Acta Informatica manuscript No.
(will be inserted by the editor)

Strong Bisimilarity of Simple Process Algebras: Complexity Lower Bounds[*]

Jiří Srba[**]

BRICS[***]
Department of Computer Science, University of Aarhus,
Ny Munkegade bld. 540, 8000 Aarhus C, Denmark
fax: +45 8942 3255
e-mail: srba@brics.dk

Received: date / Revised version: date

Abstract. In this paper we study bisimilarity problems for simple process algebras. In particular, we show PSPACE-hardness of the following problems: (i) strong bisimilarity of Basic Parallel Processes (BPP), (ii) strong bisimilarity of Basic Process Algebra (BPA), (iii) strong regularity of BPP, and (iv) strong regularity of BPA. We also demonstrate NL-hardness of strong regularity problems for the normed subclasses of BPP and BPA.

Bisimilarity problems for simple process algebras are introduced in a general framework of process rewrite systems, and a uniform description of the new techniques used for the hardness proofs is provided.

1 Introduction

An important question in the area of verification of infinite-state systems is that of *equivalence checking* [3]. In this paper we are interested in equivalence checking problems for simple process algebras, namely for the purely sequential case called *Basic Process Algebra* (BPA) and its parallel analogue called *Basic Parallel Processes* (BPP). These two

[*] A revised and extended version of [28] and [29].

[**] 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.

formalisms occupy the lowest levels in most of the process hierarchies considered in the literature so far [6,23,21].

Strong bisimilarity [25,22] is a well accepted notion of behavioural equivalence for concurrent processes. Unlike all other equivalences in van Glabbeek's spectrum (see [32,33]), strong bisimilarity is decidable for BPA [9] and BPP [8]. This challenging phenomenon was a motivation for further investigation of strong bisimilarity in the class of simple process algebras. Restricted classes of so called *normed processes* were studied (a process is normed if from every reachable state there is at least one terminating computation), with surprising results that even though language equivalence is still undecidable for normed BPA [13] and BPP [14], strong bisimilarity becomes decidable even in polynomial time [11,12]. However, the situation is different for the unrestricted classes of simple process algebras.

Despite the fact that strong bisimilarity of BPA appeared to be decidable in 2-EXPTIME [4], it is still an open problem whether there exists an elementary decision algorithm for BPP. The conjecture that strong bisimilarity of unnormed BPP is decidable (as well as in the normed case) in polynomial time was only recently proved false (unless P=NP) by Mayr. He showed that strong bisimilarity of BPP is co-NP-hard [20]. No nontrivial lower bound was known for unnormed BPA.

We improve Mayr's co-NP lower bound for BPP and show that the complexity of bisimilarity checking of BPA is indeed different (unless P=PSPACE) from the case of normed BPA by demonstrating that strong bisimilarity of BPA and BPP is PSPACE-hard. We describe polynomial time reductions from the *quantified satisfiability* (QSAT) problem (for PSPACE-completeness see e.g. [24]) to strong bisimilarity checking problems for BPA and BPP. Given an instance C of QSAT, we construct a pair of BPA (BPP) processes P_1 and P_2 such that P_1 and P_2 are strongly bisimilar if and only if C is true.

The new contribution is a general technique which enables to imitate a generation of quantified assignments of boolean formulas in the context of bisimulation games of an attacker and a defender. While the truth value of a variable prefixed by the universal quantifier is being chosen, the attacker has the possibility to decide between two alternatives in the continuation of the bisimulation game. On the other hand, while choosing the truth value for an existentially quantified variable, the defender can force the attacker to continue the bisimulation game according to his decision[1]. Satisfied clauses of the formula

[1] Similar ideas appeared independently also due to Jančar in connection with high undecidability of weak bisimilarity for Petri nets [15].

are remembered by means of process constants that are present in the current states of BPA and BPP systems. After the whole assignment of boolean variables was generated, the attacker can make a final check whether all clauses are indeed satisfied. This is easier to verify for BPP because we have a parallel access to all process constants contained in the current state. To achieve the same result for BPA, we have to encode satisfied clauses in a unary way.

Another decidability problem that has attracted much attention is that of *regularity checking*. The question is whether a given BPA (or BPP) process is strongly bisimilar to some finite-state process. Strong regularity checking is decidable in 2-EXPTIME for BPA [5,4] and in polynomial time for normed BPA and BPP [18]. Decidability of strong regularity for BPP follows from the fact that the problem is decidable even for Petri nets [16], a proper superclass of BPP. However, no elementary upper bound has been established so far. It is known that strong regularity is co-NP-hard for BPP [20] and no hardness result was available for BPA. We describe polynomial time reductions from strong bisimilarity of regular BPA (BPP) processes to strong regularity checking of BPA (BPP). By using our PSPACE-hardness of strong bisimilarity for BPA and BPP, and by the fact that the involved processes are strongly regular, we conclude that strong regularity of BPA and BPP are PSPACE-hard problems.

Finally, we also investigate the complexity of regularity checking problems for normed BPA and BPP and show their NL-completeness.

The paper is structured as follows. Basic background is introduced in Section 2 and the general idea of our reduction from QSAT to strong bisimilarity checking is explained in the beginning of Section 3. Next two subsections further develop the idea and show PSPACE-hardness of strong bisimilarity for BPP (Subsection 3.2) and then also for the more involved case of BPA (Subsection 3.3). Regularity checking problems are studied in Section 4: proofs of PSPACE-hardness for BPP and BPA are given in Subsections 4.1 and 4.2, respectively, and NL-completeness of regularity checking under the assumption of normedness is discussed in Subsection 4.3. An overview of the state of the art of bisimilarity and regularity checking problems for BPA and BPP is presented in Section 5.

2 Basic Definitions

2.1 Transition Systems, Bisimilarity

Semantics to process algebras is usually given in terms of (infinite-state) labelled transition systems [26]. Processes are understood as nodes of a certain labelled transition system and the transition relation is defined in a compositional way.

Definition 1 (Labelled transition system).
A labelled transition system T *is a triple* $T = (S, \mathcal{A}ct, \longrightarrow)$ *where*

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

As usual we extend the transition relation to the elements of $\mathcal{A}ct^*$, i.e., $\alpha \xrightarrow{\epsilon} \alpha$ for every $\alpha \in S$, and $\alpha \xrightarrow{aw} \beta$ iff $\alpha \xrightarrow{a} \alpha'$ and $\alpha' \xrightarrow{w} \beta$ for every $\alpha, \beta \in S$, $a \in \mathcal{A}ct$ and $w \in \mathcal{A}ct^*$.

We write $\alpha \longrightarrow^* \beta$ iff $\alpha \xrightarrow{w} \beta$ for some $w \in \mathcal{A}ct^*$. We also write $\alpha \overset{a}{\not\longrightarrow}$ whenever there is no β such that $\alpha \xrightarrow{a} \beta$, and $\alpha \not\longrightarrow$ whenever $\alpha \overset{a}{\not\longrightarrow}$ for all $a \in \mathcal{A}ct$.

Definition 2 (Process and its reachable states).
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* $\alpha \longrightarrow^* \beta$.

Definition 3 (Finite-state process).
Whenever (α, T) *has only finitely many reachable states, we call it a* finite-state process*.*

Definition 4 (Strong bisimilarity).
Let $T = (S, \mathcal{A}ct, \longrightarrow)$ *be a labelled transition system. A binary relation* $R \subseteq S \times S$ *is a* strong bisimulation *iff whenever* $(\alpha, \beta) \in R$ *then for each* $a \in \mathcal{A}ct$*:*

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

Processes (α_1, T) *and* (α_2, T) *are* strongly bisimilar*, written* $(\alpha_1, T) \sim (\alpha_2, T)$ *(or simply* $\alpha_1 \sim \alpha_2$ *if* T *is clear from the context), iff there is a strong 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* $(\alpha_1, T_1) \sim (\alpha_2, T_2)$ *iff* $(\alpha_1, T) \sim (\alpha_2, T)$ *where* T *is a disjoint union of* T_1 *and* T_2*.*

Definition 5 (Strong regularity).
We say that a process (α, T) *is* strongly regular *iff there exists some finite-state process bisimilar to it.*

Bisimulation equivalence has an elegant characterisation in terms of *bisimulation games.*

Definition 6 (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 of 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* $\beta_{3-i}' \in S$ *such that* $\beta_{3-i} \xrightarrow{a} \beta_{3-i}'$*.*
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* $\beta_2 \not\longrightarrow$*.*

The following proposition is a standard one (see e.g. [30,31]).

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

2.2 Process Rewrite Systems

Let $\mathcal{A}ct$ and $\mathcal{C}onst$ be sets of *actions* and *process constants*, respectively, such that $\mathcal{A}ct \cap \mathcal{C}onst = \emptyset$.

Definition 7 (Process expressions).
We define the class of process expressions $\mathcal{G}$ over $\mathcal{C}onst$ *by the following abstract syntax*

$$E \ ::= \ \epsilon \ \mid \ X \ \mid \ E.E \ \mid \ E \| E$$

where 'ϵ*' is the* empty process*,* X *ranges over* $\mathcal{C}onst$*, the operator '.' stands for a sequential composition and '*$\|$*' stands for a parallel composition.*

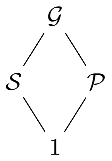

Fig. 1 Classes of Process Expressions

Definition 8 (Structural congruence).

We do not distinguish between process expressions related by a structural congruence*, which is the smallest congruence over process expressions such that the following laws hold:*

- *'$.$' is associative,*
- *'$\|$' is associative and commutative, and*
- *'ϵ' is a unit for '$.$' and '$\|$'.*

Definition 9 (Classes of process expressions).

