Undecidability of Weak Bisimilarity for PA-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 that the problem whether two PA-processes are weakly bisimilar is undecidable. We combine several proof techniques to provide a reduction from Post's correspondence problem to our problem: existential quantification technique, masking technique and deadlock elimination technique.

1 Introduction

The increasing interest in formal verification of concurrent systems has heightened the need for a systematic study of such systems. Of particular interest is the study of equivalence and model checking problems for classes of *infinite-state processes* [2]. To explore the decidability borders of automatic verification and to analyze in detail the situations where such verification is possible, is one of the main goals for theoretical research in this area. The primary focus of this paper is on equivalence checking problems, considering *bisimilarity* as the notion of behavioural equivalence.

The positive development in strong bisimilarity checking for many classes of infinite-state systems had led to the hope that extending the existing techniques to the case of weak bisimilarity might be a feasible step. Some of the recent results, however, contradict this hope. Opposed to the fact that strong bisimilarity is decidable between Petri nets (PN) and finite state systems [7], Jančar and Esparza proved in [6] that weak bisimilarity is undecidable. Similarly, strong bisimilarity is decidable for pushdown processes (PDA) [11], whereas weak bisimilarity is not [14]. Strong bisimilarity of Petri nets is undecidable [5], however, it is at the first level of the arithmetical hierarchy (Π_1^0-complete). On the other hand, weak bisimilarity of Petri nets lies beyond the arithmetical hierarchy [4].

In this paper we further confirm the inherent complexity of weak bisimilarity by showing its undecidability for *PA-processes*. PA (process algebra introduced

* 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.

by Baeten and Weijland [1]) is a formalism which combines the parallel and sequential operator but allows neither communication nor global-state control. This makes the proof more difficult than for PDA [14] and PN [5]: the undecidability argument for PDA uses a finite-state control unit and the proof for PN relies on the possibility of communication.

Our proof of undecidability of weak bisimilarity for PA is by reduction from Post's Correspondence Problem (PCP). For a given instance of PCP we construct a pair of PA processes that are weakly bisimilar if and only if the PCP instance has a solution. We use a game-theoretic characterization of weak bisimilarity (our players are called 'attacker' and 'defender') and combine several techniques to achieve our result.

- The first technique (we call it here *existential quantification*) was first used by Jančar in [4] and explicitly formulated by Srba in [13]. It makes use of the fact that the defender in the bisimulation game has a strategy to decide on a continuation of the game in case of nondeterministic branching. This enables to encode existential quantification. In our case of weak bisimilarity it moreover provides a technique for generating arbitrarily long sequences of process constants (representing solutions of a given PCP instance).
- The second technique, used by Mayr in [8] and called the *masking technique*, deals with the following phenomenon. Assume that X is an unnormed process constant that performs an action 'a' and becomes X again. Whenever X is added via parallel composition to any process expression γ, it is capable of masking every possible occurrence of the action 'a' in γ.
- Finally, we adapt the technique of *deadlock elimination* from [12] into our context, in order to make the proofs more transparent.

Many of the infinite-state systems studied so far can be uniformly defined by means of process rewrite systems (PRS) — see Figure 1 for the PRS-hierarchy from [9]. As a result of our contribution, we can now assert that weak bisimilarity is undecidable for all systems on the third level of the PRS-hierarchy, i.e., for pushdown processes (PDA), PA-processes (PA) and Petri nets (PN).

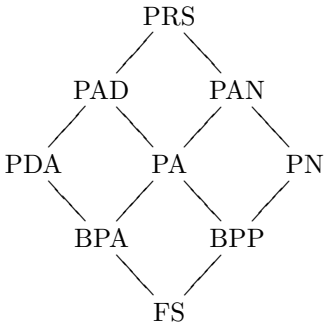

Fig. 1. PRS-hierarchy

On the other hand, the questions for the systems on the second level, namely for basic process algebra (BPA) and basic parallel processes (BPP), still remain open. The techniques used for undecidability of weak bisimilarity for PDA, PA and PN do not seem to be applicable to BPA and BPP, as these systems on the second level lack the ability of remembering global information and they do not allow to mix the sequential and parallel operator. Moreover, we think that weak bisimilarity of BPA and BPP is likely to be decidable.

2 Basic Definitions

Let $\mathcal{C}onst$ be a set of *process constants*. The class of *process expressions* over $\mathcal{C}onst$ is given by $E ::= \epsilon \mid X \mid E.E \mid E \| E$ where 'ϵ' is the *empty process*, X ranges over $\mathcal{C}onst$, '.' is the operator of *sequential composition*, and '$\|$' stands for a *parallel composition*. We do not distinguish between process expressions related by a *structural congruence*, which is the smallest congruence over process expressions such that '.' is associative, '$\|$' is associative and commutative, and 'ϵ' is a unit for '.' and '$\|$'. We shall adopt the convention that the sequential operator binds tighter than the parallel one. Thus for example $X.Y \| Z$ means $(X.Y) \| Z$.

Let $\mathcal{A}ct$ be a set of *actions* such that $\mathcal{A}ct$ contains a distinguished *silent* action τ. We call the elements of the set $\mathcal{A}ct \setminus \{\tau\}$ *visible* actions. A *PA process rewrite system* ($(1,G)$-PRS in the terminology of [9]) is a finite set Δ of *rules* of the form $X \stackrel{a}{\longrightarrow} E$, where $X \in \mathcal{C}onst$, $a \in \mathcal{A}ct$ and E is a process expression. Let us denote the set of actions and process constants that appear in Δ as $\mathcal{A}ct(\Delta)$ resp. $\mathcal{C}onst(\Delta)$ (note that these sets are finite).

