Undecidability of Weak Bisimilarity for Pushdown Processes

Jiří Srba*

BRICS**
Department of Computer Science, University of Aarhus,
Ny Munkegade bld. 540, 8000 Aarhus C, Denmark
srba@brics.dk

Abstract. We prove undecidability of the problem whether a given pair of pushdown processes is weakly bisimilar. We also show that this undecidability result extends to a subclass of pushdown processes satisfying the normedness condition.

1 Introduction

An important question in the area of verification of infinite-state systems is that of *equivalence checking* [1]. A prominent role is played by the bisimulation equivalence [17] as it possesses many pleasant properties. *Strong bisimilarity* is decidable both for Basic Process Algebra (BPA) [3] and Basic Parallel Processes (BPP) [2], two basic models of purely sequential, respectively parallel, computations. There are even polynomial time algorithms for *normed* subclasses of BPA and BPP [7, 8] (a process is normed iff from every reachable state there is a computation leading to the empty process). This strongly contrasts with the fact that all other equivalences (including language equivalence) in van Glabbeek's spectrum (see [25, 26]) are undecidable for BPA [5] and BPP [9].

The answers to the strong bisimilarity problems for processes generated by *pushdown automata* (PDA) are even more involved than those for BPA and BPP. A pushdown automaton can be seen as a BPA process extended with a finite control unit. From the language point of view, there is no difference between PDA and BPA, since both formalisms describe the class of context-free languages. On the other hand the situation is different when considering strong bisimilarity as the equivalence relation. The PDA class is strictly more expressive than BPA w.r.t. strong (and weak) bisimilarity, and hence the decidability problems are more difficult to handle. Nevertheless, Stirling proved decidability of strong bisimilarity for normed PDA [22] and the same question for the whole class of PDA was positively answered by Senizergues [19].

Let us draw our attention to the notion of *weak bisimilarity*. Weak bisimilarity is a more general equivalence than strong bisimilarity, in the sense that it allows

* The author is supported in part by the GACR, grant No. 201/00/0400.

** **B**asic **R**esearch in **C**omputer **S**cience,
Centre of the Danish National Research Foundation.

to abstract from internal behaviour of processes by introducing a *silent action* τ, which is not observable [15]. Decidability of weak bisimilarity for BPA and BPP are well known open problems. The problems are open even for normed BPA and BPP. Some partial positive results were achieved e.g. in [6, 23]. It is also known that weak bisimilarity is decidable for *deterministic* PDA (follows from [18] as mentioned e.g. in [14]).

Our contribution is the undecidability of weak bisimilarity for PDA (and even for its normed subclass). To the best of our knowledge, this is the first undecidability result for weak bisimilarity on a class of infinite-state systems where strong bisimilarity remains decidable. Similar result is only known for bisimilarity between Petri nets and finite-state systems where strong bisimilarity is decidable [12] whereas weak bisimilarity is undecidable [11].

A full proof of our result is provided in Section 3. The technique is based on an effective encoding of the halting problem for 2-counter Minsky machine into the bisimilarity checking problem for a pair of pushdown processes. We use a game-theoretic characterization of weak bisimilarity to make the proof more understandable. In this setting two processes are weakly bisimilar if and only if a player called 'defender' has a winning strategy in the bisimulation game against a player called 'attacker'. The intuition of our encoding is that a configuration of a Minsky machine, consisting of an instruction label and the values of counters, is represented by a pair of pushdown processes. The label is remembered in the control state and the values of counters are stored in the stack. The problem is, of course, that we have only sequential access to the stack but we need to enable (at least a limited) parallel access to both counters. The key idea is a technique how to manage these stack contents in such a way that the players faithfully simulate the computation of the Minsky machine. The goal is to establish that the attacker has a winning strategy in the bisimulation game iff the machine halts, or equivalently that the defender has a winning strategy iff the machine diverges.

2 Basic Definitions

A *labelled transition system* is a triple $(S, \mathcal{A}ct, \longrightarrow)$ where S is a set of *states* (or *processes*), $\mathcal{A}ct$ is a set of *labels* (or *actions*), $\longrightarrow \subseteq S \times \mathcal{A}ct \times S$ is a *transition relation*, written $\alpha \stackrel{a}{\longrightarrow} \beta$, for $(\alpha, a, \beta) \in \longrightarrow$.

As usual we extend the transition relation to the elements of $\mathcal{A}ct^*$, i.e., $\alpha \stackrel{\epsilon}{\longrightarrow} \alpha$ for every $\alpha \in S$ and $\alpha \stackrel{aw}{\longrightarrow} \beta$ if $\alpha \stackrel{a}{\longrightarrow} \alpha'$ and $\alpha' \stackrel{w}{\longrightarrow} \beta$ for every $\alpha, \beta \in S$, $a \in \mathcal{A}ct$ and $w \in \mathcal{A}ct^*$. We also write $\alpha \longrightarrow^* \beta$ whenever $\alpha \stackrel{w}{\longrightarrow} \beta$ for some $w \in \mathcal{A}ct^*$, $\alpha \stackrel{a}{\not\longrightarrow}$ whenever there is no β such that $\alpha \stackrel{a}{\longrightarrow} \beta$, and $\alpha \not\longrightarrow$ whenever $\alpha \stackrel{a}{\not\longrightarrow}$ for all $a \in \mathcal{A}ct$.

A *process* is a pair (α, T) where $T = (S, \mathcal{A}ct, \longrightarrow)$ is a labelled transition system and $\alpha \in S$. We say that $\beta \in S$ is *reachable in* (α, T) iff $\alpha \longrightarrow^* \beta$. We call (α, T) a *finite-state process* iff the set of its reachable states is finite.

Assume that the set of actions $\mathcal{A}ct$ contains a distinguished *silent action* τ. The notation $\alpha \stackrel{\tau^*}{\longrightarrow} \beta$ means that there is an integer $n \geq 0$ such that $\alpha \stackrel{\tau^n}{\longrightarrow} \beta$,

where τ^n is a word consisting of n occurrences of τ. The *weak transition relation* $\Longrightarrow$ is defined as follows:

$$\overset{a}{\Longrightarrow} \overset{\text{def}}{=} \begin{cases} \xrightarrow{\tau^*} \circ \xrightarrow{a} \circ \xrightarrow{\tau^*} & \text{if } a \in \mathcal{A}ct \setminus \{\tau\} \\ \xrightarrow{\tau^*} & \text{if } a = \tau. \end{cases}$$

Let $T = (S, \mathcal{A}ct, \longrightarrow)$ be a labelled transition system. A binary relation $R \subseteq S \times S$ is a *weak bisimulation* iff whenever $(\alpha, \beta) \in R$ then for each $a \in \mathcal{A}ct$:

- if $\alpha \xrightarrow{a} \alpha'$ then $\beta \overset{a}{\Longrightarrow} \beta'$ for some β' such that $(\alpha', \beta') \in R$
- if $\beta \xrightarrow{a} \beta'$ then $\alpha \overset{a}{\Longrightarrow} \alpha'$ for some α' such that $(\alpha', \beta') \in R$.