We distinguish the following classes of process expression:

$\mathcal{G}$*, the class of general process expressions introduced in Definition 7,*
$\mathcal{S}$*, a subclass of $\mathcal{G}$ which contains all process expressions from $\mathcal{G}$ that do not contain the '$\|$' operator,*
$\mathcal{P}$*, a subclass of $\mathcal{G}$ which contains all process expressions from $\mathcal{G}$ that do not contain the '$.$' operator, and*
1*, a subclass of $\mathcal{G}$ which contains only the process constants from $\mathcal{C}onst$ and the empty process 'ϵ'.*

Obviously, $1 \subset \mathcal{S}$, $1 \subset \mathcal{P}$, $\mathcal{S} \subset \mathcal{G}$ and $\mathcal{P} \subset \mathcal{G}$. The classes $\mathcal{S}$ and $\mathcal{P}$ are incomparable and $\mathcal{S} \cap \mathcal{P} = 1$. See Figure 1.

Remark 1. We use the notation $\mathcal{G}(\mathcal{C}onst)$, $\mathcal{S}(\mathcal{C}onst)$, $\mathcal{P}(\mathcal{C}onst)$ and $1(\mathcal{C}onst)$ whenever we need to explicitly specify from which process constants the expressions are formed.

Definition 10 (Process rewrite system (PRS) [21]).

Let $\alpha, \beta \in \{1, \mathcal{S}, \mathcal{P}, \mathcal{G}\}$ such that $\alpha \subseteq \beta$. An (α, β)-PRS is a finite set $\Delta \subseteq \alpha \times \mathcal{A}ct \times \beta$ of rewrite rules*, written $E \xrightarrow{a} F$ for $(E, a, F) \in \Delta$. Moreover, we require that $E \neq \epsilon$.*

Remark 2. Let us denote the set of actions and process constants that appear in Δ by $\mathcal{A}ct(\Delta)$ and $\mathcal{C}onst(\Delta)$, respectively. Note that $\mathcal{A}ct(\Delta)$ and $\mathcal{C}onst(\Delta)$ are finite sets.

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

Fig. 2 SOS rules

Definition 11 (Transition system $T(\Delta)$).
Let Δ be an (α, β)-PRS. The system Δ determines a labelled transition system $T(\Delta) = (\beta, \mathcal{A}ct(\Delta), \longrightarrow)$, where states *are process expressions from the class β (modulo the structural congruence introduced in Definition 8), $\mathcal{A}ct(\Delta)$ is the set of* labels, *and* transition relation *is the least relation satisfying the SOS rules from Figure 2 — recall that '$\|$' is commutative.*

Definition 12 ((α, β)-process).
*An (α, β)-*process *is a process $\big(P, T(\Delta)\big)$ — see Definition 2 — where Δ is an (α, β)-PRS and $P \in \beta$ is a process expression.*

We remind the reader of the fact that Definitions 2 to 5 define the corresponding process properties also for (α, β)-processes. Moreover, in the rest of this paper we denote an (α, β)-process $\big(P, T(\Delta)\big)$ by only (P, Δ), or even P if Δ is clear from the context.

Definition 13 (Normed (α, β)-process).
An (α, β)-process (P, Δ) is normed *iff from every reachable state E in (P, Δ) there is a terminating computation, i.e., $E \longrightarrow^* \epsilon$.*

Many classes of infinite-state systems studied so far — e.g. basic process algebra (BPA), basic parallel processes (BPP), pushdown automata (PDA), Petri nets (PN) and process algebra (PA) — are contained in the hierarchy of process rewrite systems presented in Figure 3. This hierarchy is strict w.r.t. strong bisimilarity and we refer the reader to [21] for further discussions.

In this paper we study the two bottom classes BPA and BPP.

2.3 Basic Process Algebra

Basic Process Algebra (BPA) — or equivalently $(1, \mathcal{S})$-PRS — represents the class of processes introduced by Bergstra and Klop (see [2]). This class corresponds to the transition systems associated with context-free grammars in Greibach normal form (GNF), in which only left-most derivations are allowed.

Let Δ be a BPA process rewrite system. Every rewrite rule from Δ is of the form

$$X \xrightarrow{a} E$$

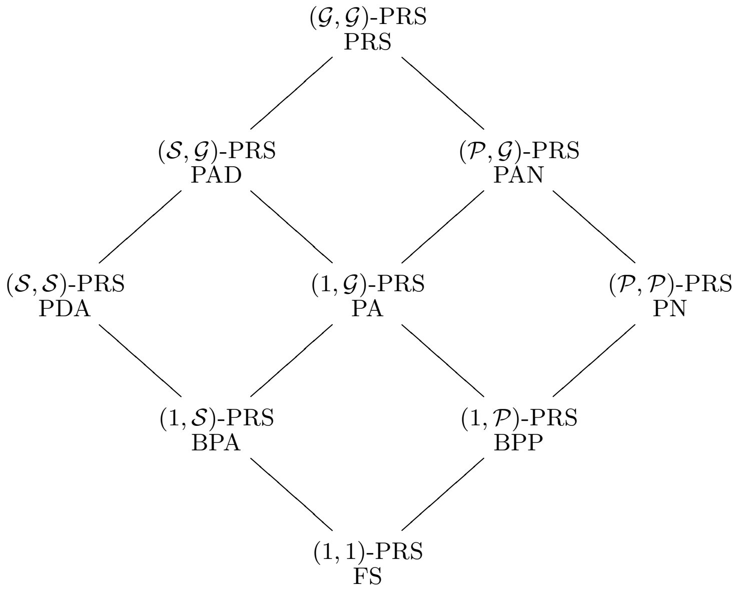
$(\mathcal{G},\mathcal{G})$-PRS
PRS
$(\mathcal{S},\mathcal{G})$-PRS
PAD
$(\mathcal{P},\mathcal{G})$-PRS
PAN
$(\mathcal{S},\mathcal{S})$-PRS
PDA
$(1,\mathcal{G})$-PRS
PA
$(\mathcal{P},\mathcal{P})$-PRS
PN
$(1,\mathcal{S})$-PRS
BPA
$(1,\mathcal{P})$-PRS
BPP
$(1,1)$-PRS
FS

Fig. 3 Hierarchy of process rewrite systems

where $X \in \mathcal{C}onst(\Delta)$, $a \in \mathcal{A}ct(\Delta)$ and $E \in \mathcal{S}(\mathcal{C}onst(\Delta))$. It is usually assumed that for each $X \in \mathcal{C}onst(\Delta)$ there is at least one rewrite rule in Δ, i.e., that there is some $a \in \mathcal{A}ct(\Delta)$ and $E \in \mathcal{S}\big(\mathcal{C}onst(\Delta)\big)$ such that $(X, a, E) \in \Delta$. If it is not the case, we say that the system contains *deadlocks*. A study of decidability problems for BPA with deadlocks is provided in [27].

Remark 3. Let m be a natural number and $A \in \mathcal{C}onst$ be a process constant. Whenever it is clear from the context that we consider only the '.' operator, we use the notation A^m for a sequential composition of m occurrences of A, i.e., $A^0 \stackrel{\text{def}}{=} \epsilon$ and $A^{m+1} \stackrel{\text{def}}{=} A^m.A$.

A simple BPA system is presented in the following example.

Example 1. Let $\mathcal{C}onst(\Delta) = \{Q_1, Q_2, \ldots, Q_k\}$ for a natural number $k > 0$ and let $\mathcal{A}ct(\Delta) = \{a\}$. Consider the following BPA system Δ containing the rewrite rules:

$$\begin{aligned} &Q_1 \xrightarrow{a} \epsilon \\ &Q_{j+1} \xrightarrow{a} Q_j \qquad \text{for all } j,\ 1 \leq j < k. \end{aligned}$$

Observe that $Q_j \xrightarrow{a^j} \epsilon$ for every j, $1 \leq j \leq k$, and no other transitions are possible. Also notice that e.g. $(Q_1^5, \Delta) \sim (Q_1.Q_2^2, \Delta)$ — see Remark 3.

Assume now that $m_1, m_2 > 0$ are natural numbers. For every ℓ_1, $1 \leq \ell_1 \leq m_1$, and every ℓ_2, $1 \leq \ell_2 \leq m_2$, let $i_{\ell_1} \in \{1, 2, \ldots, k\}$ and $j_{\ell_2} \in \{1, 2, \ldots, k\}$. It is an easy observation that

$$(Q_{i_1}.Q_{i_2}.\cdots.Q_{i_{m_1}}, \Delta) \sim (Q_{j_1}.Q_{j_2}.\cdots.Q_{j_{m_2}}, \Delta)$$

if and only if

$$\sum_{\ell_1=1}^{m_1} i_{\ell_1} = \sum_{\ell_2=1}^{m_2} j_{\ell_2}.$$

This example demonstrates that even though the BPA class is non-commutative, we can achieve a restricted commutative behaviour by assuming that $\mathcal{A}ct(\Delta)$ is a singleton set and by encoding process constants in this unary alphabet.

2.4 Basic Parallel Processes

Basic Parallel Processes (BPP) — or equivalently $(1, \mathcal{P})$-PRS — are a fragment of CCS [22] without restriction, relabelling and communication. This class was first studied by Christensen [7], and it is equivalent to the communication-free subclass of Petri nets (each transition has exactly one input place).

Let Δ be a BPP process rewrite system. Every rewrite rule from Δ is of the form

$$X \xrightarrow{a} E$$

where $X \in \mathcal{C}onst(\Delta)$, $a \in \mathcal{A}ct(\Delta)$ and $E \in \mathcal{P}(\mathcal{C}onst(\Delta))$. Unlike BPA, the presence of deadlocks in BPP systems is not essential. Assume that $D \in \mathcal{C}onst(\Delta)$ is a deadlock, i.e., $D \not\longrightarrow$. Then $(E, \Delta) \sim (E \| D, \Delta)$ for any expression $E \in \mathcal{P}\big(\mathcal{C}onst(\Delta)\big)$ and we can safely replace all occurrences of such deadlocks in Δ by the empty process 'ϵ'.

Remark 4. Let m be a natural number and $A \in \mathcal{C}onst$ be a process constant. Whenever it is clear from the context that we consider only the '$\|$' operator, we use the notation A^m for a parallel composition of m occurrences of A, i.e., $A^0 \overset{\text{def}}{=} \epsilon$ and $A^{m+1} \overset{\text{def}}{=} A^m \| A$.