A PA system Δ determines a *labelled transition system* where the states are process expressions over $\mathcal{C}onst(\Delta)$, and $\mathcal{A}ct(\Delta)$ is the set of labels. The *transition relation* $\longrightarrow$ is the least relation satisfying the following SOS rules (recall that '$\|$' is commutative).

$$\frac{(X \stackrel{a}{\longrightarrow} E) \in \Delta}{X \stackrel{a}{\longrightarrow} E} \qquad \frac{E \stackrel{a}{\longrightarrow} E'}{E.F \stackrel{a}{\longrightarrow} E'.F} \qquad \frac{E \stackrel{a}{\longrightarrow} E'}{E \| F \stackrel{a}{\longrightarrow} E' \| F}$$

As usual we extend the transition relation to the elements of $\mathcal{A}ct^*$. We write $E \longrightarrow^* E'$ whenever $E \stackrel{w}{\longrightarrow} E'$ for some $w \in \mathcal{A}ct^*$ and say that E' is *reachable from* E. The notation $E \longrightarrow E'$ means that there is an $a \in \mathcal{A}ct$ such that $E \stackrel{a}{\longrightarrow} E'$. We also write $E \stackrel{a}{\not\longrightarrow}$ if there is no E' such that $E \stackrel{a}{\longrightarrow} E'$, and $E \not\longrightarrow$ if $E \stackrel{a}{\not\longrightarrow}$ for all $a \in \mathcal{A}ct$. By $|w|$ we denote the length of w for $w \in \mathcal{A}ct^*$, and we use $|S|$ to stand for the cardinality of a set S.

A process constant $X \in \mathcal{C}onst(\Delta)$ is called a *deadlock* iff $X \not\longrightarrow$. In the usual presentation of PA it is often assumed that Δ contains no deadlocks.

A *PA process* is a pair (P, Δ) where Δ is a PA process rewrite system and P is a process expression over $\mathcal{C}onst(\Delta)$.

Let $E \stackrel{\tau^*}{\longrightarrow} E'$ mean that $E \stackrel{\tau^n}{\longrightarrow} E'$ for some $n \geq 0$. A *weak transition relation* is defined as follows: $\stackrel{a}{\Longrightarrow} \stackrel{\text{def}}{=} \stackrel{\tau^*}{\longrightarrow} \circ \stackrel{a}{\longrightarrow} \circ \stackrel{\tau^*}{\longrightarrow}$ if $a \in \mathcal{A}ct \setminus \{\tau\}$, and $\stackrel{a}{\Longrightarrow} \stackrel{\text{def}}{=} \stackrel{\tau^*}{\longrightarrow}$ if $a = \tau$. As before we extend the weak transition relation to the elements of $\mathcal{A}ct^*$ and write $E \stackrel{a}{\not\Longrightarrow}$ whenever there is no E' such that $E \stackrel{a}{\Longrightarrow} E'$.

Now we introduce the concept of *weak bisimilarity*. Let Δ be a fixed PA system. A binary relation R over process expressions is a *weak bisimulation* iff whenever $(E, F) \in R$ then for each $a \in \mathcal{A}ct(\Delta)$: if $E \stackrel{a}{\longrightarrow} E'$ then $F \stackrel{a}{\Longrightarrow} F'$ for some F' such that $(E', F') \in R$; if $F \stackrel{a}{\longrightarrow} F'$ then $E \stackrel{a}{\Longrightarrow} E'$ for some E' such that $(E', F') \in R$. Processes (P_1, Δ) and (P_2, Δ) are *weakly bisimilar*, and we write $(P_1, \Delta) \approx (P_2, \Delta)$, iff there is a weak bisimulation R such that $(P_1, P_2) \in R$. If Δ is clear from the context we write only $P_1 \approx P_2$.

Bisimulation equivalence has an elegant characterisation in terms of *bisimulation games* [16, 15]. A bisimulation game on a pair of processes (P_1, Δ) and (P_2, Δ) is a two-player game between an 'attacker' and a 'defender'. The game is played in rounds. In each round the attacker chooses one of the processes and makes an $\stackrel{a}{\longrightarrow}$-move for some $a \in \mathcal{A}ct(\Delta)$. The defender must respond by making an $\stackrel{a}{\Longrightarrow}$-move in the other process under the same action a. Now the game repeats, starting from the new processes. If one player cannot move, the other player wins. If the game is infinite, the defender wins. The processes (P_1, Δ) and (P_2, Δ) are weakly bisimilar iff the defender has a winning strategy (and nonbisimilar iff the attacker has a winning strategy).

The following proposition will be useful later and it simply rephrases a standard result that weak bisimilarity is a congruence w.r.t. to the parallel operator.

Proposition 1. *If the defender has a winning strategy from a pair E and F then he also has a winning strategy from $E\|\gamma$ and $F\|\gamma$ for any process expression γ.*

3 Undecidability of Weak Bisimilarity

We show that the problem whether $(P_1, \Delta) \approx (P_2, \Delta)$ for a given pair of PA processes (P_1, Δ) and (P_2, Δ) is undecidable. For technical convenience we use the power of deadlocks to achieve this result, however, at the end of this section we discuss a simple technique for deadlock elimination. Thus the undecidability result is valid even for PA without deadlocks.

Let us first define Post's correspondence problem (PCP): given a nonempty alphabet Σ and two lists $A = [u_1, \ldots, u_n]$ and $B = [v_1, \ldots, v_n]$ where $n > 0$ and $u_k, v_k \in \Sigma^+$ for all k, $1 \leq k \leq n$, the question is to decide whether the (A, B)-instance has a solution, i.e., whether there is an integer $m \geq 1$ and a sequence of indices $i_1, \ldots, i_m \in \{1, \ldots, n\}$ such that $u_{i_1} u_{i_2} \ldots u_{i_m} = v_{i_1} v_{i_2} \ldots v_{i_m}$.

According to the classical result due to Post, this problem is undecidable [10]. Let us consider an (A, B)-instance of PCP where

$$A = [u_1, \ldots, u_n] \quad \text{and} \quad B = [v_1, \ldots, v_n].$$

We construct a PA system Δ and a pair of processes (P_1, Δ) and (P_2, Δ) such that the (A, B)-instance has a solution if and only if $(P_1, \Delta) \approx (P_2, \Delta)$.