Processes (α_1, T) and (α_2, T) are *weakly bisimilar*, and we write $(\alpha_1, T) \approx (\alpha_2, T)$ (or simply $\alpha_1 \approx \alpha_2$ if T is clear from the context), iff there is a weak bisimulation R such that $(\alpha_1, \alpha_2) \in R$. Given a pair of processes (α_1, T_1) and (α_2, T_2) such that T_1 and T_2 are different labelled transition systems, we write $(\alpha_1, T_1) \approx (\alpha_2, T_2)$ iff $(\alpha_1, T) \approx (\alpha_2, T)$ where T is the disjoint union of T_1 and T_2.

Remark 1. If we assume that τ does not appear in the set of actions $\mathcal{A}ct$ then the relations $\Longrightarrow$ and $\longrightarrow$ coincide. We call the corresponding notion of bisimilarity *strong bisimilarity* and denote it by $\sim$.

Weak bisimilarity has an elegant characterisation in terms of *bisimulation games*.

Definition 1 (Bisimulation game).
A bisimulation game *on a pair of processes* (α_1, T) *and* (α_2, T) *where* $T = (S, \mathcal{A}ct, \longrightarrow)$ *is a two-player game between an 'attacker' and a 'defender'. The game is played in* rounds *on pairs of states from* $S \times S$. *In each round the players change the* current states β_1 *and* β_2 *(initially* α_1 *and* α_2*) according to the following rule.*

1. *The attacker chooses an* $i \in \{1, 2\}$, $a \in \mathcal{A}ct$ *and* $\beta_i' \in S$ *such that* $\beta_i \xrightarrow{a} \beta_i'$.
2. *The defender responds by choosing a* $\beta_{3-i}' \in S$ *such that* $\beta_{3-i} \overset{a}{\Longrightarrow} \beta_{3-i}'$.
3. *The states* β_1' *and* β_2' *become the current states.*

A play *is a maximal sequence of pairs of states formed by the players according to the rule described above, and starting from the initial states* α_1 *and* α_2. *The defender is the winner in every infinite play. A finite play is lost by the player who is stuck. Note that the attacker gets stuck in current states* β_1 *and* β_2 *if and only if both* $\beta_1 \not\longrightarrow$ *and* $\beta_2 \not\longrightarrow$.

We remind the reader of the fact that if the attacker chooses a move under the action τ in one of the processes, the defender can (as one possibility) simply answer by doing "nothing", i.e., by staying in the same state of the other process. The following proposition is a standard one (see e.g. [21, 24]).

Proposition 1. *Processes (α_1, T) and (α_2, T) are weakly bisimilar iff the defender has a winning strategy (and nonbisimilar iff the attacker has a winning strategy).*

Let $Q = \{p, q, \ldots\}$, $\Gamma = \{X, Y, \ldots\}$ and $\mathcal{A}ct = \{a, b, \ldots\}$ be finite sets of *control states*, *stack symbols* and *actions*, respectively, such that $Q \cap \Gamma = \emptyset$ and $\tau \in \mathcal{A}ct$ is the distinguished *silent action*. A *pushdown automaton* (PDA) is a finite set

$$\Delta \subseteq Q \times \Gamma \times \mathcal{A}ct \times Q \times \Gamma^*$$

of *rewrite rules*, written $pX \stackrel{a}{\longrightarrow} q\alpha$ for $(p, X, a, q, \alpha) \in \Delta$. A pushdown automaton Δ generates a labelled transition system $T(\Delta) = (Q \times \Gamma^*, \mathcal{A}ct, \longrightarrow)$ where $Q \times \Gamma^*$ is the set of states[1], $\mathcal{A}ct$ is the set of actions, and the transition relation $\longrightarrow$ is defined by

$$pX\beta \stackrel{a}{\longrightarrow} q\alpha\beta \quad \text{iff} \quad (pX \stackrel{a}{\longrightarrow} q\alpha) \in \Delta$$

for all $\beta \in \Gamma^*$.

A *pushdown process* (or simply a *process*) is a pair $\big(p\alpha, T(\Delta)\big)$ where $T(\Delta)$ is the transition system generated by a pushdown automaton Δ and $p\alpha$ is a state of $T(\Delta)$. We often abbreviate the notation $\big(p\alpha, T(\Delta)\big)$ to $(p\alpha, \Delta)$ or even to $p\alpha$ if Δ is clear from the context.

A process $(p\alpha, \Delta)$ is *normed* iff for every reachable state $q\beta$ there is a finite computation which empties the stack, i.e., there is a state $r\epsilon \in Q \times \Gamma^*$ such that $q\beta \longrightarrow^* r\epsilon$. We say that $(p\alpha, \Delta)$ is *weakly (or strongly) regular* iff there is some finite-state process (γ, T) such that $(p\alpha, \Delta) \approx (\gamma, T)$ (or $(p\alpha, \Delta) \sim (\gamma, T)$).

Notation 1. Let i be a natural number and $A \in \Gamma$. We use the notation A^i for a sequence of i occurrences of A, i.e., $A^0 \stackrel{\text{def}}{=} \epsilon$ and $A^{i+1} \stackrel{\text{def}}{=} A^i A$. For example pX^2Y^3 is an abbreviation for $pXXYYY$.

Example 1. Let $Q \stackrel{\text{def}}{=} \{p, p_1, p', p'_1, p'_2, p'_3\}$, $\Gamma \stackrel{\text{def}}{=} \{X, Y\}$ and $\mathcal{A}ct \stackrel{\text{def}}{=} \{a, b, c, \tau\}$ and let Δ be the following pushdown automaton.

$$\begin{array}{lll}
pX \stackrel{a}{\longrightarrow} p_1X & pX \stackrel{\tau}{\longrightarrow} p'X & p_1X \stackrel{c}{\longrightarrow} p_1X \\
\\
p'X \stackrel{a}{\longrightarrow} p'_1X & p'_1X \stackrel{c}{\longrightarrow} p'_1X & \\
p'X \stackrel{\tau}{\longrightarrow} p'_2X & p'_2X \stackrel{c}{\longrightarrow} p'_3 & \\
p'_2X \stackrel{\tau}{\longrightarrow} p'_2YX & p'_2Y \stackrel{\tau}{\longrightarrow} p'_2YY & p'_2Y \stackrel{b}{\longrightarrow} p'_2
\end{array}$$

A fraction of the transition system $T(\Delta)$ is depicted in Figure 1. Let us consider processes (pX, Δ) and $(p'X, \Delta)$. We show that $(pX, \Delta) \approx (p'X, \Delta)$ by describing a winning strategy for the defender in the bisimulation game starting from pX and $p'X$. The attacker has the following four possibilities in the first round: $pX \stackrel{a}{\longrightarrow} p_1X$, or $pX \stackrel{\tau}{\longrightarrow} p'X$, or $p'X \stackrel{a}{\longrightarrow} p'_1X$, or $p'X \stackrel{\tau}{\longrightarrow} p'_2X$. In order to

[1] We write $p\alpha$ instead of $(p, \alpha) \in Q \times \Gamma^*$. A state $p\epsilon \in Q \times \Gamma^*$, where ϵ is the symbol for *empty stack*, is usually written only as p.