The following example aims to demonstrate that the operator '$\|$' in BPP systems allows a parallel access to all process constants contained in the current state.

Example 2. Let $\mathcal{C}onst(\Delta) = \{Q_1, Q_2, \ldots, Q_k\}$ for a natural number $k > 0$ and let $\mathcal{A}ct(\Delta) = \{q_1, q_2, \ldots, q_k\}$. The set of rewrite rules Δ is defined by:

$$Q_j \xrightarrow{q_j} Q_j \qquad \text{for all } j,\ 1 \le j \le k.$$

Assume now that $m_1, m_2 > 0$ are natural numbers. For every ℓ_1, $1 \le \ell_1 \le m_1$, and every ℓ_2, $1 \le \ell_2 \le m_2$, let $i_{\ell_1} \in \{1, 2, \ldots, k\}$ and $j_{\ell_2} \in \{1, 2, \ldots, k\}$. We conclude that

$$(Q_{i_1} \| Q_{i_2} \| \cdots \| Q_{i_{m_1}}, \Delta) \sim (Q_{j_1} \| Q_{j_2} \| \cdots \| Q_{j_{m_2}}, \Delta)$$

if and only if

$$\{i_1, i_2, \ldots, i_{m_1}\} = \{j_1, j_2, \ldots, j_{m_2}\}.$$

In other words, the processes are strongly bisimilar if and only if for all j, $1 \le j \le k$, the process constant Q_j appears either in both sides of the processes or in neither of them. In the first case the number of occurrences of Q_j is irrelevant since $(Q_j^m, \Delta) \sim (Q_j, \Delta)$ for any natural number $m > 0$ — see Remark 4.

2.5 Definitions of Problems

Problem:	Strong bisimilarity of BPA (BPP)
Instance:	Two BPA (BPP) processes (P_1, Δ) and (P_2, Δ).
Question:	$(P_1, \Delta) \sim (P_2, \Delta)$?

Problem:	Strong regularity of BPA (BPP)
Instance:	A BPA (BPP) process (P, Δ).
Question:	Is there a finite-state process (F, Δ') such that $(P, \Delta) \sim (F, \Delta')$?

The main results of Section 3 are proved by polynomial time reductions from a PSPACE-complete problem called *quantified satisfiability* (QSAT) [24]. We use a version where the prefix of quantifiers starts with the existential one.

Problem:	QSAT
Instance:	A natural number $n > 0$ and a boolean formula ϕ in conjunctive normal form with boolean variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$.
Question:	Is $\exists x_1 \forall y_1 \exists x_2 \forall y_2 \ldots \exists x_n \forall y_n . \phi$ true?

A *literal* is a variable or the negation of a variable. An instance of QSAT is a formula C of the form

$$C \equiv \exists x_1 \forall y_1 \exists x_2 \forall y_2 \ldots \exists x_n \forall y_n .\ C_1 \wedge C_2 \wedge \ldots \wedge C_k$$

where each *clause* C_j, $1 \leq j \leq k$, is a disjunction of literals.

3 Lower Bounds for Strong Bisimilarity

In this section we study strong bisimilarity problems for BPA and BPP.

3.1 The Main Idea

We try to explain here the main idea of PSPACE-hardness proofs given in Subsections 3.2 and 3.3. Our aim is to make the rewrite rules defined in this paper more readable, by demonstrating a general pattern used heavily (with small modifications) later on. Let us consider the following process rewrite system Δ where $\alpha_{\texttt{one}}$ and $\alpha_{\texttt{two}}$ are some fixed process expressions and $\oplus$ is either a sequential or parallel composition.

$$\begin{array}{ll}
X \xrightarrow{a} Y^{\texttt{choice}} & \\
X \xrightarrow{a} Y^{\texttt{one}} & X' \xrightarrow{a} Y^{\texttt{one}} \\
X \xrightarrow{a} Y^{\texttt{two}} & X' \xrightarrow{a} Y^{\texttt{two}} \\
 & \\
Y^{\texttt{choice}} \xrightarrow{\texttt{one}} Z \oplus \alpha_{\texttt{one}} & Y^{\texttt{one}} \xrightarrow{\texttt{one}} Z' \oplus \alpha_{\texttt{one}} \\
Y^{\texttt{choice}} \xrightarrow{\texttt{two}} Z \oplus \alpha_{\texttt{two}} & Y^{\texttt{two}} \xrightarrow{\texttt{two}} Z' \oplus \alpha_{\texttt{two}} \\
Y^{\texttt{one}} \xrightarrow{\texttt{two}} Z \oplus \alpha_{\texttt{two}} & Y^{\texttt{two}} \xrightarrow{\texttt{one}} Z \oplus \alpha_{\texttt{one}}
\end{array}$$

Transition systems generated by processes (X, Δ) and (X', Δ) are depicted in Figure 4. The intuition behind the construction can be nicely explained in terms of bisimulation games. Consider a bisimulation game starting from the pair X and X'.

The attacker is forced to make the first move by playing $X \xrightarrow{a} Y^{\texttt{choice}}$ because in all other possible moves, either from X or X', the defender can make the resulting processes syntactically equal and hence bisimilar. The defender's answer to the move $X \xrightarrow{a} Y^{\texttt{choice}}$ is either (i) $X' \xrightarrow{a} Y^{\texttt{one}}$ or (ii) $X' \xrightarrow{a} Y^{\texttt{two}}$.

In the next round starting from (i) $Y^{\texttt{choice}}$ and $Y^{\texttt{one}}$ or (ii) $Y^{\texttt{choice}}$ and $Y^{\texttt{two}}$, the attacker can use either the action `one` or `two` — obviously it is irrelevant whether the chosen action is performed in the

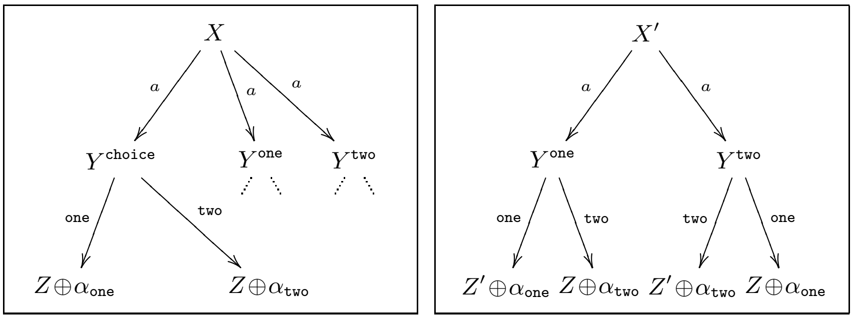
X
a
a
a
$Y^{\texttt{choice}}$
$Y^{\texttt{one}}$
$Y^{\texttt{two}}$
one
two
$Z \oplus \alpha_{\texttt{one}}$
$Z \oplus \alpha_{\texttt{two}}$
X'
a
a
$Y^{\texttt{one}}$
$Y^{\texttt{two}}$
one
two
two
one
$Z' \oplus \alpha_{\texttt{one}}$
$Z \oplus \alpha_{\texttt{two}}$
$Z' \oplus \alpha_{\texttt{two}}$
$Z \oplus \alpha_{\texttt{one}}$

Fig. 4 Processes (X, Δ) and (X', Δ)

first or in the second process. In case (i), if the attacker chooses the action `one` then the players reach the pair $Z \oplus \alpha_{\texttt{one}}$ and $Z' \oplus \alpha_{\texttt{one}}$. If he chooses the action `two` then the players reach a pair of syntactically equal states, namely $Z \oplus \alpha_{\texttt{two}}$ and $Z \oplus \alpha_{\texttt{two}}$, from which the defender has an obvious winning strategy. In case (ii), if the attacker chooses the action `two` then the players reach the pair $Z \oplus \alpha_{\texttt{two}}$ and $Z' \oplus \alpha_{\texttt{two}}$. If he chooses the action `one` then he loses as in case (i). Now, either the defender won by reaching syntactically equal states, or the resulting processes after two rounds are (i) $Z \oplus \alpha_{\texttt{one}}$ and $Z' \oplus \alpha_{\texttt{one}}$ or (ii) $Z \oplus \alpha_{\texttt{two}}$ and $Z' \oplus \alpha_{\texttt{two}}$. Note that it was the defender who had the possibility to decide between adding $\alpha_{\texttt{one}}$ or $\alpha_{\texttt{two}}$.

We can repeat this construction several times in a row, which is explained in more detail in the following two subsections.

3.2 Strong Bisimilarity of BPP

In this subsection we show that strong bisimilarity of BPP is a PSPACE-hard problem. We prove it by reduction from QSAT. Let

$$C \equiv \exists x_1 \forall y_1 \exists x_2 \forall y_2 \ldots \exists x_n \forall y_n .\ C_1 \wedge C_2 \wedge \ldots \wedge C_k$$

be an instance of QSAT. We define the following BPP processes (X_1, Δ) and (X'_1, Δ) where

$$\begin{aligned} \mathcal{C}onst(\Delta) \stackrel{\text{def}}{=} \ & \{Q_1, \ldots, Q_k\} \cup \\ & \{X_1, \ldots, X_{n+1}, Y_1^{\texttt{choice}}, \ldots, Y_n^{\texttt{choice}}, Z_1, \ldots, Z_n\} \cup \\ & \{X'_1, \ldots, X'_{n+1}, Y_1^{\texttt{tt}}, \ldots, Y_n^{\texttt{tt}}, Y_1^{\texttt{ff}}, \ldots, Y_n^{\texttt{ff}}, Z'_1, \ldots, Z'_n\} \end{aligned}$$

and $\mathcal{A}ct(\Delta) \stackrel{\text{def}}{=} \{q_1, \ldots, q_k, a, \texttt{tt}, \texttt{ff}\}$. For each i, $1 \le i \le n$, let

α_i be a parallel composition $Q_{i_1} \| Q_{i_2} \| \cdots \| Q_{i_\ell}$ such that
$1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and
$C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}$ are all the clauses where x_i occurs positively,

$\overline{\alpha_i}$ be a parallel composition $Q_{i_1} \| Q_{i_2} \| \cdots \| Q_{i_\ell}$ such that
$1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and
$C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}$ are all the clauses where x_i occurs negatively,

β_i be a parallel composition $Q_{i_1} \| Q_{i_2} \| \cdots \| Q_{i_\ell}$ such that
$1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and
$C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}$ are all the clauses where y_i occurs positively, and

$\overline{\beta_i}$ be a parallel composition $Q_{i_1} \| Q_{i_2} \| \cdots \| Q_{i_\ell}$ such that
$1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and
$C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}$ are all the clauses where y_i occurs negatively.