Let $\mathcal{SF}(\alpha)$ denote the set of all suffixes of a sequence $\alpha \in \Sigma^*$, i.e., $\mathcal{SF}(\alpha) \stackrel{\text{def}}{=} \{\alpha' \in \Sigma^* \mid \exists \alpha'' \in \Sigma^* \text{ such that } \alpha = \alpha''\alpha'\}$. Note that $\epsilon \in \mathcal{SF}(\alpha)$ for any α. We can now define the set of process constants $\mathcal{C}onst(\Delta)$ and actions $\mathcal{A}ct(\Delta)$ by

$$\begin{aligned} \mathcal{C}onst(\Delta) \stackrel{\text{def}}{=} & \{U^{u_k} \mid 1 \leq k \leq n\} \cup \{V^{v_k} \mid 1 \leq k \leq n\} \cup \\ & \{T^w \mid w \in \bigcup_{k=1}^{n} \mathcal{SF}(u_k) \cup \bigcup_{k=1}^{n} \mathcal{SF}(v_k)\} \cup \\ & \{X, X', X'_1, Y, Y', Y_1, Z, C, C_1, C_2, W, D\} \end{aligned}$$

$$\begin{aligned} \mathcal{A}ct(\Delta) \stackrel{\text{def}}{=} & \{a \mid a \in \Sigma\} \cup \{\iota_k \mid 1 \leq k \leq n\} \cup \\ & \{x, y, z, c_1, c_2, \tau\}. \end{aligned}$$

Remark 1. In what follows D will be a distinguished process constant with no rules associated to it (deadlock). Hence in particular $\alpha.D.\beta \parallel \gamma \approx \alpha \parallel \gamma$ for any process expressions α, β and γ.

To make the rewrite rules introduced in this section more understandable, we define the system Δ in four stages. It is important to remark here that whenever we define the rules for some process constant $Q \in \mathcal{C}onst(\Delta)$, we always give all the rules for Q at the same stage. Our ultimate goal is to show that $(X\|C, \Delta) \approx (X'\|C, \Delta)$ if and only if the given (A, B)-instance of PCP has a solution. The first part of the system Δ is given by the following rules:

$$
\begin{array}{lll}
U^{u_k} \xrightarrow{\iota_k} \epsilon & U^{u_k} \xrightarrow{\tau} T^{u_k} & \text{for all } k \in \{1, \ldots, n\} \\
V^{v_k} \xrightarrow{\iota_k} \epsilon & V^{v_k} \xrightarrow{\tau} T^{v_k} & \text{for all } k \in \{1, \ldots, n\} \\
\\
T^{aw} \xrightarrow{a} T^{w} & T^{aw} \xrightarrow{\tau} T^{w} & \text{for all } a \in \Sigma \text{ and } w \in \Sigma^* \text{ such that} \\
 & & aw \in \bigcup_{k=1}^{n} \mathcal{SF}(u_k) \cup \bigcup_{k=1}^{n} \mathcal{SF}(v_k) \\
T^{\epsilon} \xrightarrow{\tau} \epsilon. & &
\end{array}
$$

This means that for a given $k \in \{1, \ldots, n\}$ the process constants U^{u_k} and V^{v_k} can perform e.g. the following transitions (or transition sequences): $U^{u_k} \xrightarrow{\iota_k} \epsilon$, $V^{v_k} \xrightarrow{\iota_k} \epsilon$, $U^{u_k} \stackrel{u_k}{\Longrightarrow} \epsilon$, $V^{v_k} \stackrel{v_k}{\Longrightarrow} \epsilon$, $U^{u_k} \stackrel{\tau}{\Longrightarrow} \epsilon$, $V^{v_k} \stackrel{\tau}{\Longrightarrow} \epsilon$. The intuition is that a solution $i_1, \ldots, i_m \in \{1, \ldots, n\}$ of the (A, B)-instance is represented by a pair of processes $U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$ and $V^{v_{i_1}}.V^{v_{i_2}}.\cdots.V^{v_{i_m}}$. These processes can perform the sequences of visible actions $u_{i_1}u_{i_2}\ldots u_{i_m}$ and $v_{i_1}v_{i_2}\ldots v_{i_m}$, respectively, or they can perform the actions corresponding to the indices, namely $\iota_{i_1}\iota_{i_2}\ldots\iota_{i_m}$. Moreover, since there is no global state control, the processes can produce also a combination of the actions from Σ and $\{\iota_1, \ldots, \iota_n\}$. In order to avoid this undesirable behaviour, we add (via parallel composition) a process constant C_1 or C_2 such that C_1 masks all the actions from Σ and C_2 masks all the actions testing the indices. The reason for adding Z will become clear later.

The rewrite rules for C_1, C_2 and Z are given by:

$$
\begin{array}{ll}
C_1 \xrightarrow{a} C_1 & \text{for all } a \in \Sigma \\
C_2 \xrightarrow{\iota_k} C_2 & \text{for all } k \in \{1, \ldots, n\} \\
Z \xrightarrow{z} \epsilon & Z \xrightarrow{\tau} D.
\end{array}
$$

Lemma 1. *It holds that*

$$Z.U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}} \parallel C_1 \approx Z.V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}} \parallel C_1$$

if and only if

$$m = m' \text{ and } i_\ell = j_\ell \text{ for all } \ell,\ 1 \leq \ell \leq m = m'.$$