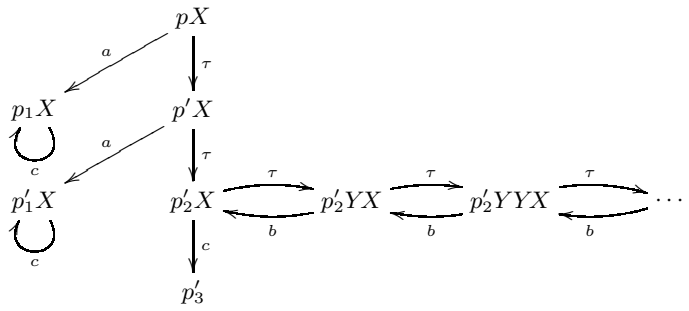

Fig. 1. Transition system generated by Δ

avoid the situation where the defender can reach a pair of syntactically equal states, the attacker is forced to choose the first move, namely $pX \xrightarrow{a} p_1X$.

The defender answers by playing $p'X \stackrel{a}{\Longrightarrow} p_1'X$. Now the game continues from p_1X and $p_1'X$, however, these two states are obviously weakly bisimilar and hence the defender has a winning strategy. This implies that $(pX, \Delta) \approx (p'X, \Delta)$.

In this example we also recall some of our previous definitions. The process (pX, Δ) can terminate (empty its stack) since $pX \longrightarrow^* p_3'$ but it is not normed — it can reach e.g. the state p_1X from which there is no terminating computation. On the other hand the process $(p_2'Y^iX, \Delta)$ is normed for any $i \geq 0$.

Also note that (pX, Δ) is not weakly regular because it has infinitely many reachable and weakly nonbisimilar states. Consider the states $p_2'Y^iX$ and $p_2'Y^jX$ for $i \neq j$. Obviously, $pX \longrightarrow^* p_2'Y^iX$ and $pX \longrightarrow^* p_2'Y^jX$. We leave it to the reader to find a winning strategy for the attacker from the pair $p_2'Y^iX$ and $p_2'Y^jX$ and thus show that $(p_2'Y^iX, \Delta) \not\approx (p_2'Y^jX, \Delta)$.

3 Undecidability of Weak Bisimilarity

We shall prove that weak bisimilarity of pushdown processes is undecidable.

Problem:	Weak bisimilarity of pushdown processes
Instance:	A pushdown automaton Δ and a pair of processes $(p_1\alpha_1, \Delta)$ and $(p_2\alpha_2, \Delta)$.
Question:	$(p_1\alpha_1, \Delta) \approx (p_2\alpha_2, \Delta)$?

The proof is by reduction from the halting problem of Minsky machine [16] with two counters.

Definition 2 (Minsky machine with two counters).
A Minsky machine R *with two counters* c_1 *and* c_2 *is a finite sequence*

$$R = (L_1 : I_1, \quad L_2 : I_2, \quad \ldots, \quad L_{n-1} : I_{n-1}, \quad L_n : \texttt{halt})$$

where $n \geq 1$, $L_1, \ldots, L_n$ *are pairwise different* labels, *and* $I_1, \ldots, I_{n-1}$ *are* instructions *of the following two types:*

– *increment:* $c_r := c_r + 1$; `goto` L_j
– *test and decrement:* `if` $c_r = 0$ `then goto` L_j `else` $c_r := c_r - 1$; `goto` L_k

where $1 \leq r \leq 2$ *and* $1 \leq j, k \leq n$.

A *configuration* of a Minsky machine R is a triple (L_i, v_1, v_2) where L_i is the instruction label ($1 \leq i \leq n$), and $v_1, v_2 \in \mathbb{N}$ are nonnegative integers representing the values of counters c_1 and c_2, respectively. Let $\mathcal{C}onf$ be the set of all configurations of R. The transition relation $\hookrightarrow \subseteq \mathcal{C}onf \times \mathcal{C}onf$ between configurations is defined in the obvious and natural way. We remind the reader of the fact that the computation of the machine R is deterministic, i.e., if $c \hookrightarrow d$ and $c \hookrightarrow e$ then $d = e$ for all $c, d, e \in \mathcal{C}onf$.

It is a well known fact that the problem whether a Minsky machine R *halts* with the initial counter values set to zero (in other words the problem whether $(L_1, 0, 0) \hookrightarrow^* (L_n, v_1, v_2)$ for some $v_1, v_2 \in \mathbb{N}$) is undecidable [16]. If R does not halt we say that it *diverges*.

Our aim is to show that there is an effective construction such that given a Minsky machine R it defines a pushdown automaton Δ and a pair of processes $p_1\alpha_1$ and $p_2\alpha_2$ with the property that R halts if and only if $(p_1\alpha_1, \Delta) \not\approx (p_2\alpha_2, \Delta)$. This proves that weak bisimilarity of pushdown processes is an undecidable problem.

Let us fix a Minsky machine

$$R = (L_1 : I_1, \quad L_2 : I_2, \quad \ldots, \quad L_{n-1} : I_{n-1}, \quad L_n : \texttt{halt}).$$

We construct Δ in stages. First, we define the sets of control states, stack symbols and actions. Let $\mathcal{I}nc \stackrel{\text{def}}{=} \{i \mid 1 \leq i < n$ and I_i is of the type 'increment'$\}$ and $\mathcal{D}ec \stackrel{\text{def}}{=} \{i \mid 1 \leq i < n$ and I_i is of the type 'test and decrement'$\}$.

$$Q \stackrel{\text{def}}{=} \{\mathsf{equal}, \mathsf{equal}_1, \mathsf{equal}_2, \mathsf{empty}_1, \mathsf{empty}_2, \mathsf{empty}'_1, \mathsf{empty}'_2\} \cup \bigcup_{i \in \mathcal{I}nc} \{p_i, p'_i\} \cup \bigcup_{i \in \mathcal{D}ec} \{p_i, p'_i, u_i, u'_i, q_i, q'_i, t_i, t'_i\} \cup \{p_n, p'_n\}$$

$$\Gamma \stackrel{\text{def}}{=} \{C_1, C_2, S\}$$

$$\mathcal{A}ct \stackrel{\text{def}}{=} \{a, b, c, d, e, c_1, c_2, c'_1, c'_2, \mathsf{halt}, \tau\}$$