Example 3. Let us consider a quantified boolean formula

$$\exists x_1 \forall y_1 \exists x_2 \forall y_2 .\ (x_1 \vee \neg y_1 \vee y_2) \wedge (\neg x_1 \vee y_1 \vee y_2) \wedge (x_1 \vee y_1 \vee y_2 \vee \neg y_2)$$

where $n = 2$, $k = 3$, $C_1 = x_1 \vee \neg y_1 \vee y_2$, $C_2 = \neg x_1 \vee y_1 \vee y_2$ and $C_3 = x_1 \vee y_1 \vee y_2 \vee \neg y_2$. Then

$$\begin{array}{llll} \alpha_1 = Q_1 \| Q_3 & \overline{\alpha_1} = Q_2 & \beta_1 = Q_2 \| Q_3 & \overline{\beta_1} = Q_1 \\ \alpha_2 = \epsilon & \overline{\alpha_2} = \epsilon & \beta_2 = Q_1 \| Q_2 \| Q_3 & \overline{\beta_2} = Q_3 . \end{array}$$

The set Δ is given by the following rewrite rules:

- for all j such that $1 \leq j \leq k$:

$$Q_j \xrightarrow{q_j} Q_j$$

- for all i such that $1 \leq i \leq n$:

$$\begin{array}{ll} X_i \xrightarrow{a} Y_i^{\texttt{choice}} & \\ X_i \xrightarrow{a} Y_i^{\texttt{tt}} & X_i' \xrightarrow{a} Y_i^{\texttt{tt}} \\ X_i \xrightarrow{a} Y_i^{\texttt{ff}} & X_i' \xrightarrow{a} Y_i^{\texttt{ff}} \\ & \\ Y_i^{\texttt{choice}} \xrightarrow{\texttt{tt}} Z_i \| \alpha_i & Y_i^{\texttt{tt}} \xrightarrow{\texttt{tt}} Z_i' \| \alpha_i \\ Y_i^{\texttt{choice}} \xrightarrow{\texttt{ff}} Z_i \| \overline{\alpha_i} & Y_i^{\texttt{ff}} \xrightarrow{\texttt{ff}} Z_i' \| \overline{\alpha_i} \\ Y_i^{\texttt{tt}} \xrightarrow{\texttt{ff}} Z_i \| \overline{\alpha_i} & Y_i^{\texttt{ff}} \xrightarrow{\texttt{tt}} Z_i \| \alpha_i \\ & \\ Z_i \xrightarrow{\texttt{tt}} X_{i+1} \| \beta_i & Z_i' \xrightarrow{\texttt{tt}} X_{i+1}' \| \beta_i \\ Z_i \xrightarrow{\texttt{ff}} X_{i+1} \| \overline{\beta_i} & Z_i' \xrightarrow{\texttt{ff}} X_{i+1}' \| \overline{\beta_i} \end{array}$$

– and

$$X_{n+1} \xrightarrow{a} Q_1 \| Q_2 \| \dots \| Q_{k-1} \| Q_k \qquad X'_{n+1} \xrightarrow{a} \epsilon.$$

We can see the processes (X_1, Δ) and (X'_1, Δ) in Figure 5 (if we set $i = 1$ and $\gamma_1 = \epsilon$). The intuition behind the construction will be explained in terms of bisimulation games and follows the main idea from Subsection 3.1. Consider a bisimulation game starting from the pair of processes X_1 and X'_1.

The attacker is forced to make the first move by playing $X_1 \xrightarrow{a} Y_1^{\mathtt{choice}}$ because in all other possible moves, either from X_1 or X'_1, the defender can make the resulting processes syntactically equal and hence bisimilar. The defender's answer to the move $X_1 \xrightarrow{a} Y_1^{\mathtt{choice}}$ is either (i) $X'_1 \xrightarrow{a} Y_1^{\mathtt{tt}}$ (this corresponds to setting the variable x_1 to true) or (ii) $X'_1 \xrightarrow{a} Y_1^{\mathtt{ff}}$ (this corresponds to setting the variable x_1 to false).

In the next round the attacker is forced to take the action (i) $\mathtt{tt}$ or (ii) $\mathtt{ff}$, according to the defender's choice in the first round. Otherwise the attacker loses. The defender can only imitate the same action in the other process. The resulting processes after two rounds are (i) $Z_1 \| \alpha_1$ and $Z'_1 \| \alpha_1$ or (ii) $Z_1 \| \overline{\alpha_1}$ and $Z'_1 \| \overline{\alpha_1}$. Note that it was the defender who had the possibility to decide between adding α_1 (i.e. setting x_1 to true) or $\overline{\alpha_1}$ (i.e. setting x_1 to false).

In the third round the attacker has the choice of playing either along the action $\mathtt{tt}$ or $\mathtt{ff}$, which corresponds to the universal quantifier in front of y_1. It does not matter in which process the attacker performs the move. The defender has only one possibility how to answer to this move — he must imitate the corresponding move in the other process. The resulting processes are $X_2 \| \gamma_2$ and $X'_2 \| \gamma_2$ such that $\gamma_2 = \widetilde{\alpha}_1 \| \widetilde{\beta}_1$ where $\widetilde{\alpha}_1 \in \{\alpha_1, \overline{\alpha_1}\}$ and $\widetilde{\beta}_1 \in \{\beta_1, \overline{\beta_1}\}$ according to the truth values chosen for x_1 (by the defender) and for y_1 (by the attacker). Now the game continues in similar fashion from $X_2 \| \gamma_2$ and $X'_2 \| \gamma_2$. Playing some of the actions $q_1, \dots, q_k$ cannot make the attacker win since the defender has always the possibility to imitate the same move in the other processes.

Hence if the attacker wants to win he has to reach eventually the states $X_{n+1} \| \gamma_{n+1}$ and $X'_{n+1} \| \gamma_{n+1}$, and then he performs the move $X_{n+1} \| \gamma_{n+1} \xrightarrow{a} Q_1 \| Q_2 \| \dots \| Q_{k-1} \| Q_k \| \gamma_{n+1}$ to which the defender has only one answer, namely $X'_{n+1} \| \gamma_{n+1} \xrightarrow{a} \gamma_{n+1}$. From the states $Q_1 \| Q_2 \| \dots \| Q_{k-1} \| Q_k \| \gamma_{n+1}$ and γ_{n+1} the attacker has the possibility to check whether every clause C_j, $1 \leq j \leq k$, in C is indeed satisfied under the generated truth assignment by using the

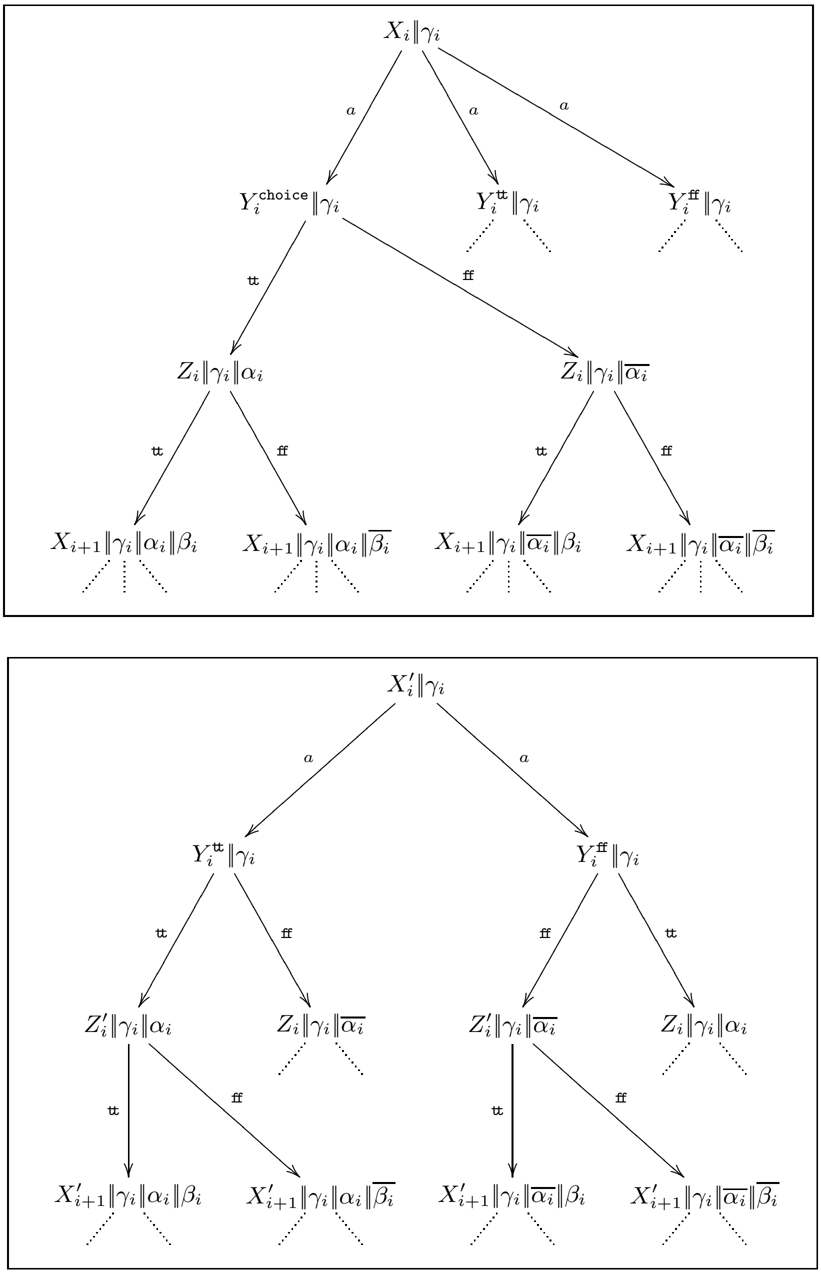
$X_i\|\gamma_i$
a
a
a
$Y_i^{\texttt{choice}}\|\gamma_i$
$Y_i^{\texttt{tt}}\|\gamma_i$
$Y_i^{\texttt{ff}}\|\gamma_i$
tt
ff
$Z_i\|\gamma_i\|\alpha_i$
$Z_i\|\gamma_i\|\overline{\alpha_i}$
tt
ff
tt
ff
$X_{i+1}\|\gamma_i\|\alpha_i\|\beta_i$
$X_{i+1}\|\gamma_i\|\alpha_i\|\overline{\beta_i}$
$X_{i+1}\|\gamma_i\|\overline{\alpha_i}\|\beta_i$
$X_{i+1}\|\gamma_i\|\overline{\alpha_i}\|\overline{\beta_i}$
$X'_i\|\gamma_i$
a
a
$Y_i^{\texttt{tt}}\|\gamma_i$
$Y_i^{\texttt{ff}}\|\gamma_i$
tt
ff
ff
tt
$Z'_i\|\gamma_i\|\alpha_i$
$Z_i\|\gamma_i\|\overline{\alpha_i}$
$Z'_i\|\gamma_i\|\overline{\alpha_i}$
$Z_i\|\gamma_i\|\alpha_i$
tt
ff
tt
ff
$X'_{i+1}\|\gamma_i\|\alpha_i\|\beta_i$
$X'_{i+1}\|\gamma_i\|\alpha_i\|\overline{\beta_i}$
$X'_{i+1}\|\gamma_i\|\overline{\alpha_i}\|\beta_i$
$X'_{i+1}\|\gamma_i\|\overline{\alpha_i}\|\overline{\beta_i}$

