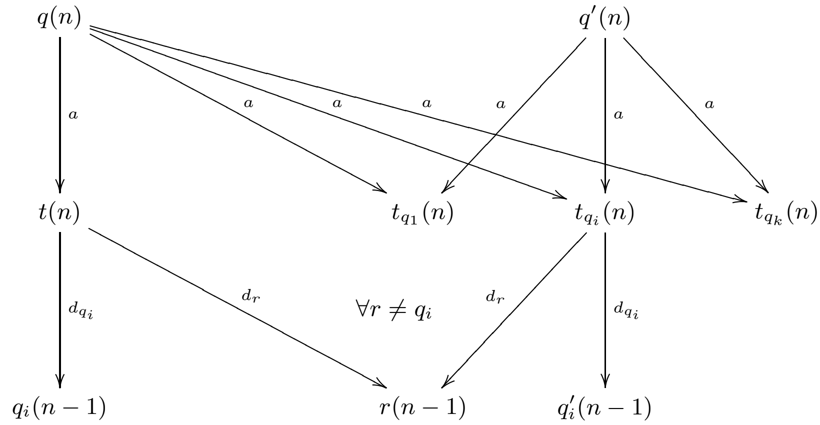


Fig. 3. Defender's Choice: $q \in Q_\forall$ and $\delta(q) = \{q_1, \dots, q_k\}$

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. [17]). We shall explain the idea by using Figure 3. Assume that $q \in Q_\forall$ and $\delta(q) = \{q_1, \dots, q_k\}$. 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 $i \in \{1, \dots, k\}$. 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 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 Δ is a visibly one-counter automaton and we shall now argue for the correctness of the reduction.

Assume that $L(A) = \emptyset$. We shall argue that Defender has a winning strategy in the bisimulation game starting from pZ and $p'Z$. In the first phase Attacker can generate an arbitrary number of the symbols I on the stacks. At some point he has to perform the action a (because Defender wins any infinite game) and switch to $q_0(n)$ and $q'_0(n)$ for some n. The players now remove the symbols I one by one and change the control states according to the function δ. As $L(A) = \emptyset$, we know that no computational tree can be accepting. This means that whatever choices Attacker makes in existential states, Defender can still select suitable successors of universal states such that when the players empty the whole counter and arrive to the pair qZ and $q'Z$, Defender guarantees that $q \notin F$. Therefore qZ and $q'Z$ are stuck and thus Defender has a winning strategy in the bisimulation game.

On the other hand, if $L(A) \neq \emptyset$, we will demonstrate Attacker's winning strategy in the simulation preorder game starting from pZ and $p'Z$. Attacker first forces (by repeatedly performing the action i followed by one action a) to reach a pair of states $q_0(n)$ and $q'_0(n)$ such that $I^n \in L(A)$. In the checking phase, there is an accepting computational tree for the word I^n and hence Attacker can make existential choices such that whatever universal choices Defender makes, the players arrive to the situation qZ and $q'Z$ for some $q \in F$. Now the attacker wins by playing $qZ \xrightarrow{e} qZ$ to which Defender has no answer. Notice that during the whole game (and particularly during the part where Defender chooses a successor of a universal state) Attacker can make his moves only on the left-hand side. Therefore $pZ \not\sqsubseteq_s p'Z$ as required. □

Remark 4. 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. [15]). 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 [15] 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 preorders/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 [25] where it was very recently showed that model checking modal μ-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 [25]. □

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 $\mathcal{A}ct = \mathcal{A}ct_c \cup \mathcal{A}ct_r \cup \mathcal{A}ct_i$ be the set of actions of a given vPDA. We define a function $h : \mathcal{A}ct \to \{-1, 0, +1\}$ by $h(a) = +1$ for all $a \in \mathcal{A}ct_c$, $h(a) = -1$ for all $a \in \mathcal{A}ct_r$, and $h(a) = 0$ for all $a \in \mathcal{A}ct_i$. The function h can be naturally extended to sequences of actions by $h(a_1 \dots a_n) = \sum_{i \in \{1,\dots,n\}} h(a_i)$. 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 $w \in \mathcal{A}ct^*$ such that $h(w) \leq -d$ 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 $h(w_2) \leq -n$. 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 $h(w) \geq -1$). This is a contradiction. $\square$

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). $\square$

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. $\square$

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, 10].