Proof. "⇒": Assume that (i) $m \neq m'$, or (ii) $m = m'$ and let ℓ, $1 \leq \ell \leq m = m'$, be the smallest number such that $i_\ell \neq j_\ell$. It is easy to show that $Z.U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}} \parallel C_1 \not\approx Z.V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}} \parallel C_1$. In case (i), assuming w.l.o.g. that $m > m'$, the attacker can perform in the first process a sequence of actions $z\iota_{i_1}\iota_{i_2}\ldots\iota_{i_m}$ of length $m+1$ and the defender cannot answer by any corresponding sequence of the same length from the second process ($m > m'$). Hence the attacker wins. In case (ii), the attacker again performs the sequence $z\iota_{i_1}\iota_{i_2}\ldots\iota_{i_m}$ in the first process. The only appropriate sequence of the same length that the defender can perform in the second process is $z\iota_{j_1}\iota_{j_2}\ldots\iota_{j_{m'}}$ — obviously no τ rules can be used otherwise the defender loses (his sequence gets shorter). The attacker wins because $\iota_{i_\ell} \neq \iota_{j_\ell}$.
"⇐": We show that $Z.U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}} \parallel C_1 \approx Z.V^{v_{i_1}}.V^{v_{i_2}}.\cdots.V^{v_{i_m}} \parallel C_1$. Let $U(\ell) \stackrel{\text{def}}{=} U^{u_{i_\ell}}.U^{u_{i_{\ell+1}}}.\cdots.U^{u_{i_m}}$ and $V(\ell) \stackrel{\text{def}}{=} V^{v_{i_\ell}}.V^{v_{i_{\ell+1}}}.\cdots.V^{v_{i_m}}$ for all ℓ, $1 \leq \ell \leq m$. By definition $U(m+1) \stackrel{\text{def}}{=} \epsilon$ and $V(m+1) \stackrel{\text{def}}{=} \epsilon$. Let us consider the following relation R.

$$
\begin{array}{l}
\{\ (\ \ Z.U(1)\|C_1\ ,\ Z.V(1)\|C_1\ \ )\ \} \cup \\
\{\ (\ \ D.U(1)\|C_1\ ,\ D.V(1)\|C_1\ \ )\ \} \cup \\
\{\ (\ \ \ \ U(\ell)\|C_1\ ,\ V(\ell)\|C_1\ \ \ \ )\mid 1 \leq \ell \leq m+1\ \} \cup \\
\{\ (\ T^w.U(\ell)\|C_1\ ,\ V(\ell)\|C_1\ \ \ \ )\mid 2 \leq \ell \leq m+1\ \wedge\ w \in \mathcal{SF}(u_{\ell-1})\} \cup \\
\{\ (\ \ \ \ U(\ell)\|C_1\ ,\ T^w.V(\ell)\|C_1\ )\mid 2 \leq \ell \leq m+1\ \wedge\ w \in \mathcal{SF}(v_{\ell-1})\}
\end{array}
$$

It is a routine exercise to check that R is a weak bisimulation. Moreover, it satisfies that $(Z.U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}} \parallel C_1,\ Z.V^{v_{i_1}}.V^{v_{i_2}}.\cdots.V^{v_{i_m}} \parallel C_1) \in R$. □

Lemma 2. *It holds that*

$$Z.U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}} \parallel C_2 \approx Z.V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}} \parallel C_2$$

if and only if

$$u_{i_1}u_{i_2}\ldots u_{i_m} = v_{j_1}v_{j_2}\ldots v_{j_{m'}}.$$

Proof. "⇒": Let $\sigma \stackrel{\text{def}}{=} U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$ and $\omega \stackrel{\text{def}}{=} V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}}$, and let $u \stackrel{\text{def}}{=} u_{i_1}u_{i_2}\ldots u_{i_m}$ and $v \stackrel{\text{def}}{=} v_{j_1}v_{j_2}\ldots v_{j_{m'}}$. Hence $\sigma \stackrel{u}{\Longrightarrow} \epsilon$ and $\omega \stackrel{v}{\Longrightarrow} \epsilon$, and u and v are the longest sequences (and unique ones among the sequences of the length $|u|$ resp. $|v|$) of visible actions from Σ that σ and ω can perform. From the assumption that $u \neq v$ it is easy to see that $Z.\sigma\|C_2 \not\approx Z.\omega\|C_2$.
"⇐": We show that $Z.U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}} \parallel C_2 \approx Z.V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}} \parallel C_2$ assuming that $u_{i_1}u_{i_2}\ldots u_{i_m} = v_{j_1}v_{j_2}\ldots v_{j_{m'}}$. Let $\alpha \in \mathcal{SF}(u_{i_1}u_{i_2}\ldots u_{i_m}) = \mathcal{SF}(v_{j_1}v_{j_2}\ldots v_{j_{m'}})$. We define two sets $U(\alpha)$ and $V(\alpha)$. The intuition is that $U(\alpha)$ contains all the states reachable from $U^{u_{i_1}}.\cdots.U^{u_{i_m}}$ such that α is the longest sequence of visible actions from Σ that these states can perform, and similarly for $V(\alpha)$.

Let us fix the following notation: $U^{u_{m+1}}.\cdots.U^{u_m} \stackrel{\text{def}}{=} \epsilon$, $V^{v_{m'+1}}.\cdots.V^{v_{m'}} \stackrel{\text{def}}{=} \epsilon$ (here 'ϵ' stands for the empty process), and $u_{m+1}\ldots u_m \stackrel{\text{def}}{=} \epsilon$, $v_{m'+1}\ldots v_{m'} \stackrel{\text{def}}{=} \epsilon$ (here 'ϵ' means the empty sequence of actions).

$$U(\alpha) \stackrel{\text{def}}{=} \{U^{u_{i_\ell}}.U^{u_{i_{\ell+1}}}.\cdots.U^{u_{i_m}} \mid 1 \le \ell \le m \ \wedge\ u_{i_\ell}u_{i_{\ell+1}}\ldots u_{i_m} = \alpha\} \cup \{T^w.U^{u_{i_\ell}}.U^{u_{i_{\ell+1}}}.\cdots.U^{u_{i_m}} \mid 2 \le \ell \le m+1 \ \wedge\ w \in \mathcal{SF}(u_{i_{\ell-1}}) \wedge wu_{i_\ell}u_{i_{\ell+1}}\ldots u_{i_m} = \alpha\}$$

$$V(\alpha) \stackrel{\text{def}}{=} \{V^{v_{j_\ell}}.V^{v_{j_{\ell+1}}}.\cdots.V^{v_{j_{m'}}} \mid 1 \le \ell \le m' \ \wedge\ v_{j_\ell}v_{j_{\ell+1}}\ldots v_{j_{m'}} = \alpha\} \cup \{T^w.V^{v_{j_\ell}}.V^{v_{j_{\ell+1}}}.\cdots.V^{v_{j_{m'}}} \mid 2 \le \ell \le m'+1 \ \wedge\ w \in \mathcal{SF}(v_{j_{\ell-1}}) \wedge wv_{j_\ell}v_{j_{\ell+1}}\ldots v_{j_{m'}} = \alpha\}.$$