Fig. 5 Processes $(X_i\|\gamma_i, \Delta)$ and $(X'_i\|\gamma_i, \Delta)$

rule $Q_j \xrightarrow{q_j} Q_j$ in the first process. If the clause C_j is not satisfied then Q_j does not appear in γ_{n+1} and the defender loses. If all the clauses in C are satisfied then $Q_1 \| Q_2 \| \dots \| Q_{k-1} \| Q_k \| \gamma_{n+1} \sim \gamma_{n+1}$ and the defender wins.

In what follows we formally prove that C is true iff $(X_1, \Delta) \sim (X_1', \Delta)$.

Lemma 1. *If $(X_1, \Delta) \sim (X_1', \Delta)$ then the formula C is true.*

Proof. We show that $(X_1, \Delta) \not\sim (X_1', \Delta)$ under the assumption that C is false. If C is false then C' defined by

$$C' \stackrel{\text{def}}{=} \forall x_1 \exists y_1 \forall x_2 \exists y_2 \dots \forall x_n \exists y_n .\ \neg(C_1 \wedge C_2 \wedge \dots \wedge C_k)$$

is true and we claim that the attacker has a winning strategy in the bisimulation game starting from (X_1, Δ) and (X_1', Δ). The attacker's strategy starts with performing a sequence of actions

$$a, \widetilde{x}_1, \widetilde{y}_1, \dots, a, \widetilde{x}_i, \widetilde{y}_i, \dots, a, \widetilde{x}_n, \widetilde{y}_n, a$$

where $\widetilde{x}_i, \widetilde{y}_i \in \{\mathtt{tt}, \mathtt{ff}\}$ for all i, $1 \le i \le n$. The attacker is playing only in the first process (X_1, Δ). The choice of $\widetilde{x}_i$ is done by the defender and of $\widetilde{y}_i$ by the attacker — see the discussion above. This means that whatever values for $\widetilde{x}_1, \dots, \widetilde{x}_n$ are chosen by the defender, the attacker can still decide on values for $\widetilde{y}_1, \dots, \widetilde{y}_n$ such that the generated assignment satisfies the formula $\neg(C_1 \wedge C_2 \wedge \dots \wedge C_k)$. Hence there must be some j, $1 \le j \le k$, such that the clause C_j is not satisfied. This implies that Q_j does not occur in the second process. However, the attacker can perform the action q_j in the first process by using the rule $Q_j \xrightarrow{q_j} Q_j$. Thus the attacker has a winning strategy in the bisimulation game and $(X_1, \Delta) \not\sim (X_1', \Delta)$. □

Lemma 2. *If the formula C is true then $(X_1, \Delta) \sim (X_1', \Delta)$.*

Proof. Let us define sets $\mathcal{AS}_i$, corresponding to the assignments of variables from x_1, y_1 to x_i, y_i, $1 \le i \le n$, such that the formula

$$\exists x_{i+1} \forall y_{i+1} \dots \exists x_n \forall y_n .\ C_1 \wedge C_2 \wedge \dots \wedge C_k$$

is still true. For $i \in \{1, \dots, n\}$ let $\mathcal{AS}_i \stackrel{\text{def}}{=}$

$$\Big\{\widetilde{\alpha}_1\|\widetilde{\beta}_1\|\widetilde{\alpha}_2\|\widetilde{\beta}_2\|\ldots\|\widetilde{\alpha}_i\|\widetilde{\beta}_i \ \ |$$

such that for all j, $1 \le j \le i$, it holds that $\widetilde{\alpha}_j \in \{\alpha_j, \overline{\alpha_j}\}$ and $\widetilde{\beta}_j \in \{\beta_j, \overline{\beta_j}\}$, and under the assignment

$$x_j = \begin{cases} \mathtt{tt} & \text{if } \widetilde{\alpha}_j = \alpha_j \\ \mathtt{ff} & \text{if } \widetilde{\alpha}_j = \overline{\alpha_j} \end{cases} \quad \text{and} \quad y_j = \begin{cases} \mathtt{tt} & \text{if } \widetilde{\beta}_j = \beta_j \\ \mathtt{ff} & \text{if } \widetilde{\beta}_j = \overline{\beta_j} \end{cases}$$

the formula $\exists x_{i+1} \forall y_{i+1} \ldots \exists x_n \forall y_n.\ C_1 \wedge C_2 \wedge \ldots \wedge C_k$ is true $\Big\}$.

By definition $\mathcal{AS}_0 \overset{\text{def}}{=} \{\epsilon\}$. In particular, $\mathcal{AS}_n$ contains all the assignments for which the unquantified formula $C_1 \wedge C_2 \wedge \ldots \wedge C_k$ is true. The following relation is a strong bisimulation (assuming that the index i ranges over the set $\{1, \ldots, n\}$).

$$\begin{aligned}
&\{(X_i\|\gamma_i, X'_i\|\gamma_i) \mid \gamma_i \in \mathcal{AS}_{i-1}\} \cup \\
&\{(Y_i^{\mathtt{choice}}\|\gamma_i, Y_i^{\mathtt{tt}}\|\gamma_i) \mid \gamma_i\|\alpha_i\|\beta_i \in \mathcal{AS}_i \ \wedge\ \gamma_i\|\alpha_i\|\overline{\beta_i} \in \mathcal{AS}_i\} \cup \\
&\{(Y_i^{\mathtt{choice}}\|\gamma_i, Y_i^{\mathtt{ff}}\|\gamma_i) \mid \gamma_i\|\overline{\alpha_i}\|\beta_i \in \mathcal{AS}_i \ \wedge\ \gamma_i\|\overline{\alpha_i}\|\overline{\beta_i} \in \mathcal{AS}_i\} \cup \\
&\{(Z_i\|\gamma_i\|\alpha_i, Z'_i\|\gamma_i\|\alpha_i) \mid \gamma_i\|\alpha_i\|\beta_i \in \mathcal{AS}_i \ \wedge\ \gamma_i\|\alpha_i\|\overline{\beta_i} \in \mathcal{AS}_i\} \cup \\
&\{(Z_i\|\gamma_i\|\overline{\alpha_i}, Z'_i\|\gamma_i\|\overline{\alpha_i}) \mid \gamma_i\|\overline{\alpha_i}\|\beta_i \in \mathcal{AS}_i \ \wedge\ \gamma_i\|\overline{\alpha_i}\|\overline{\beta_i} \in \mathcal{AS}_i\} \cup \\
&\{(X_{n+1}\|\gamma_{n+1}, X'_{n+1}\|\gamma_{n+1}) \mid \gamma_{n+1} \in \mathcal{AS}_n\} \cup \\
&\{(Q_1\|Q_2\|\ldots\|Q_{k-1}\|Q_k\|\gamma_{n+1}, \gamma_{n+1}) \mid \gamma_{n+1} \in \mathcal{AS}_n\} \cup \\
&\{(E, E) \mid E \in \mathcal{P}\big(\mathcal{C}onst(\Delta)\big)\}
\end{aligned}$$

Since $\mathcal{AS}_0 = \{\epsilon\}$, we get that the pair (X_1, X'_1) is an element of this relation. Hence we proved that $(X_1, \Delta) \sim (X'_1, \Delta)$. □

Theorem 1. *Strong bisimilarity of BPP is PSPACE-hard.*

Proof. By Lemma 1 and Lemma 2. □

Remark 5. Notice that there are only finitely many reachable states from (X_1, Δ) and (X'_1, Δ). Hence (X_1, Δ) and (X'_1, Δ) are strongly regular processes.

Remark 6. Theorem 1 can be easily extended to 1-safe Petri nets where each transition has exactly one input place (for related results about 1-safe Petri nets see e.g. [17]). It is enough to introduce for each α_i, $\overline{\alpha_i}$, β_i and $\overline{\beta_i}$, $1 \le i \le n$, a new set of process constants $\{Q_1, \ldots, Q_k\}$ to ensure that in each reachable marking there is at most one token in every place.