The intuition is that a configuration $(L_i, v_1, v_2) \in \mathcal{C}onf$ is represented by a pair of processes $p_i\gamma S$ and $p'_i\gamma' S$ where $\gamma, \gamma' \in \{C_1, C_2\}^*$ such that the number of occurrences of C_1 and C_2 in γ (and also in γ') is equal to v_1 and v_2, respectively. Using this representation, our task is now to design rewrite rules to simulate step by step the computation of R. Let us define formally a mapping $\mathit{value} : \{C_1, C_2\}^* \mapsto \mathbb{N} \times \mathbb{N}$ by the following inductive definition (the operation of addition is component-wise).

$$\begin{aligned} \mathit{value}(\epsilon) &\stackrel{\text{def}}{=} (0,0) \\ \mathit{value}(C_1\gamma) &\stackrel{\text{def}}{=} \mathit{value}(\gamma) + (1,0) && \text{for all } \gamma \in \{C_1, C_2\}^* \\ \mathit{value}(C_2\gamma) &\stackrel{\text{def}}{=} \mathit{value}(\gamma) + (0,1) && \text{for all } \gamma \in \{C_1, C_2\}^* \end{aligned}$$

As a part of the bisimulation game we will find useful the following rewrite rules which enable to check whether two given stacks contain the same number of occurrences of C_1 and C_2. In the rules below X ranges over the set $\{C_1, C_2, S\}$.

$$\mathsf{equal}\ X \xrightarrow{a} \mathsf{equal}_1\ X \qquad \mathsf{equal}\ X \xrightarrow{b} \mathsf{equal}_2\ X$$

$$\begin{array}{ll} \mathsf{equal}_1\ C_1 \xrightarrow{c_1} \mathsf{equal}_1 & \mathsf{equal}_1\ C_2 \xrightarrow{\tau} \mathsf{equal}_1 \\ \mathsf{equal}_2\ C_2 \xrightarrow{c_2} \mathsf{equal}_2 & \mathsf{equal}_2\ C_1 \xrightarrow{\tau} \mathsf{equal}_2 \end{array}$$

Proposition 2. *Let $\gamma, \gamma' \in \{C_1, C_2\}^*$. Then*

$$\mathsf{equal}\ \gamma S \approx \mathsf{equal}\ \gamma' S \quad \textit{iff} \quad \textit{value}(\gamma) = \textit{value}(\gamma').$$

We continue by defining further rewrite rules to check whether the number of occurences of C_1 (or C_2) is zero.

$$\begin{array}{ll} \mathsf{empty}_1\ C_1 \xrightarrow{c_1} \mathsf{empty}_1 & \mathsf{empty}_1\ C_2 \xrightarrow{c_2} \mathsf{empty}_1 \\ \mathsf{empty}'_1\ C_1 \xrightarrow{c'_1} \mathsf{empty}'_1 & \mathsf{empty}'_1\ C_2 \xrightarrow{c_2} \mathsf{empty}'_1 \\ \\ \mathsf{empty}_2\ C_1 \xrightarrow{c_1} \mathsf{empty}_2 & \mathsf{empty}_2\ C_2 \xrightarrow{c_2} \mathsf{empty}_2 \\ \mathsf{empty}'_2\ C_1 \xrightarrow{c_1} \mathsf{empty}'_2 & \mathsf{empty}'_2\ C_2 \xrightarrow{c'_2} \mathsf{empty}'_2 \end{array}$$

Proposition 3. *Let $\gamma, \gamma' \in \{C_1, C_2\}^*$ be such that $\textit{value}(\gamma) = \textit{value}(\gamma') = (v_1, v_2)$ for some $v_1, v_2 \in \mathbb{N}$. Let $r \in \{1, 2\}$. Then*

$$\mathsf{empty}_r\ \gamma S \approx \mathsf{empty}'_r\ \gamma' S \quad \textit{iff} \quad v_r = 0.$$

Let us now define the rewrite rules that are connected with the increment instructions of R. Assume again that X ranges over the set $\{C_1, C_2, S\}$. For all $i \in \mathcal{I}nc$ such that I_i is of the type

$$L_i\texttt{:}\ c_r := c_r + 1\texttt{;}\ \texttt{goto}\ L_j$$

where $1 \leq j \leq n$ and $1 \leq r \leq 2$, we add the following two rules.

$$p_i X \xrightarrow{a} p_j C_r X \qquad p'_i X \xrightarrow{a} p'_j C_r X$$

Lemma 1. *Let $(L_i, v_1, v_2) \in \mathcal{C}onf$ be such that I_i is the 'increment' instruction and $(L_i, v_1, v_2) \hookrightarrow (L_j, v'_1, v'_2)$. Let $\gamma, \gamma' \in \{C_1, C_2\}^*$ be such that $\textit{value}(\gamma) = \textit{value}(\gamma') = (v_1, v_2)$. There is a unique continuation of the bisimulation game from the pair $p_i \gamma S$ and $p'_i \gamma' S$ such that after one round the players reach the pair $p_j \overline{\gamma} S$ and $p'_j \overline{\gamma'} S$ satisfying $\textit{value}(\overline{\gamma}) = \textit{value}(\overline{\gamma'}) = (v'_1, v'_2)$.*

Proof. Obvious — see Figure 2. □

Fig. 2. Instruction L_i: $c_r := c_r + 1$; goto L_j

We proceed by giving the rules for the 'test and decrement' instructions. For all $i \in \mathcal{D}ec$ such that I_i is of the type

$$L_i\texttt{: if } c_r = 0 \texttt{ then goto } L_j \texttt{ else } c_r := c_r - 1\texttt{; goto } L_k$$

where $1 \leq j, k \leq n$ and $1 \leq r \leq 2$, we define the rewrite rules in three parts. The intuitive meaning is that if $v_r \neq 0$ and the stacks γS and $\gamma' S$ contain on their tops the symbol C_r, we can do immediately the branching according to the rules defined later in the third part. However, if it is not the case, the first two parts of the rewrite rules enable the defender to rearrange the stack contents (while preserving the number of occurrences of C_1 and C_2) in such a way that C_r will appear as the first symbol on the stacks. Recall that X ranges over the set $\{C_1, C_2, S\}$.

$$p_iX \xrightarrow{a} q_iX \qquad p_iX \xrightarrow{a} u'_iX$$

$$\begin{array}{lll} p'_iX \xrightarrow{a} u'_iX & u'_iX \xrightarrow{\tau} q'_iX & u'_iX \xrightarrow{e} u'_iX \\ u'_iC_1 \xrightarrow{\tau} u'_i & u'_iC_2 \xrightarrow{\tau} u'_i & \\ u'_iX \xrightarrow{\tau} u'_iC_1X & u'_iX \xrightarrow{\tau} u'_iC_2X & \end{array}$$

$$q_iX \xrightarrow{c} \mathsf{equal}\ X \qquad q'_iX \xrightarrow{c} \mathsf{equal}\ X$$