We remind the reader of the fact that $1 \le |U(\alpha)|, |V(\alpha)| \le 3$ for all α. For example if $m \ge 2$ then $U(u_{i_m}) = \{U^{u_{i_m}},\ T^\epsilon.U^{u_{i_m}},\ T^{u_{i_m}}\}$. Moreover, if $E \in U(\alpha)$ and $F \in V(\alpha)$ then $E \stackrel{\alpha}{\Longrightarrow} \epsilon$ and $F \stackrel{\alpha}{\Longrightarrow} \epsilon$, and α is the longest sequence of actions from Σ satisfying this property. Let us consider the following relation R where $U(1) \stackrel{\text{def}}{=} U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$, $V(1) \stackrel{\text{def}}{=} V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}}$, and $\beta \stackrel{\text{def}}{=} u_{i_1}u_{i_2}\ldots u_{i_m} = v_{j_1}v_{j_2}\ldots v_{j_{m'}}$.

$$\begin{array}{l}
\{\ (\ Z.U(1)\|C_2\ ,\ Z.V(1)\|C_2\ )\ \} \cup \\
\{\ (\ D.U(1)\|C_2\ ,\ D.V(1)\|C_2\ )\ \} \cup \\
\{\ (\quad E\|C_2\ ,\ F\|C_2\quad)\mid E \in U(\alpha)\ \wedge\ F \in V(\alpha)\ \wedge\ \alpha \in \mathcal{SF}(\beta)\} \cup \\
\{\ (\quad\ C_2\ ,\ C_2\quad\ )\ \}
\end{array}$$

As in the previous lemma, it is easy to check that R is a weak bisimulation. Moreover $(Z.U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}} \parallel C_2,\ Z.V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}} \parallel C_2) \in R$. □

We continue with the definition of Δ by adding rules which enable the defender to generate a solution of the (A, B)-instance (if it exists).

$$\begin{array}{lll}
X \stackrel{x}{\longrightarrow} Y & X' \stackrel{x}{\longrightarrow} X'_1 & \\
X \stackrel{x}{\longrightarrow} X'_1 & & \\
 & X'_1 \stackrel{\tau}{\longrightarrow} X'_1.V^{v_k} & \text{for all } k \in \{1,\ldots,n\} \\
 & X'_1 \stackrel{\tau}{\longrightarrow} Y'.V^{v_k} & \text{for all } k \in \{1,\ldots,n\} \\
 & & \\
Y \stackrel{y}{\longrightarrow} Y_1 & Y' \stackrel{y}{\longrightarrow} Z & \\
 & Y' \stackrel{y}{\longrightarrow} Y_1.D & \\
Y_1 \stackrel{\tau}{\longrightarrow} Y_1.U^{u_k} & & \text{for all } k \in \{1,\ldots,n\} \\
Y_1 \stackrel{\tau}{\longrightarrow} Z.U^{u_k} & & \text{for all } k \in \{1,\ldots,n\}
\end{array}$$

See Figure 2 for fragments of transition systems generated by (X, Δ) and (X', Δ). The following lemma explains the purpose of the rules defined above.

Lemma 3. *Consider a bisimulation game from (X, Δ) and (X', Δ). The defender has a strategy such that after two rounds the players reach a pair of states $Z.\sigma$ and $Z.\omega$ where $\sigma = U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$ and $\omega = V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}}$ $(m, m' \ge 1)$, and where σ and ω were chosen by the defender; or the defender wins by reaching a pair of weakly bisimilar states.*

Proof. In the first round of the bisimulation game played from (X, Δ) and (X', Δ) the attacker has only one possible move: $X \stackrel{x}{\longrightarrow} Y$. If the attacker

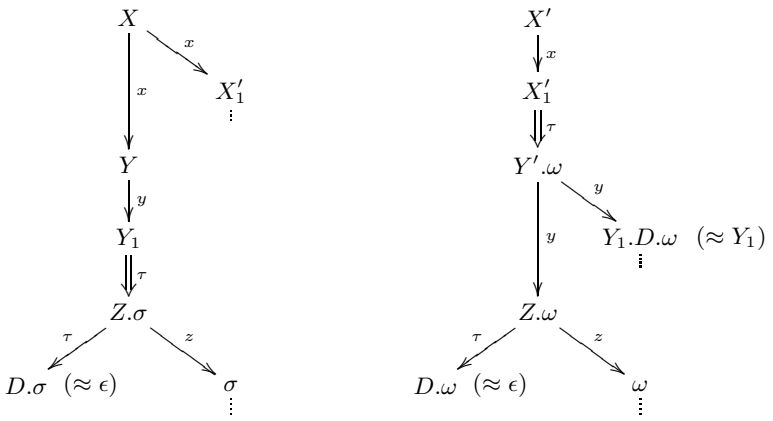

Fig. 2. Fragments of transition systems generated by (X, Δ) and (X', Δ)

plays any other move ($X \xrightarrow{x} X_1'$ or $X' \xrightarrow{x} X_1'$) then the defender can make the resulting processes syntactically equal and he wins. The defender's answer to the move $X \xrightarrow{x} Y$ is by $X' \stackrel{x}{\Longrightarrow} Y'.\omega$ for some $\omega = V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}}$ such that $m' \geq 1$.