We shall now argue that regularity for vPDA and vBPA is P-hard. Let Δ be a BPA system over the set of actions $\mathcal{A}ct$ and a stack alphabet Γ. Let $X \in \Gamma$. The language of X recognized by the empty stack is defined as $L(X) = \{w \in \mathcal{A}ct^* \mid X \xrightarrow{w} \epsilon\}$. It is known that the problem whether $L(X) = \emptyset$ (which we shall call BPA-EMPTY) is P-hard, even under the assumption that there are only finitely many states reachable from the process X. This follows from a simple logarithmic space reduction from the Monotone Circuit Value problem (see [11, p. 177]). We shall reduce BPA-EMPTY to regularity checking on vBPA. Let Δ together with a stack symbol $X \in \Gamma$, which has finitely many reachable states, be a given instance of BPA-EMPTY. We construct (in logarithmic space) a vBPA system Δ' over the partitioned action alphabet $\mathcal{A}ct_c = \{c\}$, $\mathcal{A}ct_r = \{r\}$, $\mathcal{A}ct_i = \{i, e\}$ and the stack alphabet $\Gamma' = \Gamma \cup \{X', B, C, D\}$, where X', B, C and D are fresh stack symbols, such that

- if $L(X) = \emptyset$ in Δ then X' is a regular process in Δ' w.r.t. isomorphism, and
- if $L(X) \neq \emptyset$ in Δ then X' is not a regular process in Δ' w.r.t. trace equivalence.

We build Δ' from Δ as follows:

- for every $(Y \xrightarrow{a} \alpha) \in \Delta$ where $|\alpha| = 2$ we add to Δ' the rule $Y \xrightarrow{c} \alpha$,
- for every $(Y \xrightarrow{a} \alpha) \in \Delta$ where $|\alpha| = 0$ we add to Δ' the rule $Y \xrightarrow{r} \alpha$, and
- for every $(Y \xrightarrow{a} \alpha) \in \Delta$ where $|\alpha| = 1$ we add to Δ' the rule $Y \xrightarrow{i} \alpha$.

This does not change the answer to the emptiness problem and the system Δ' becomes visibly BPA. If we now add the following rules to Δ'

$$X' \xrightarrow{c} XB \qquad B \xrightarrow{e} C \qquad C \xrightarrow{c} CD \qquad C \xrightarrow{r} \epsilon \qquad D \xrightarrow{r} \epsilon$$

then it is easy to see that X' is regular if and only if $L(X) = \emptyset$. Obviously, Δ' is visibly BPA. Hence regularity checking on vBPA (and vPDA) is P-hard.

Finally, we show that regularity checking on visibly one-counter automata is NL-complete. NL-hardness follows immediately from the fact that the regularity checking problem naturally contains the reachability problem on finite-state systems (by a similar construction as showed above). The containment in nondeterministic logarithmic space is by the observation that a given visibly one-counter process $q_0(0)$ in Δ, where Δ has n control states, provides unbounded popping if and only if there exist two control states p and p' such that

1. $q_0(0) \xrightarrow{w_1} p(n_1)$ for some w_1 and n_1 such that $n_1 \geq n$ and $h(w') \leq n^2 + 2n$ for every prefix w' of w_1,

2. $p(n) \xrightarrow{w_2} p(n_2)$ for some w_2 and n_2 such that $n_2 > n$ and $|w_2| \leq n$,
3. $p(n) \xrightarrow{w_3} p'(n_3)$ for some w_3 and n_3 such that $|w_3| \leq n$, and
4. $p'(n) \xrightarrow{w_4} p'(n_4)$ for some w_4 and n_4 such that $n_4 < n$ and $|w_4| \leq n$.

Note that due to the restrictions on the lengths of the action sequences in points 2., 3. and 4., no transition is performed from any configuration where the counter value is zero. Hence the same computations are possible also for any greater initial counter value.

The idea is that the process $q_0(0)$ provides unbounded popping iff there is a possibility to arbitrarily increase the counter value (by means of the cycle from the control state p in condition 2.) and then reach a control state p' (condition 3.) in which the counter value can be arbitrarily decreased (condition 4.). Initially, condition 1. guarantees that the state p can be reached with a sufficiently large counter value. The extra requirement $h(w') \leq n^2 + 2n$ for any prefix w' of w_1 in condition 1. is harmless because if the state $p(n_1)$ is reachable then it is not necessary that the counter value during the computation grows to more than n^2+2n. To show that, we first observe that we can require that the counter value n_1 satisfies $n \leq n_1 \leq 2n$. For the sake of contradiction, let the control state p be reachable from the initial configuration such that the minimal counter value n_1 is greater than $2n$. We will show that we can then reach p with a counter value strictly smaller (while still at least n). Let us consider the suffix of this computation where all configurations have the counter values greater or equal to n. Now, in the region of configurations with the counter values between n and $2n$, there are necessarily two configurations $r(n')$ and $r(n'')$ for some control state r such that $n \leq n' < n'' \leq 2n$ and $r(n')$ precedes $r(n'')$. By removing the part of the computation between $r(n')$ and $r(n'')$ we achieve a computation that reaches the control state p with a strictly lower counter value.