Assume a bisimulation game played from $p_i\gamma S$ and $p'_i\gamma' S$. The purpose of the previously defined rules is to enable the defender to rearrange the sequence of C_1 and C_2 in γ'. Details are discussed in the proof of Lemma 2, here we give only a short description. If the attacker plays $p_i\gamma S \xrightarrow{a} q_i\gamma S$, the defender must answer by $p'_i\gamma' S \stackrel{a}{\Longrightarrow} q'_i\overline{\gamma'}S$ for some $\overline{\gamma'} \in \{C_1, C_2\}^*$. Now the attacker can check the invariant that $\mathit{value}(\gamma) = \mathit{value}(\overline{\gamma'})$ by using the rules $q_iX \xrightarrow{c} \mathsf{equal}\ X$ and $q'_iX \xrightarrow{c} \mathsf{equal}\ X$.

$$q'_iX \xrightarrow{a} t'_iX \qquad q'_iX \xrightarrow{a} u_iX$$

$$\begin{array}{lll} q_iX \xrightarrow{a} u_iX & u_iX \xrightarrow{\tau} t_iX & u_iX \xrightarrow{e} u_iX \\ u_iC_1 \xrightarrow{\tau} u_i & u_iC_2 \xrightarrow{\tau} u_i & \\ u_iX \xrightarrow{\tau} u_iC_1X & u_iX \xrightarrow{\tau} u_iC_2X & \end{array}$$

$$t_iX \xrightarrow{c} \mathsf{equal}\ X \qquad t'_iX \xrightarrow{c} \mathsf{equal}\ X$$

These rules are completely symmetric to the previous ones. In the bisimulation game starting from $q_i\gamma S$ and $q'_i\overline{\gamma'}S$, if the attacker plays $q'_i\overline{\gamma'}S \xrightarrow{a} t'_i\overline{\gamma'}S$, the defender must choose some $\overline{\gamma} \in \{C_1, C_2\}^*$ and play $q_i\gamma S \stackrel{a}{\Longrightarrow} t_i\overline{\gamma}S$. The attacker can again check whether $value(\overline{\gamma}) = value(\overline{\gamma'})$. The current states become $t_i\overline{\gamma}S$ and $t'_i\overline{\gamma'}S$ satisfying $value(\gamma) = value(\gamma') = value(\overline{\gamma}) = value(\overline{\gamma'})$.

The third part of the rewrite rules defined below is here to perform a branching according to whether C_r occurs in γ and γ' or not. The correctness is discussed later.

$$t_iC_r \xrightarrow{a} p_k \qquad\qquad t'_iC_r \xrightarrow{a} p'_k$$

$$t_iC_{3-r} \xrightarrow{b} p_jC_{3-r} \qquad\qquad t'_iC_{3-r} \xrightarrow{b} p'_jC_{3-r}$$
$$t_iS \xrightarrow{b} p_jS \qquad\qquad t'_iS \xrightarrow{b} p'_jS$$

$$t_iC_{3-r} \xrightarrow{d} \mathsf{empty}_r\ C_{3-r} \qquad\qquad t'_iC_{3-r} \xrightarrow{d} \mathsf{empty}'_r\ C_{3-r}$$

Finally, we add one extra rule to distinguish whether the last instruction `halt` was reached. Recall that X ranges over the set $\{C_1, C_2, S\}$.

$$p_nX \xrightarrow{\mathsf{halt}} p_nX$$

Lemma 2. *Let $(L_i, v_1, v_2) \in \mathcal{C}onf$ be such that I_i is the 'test and decrement' instruction*

$$L_i\texttt{: if } c_r = 0 \texttt{ then goto } L_j \texttt{ else } c_r := c_r - 1\texttt{; goto } L_k$$

and let $\gamma, \gamma' \in \{C_1, C_2\}^*$ be such that $value(\gamma) = value(\gamma') = (v_1, v_2)$. Assume a bisimulation game played from the pair

$$p_i\gamma S \quad and \quad p'_i\gamma' S.$$

a) The attacker has a strategy such that he either wins, or after three rounds the players reach the states
   1. *$p_k\overline{\gamma}S$ and $p'_k\overline{\gamma'}S$ — if $v_r \neq 0$ and $(L_i, v_1, v_2) \hookrightarrow (L_k, v'_1, v'_2)$ — where $value(\overline{\gamma}) = value(\overline{\gamma'}) = (v'_1, v'_2)$, or*
   2. *$p_j\overline{\gamma}S$ and $p'_j\overline{\gamma'}S$ — if $v_r = 0$ and $(L_i, v_1, v_2) \hookrightarrow (L_j, v_1, v_2)$ — where $value(\overline{\gamma}) = value(\overline{\gamma'}) = (v_1, v_2)$.*

b) The defender has a strategy such that he either wins, or after three rounds the players reach the states
   1. *$p_k\overline{\gamma}S$ and $p'_k\overline{\gamma'}S$ — if $v_r \neq 0$ and $(L_i, v_1, v_2) \hookrightarrow (L_k, v'_1, v'_2)$ — where $value(\overline{\gamma}) = value(\overline{\gamma'}) = (v'_1, v'_2)$, or*
   2. *$p_j\overline{\gamma}S$ and $p'_j\overline{\gamma'}S$ — if $v_r = 0$ and $(L_i, v_1, v_2) \hookrightarrow (L_j, v_1, v_2)$ — where $value(\overline{\gamma}) = value(\overline{\gamma'}) = (v_1, v_2)$.*

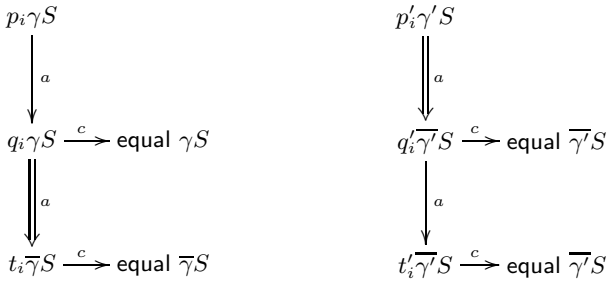

Fig. 3. Instruction 'test and decrement' — first two rounds

Proof. We begin with part a). First two rounds of the bisimulation game are depicted in Figure 3. The game starts from $p_i\gamma S$ and $p'_i\gamma' S$ such that $\mathit{value}(\gamma) = \mathit{value}(\gamma') = (v_1, v_2)$. We show that after two attacker's moves the players either reach a pair $t_i\overline{\gamma}S$ and $t'_i\overline{\gamma'}S$ such that $\overline{\gamma}, \overline{\gamma'} \in \{C_1, C_2\}^*$ and $\mathit{value}(\overline{\gamma}) = \mathit{value}(\overline{\gamma'}) = (v_1, v_2)$, or the attacker has an immediate winning strategy. The attacker starts by playing $p_i\gamma S \xrightarrow{a} q_i\gamma S$. The defender must respond by playing $p'_i\gamma' S \stackrel{a}{\Longrightarrow} q'_i\overline{\gamma'}S$ for some $\overline{\gamma'} \in \{C_1, C_2\}^*$ because of the following remark.