3.3 Strong Bisimilarity of BPA

In this subsection we show that strong bisimilarity of BPA is PSPACE-hard. The proof is again by reduction from QSAT, using the main idea from Subsection 3.1. However, there is a substantial difference from the proof for BPP explained in the previous subsection. In case of BPP it is easier to check which clauses of a given boolean formula are satisfied because we have a parallel access to all process constants contained in the current state. This technique has to be modified to work for BPA since we have only a sequential access to the process constants contained in the current state, and there is no possibility of remembering any information in e.g. a finite-state control unit as in pushdown systems. Hence we have to encode the information about satisfied clauses in a unary way to achieve our result.

Let

$$C \equiv \exists x_1 \forall y_1 \exists x_2 \forall y_2 \dots \exists x_n \forall y_n .\ C_1 \wedge C_2 \wedge \dots \wedge C_k$$

be an instance of QSAT. Assume that $Q_1, \dots, Q_k$ are process constants. For each i, $1 \leq i \leq n$, let

α_i be a sequential composition $Q_{i_1}.Q_{i_2}.\cdots.Q_{i_\ell}$ such that
$1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and
$C_{i_1}, C_{i_2}, \dots, C_{i_\ell}$ are all the clauses where x_i occurs positively,

$\overline{\alpha_i}$ be a sequential composition $Q_{i_1}.Q_{i_2}.\cdots.Q_{i_\ell}$ such that
$1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and
$C_{i_1}, C_{i_2}, \dots, C_{i_\ell}$ are all the clauses where x_i occurs negatively,

β_i be a sequential composition $Q_{i_1}.Q_{i_2}.\cdots.Q_{i_\ell}$ such that
$1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and
$C_{i_1}, C_{i_2}, \dots, C_{i_\ell}$ are all the clauses where y_i occurs positively, and

$\overline{\beta_i}$ be a sequential composition $Q_{i_1}.Q_{i_2}.\cdots.Q_{i_\ell}$ such that
$1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and
$C_{i_1}, C_{i_2}, \dots, C_{i_\ell}$ are all the clauses where y_i occurs negatively.

Example 4. Let us consider a quantified formula

$$\exists x_1 \forall y_1 \exists x_2 \forall y_2 .\ (x_1 \vee \neg y_1 \vee y_2) \wedge (\neg x_1 \vee y_1 \vee y_2) \wedge (x_1 \vee y_1 \vee y_2 \vee \neg y_2)$$

where $n = 2$, $k = 3$, $C_1 = x_1 \vee \neg y_1 \vee y_2$, $C_2 = \neg x_1 \vee y_1 \vee y_2$ and $C_3 = x_1 \vee y_1 \vee y_2 \vee \neg y_2$. Then

$$\begin{array}{llll}
\alpha_1 = Q_1.Q_3 & \overline{\alpha_1} = Q_2 & \beta_1 = Q_2.Q_3 & \overline{\beta_1} = Q_1 \\
\alpha_2 = \epsilon & \overline{\alpha_2} = \epsilon & \beta_2 = Q_1.Q_2.Q_3 & \overline{\beta_2} = Q_3 .
\end{array}$$

Let $\mathcal{SF}(\gamma)$ be the set of all suffixes of a sequential composition $\gamma \in \mathcal{S}(\{Q_1, \ldots, Q_k\})$, i.e., $\mathcal{SF}(\gamma) \stackrel{\text{def}}{=} \{\gamma' \mid \exists \gamma'' \text{ such that } \gamma''.\gamma' = \gamma\}$. Let M be the least natural number such that $M \geq 2n{+}1$ and $M = 2^K$ for some natural number $K > 0$. Of course, $M > 1$.

We define processes $(X_1.S, \Delta)$ and $(X'_1.S, \Delta)$ where $\mathcal{C}onst(\Delta) \stackrel{\text{def}}{=}$

$$\begin{array}{l}
\{A_0, A_1, A_2, \ldots, A_{kK-1}\} \;\cup\; \{Q_1, \ldots, Q_k\} \;\cup \\
\{V_i^{\gamma}, V_i^{\gamma,\mathtt{choice}} \mid 1 \leq i \leq n \;\wedge\; \gamma \in \mathcal{SF}(\alpha_i) \cup \mathcal{SF}(\overline{\alpha_i})\} \;\cup \\
\{V_i^{\prime\gamma}, V_i^{\gamma,\mathtt{yes}}, V_i^{\gamma,\mathtt{no}} \mid 1 \leq i \leq n \;\wedge\; \gamma \in \mathcal{SF}(\alpha_i) \cup \mathcal{SF}(\overline{\alpha_i})\} \;\cup \\
\{W_i^{\gamma}, W_i^{\gamma,\mathtt{choice}} \mid 1 \leq i \leq n \;\wedge\; \gamma \in \mathcal{SF}(\beta_i) \cup \mathcal{SF}(\overline{\beta_i})\} \;\cup \\
\{W_i^{\prime\gamma}, W_i^{\gamma,\mathtt{yes}}, W_i^{\gamma,\mathtt{no}} \mid 1 \leq i \leq n \;\wedge\; \gamma \in \mathcal{SF}(\beta_i) \cup \mathcal{SF}(\overline{\beta_i})\} \;\cup \\
\{X_i, Y_i^{\mathtt{choice}}, Z_i, X'_i, Y_i^{\mathtt{tt}}, Y_i^{\mathtt{ff}}, Z'_i \mid 1 \leq i \leq n\} \;\cup\; \{X_{n+1}, X'_{n+1}, S\}
\end{array}$$

and $\mathcal{A}ct(\Delta) \stackrel{\text{def}}{=} \{a, c, \mathtt{tt}, \mathtt{ff}, \mathtt{yes}, \mathtt{no}, s\}$. The first part of the rewrite system Δ consists of the rewrite rules:

$$\begin{array}{ll}
A_0 \xrightarrow{c} \epsilon & \\
A_\ell \xrightarrow{c} A_{\ell-1}.A_{\ell-2}.\cdots.A_1.A_0 & \text{for all } \ell,\ 1 \leq \ell \leq kK-1 \\
& \\
Q_j \xrightarrow{c} A_{jK-1}.A_{jK-2}.\cdots.A_1.A_0 & \text{for all } j,\ 1 \leq j \leq k.
\end{array}$$

Remark 7. Notice that the size of the previously introduced rewrite rules is polynomial w.r.t. the size of the formula C. Moreover $A_\ell \xrightarrow{c^{2^\ell}} \epsilon$ for all ℓ, $0 \leq \ell \leq kK-1$, which implies by using the equation $2^{jK} = M^j$ that $Q_j \xrightarrow{c^{M^j}} \epsilon$ for all j, $1 \leq j \leq k$. Hence Q_j can perform exactly M^j transitions labelled by the "counting" action c and then disappears.

The intuition is that each clause C_j, $1 \leq j \leq k$, is coded by the process constant Q_j, which enables to perform exactly M^j of c actions. The key idea of our proof is then that the defender and the attacker will choose truth values for the variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, respectively. During this process some of the clauses $C_1, \ldots, C_k$ become satisfied, and the defender will have the possibility to add the corresponding process constants $Q_1, \ldots, Q_k$ to the current state.

Moreover, the defender will be able to select which of the process constants (corresponding to the satisfied clauses) appear in the current state in such a way that each of them appears there exactly once.

The following lemma shall be essential for proving our reduction correct.

Lemma 3. *Assume that M and k are constants introduced above, i.e., $M > 1$ and $k > 0$. Let a_j, $1 \leq j \leq k$, be natural numbers such that $0 \leq a_j \leq M-1$ for all j. The following two statements are equivalent:*

$$(i)\ \sum_{j=1}^{k} a_j M^j = \sum_{j=1}^{k} M^j \qquad (ii)\ a_j = 1 \textit{ for all } j,\ 1 \leq j \leq k.$$

Proof. By uniqueness of M-ary representations. Obviously (ii) implies (i). By induction on k we prove the other direction. If $k = 1$ then $a_1 M = M$ immediately gives that $a_1 = 1$. Let $\sum_{j=1}^{k+1} a_j M^j = \sum_{j=1}^{k+1} M^j$. Since $a_j \leq M-1$ for all j, $1 \leq j \leq k+1$, we get that

$$\sum_{j=1}^{k} a_j M^j \leq \sum_{j=1}^{k} (M-1)M^j = (M-1)M \cdot \sum_{j=0}^{k-1} M^j =$$

$$(M-1)M \cdot \frac{M^k - 1}{M-1} = M^{k+1} - M < M^{k+1}.$$

Let us now consider the equation

$$\sum_{j=1}^{k} a_j M^j + a_{k+1} M^{k+1} = \sum_{j=1}^{k} M^j + M^{k+1}.$$

The fact that $\sum_{j=1}^{k} a_j M^j < M^{k+1}$ gives that $a_{k+1} \geq 1$. On the other hand $\sum_{j=1}^{k} M^j = M \cdot \sum_{j=0}^{k-1} M^j = M \cdot \frac{M^k-1}{M-1} < M^{k+1}$, which implies that $a_{k+1} \leq 1$. Hence $a_{k+1} = 1$ and the following equation must be satisfied

$$\sum_{j=1}^{k} a_j M^j = \sum_{j=1}^{k} M^j.$$

By induction hypothesis $a_j = 1$ also for all j, $1 \leq j \leq k$. □

We continue with the definition of the set of rewrite rules Δ. For all i, $1 \leq i \leq n$, and $Q.\gamma \in \mathcal{SF}(\alpha_i) \cup \mathcal{SF}(\overline{\alpha_i})$ where $Q \in \{Q_1, \ldots, Q_k\}$ and $\gamma \in \mathcal{S}(\{Q_1, \ldots, Q_k\})$, Δ contains the rules:

$$\begin{array}{ll}
V_i^{Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\texttt{choice}} & \\
V_i^{Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\texttt{yes}} & V_i'^{Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\texttt{yes}} \\
V_i^{Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\texttt{no}} & V_i'^{Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\texttt{no}} \\
 & \\
V_i^{Q.\gamma,\texttt{choice}} \xrightarrow{\texttt{yes}} V_i^{\gamma}.Q & V_i^{Q.\gamma,\texttt{yes}} \xrightarrow{\texttt{yes}} V_i'^{\gamma}.Q \\
V_i^{Q.\gamma,\texttt{choice}} \xrightarrow{\texttt{no}} V_i^{\gamma} & V_i^{Q.\gamma,\texttt{no}} \xrightarrow{\texttt{no}} V_i'^{\gamma} \\
V_i^{Q.\gamma,\texttt{yes}} \xrightarrow{\texttt{no}} V_i^{\gamma} & V_i^{Q.\gamma,\texttt{no}} \xrightarrow{\texttt{yes}} V_i^{\gamma}.Q.
\end{array}$$