In the next round played from Y and $Y'.\omega$ the attacker is forced to continue by $Y'.\omega \xrightarrow{y} Z.\omega$. Similarly as in the first round: if the attacker chooses any other move, the defender can make the resulting processes weakly bisimilar (here we use the fact that $Y_1 \approx Y_1.D.\omega$). The defender can now choose some $\sigma = U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$ such that $m \geq 1$ and plays $Y \stackrel{y}{\Longrightarrow} Z.\sigma$. Hence the defender either won or he chose nonempty σ and ω and forced the attacker in two rounds to reach the pair $Z.\sigma$ and $Z.\omega$. □

We finish the definition of Δ by adding the rules:

$$C \xrightarrow{c_1} C_1 \qquad C \xrightarrow{c_2} C_2 \qquad C \xrightarrow{z} C\|W$$

$$W \xrightarrow{\tau} W.U^{u_k} \qquad W \xrightarrow{\tau} W.V^{v_k} \qquad \text{for all } k \in \{1, \ldots, n\}$$
$$W \xrightarrow{\tau} \epsilon.$$

The intuition is that while playing a bisimulation game from $X\|C$ and $X'\|C$, the defender can generate a solution of the (A, B)-instance by forcing the attacker to reach the states $Z.\sigma\|C$ and $Z.\omega\|C$ (see Lemma 3) such that $\sigma = U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$ and $\omega = V^{v_{i_1}}.V^{v_{i_2}}.\cdots.V^{v_{i_m}}$ where $i_1, \ldots, i_m$ is a solution of the (A, B)-instance (if it exists). The attacker waits with using the rule $C \xrightarrow{c_1} C_1$ or $C \xrightarrow{c_2} C_2$ until the pair $Z.\sigma\|C$ and $Z.\omega\|C$ is reached and then he can check that the sequence $i_1, \ldots, i_m$ is indeed a solution: from $Z.\sigma\|C_1$ and $Z.\omega\|C_1$ he checks whether the defender generated the same indices in both σ and ω, and from $Z.\sigma\|C_2$ and $Z.\omega\|C_2$ he checks whether $u_{i_1}u_{i_2}\ldots u_{i_m} = v_{i_1}v_{i_2}\ldots v_{i_m}$. The purpose of the rules for the process constant W is explained later.

Lemma 4. *If* $(X\|C, \Delta) \approx (X'\|C, \Delta)$ *then the* (A, B)*-instance has a solution.*

Proof. Assume that the (A, B)-instance has no solution, i.e., for every sequence of indices $i_1, \ldots, i_m \in \{1, \ldots, n\}$ where $m \geq 1$ it is the case that $u_{i_1} u_{i_2} \ldots u_{i_m} \neq v_{i_1} v_{i_2} \ldots v_{i_m}$. We show that the attacker has a winning strategy from the pair $X\|C$ and $X'\|C$. In the first round the attacker plays $X\|C \xrightarrow{x} Y\|C$. The defender can only answer by $X'\|C \xrightarrow{x} X'_1\|C$ followed by a finite number of τ actions, thus reaching a state $X'_1.\omega\|C$ or $Y'.\omega\|C$ for some ω. In the first case the attacker switches the processes and uses e.g. the rule $X'_1 \xrightarrow{\tau} Y'.V^{v_1}$. Since $Y\|C \not\xrightarrow{\tau}$, the defender can only stay at the state $Y\|C$. In the second case the state is already of the form $Y'.\omega\|C$.

The game now continues from the pair of states $Y\|C$ and $Y'.\omega\|C$ for some ω. The attacker chooses the move $Y'.\omega\|C \xrightarrow{y} Z.\omega\|C$. The defender has to answer by $Y\|C \xrightarrow{y} Y_1\|C$ followed by a finite number of τ actions. This means that he can reach a state $Y_1.\sigma\|C$, or $Z.\sigma\|C$, or $D.\sigma\|C$ for some σ. The attacker wants to force the defender to reach the second possibility. We show later that if the defender reaches $D.\sigma\|C$ then he loses. Moreover, if the defender reaches $Y_1.\sigma\|C$ then the attacker can use e.g. the rule $Y_1 \xrightarrow{\tau} Z.U^{u_1}$ and the defender can only respond by staying in $Z.\omega\|C$, or by the move $Z.\omega\|C \xrightarrow{\tau} D.\omega\|C$. As we want the game to continue from $Z.\sigma\|C$ and $Z.\omega\|C$, it is enough to show that the attacker has a winning strategy from $D.\sigma\|C$ and $Z.\omega\|C$, and from $Z.\sigma\|C$ and $D.\omega\|C$. We show how the attacker wins from $D.\sigma\|C$ and $Z.\omega\|C$ (the situation from $Z.\sigma\|C$ and $D.\omega\|C$ is completely symmetric). The attacker plays in the second state: $Z.\omega\|C \xrightarrow{c_1} Z.\omega\|C_1$. The defender can only respond by $D.\sigma\|C \xrightarrow{c_1} D.\sigma\|C_1$. Now, $Z.\omega\|C_1 \xrightarrow{z} \omega\|C_1$ but $D.\sigma\|C_1 \overset{z}{\not\Longrightarrow}$. Hence the attacker wins.

To sum up, either the attacker wins or the game continues from the pair $Z.\sigma\|C$ and $Z.\omega\|C$ for some $\sigma = U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$ and $\omega = V^{v_{j_1}}.V^{v_{j_2}}.\cdots.V^{v_{j_{m'}}}$ where $m, m' \geq 1$. There are two cases.

- If $m = m'$ and $i_\ell = j_\ell$ for all ℓ, $1 \leq \ell \leq m = m'$, then using our assumption that the (A, B)-instance has no solution and by the fact that $m, m' \geq 1$ we get that $u_{i_1} u_{i_2} \ldots u_{i_m} \neq v_{i_1} v_{i_2} \ldots v_{i_m}$. The attacker plays $Z.\sigma\|C \xrightarrow{c_2} Z.\sigma\|C_2$ and the defender has to answer by $Z.\omega\|C \xrightarrow{c_2} Z.\omega\|C_2$ or $Z.\omega\|C \overset{c_2}{\Longrightarrow} D.\omega\|C_2$. From the pair $Z.\sigma\|C_2$ and $Z.\omega\|C_2$ the attacker has a winning strategy because of Lemma 2 and the attacker's strategy from the pair $Z.\sigma\|C_2$ and $D.\omega\|C_2$ is obvious: $Z.\sigma\|C_2 \xrightarrow{z} \sigma\|C_2$ but $D.\omega\|C_2 \overset{z}{\not\Longrightarrow}$.
- If it is not the case that $m = m'$ and $i_\ell = j_\ell$ for all ℓ, $1 \leq \ell \leq m = m'$, the attacker plays $Z.\sigma\|C \xrightarrow{c_1} Z.\sigma\|C_1$ and the defender must respond by $Z.\omega\|C \xrightarrow{c_1} Z.\omega\|C_1$ or $Z.\omega\|C \overset{c_1}{\Longrightarrow} D.\omega\|C_1$. By Lemma 1 the attacker has a winning strategy from $Z.\sigma\|C_1$ and $Z.\omega\|C_1$. The argument for the attacker's winning strategy from $Z.\sigma\|C_1$ and $D.\omega\|C_1$ is as in the previous case.