Remark 2. The defender's $\stackrel{a}{\Longrightarrow}$-answer must start with the transition $p'_i\gamma' S \xrightarrow{a} u'_i\gamma' S$, followed by a finite number of τ-labelled transitions using the rules that enable to remove an arbitrary part of the stack $\gamma' S$ and add an arbitrary sequence from the symbols C_1 and C_2. Thus the defender can reach the state $u'_i\overline{\gamma'}S$ for any sequence $\overline{\gamma'} \in \{C_1, C_2\}^*$. Also note that he must finish the sequence of τ-moves by $u'_i\overline{\gamma'}S \xrightarrow{\tau} q'_i\overline{\gamma'}S$. If not, then the attacker has an immediate winning move in the next round by playing $u'_i\overline{\gamma'}S \xrightarrow{e} u'_i\overline{\gamma'}S$ to which the defender has no answer because there is no $\stackrel{e}{\Longrightarrow}$-move from $q_i\gamma S$.

The bisimulation game continues from the states $q_i\gamma S$ and $q'_i\overline{\gamma'}S$. Whenever $\mathit{value}(\gamma) \neq \mathit{value}(\overline{\gamma'})$ then the attacker plays $q_i\gamma S \xrightarrow{c} \mathsf{equal}\ \gamma S$ to which the defender has only one possible answer $q'_i\overline{\gamma'}S \xrightarrow{c} \mathsf{equal}\ \overline{\gamma'}S$. Now the attacker has a winning strategy because of Proposition 2.

Let us so assume that $\mathit{value}(\gamma) = \mathit{value}(\overline{\gamma'})$. In the second round the attacker switches the states and performs the move $q'_i\overline{\gamma'}S \xrightarrow{a} t'_i\overline{\gamma'}S$. The game is now completely symmetric to the situation in the first round. The defender must answer with $q_i\gamma S \stackrel{a}{\Longrightarrow} t_i\overline{\gamma}S$ for some $\overline{\gamma} \in \{C_1, C_2\}^*$ such that $\mathit{value}(\overline{\gamma}) = \mathit{value}(\overline{\gamma'}) = (v_1, v_2)$.

In the third round played from $t_i\overline{\gamma}S$ and $t'_i\overline{\gamma'}S$ the attacker's strategy splits into two parts, according to whether $v_r \neq 0$ or $v_r = 0$.

1. Let $v_r \neq 0$ and hence $(L_i, v_1, v_2) \hookrightarrow (L_k, v'_1, v'_2)$. See Figure 4.
   - If $\overline{\gamma} = C_r\overline{\overline{\gamma}}$ for some $\overline{\overline{\gamma}}$ then the attacker plays $t_i\overline{\gamma}S \xrightarrow{a} p_k\overline{\overline{\gamma}}S$ and the defender must answer by $t'_i\overline{\gamma'}S \xrightarrow{a} p'_k\overline{\overline{\gamma'}}S$ where $\overline{\gamma'} = C_r\overline{\overline{\gamma'}}$. (If $\overline{\gamma'} = C_{3-r}\overline{\overline{\gamma'}}$ then the attacker wins immediately since $t'_i\overline{\gamma'}S$ cannot perform

Fig. 4. Case $v_r \neq 0$, i.e., $(L_i, v_1, v_2) \hookrightarrow (L_k, v'_1, v'_2)$

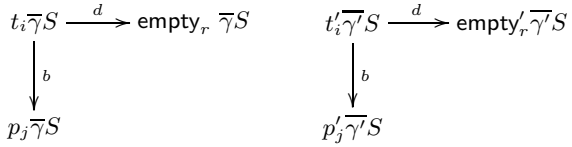

Fig. 5. Case $v_r = 0$, i.e., $(L_i, v_1, v_2) \hookrightarrow (L_j, v_1, v_2)$

any $\stackrel{a}{\Longrightarrow}$-move.) Now the players reached the pair $p_k \overline{\overline{\gamma}} S$ and $p'_k \overline{\overline{\gamma'}} S$ as required. Obviously $\mathit{value}(\overline{\overline{\gamma}}) = \mathit{value}(\overline{\overline{\gamma'}}) = (v'_1, v'_2)$.

- If $\overline{\gamma} = C_{3-r} \overline{\overline{\gamma}}$ for some $\overline{\overline{\gamma}}$ then the attacker plays $t_i \overline{\gamma} S \xrightarrow{d} \mathsf{empty}_r\ \overline{\gamma} S$ to which the defender has only one possible answer (if any), namely $t'_i \overline{\gamma'} S \xrightarrow{d} \mathsf{empty}'_r\ \overline{\gamma'} S$. Since $\mathit{value}(\overline{\gamma}) = \mathit{value}(\overline{\gamma'}) = (v_1, v_2)$ and $v_r \neq 0$, the attacker has a winning strategy because of Proposition 3.
- The case $\overline{\gamma} = \epsilon$ is impossible since we assume that $v_r \neq 0$.

2. Let $v_r = 0$ and hence $(L_i, v_1, v_2) \hookrightarrow (L_j, v_1, v_2)$. See Figure 5. The assumption $v_r = 0$ implies that $\overline{\gamma}, \overline{\gamma'} \in \{C_{3-r}\}^*$. Hence the attacker can play $t_i \overline{\gamma} S \xrightarrow{b} p_j \overline{\gamma} S$ and the defender has only one answer $t'_i \overline{\gamma'} S \xrightarrow{b} p'_j \overline{\gamma'} S$. The players reached the pair $p_j \overline{\gamma} S$ and $p'_j \overline{\gamma'} S$ as required. Recall that $\mathit{value}(\overline{\gamma}) = \mathit{value}(\overline{\gamma'}) = (v_1, v_2)$.

Let us now prove part b). First two rounds can be seen again in Figure 3. The initial states are $p_i \gamma S$ and $p'_i \gamma' S$ such that $\mathit{value}(\gamma) = \mathit{value}(\gamma') = (v_1, v_2)$. We claim that the defender has a strategy such that he either wins, or after two rounds the players reach the states $t_i C_r^{v_r} C_{3-r}^{v_{3-r}} S$ and $t'_i C_r^{v_r} C_{3-r}^{v_{3-r}} S$ (for definitions of $C_r^{v_r}$ and $C_{3-r}^{v_{3-r}}$ see Notation 1). In the first round the attacker has three possible moves: (i) $p_i \gamma S \xrightarrow{a} q_i \gamma S$, (ii) $p_i \gamma S \xrightarrow{a} u'_i \gamma S$ or (iii) $p'_i \gamma' S \xrightarrow{a} u'_i \gamma' S$. The moves (ii) and (iii) are good for the defender since he can immediately win by playing (ii) $p'_i \gamma' S \stackrel{a}{\Longrightarrow} u'_i \gamma S$ or (iii) $p_i \gamma S \stackrel{a}{\Longrightarrow} u'_i \gamma' S$. Obviously, two syntactically equal states are also weakly bisimilar. Hence we can assume that the attacker's first move is $p_i \gamma S \xrightarrow{a} q_i \gamma S$. The defender answers by $p'_i \gamma' S \stackrel{a}{\Longrightarrow} q'_i C_r^{v_r} C_{3-r}^{v_{3-r}} S$. Recall that $\mathit{value}(\gamma) = (v_1, v_2)$ and thus the attacker loses by taking (i) $q_i \gamma S \xrightarrow{c} \mathsf{equal}\ \gamma S$ or (ii) $q'_i C_r^{v_r} C_{3-r}^{v_{3-r}} S \xrightarrow{c} \mathsf{equal}\ C_r^{v_r} C_{3-r}^{v_{3-r}} S$ as his next move since the defender can respond by playing (i) $q'_i C_r^{v_r} C_{3-r}^{v_{3-r}} S \xrightarrow{c} \mathsf{equal}\ C_r^{v_r} C_{3-r}^{v_{3-r}} S$ or (ii)