Similarly, for all i, $1 \leq i \leq n$, and $Q.\gamma \in \mathcal{SF}(\beta_i) \cup \mathcal{SF}(\overline{\beta_i})$ where $Q \in \{Q_1, \ldots, Q_k\}$ and $\gamma \in \mathcal{S}(\{Q_1, \ldots, Q_k\})$, Δ contains the rules:

$$\begin{array}{ll}
W_i^{Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\texttt{choice}} & \\
W_i^{Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\texttt{yes}} & W_i'^{Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\texttt{yes}} \\
W_i^{Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\texttt{no}} & W_i'^{Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\texttt{no}} \\
 & \\
W_i^{Q.\gamma,\texttt{choice}} \xrightarrow{\texttt{yes}} W_i^{\gamma}.Q & W_i^{Q.\gamma,\texttt{yes}} \xrightarrow{\texttt{yes}} W_i'^{\gamma}.Q \\
W_i^{Q.\gamma,\texttt{choice}} \xrightarrow{\texttt{no}} W_i^{\gamma} & W_i^{Q.\gamma,\texttt{no}} \xrightarrow{\texttt{no}} W_i'^{\gamma} \\
W_i^{Q.\gamma,\texttt{yes}} \xrightarrow{\texttt{no}} W_i^{\gamma} & W_i^{Q.\gamma,\texttt{no}} \xrightarrow{\texttt{yes}} W_i^{\gamma}.Q.
\end{array}$$

Assume now a bisimulation game starting from the pair $(V_i^{\alpha_i}, \Delta)$ and $(V_i'^{\alpha_i}, \Delta)$. As shown in Subsection 3.1, either in some round the states become syntactically equal, or the defender has the possibility to choose in the first round the next states (i) $V_i^{\alpha_i,\texttt{choice}}$ and $V_i^{\alpha_i,\texttt{yes}}$ or (ii) $V_i^{\alpha_i,\texttt{choice}}$ and $V_i^{\alpha_i,\texttt{no}}$. This means that in the next round a process constant Q such that $\alpha_i = Q.\alpha_i'$ for some α_i' is either (i) added to the current state or (ii) left out. Now the game continues either from (i) $V_i^{\alpha_i'}.Q$ and $V_i'^{\alpha_i'}.Q$ or from (ii) $V_i^{\alpha_i'}$ and $V_i'^{\alpha_i'}$. This repeats in similar fashion until the states $V_i^{\epsilon}.\gamma_i$ and $V_i'^{\epsilon}.\gamma_i$ are reached, such that γ_i is some subsequence of α_i (in a reverse order) and it was the defender who had the possibility to decide which of the process constants contained in α_i appear also in γ_i.

The same happens if we start playing the bisimulation game from $(V_i^{\overline{\alpha_i}}, \Delta)$ and $(V_i'^{\overline{\alpha_i}}, \Delta)$, or $(W_i^{\beta_i}, \Delta)$ and $(W_i'^{\beta_i}, \Delta)$, or $(W_i^{\overline{\beta_i}}, \Delta)$ and $(W_i'^{\overline{\beta_i}}, \Delta)$.

We finish the definition of Δ by adding the rules:

- for all i, $1 \leq i \leq n$:

$$\begin{array}{ll}
X_i \xrightarrow{a} Y_i^{\texttt{choice}} & \\
X_i \xrightarrow{a} Y_i^{\texttt{tt}} & X_i' \xrightarrow{a} Y_i^{\texttt{tt}} \\
X_i \xrightarrow{a} Y_i^{\texttt{ff}} & X_i' \xrightarrow{a} Y_i^{\texttt{ff}} \\
 & \\
Y_i^{\texttt{choice}} \xrightarrow{\texttt{tt}} V_i^{\alpha_i} & Y_i^{\texttt{tt}} \xrightarrow{\texttt{tt}} V_i'^{\alpha_i} \\
Y_i^{\texttt{choice}} \xrightarrow{\texttt{ff}} V_i^{\overline{\alpha_i}} & Y_i^{\texttt{ff}} \xrightarrow{\texttt{ff}} V_i'^{\overline{\alpha_i}} \\
Y_i^{\texttt{tt}} \xrightarrow{\texttt{ff}} V_i^{\overline{\alpha_i}} & Y_i^{\texttt{ff}} \xrightarrow{\texttt{tt}} V_i^{\alpha_i} \\
 & \\
V_i^{\epsilon} \xrightarrow{a} Z_i & V_i'^{\epsilon} \xrightarrow{a} Z_i' \\
 & \\
Z_i \xrightarrow{\texttt{tt}} W_i^{\beta_i} & Z_i' \xrightarrow{\texttt{tt}} W_i'^{\beta_i} \\
Z_i \xrightarrow{\texttt{ff}} W_i^{\overline{\beta_i}} & Z_i' \xrightarrow{\texttt{ff}} W_i'^{\overline{\beta_i}} \\
 & \\
W_i^{\epsilon} \xrightarrow{a} X_{i+1} & W_i'^{\epsilon} \xrightarrow{a} X_{i+1}'
\end{array}$$

– and

$$X_{n+1} \xrightarrow{a} Q_1.Q_2.\ldots.Q_{k-1}.Q_k.S \qquad X_{n+1}' \xrightarrow{a} \epsilon \qquad S \xrightarrow{s} S.$$

Lemma 4. *If $(X_1.S, \Delta) \sim (X_1'.S, \Delta)$ then the formula C is true.*

Proof. We show that $(X_1.S, \Delta) \not\sim (X_1'.S, \Delta)$ under the assumption that C is false. If C is false then C' defined by

$$C' \overset{\text{def}}{=} \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots \forall x_n \exists y_n.\ \neg(C_1 \wedge C_2 \wedge \ldots \wedge C_k)$$

is true and we claim that the attacker has a winning strategy in the bisimulation game starting from $(X_1.S, \Delta)$ and $(X_1'.S, \Delta)$. As mentioned in Subsection 3.1, in the first round the attacker is forced to perform the move $X_1.S \xrightarrow{a} Y_1^{\texttt{choice}}.S$. The defender can respond either by (i) $X_1'.S \xrightarrow{a} Y_1^{\texttt{tt}}.S$ (which corresponds to setting the variable x_1 to true) or by (ii) $X_1'.S \xrightarrow{a} Y_1^{\texttt{ff}}.S$ (which corresponds to setting the variable x_1 to false). In the second round the attacker performs the action (i) $\texttt{tt}$ or (ii) $\texttt{ff}$, and the defender must answer by the same action in the other process. Now the game continues from (i) $V_1^{\alpha_1}.S$ and $V_1'^{\alpha_1}.S$ or (ii) $V_1^{\overline{\alpha_1}}.S$ and $V_1'^{\overline{\alpha_1}}.S$. Within the next (i) $2 \cdot |\alpha_1|$ or (ii) $2 \cdot |\overline{\alpha_1}|$ rounds (where $|w|$ is the length of w) the defender has the possibility to choose some subsequence of (i) α_1 or (ii) $\overline{\alpha_1}$ and add it in a reverse order to the current state. Then the game continues either from (i) $V_1^{\epsilon}.\gamma_1.S$ and $V_1'^{\epsilon}.\gamma_1.S$ or (ii) $V_1^{\epsilon}.\overline{\gamma_1}.S$ and $V_1'^{\epsilon}.\overline{\gamma_1}.S$, such that (i) γ_1 is a subsequence (in a reverse order and chosen by the defender) of α_1 or (ii) $\overline{\gamma_1}$ is a subsequence (in a reverse order and chosen by the defender) of $\overline{\alpha_1}$. The players have only one possible continuation of the game by using the rewrite rules $V_1^{\epsilon} \xrightarrow{a} Z_1$ and $V_1'^{\epsilon} \xrightarrow{a} Z_1'$, thus reaching the states (i) $Z_1.\gamma_1.S$ and $Z_1'.\gamma_1.S$ or (ii) $Z_1.\overline{\gamma_1}.S$ and $Z_1'.\overline{\gamma_1}.S$.

Now, it is the attacker who has the possibility of making a choice between the rewrite rules $Z_1 \xrightarrow{\mathtt{tt}} W_1^{\beta_1}$ or $Z_1 \xrightarrow{\mathtt{ff}} W_1^{\overline{\beta_1}}$ in the first process. This corresponds to setting the variable y_1 to true or false. The defender can only imitate the same action by using the rules $Z_1' \xrightarrow{\mathtt{tt}} W_1'^{\beta_1}$ or $Z_1' \xrightarrow{\mathtt{ff}} W_1'^{\overline{\beta_1}}$ in the other process. From the current states starting with $W_1^{\beta_1}$ and $W_1'^{\beta_1}$, or $W_1^{\overline{\beta_1}}$ and $W_1'^{\overline{\beta_1}}$, the same happens as before: the defender has the possibility of choosing a subsequence δ_1 (in a reverse order) of β_1 or a subsequence $\overline{\delta_1}$ (in a reverse order) of $\overline{\beta_1}$. So precisely after $2 \cdot |\beta_1|$ or $2 \cdot |\overline{\beta_1}|$ rounds the following four possible pairs of states can be reached: (1) $W_1^{\epsilon}.\delta_1.\gamma_1.S$ and $W_1'^{\epsilon}.\delta_1.\gamma_1.S$, or (2) $W_1^{\epsilon}.\delta_1.\overline{\gamma_1}.S$ and $W_1'^{\epsilon}.\delta_1.\overline{\gamma_1}.S$, or (3) $W_1^{\epsilon}.\overline{\delta_1}.\gamma_1.S$ and $W_1'^{\epsilon}.\overline{\delta_1}.\gamma_1.S$, or (4) $W_1^{\epsilon}.\overline{\delta_1}.\overline{\gamma_1}.S$ and $W_1'^{\epsilon}.\overline{\delta_1}.\overline{\gamma_1}.S$. We have now only one possible continuation of the game in the next round, reaching the states (1) $X_2.\delta_1.\gamma_1.S$ and $X_2'.\delta_1.\gamma_1.S$, or (2) $X_2.\delta_1.\overline{\gamma_1}.S$ and $X_2'.\delta_1.\overline{\gamma_1}.S$, or (3) $X_2.\overline{\delta_1}.\gamma_1.S$ and $X_2'.\overline{\delta_1}.\gamma_1.S$, or (4) $X_2.\overline{\delta_1}.\overline{\gamma_1}.S$ and $X_2'.\overline{\delta_1}.\overline{\gamma_1}.S$.