□

Lemma 5. *If the* (A, B)*-instance has a solution then* $(X\|C, \Delta) \approx (X'\|C, \Delta)$.

Proof. Let $i_1, \ldots, i_m \in \{1, \ldots n\}$ where $m \geq 1$ be a solution of the (A, B)-instance. We show that the defender has a winning strategy from the pair $X \| C$ and $X' \| C$.

As it was already proved in Lemma 3, in the bisimulation game played from X and X' the defender can force the attacker to reach the pair $Z.\sigma$ and $Z.\omega$, or the defender has a winning strategy. In particular, the defender can make sure that the players reach the pair $Z.\sigma$ and $Z.\omega$ where σ and ω correspond to the solution of the (A, B)-instance, i.e., $\sigma = U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$ and $\omega = V^{v_{i_1}}.V^{v_{i_2}}.\cdots.V^{v_{i_m}}$.

The situation in this lemma, however, requires that the players start playing from $X \| C$ and $X' \| C$. We have to extend the defender's strategy by defining his responses to the attacks from the process constant C, or more generally from any context γ reachable from C (see the last part of the definition of Δ). To any attacker's move $X \| \gamma \longrightarrow X \| \gamma'$ or $X' \| \gamma \longrightarrow X' \| \gamma'$ the defender answers simply by imitating the same move in the other process. The bisimulation game then continues from the pair $X \| \gamma'$ and $X' \| \gamma'$. Since any infinite game is a winning one for the defender, the attacker must eventually use some rules for X or X'. In this case the defender uses the strategy from Lemma 3. The attacker is forced to play $X \| \gamma \stackrel{x}{\longrightarrow} Y \| \gamma$ and the defender answers by $X' \| \gamma \stackrel{x}{\Longrightarrow} Y'.\omega \| \gamma$ where $\omega = V^{v_{i_1}}.V^{v_{i_2}}.\cdots.V^{v_{i_m}}$. From the states $Y \| \gamma$ and $Y'.\omega \| \gamma$, again the defender imitates any attacks from the context γ. Thus the attacker must eventually play $Y'.\omega \| \gamma \stackrel{y}{\longrightarrow} Z.\omega \| \gamma$ and the defender answers by $Y \| \gamma \stackrel{y}{\Longrightarrow} Z.\sigma \| \gamma$ where $\sigma = U^{u_{i_1}}.U^{u_{i_2}}.\cdots.U^{u_{i_m}}$.

By inspecting the rules for C we can see that the context γ always contains either the process constant (i) C, (ii) C_1, or (iii) C_2. Hence γ can be written as (i) $C \| \gamma'$, (ii) $C_1 \| \gamma'$, or (iii) $C_2 \| \gamma'$ for some context γ'. In case (ii) the bisimulation game continues from the pair $Z.\sigma \| C_1 \| \gamma'$ and $Z.\omega \| C_1 \| \gamma'$, and the defender has a winning strategy because of Lemma 1 and Proposition 1. In case (iii) the bisimulation game continues from the pair $Z.\sigma \| C_2 \| \gamma'$ and $Z.\omega \| C_2 \| \gamma'$, and the defender has a winning strategy because of Lemma 2 and Proposition 1. It remains to demonstrate that the defender has a winning strategy also in case (i). Hence assume that the game continues from $Z.\sigma \| C \| \gamma'$ and $Z.\omega \| C \| \gamma'$. By Proposition 1 it is enough to show that $Z.\sigma \| C \approx Z.\omega \| C$. We will analyze the attacker's moves from $Z.\sigma \| C$. The arguments for the moves from $Z.\omega \| C$ are completely symmetric. The attacker has the following moves available.

$$
\begin{array}{rl}
\text{(i)} & Z.\sigma \| C \stackrel{c_1}{\longrightarrow} Z.\sigma \| C_1 \\
\text{(ii)} & Z.\sigma \| C \stackrel{c_2}{\longrightarrow} Z.\sigma \| C_2 \\
\text{(iii)} & Z.\sigma \| C \stackrel{z}{\longrightarrow} Z.\sigma \| C \| W \\
\text{(iv)} & Z.\sigma \| C \stackrel{\tau}{\longrightarrow} D.\sigma \| C \\
\text{(v)} & Z.\sigma \| C \stackrel{z}{\longrightarrow} \sigma \| C
\end{array}
$$

In case (i) the defender answers by $Z.\omega \| C \stackrel{c_1}{\longrightarrow} Z.\omega \| C_1$ and wins because of Lemma 1. In case (ii) the defender answers by $Z.\omega \| C \stackrel{c_2}{\longrightarrow} Z.\omega \| C_2$ and wins because of Lemma 2. In case (iii) the defender answers by $Z.\omega \| C \stackrel{z}{\longrightarrow} Z.\omega \| C \| W$. By Proposition 1 this case is already covered by the discussion of the defender's

strategy from $Z.\sigma \| C$ and $Z.\omega \| C$. In case (iv) the defender answers by $Z.\omega \| C \xrightarrow{\tau} D.\omega \| C$ and he wins since $D.\sigma \| C \not\approx C \approx D.\omega \| C$. Case (v) is the only case where we need the rules for the process constant W. The defender answers by the following sequence:

$$Z.\omega \| C \xrightarrow{\tau} D.\omega \| C \xrightarrow{z} D.\omega \| C \| W \stackrel{\tau}{\Longrightarrow} D.\omega \| C \| \sigma.$$

This can be written in one step as $Z.\omega \| C \stackrel{z}{\Longrightarrow} D.\omega \| C \| \sigma$. Now the game continues from the pair $\sigma \| C$ and $D.\omega \| C \| \sigma$, however, $\sigma \| C \not\approx D.\omega \| C \| \sigma$. This implies that the defender has a winning strategy also in this case. □

Theorem 1. *Weak bisimilarity of PA with deadlocks is undecidable.*

Proof. Immediately from Lemmas 4 and 5. □

In the rest of this section we show that the presence of the deadlock D in Δ is not an essential requirement. We build upon the technique of deadlock elimination described (for the case of BPA) in [12].