$q_i\gamma S \xrightarrow{c} \mathsf{equal}\ \gamma S$. The pair of states $\mathsf{equal}\ \gamma S$ and $\mathsf{equal}\ C_r^{v_r}C_{3-r}^{v_{3-r}}S$ is weakly bisimilar because of Proposition 2 and the defender has a winning strategy.

From the pair $q_i\gamma S$ and $q_i'C_r^{v_r}C_{3-r}^{v_{3-r}}S$ we have a symmetric situation to the previous one. So after the second round either the defender can win, or he can force the attacker to reach the states $t_iC_r^{v_r}C_{3-r}^{v_{3-r}}S$ and $t_i'C_r^{v_r}C_{3-r}^{v_{3-r}}S$. Now the game splits into two parts according to whether $v_r \neq 0$ or $v_r = 0$.

1. Let $v_r \neq 0$ and hence $(L_i, v_1, v_2) \hookrightarrow (L_k, v_1', v_2')$. See Figure 4. Then there is a unique continuation of the game reaching the states $p_kC_r^{v_r-1}C_{3-r}^{v_{3-r}}S$ and $p_k'C_r^{v_r-1}C_{3-r}^{v_{3-r}}S$. Obviously $\mathit{value}(C_r^{v_r-1}C_{3-r}^{v_{3-r}}) = (v_1', v_2')$.
2. Let $v_r = 0$ and hence $(L_i, v_1, v_2) \hookrightarrow (L_j, v_1, v_2)$. See Figure 5. Consider the game starting from $t_iC_{3-r}^{v_{3-r}}S$ and $t_i'C_{3-r}^{v_{3-r}}S$ (note that $C_r^{v_r} = C_r^0$ is the empty string here). There is either a continuation of the game such that the players reach the states $p_jC_{3-r}^{v_{3-r}}S$ and $p_j'C_{3-r}^{v_{3-r}}S$, and $\mathit{value}(C_{3-r}^{v_{3-r}}) = (v_1, v_2)$ — or the attacker performs the $\xrightarrow{d}$-move but then the defender wins because of Proposition 3.

□

We arrived at the point where we are ready to prove our main theorem.

Theorem 1. *Weak bisimilarity of pushdown processes is undecidable.*

Proof. Let R be a Minsky machine and let Δ be the pushdown automaton constructed above. We prove that R halts if and only if $p_1S \not\approx p_1'S$.

Assume that R halts, i.e., $(L_1, 0, 0) \hookrightarrow^* (L_n, v_1, v_2)$ for some $v_1, v_2 \in \mathbb{N}$. Then the attacker has a winning strategy starting from p_1S and $p_1'S$. Using repeatedly Lemma 1 and part a) of Lemma 2 we can easily see that the attacker either wins, or the players reach the states $p_n\gamma S$ and $p_n'\gamma' S$ for some $\gamma, \gamma' \in \{C_1, C_2\}^*$ such that $\mathit{value}(\gamma) = \mathit{value}(\gamma') = (v_1, v_2)$. From the pair $p_n\gamma S$ and $p_n'\gamma' S$ the attacker immediately wins by playing $p_n\gamma S \xrightarrow{\mathsf{halt}} p_n\gamma S$ to which the defender has no answer from $p_n'\gamma' S$. Hence $p_1S \not\approx p_1'S$.

On the other hand if R diverges, i.e., there is an infinite computation starting from $(L_1, 0, 0)$, the defender has a winning strategy. Using repeatedly Lemma 1 and part b) of Lemma 2 he can force the attacker to simulate the computation of R in the bisimulation game. Because the computation of R is infinite, so is the bisimulation game starting from p_1S and $p_1'S$. Since any infinite bisimulation game is won by the defender (Definition 1), we get that $p_1S \approx p_1'S$. □

Let us now study the rewrite rules defined above to see whether we can prove the even stronger undecidability result for the normed subclass of pushdown processes. As it can be observed, the pushdown processes p_1S and $p_1'S$ are almost normed. There are only a few exceptions: computations of the pushdown automaton from p_1S and $p_1'S$ can get stuck with nonempty stacks by reaching e.g. the states $p_n'\gamma' S$, $\mathsf{equal}_1 S$, $\mathsf{equal}_2 S$, $\mathsf{empty}_1 S$, or there is an infinite loop where only the increment instructions appear.

It would be easy to fix these problems by adding some extra rules but we didn't want to confuse the reader by mentioning these rules during the development of the undecidability proof. In fact, we can derive undecidability of weak bisimilarity for normed pushdown processes from the following lemma.

Lemma 3. *Let Δ be a pushdown automaton, and $(p_1\alpha_1, \Delta)$ and $(p_2\alpha_2, \Delta)$ a pair of processes. We can construct in polynomial time a pushdown automaton Δ' and a pair of normed processes $(p_1\alpha_1', \Delta')$ and $(p_2\alpha_2', \Delta')$ such that*

$$(p_1\alpha_1, \Delta) \approx (p_2\alpha_2, \Delta) \quad \textit{if and only if} \quad (p_1\alpha_1', \Delta') \approx (p_2\alpha_2', \Delta').$$

Proof. Let Δ be a pushdown automaton with the set of control state Q, stack symbols Γ and actions $\mathcal{A}ct$. We define Δ' with the corresponding sets $Q' \stackrel{\text{def}}{=} Q \cup \{p_d\}$, $\Gamma' \stackrel{\text{def}}{=} \Gamma \cup \{D\}$ and $\mathcal{A}ct' \stackrel{\text{def}}{=} \mathcal{A}ct \cup \{f\}$ such that p_d, D and f are new symbols. In particular, D is the symbol for a new bottom of the stack. Let $\Delta' \stackrel{\text{def}}{=} \Delta \cup \{pX \stackrel{f}{\longrightarrow} p_d \mid p \in Q \text{ and } X \in \Gamma'\} \cup \{p_dX \stackrel{\tau}{\longrightarrow} p_d \mid X \in \Gamma'\}$. We define $\alpha_1' \stackrel{\text{def}}{=} \alpha_1 D$ and $\alpha_2' \stackrel{\text{def}}{=} \alpha_2 D$. Obviously, $(p_1\alpha_1', \Delta')$ and $(p_2\alpha_2', \Delta')$ are normed processes. The validity of $(p_1\alpha_1, \Delta) \approx (p_2\alpha_2, \Delta)$ iff $(p_1\alpha_1', \Delta') \approx (p_2\alpha_2', \Delta')$ is easy to see from the fact that $(p_d\gamma, \Delta') \approx (p_d\gamma', \Delta')$ for any $\gamma, \gamma' \in \Gamma'^*$. □