We can hence assume that $p(n_1)$ is reachable such that $n \leq n_1 \leq 2n$. Should the counter grow to more than n^2+2n during this computation then there would necessarily appear two configurations with the same counter value (greater than $2n$) and the same control state and hence we could find a shorter sequence of actions to reach $p(n_1)$.

We shall now argue that the extra restrictions in conditions 2., 3. and 4. are harmless too. In condition 2., for the sake of contradiction, assume that from the control state p we can reach p with higher counter value (and never test for zero during the computation) such that the shortest sequence of actions to achieve this is strictly longer than n. On such a sequence, there are necessarily two configurations $r(n')$ and $r(n'')$ with the same control state r such that $r(n')$ precedes $r(n'')$. If $n' \geq n''$ than we can simply remove the part of the computation between these two configurations and reach the control state p with a possibly even greater counter value than before. If $n' < n''$ then we could have initially selected the control state r instead of p, because there is a loop on the control state r which increases the counter value. Similarly, we can show that the restrictions in points 3. and 4. are harmless too.

Finally, we finish be noting that the control states p and p' above can be nondeterministically guessed and the conditions 1. - 4. verified in nondeterministic logarithmic space. Hence the regularity checking problem for visibly one-counter automata is in NL. □

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 [24] EXPTIME-hard [19]	undecidable [12]	? EXPTIME-hard [19, 26]
vPDA	**in EXPTIME** EXPTIME-hard [19]	**in EXPTIME** EXPTIME-hard [19]	**P-compl.**
1CA	decidable [14] **PSPACE-hard**	undecidable [16]	decidable [14] P-hard [5, 26]
v1CA	**PSPACE-compl.**	**PSPACE-compl.**	**NL-compl.**
BPA	in 2-EXPTIME [8] PSPACE-hard [26]	undecidable [12]	in 2-EXPTIME [9, 8] PSPACE-hard [26]
vBPA	**in P** P-hard [5]	**in P** P-hard [23]	**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 4, 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.

Acknowledgments. I would like to thank Markus Lohrey for a discussion at ETAPS'06 and for a reference to PSPACE-completeness of the emptiness problem for alternating automata over one-letter alphabet. My thanks go also to the referees of CSL'06 for their useful comments and for suggesting the P-hardness proof of regularity checking for vPDA and vBPA.

References

1. L. Aceto, W. Fokkink, and A. Ingólfsdóttir. 2-nested simulation is not finitely equationally axiomatizable. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS'01)*, volume 2010 of *LNCS*, pages 39–50. Springer-Verlag, 2001.
2. R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, volume 2988 of *LNCS*, pages 467–481. Springer-Verlag, 2004.
3. R. Alur, V. Kumar, P. Madhusudan, and M. Viswanathan. Congruences for visibly pushdown languages. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 1102–1114. Springer-Verlag, 2005.
4. R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*, pages 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 *Proceedings of the 23rd Annual Symposioum on Theoretical Aspects of Computer Science (STACS'06)*, volume 3884 of *LNCS*, pages 420–431. Springer-Verlag, 2006.
7. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In *Proceedings of the 8th International Conference on Concurrency Theory (CONCUR'97)*, volume 1243 of *LNCS*, pages 135–150. Springer-Verlag, 1997.
8. O. Burkart, D. Caucal, and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *LNCS*, pages 423–433. Springer-Verlag, 1995.