Lemma 6. *There is a (polynomial time) reduction from weak bisimilarity of PA with deadlocks to weak bisimilarity of PA without deadlocks.*

Proof. Let Δ be a PA system. By $\mathcal{D}(\Delta)$ we denote the set of all process constants which have no rewrite rule in Δ, i.e., $\mathcal{D}(\Delta) = \{X \in \mathcal{C}onst(\Delta) \mid X \not\longrightarrow\}$. Let us consider a PA system Δ' such that $\mathcal{C}onst(\Delta') \stackrel{\text{def}}{=} \mathcal{C}onst(\Delta) \setminus \mathcal{D}(\Delta) \cup \{D\}$ and $\mathcal{A}ct(\Delta') \stackrel{\text{def}}{=} \mathcal{A}ct(\Delta) \cup \{d\}$ where D is a new process constant and d is a new action. Let $\Delta' \stackrel{\text{def}}{=} \{X \xrightarrow{a} \overline{E} \mid (X \xrightarrow{a} E) \in \Delta\} \cup \{D \xrightarrow{d} D\}$ such that $\overline{\epsilon} \stackrel{\text{def}}{=} \epsilon$, $\overline{X} \stackrel{\text{def}}{=} X$ if $X \notin \mathcal{D}(\Delta)$, $\overline{X} \stackrel{\text{def}}{=} D$ if $X \in \mathcal{D}(\Delta)$, $\overline{E.F} \stackrel{\text{def}}{=} \overline{E}.\overline{F}$, and $\overline{E \| F} \stackrel{\text{def}}{=} \overline{E} \| \overline{F}$, where X is a process constant and E, F are process expressions. Obviously $\mathcal{D}(\Delta') = \emptyset$ and it is easy to verify that $(E, \Delta) \approx (F, \Delta)$ if and only if $(\overline{E} \| D, \Delta') \approx (\overline{F} \| D, \Delta')$ for any process expressions E and F. □

Corollary 1. *Weak bisimilarity of PA (without deadlocks) is undecidable.*

4 Conclusion

We proved that weak bisimilarity of PA-processes is undecidable. In our proof we used the notion of deadlocks to make the reduction more understandable, and we also showed that the result can be easily generalized to PA without deadlocks. We took advantage of several new techniques recently developed, in particular the existential quantification technique and the masking technique.

The undecidability result of weak bisimilarity for PA contrasts to the situation of strong bisimilaririty for normed PA, which is known to be decidable in 2-NEXPTIME [3]. The problems of strong bisimilarity for unnormed PA and of weak bisimilarity for normed PA still remain open. Another question to be considered is, whether the problem of weak bisimilarity for PA is highly undecidable. In particular, we do not know whether it lies inside the arithmetical hierarchy, or whether it is beyond the hierarchy, as it is in the case of PDA (Remark 4 in [14]) and PN [4].

Acknowledgements. I would like to thank my advisor Mogens Nielsen for his kind supervision, Petr Jančar for his suggestions, and the referees for their comments.

References

[1] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[2] 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.

[3] Y. Hirshfeld and M. Jerrum. Bisimulation equivalence is decidable for normed process algebra. In *Proc. of Automata, Languages and Programming, 26th International Colloquium (ICALP'99)*, volume 1644 of *LNCS*, pages 412–421. Springer-Verlag, 1999.

[4] P. Jančar. High undecidability of weak bisimilarity for Petri nets. In *Proc. of Colloquium on Trees in Algebra and Programming (CAAP'95)*, volume 915 of *LNCS*, pages 349–363. Springer-Verlag, 1995.

[5] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.

[6] P. Jančar and J. Esparza. Deciding finiteness of Petri nets up to bisimulation. In *Proc. of 23rd International Colloquium on Automata, Languages, and Programming (ICALP'96)*, volume 1099 of *LNCS*, pages 478–489. Springer-Verlag, 1996.

[7] P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proc. of 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *LNCS*, pages 348–362. Springer-Verlag, 1995.

[8] R. Mayr. On the complexity of bisimulation problems for basic parallel processes. In *Proc. of 27st International Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 329–341. Springer-Verlag, 2000.

[9] R. Mayr. Process rewrite systems. *Information and Comp.*, 156(1):264–286, 2000.

[10] E.L. Post. A variant of a recursively unsolvable problem. *Bulletion of the American Mathematical Society*, 52:264–268, 1946.

[11] G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proc. of the 39th Annual Symposium on Foundations of Computer Science(FOCS'98)*, pages 120–129. IEEE Computer Society, 1998.

[12] J. Srba. Basic process algebra with deadlocking states. *Theoretical Computer Science*, 266(1–2):605–630, 2001.

[13] J. Srba. Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard. In *Proc. of 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *LNCS*, pages 535–546. Springer-Verlag, 2002.

[14] J. Srba. Undecidability of weak bisimilarity for pushdown processes. In *Proc. of 13th International Conference on Concurrency Theory (CONCUR'02)*, volume 2421 of *LNCS*, pages 579–593. Springer-Verlag, 2002.

[15] C. Stirling. Local model checking games. In *Proc. of 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *LNCS*, pages 1–11. Springer-Verlag, 1995.

[16] W. Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science (extended abstract). In *Proc. of 4th International Joint Conference CAAP/FASE, Theory and Practice of Software Development (TAPSOFT'93)*, volume 668 of *LNCS*, pages 559–568. Springer-Verlag, 1993.