Corollary 1. *Weak bisimilarity of normed pushdown processes is undecidable.*

Remark 3. Observe that the construction in Lemma 3 gives immediately a polynomial time reduction from weak bisimilarity between pushdown processes and finite-state processes to the normed instances of the problems. It is also easy to see that it preserves the property of being weakly regular, i.e., $(p_1\alpha_1, \Delta)$ is weakly regular iff $(p_1\alpha_1', \Delta')$ is weakly regular.

4 Conclusion

We proved that weak bisimilarity of pushdown processes is undecidable. This result confirms that decidability issues for weak bisimilarity are more complex than those for strong bisimilarity, even though not many examples of infinite-state systems which give similar conclusions have been found so far. In particular, the decidability questions of weak bisimilarity for BPA and BPP are still open. Another interesting problem is decidability of strong/weak regularity for PDA.

Remark 4. It is obvious that the presented reduction from 2-counter machines to weak bisimilarity of pushdown processes can be extended to work for an arbitrary number of counters and hence we think that the problem lies beyond arithmetical hierarhy: the technique of Jančar [10] for showing high undecidability of weak bisimilarity for Petri nets can be adapted also to our case.

In the following table we provide a summary of the state of the art for bisimilarity problems of pushdown processes. The notation $\sim$ FS ($\approx$ FS) stands for strong (weak) bisimilarity checking between pushdown processes and finite-state processes.

	PDA	normed PDA
strong bisimilarity	decidable [19] EXPTIME-hard [13]	decidable [22] EXPTIME-hard [13]
weak bisimilarity	undecidable	undecidable
$\sim$ FS	$\in$ PSPACE [13] PSPACE-hard [14]	$\in$ PSPACE [13] PSPACE-hard [14], Remark 5
$\approx$ FS	$\in$ PSPACE [13] PSPACE-hard [14]	$\in$ PSPACE [13] PSPACE-hard [14], Remark 3
strong regularity	? PSPACE-hard [14]	$\in$ P [4], Remark 6 NL-hard [20]
weak regularity	? PSPACE-hard [14]	? PSPACE-hard [14], Remark 3

Remark 5. The reduction from [14] (Theorem 8) uses unnormed processes but can be modified to work also for the normed case. An important observation is that the stack size of the PDA from Theorem 8 is bounded by the number of variables in the instance of QSAT from which the reduction is done.

Remark 6. Strong regularity of normed PDA is equivalent to the boundedness problem. Boundedness (even for unnormed PDA) is decidable in polynomial time using the fact that the set of all reachable configurations of a pushdown process is a regular language L and a finite automaton recognizing L can be constructed in polynomial time [4].

Acknowledgement. I would like to thank my advisor Mogens Nielsen for his comments and suggestions. I also thank Marco Carbone for useful remarks, Petr Jančar for drawing my attention to high undecidability issues mentioned in Remark 4, and Richard Mayr for several discussions concerning Remark 5. Finally, my thanks go to the anonymous referees for their detailed reviews.

References

[1] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, chapter 9, pages 545–623. Elsevier Science, 2001.

[2] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for basic parallel processes. In *Proc. of CONCUR'93*, volume 715 of *LNCS*, pages 143–157. Springer-Verlag, 1993.

[3] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.

[4] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proc. of CAV'00*, volume 1855 of *LNCS*, pages 232–247. Springer-Verlag, 2000.

[5] J.F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 115(2):353–371, 1994.
[6] Y. Hirshfeld. Bisimulation trees and the decidability of weak bisimulations. In *Proc. of INFINITY'96*, volume 5 of *ENTCS*. Springer-Verlag, 1996.
[7] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science*, 158(1–2):143–159, 1996.
[8] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial-time algorithm for deciding bisimulation equivalence of normed basic parallel processes. *Mathematical Structures in Computer Science*, 6(3):251–259, 1996.
[9] H. Hüttel. Undecidable equivalences for basic parallel processes. In *Proc. of TACS'94*, volume 789 of *LNCS*, pages 454–464. Springer-Verlag, 1994.
[10] P. Jančar. High undecidability of weak bisimilarity for Petri nets. In *Proc. of CAAP'95*, volume 915 of *LNCS*, pages 349–363. Springer-Verlag, 1995.
[11] P. Jančar and J. Esparza. Deciding finiteness of Petri nets up to bisimulation. In *Proc. of ICALP'96*, volume 1099 of *LNCS*, pages 478–489. Springer-Verlag, 1996.
[12] P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proc. of CONCUR'95*, volume 962 of *LNCS*, pages 348–362. Springer-Verlag, 1995.
[13] A. Kučera and R. Mayr. On the complexity of semantic equivalences for pushdown automata and BPA. In *Proc. of MFCS'02*, LNCS. Springer-Verlag, 2002. To appear.
[14] R. Mayr. On the complexity of bisimulation problems for pushdown automata. In *Proc. of IFIP TCS'00*, volume 1872 of *LNCS*. Springer-Verlag, 2000.
[15] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
[16] M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.
[17] D.M.R. Park. Concurrency and automata on infinite sequences. In *Proc. of* 5^{th} *GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.
[18] G. Sénizergues. The equivalence problem for deterministic pushdown automata is decidable. In *Proc. of ICALP'97*, volume 1256 of *LNCS*, pages 671–681. Springer-Verlag, 1997.
[19] G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proc. of FOCS'98*, pages 120–129. IEEE Computer Society, 1998.
[20] J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proc. of ICALP'02*, LNCS. Springer-Verlag, 2002. To appear.
[21] C. Stirling. Local model checking games. In *Proc. of CONCUR'95*, volume 962 of *LNCS*, pages 1–11. Springer-Verlag, 1995.
[22] C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science*, 195(2):113–131, 1998.
[23] C. Stirling. Decidability of weak bisimilarity for a subset of basic parallel processes. In *Proc. of FOSSACS'01*, volume 2030 of *LNCS*, pages 379–393. Springer-Verlag, 2001.
[24] W. Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science (extended abstract). In *Proc. of TAPSOFT'93*, volume 668 of *LNCS*, pages 559–568. Springer-Verlag, 1993.
[25] R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, CWI/Vrije Universiteit, 1990.
[26] R.J. van Glabbeek. The linear time—branching time spectrum. In *Proc. of CONCUR'90*, volume 458 of *LNCS*, pages 278–297. Springer-Verlag, 1990.