9. O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR'96)*, volume 1119 of *LNCS*, pages 247–262. Springer-Verlag, 1996.
10. J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV'00)*, volume 1855 of *LNCS*, pages 232–247. Springer-Verlag, 2000.
11. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 1995.
12. J.F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 115(2):353–371, 1994.
13. M. Holzer. On emptiness and counting for alternating finite automata. In *Proceedings of the 2nd International Conference on Developments in Language Theory (DLT'95)*, pages 88–97. World Scientific, 1996.
14. P. Jančar. Decidability of bisimilarity for one-counter processes. *Information and Computation*, 158(1):1–17, 2000.
15. 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.
16. P. Jančar, F. Moller, and Z. Sawa. Simulation problems for one-counter machines. In *Proceedings of the 26th Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'99)*, volume 1725 of *LNCS*, pages 404–413. Springer-Verlag, 1999.
17. P. Jančar and J. Srba. Highly undecidable questions for process algebras. In *Proceedings of the 3rd IFIP International Conference on Theoretical Computer Science (TCS'04)*, Exploring New Frontiers of Theoretical Informatics, pages 507–520. Kluwer Academic Publishers, 2004.
18. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
19. A. Kučera and R. Mayr. On the complexity of semantic equivalences for pushdown automata and BPA. In *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science (MFCS'02)*, volume 2420 of *LNCS*, pages 433–445. Springer-Verlag, 2002.
20. A. Murawski and I. Walukiewicz. Third-order idealized algol with iteration is decidable. In *Proceedings of the 8th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'05)*, volume 3441 of *LNCS*, pages 202–218, 2005.
21. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, December 1987.
22. C. Pitcher. Visibly pushdown expression effects for XML stream processing. In *Proceedings of Programming Language Technologies for XML (PLAN-X)*, pages 5–19, 2005.
23. Z. Sawa and P. Jančar. P-hardness of equivalence testing on finite-state processes. In *Proceedings of the 28th Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'01)*, volume 2234 of *LNCS*, pages 326–345. Springer-Verlag, 2001.
24. G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science(FOCS'98)*, pages 120–129. IEEE Computer Society, 1998.

25. O. Serre. Parity games played on transition graphs of one-counter processes. In *Proceedings of the 9th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'06)*, volume 3921 of *LNCS*, pages 337–351. Springer-Verlag, 2006.
26. J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP'02)*, volume 2380 of *LNCS*, pages 716–727. Springer-Verlag, 2002.
27. C. Stirling. Local model checking games. In *Proceedings of the 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *LNCS*, pages 1–11. Springer-Verlag, 1995.
28. W. Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science (extended abstract). In *Proceedings of the 4th International Joint Conference CAAP/FASE, Theory and Practice of Software Development (TAPSOFT'93)*, volume 668 of *LNCS*, pages 559–568. Springer-Verlag, 1993.
29. R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, CWI/Vrije Universiteit, 1990.
30. R.J. van Glabbeek. The linear time—branching time spectrum. In *Proceedings of the 1st Internatinal Conference on Theories of Concurrency: Unification and Extension (CONCUR'90)*, volume 458 of *LNCS*, pages 278–297. Springer-Verlag, 1990.
31. I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.

Recent BRICS Report Series Publications

RS-06-13 **Jiří Srba.** ***Visibly Pushdown Automata: From Language Equivalence to Simulation and Bisimulation*****. July 2006. To appear in CSL '06.**

RS-06-12 **Kristian Støvring.** ***Higher-Order Beta Matching with Solutions in Long Beta-Eta Normal Form*****. June 2006. 13 pp. To appear in** ***Nordic Journal of Computing*****, 2006.**

RS-06-11 **Kim G. Larsen, Ulrik Nyman, and Andrzej Wasowski.** ***An Interface Theory for Input/Output Automata*****. June 2006. 40 pp. Appears in Misra, Nipkow and Sekerinski, editors,** ***Formal Methods: 14th International Symposium*****, FM '06 Proceedings, LNCS 4085, 2006, pages 82–97.**

RS-06-10 **Christian Kirkegaard and Anders Møller.** ***Static Analysis for Java Servlets and JSP*****. June 2006. 23 pp. Full version of paper presented at SAS '06.**

RS-06-9 **Claus Brabrand, Robert Giegerich, and Anders Møller.** ***Analyzing Ambiguity of Context-Free Grammars*****. April 2006. 19 pp.**

RS-06-8 **Christian Kirkegaard and Anders Møller.** ***Static Analysis for Java Servlets and JSP*****. April 2006. 22 pp.**

RS-06-7 **Petr Jančar and Jiří Srba.** ***Undecidability Results for Bisimilarity on Prefix Rewrite Systems*****. April 2006. 20 pp. Presented at** ***FoSSaCS 2006*****, LNCS 3921:277–291.**

RS-06-6 **Luca Aceto, Willem Jan Fokkink, Anna Ingólfsdóttir, and Bas Luttik.** ***A Finite Equational Base for CCS with Left Merge and Communication Merge*****. March 2006. 22 pp.**

RS-06-5 **Kristian Støvring.** ***Extending the Extensional Lambda Calculus with Surjective Pairing is Conservative*****. March 2006. 18 pp. To appear in** ***Logical Methods in Computer Science*****. Supersedes RS-05-35.**

RS-06-4 **Olivier Danvy and Kevin Millikin.** ***A Rational Deconstruction of Landin's J Operator*****. February 2006. ii+26 pp. To appear in the post-reviewed proceedings of the 17th International Workshop on the** ***Implementation and Application of Functional Languages*** **(IFL'05), Dublin, Ireland, September 2005.**