We remind the reader of the fact that the defender had the possibility to set the variable x_1 to true or false, and the attacker decided on the truth value for the variable y_1. In the meantime, all the process constants from $\{Q_1, \ldots, Q_k\}$ corresponding to the clauses that became satisfied by this assignment could have been potentially added to the current state, but it was the defender who had the possibility to filter some of them out.

In the next rounds the same schema of the game repeats, until we reach the states $X_{n+1}.\omega.S$ and $X_{n+1}'.\omega.S$. The defender decides on the truth values for each of the variables $x_2, \ldots, x_n$, and the attacker has the possibility to respond by choosing the truth values for the variables $y_2, \ldots, y_n$. During this some of the clauses appear to be satisfied and ω consists of a selection (made by the defender) of process constants corresponding to these clauses.

Since we assume that the formula C' is true, the attacker can decide on the truth values for $y_1, \ldots, y_n$ in such a way that at least one of the clauses $C_1, \ldots, C_k$ is not satisfied. Let us suppose that it is C_m for some m, $1 \leq m \leq k$, that is not satisfied. Hence Q_m cannot appear in ω and the attacker has the following winning strategy. He plays $X_{n+1}.\omega.S \xrightarrow{a} Q_1.Q_2.\ldots.Q_{k-1}.Q_k.S.\omega.S$, to which the defender can only answer by $X_{n+1}'.\omega.S \xrightarrow{a} \omega.S$.

The state $Q_1.Q_2.\ldots.Q_{k-1}.Q_k.S.\omega.S$ can perform exactly $\sum_{j=1}^{k} M^j$ of actions c (see Remark 7) followed by an infinite sequence of actions s. On the other hand, $\omega.S$ can never perform exactly $\sum_{j=1}^{k} M^j$ of actions c and then the infinite sequence of actions s. This follows from the fact that Q_m does not appear in ω and from Lemma 3 — obviously, any process constant from $\{Q_1,\ldots,Q_k\}$ can occur at most $2n$ times in ω ($2n \leq M-1$), which justifies the assumption of Lemma 3.

Hence the attacker has a winning strategy which implies that $(X_1.S,\Delta) \not\sim (X'_1.S,\Delta)$. □

Lemma 5. *If the formula C is true then $(X_1.S,\Delta) \sim (X'_1.S,\Delta)$.*

Proof. Assume a bisimulation game starting from the pair $(X_1.S,\Delta)$ and $(X'_1.S,\Delta)$. We show that the defender has a winning strategy. As mentioned in Subsection 3.1 and in the proof above, the attacker is forced to play according to a strictly defined strategy, otherwise the defender can make the resulting processes immediately syntactically equal and hence bisimilar. As shown before the defender can make the choices between setting the variables $x_1,\ldots,x_n$ to true or false, whereas the attacker can decide on truth values for $y_1,\ldots,y_n$. Thus the defender can play the bisimulation game such that finally every clause $C_1,\ldots,C_k$ in C is satisfied. The defender has the possibility to add the corresponding process constants $Q_1,\ldots,Q_k$ to the current state in such a way that when reaching the states $X_{n+1}.\omega.S$ and $X'_{n+1}.\omega.S$, the sequential composition ω contains every Q_j exactly once for each j, $1 \leq j \leq k$. This can be easily achieved by following the strategy: "add Q_j to the current state if and only if it is not already present there". After performing the moves $X_{n+1}.\omega.S \xrightarrow{a} Q_1.Q_2.\ldots.Q_{k-1}.Q_k.S.\omega.S$ and $X'_{n+1}.\omega.S \xrightarrow{a} \omega.S$, the defender wins since $S.\omega.S \sim S$ and both $Q_1.Q_2.\ldots.Q_{k-1}.Q_k$ and ω can perform the same number of actions c and then become ϵ. Hence $(X_1.S,\Delta) \sim (X'_1.S,\Delta)$. □

Theorem 2. *Strong bisimilarity of BPA is PSPACE-hard.*

Proof. By Lemma 4 and Lemma 5. □

Remark 8. Notice that there are only finitely many reachable states from $(X_1.S,\Delta)$ and $(X'_1.S,\Delta)$. Hence $(X_1.S,\Delta)$ and $(X'_1.S,\Delta)$ are strongly regular processes.

4 Lower Bounds for Strong Regularity

The idea to reduce bisimilarity to regularity first appeared in the literature due to Mayr [20]. He showed a technique for reducing weak bisimilarity of regular BPP to weak regularity of BPP. However, in his reduction τ actions are used. Building upon Mayr's approach we provide a polynomial time reduction from strong bisimilarity of regular BPP (BPA) to strong regularity of BPP (BPA).

4.1 Strong Regularity of BPP

Theorem 3 (Reduction from bisimilarity to regularity). *Let (P_1, Δ) and (P_2, Δ) be strongly regular BPP processes. We can construct in polynomial time a BPP process (P, Δ') such that*

$$(P_1, \Delta) \sim (P_2, \Delta) \quad \textit{if and only if} \quad (P, \Delta') \textit{ is strongly regular.}$$

Proof. Assume that (P_1, Δ) and (P_2, Δ) are strongly regular. We construct a BPP process (P, Δ') with

$$\mathcal{C}onst(\Delta') \stackrel{\text{def}}{=} \mathcal{C}onst(\Delta) \cup \{X, A, A_c, B_c, P_1', P_2'\}$$

and

$$\mathcal{A}ct(\Delta') \stackrel{\text{def}}{=} \mathcal{A}ct(\Delta) \cup \{a, b\}$$

where $X, A, A_c, B_c, P_1', P_2'$ are new process constants and a, b are new actions. We define $\Delta' \stackrel{\text{def}}{=} \Delta \cup \Delta^1 \cup \Delta^2$ where the set of rewrite rules Δ^1 is given by

$$\begin{array}{llll} X \xrightarrow{b} X\|A & A \xrightarrow{a} \epsilon & A_c \xrightarrow{a} A_c & B_c \xrightarrow{b} B_c \\ X \xrightarrow{a} P_1'\|A_c & X \xrightarrow{a} P_1\|A_c & P_1' \xrightarrow{a} P_1 & \end{array}$$

and Δ^2 is given by

$$X \xrightarrow{a} P_2'\|A_c \qquad X \xrightarrow{a} P_2\|A_c \qquad P_2' \xrightarrow{a} P_2.$$

Let $P \stackrel{\text{def}}{=} X\|B_c$.

Lemma 6. *If $(P_1, \Delta) \not\sim (P_2, \Delta)$ then (P, Δ') is not strongly regular.*

Proof. Let $(P_1, \Delta) \not\sim (P_2, \Delta)$. For simplicity (and without loss of generality) we assume that $P_1 \not\sim \epsilon$ and $P_2 \not\sim \epsilon$. We demonstrate that there are infinitely many strongly nonbisimilar states reachable from (P, Δ').

Let us consider an infinite number of states of the form $X\|A^i\|B_c$ for any natural number i. Of course $P \longrightarrow^* X\|A^i\|B_c$ and we claim that $(X\|A^i\|B_c, \Delta') \not\sim (X\|A^j\|B_c, \Delta')$ for any $i \neq j$. Without loss of generality assume that $i < j$. The attacker has the following winning strategy (playing only in the second process — see Figure 6).

$$\begin{tikzcd}
X\|A^i\|B_c \arrow[d, "a^{i'}"] \arrow[r, phantom, "\sim^?"] & & X\|A^j\|B_c \arrow[d, "a^{i'}"] \\
P_1'\|A^{i-i'+1}\|A_c\|B_c \arrow[d, "a^{j-i'}"] & & X\|A^{j-i'}\|B_c \arrow[d, "a^{j-i'}"] \\
P_1'\|A^{i''}\|A_c\|B_c \arrow[d, "a"] & & X\|B_c \arrow[d, "a"] \\
P_1\|A^{i''}\|A_c\|B_c \arrow[r, phantom, "\not\sim"] & & P_2\|A_c\|B_c
\end{tikzcd}$$

Fig. 6 Winning strategy for the attacker $(i < j)$

He performs a sequence of j actions a from $X\|A^j\|B_c$, thus reaching a state $X\|B_c$. The defender playing from $X\|A^i\|B_c$ cannot do this sequence of a-actions without using some rule for X. This is because $B_c \overset{a}{\not\longrightarrow}$ and A^i can perform at most i a-actions $(i < j)$. As we assume that $P_1 \not\sim \epsilon$ and $P_2 \not\sim \epsilon$, process constants P_1 and P_2 cannot appear in the defender's process during the first j rounds, otherwise he loses immediately. So the defender has to make a choice between the rules $X \xrightarrow{a} P_1'\|A_c$ and $X \xrightarrow{a} P_2'\|A_c$ sometime within the first j moves (let us say in round i' where $i' \leq i+1$). Assume that the defender chooses $X \xrightarrow{a} P_1'\|A_c$ — the other case is symmetric. Now, the defender must perform $j - i'$ of a-actions by using the rules $A_c \xrightarrow{a} A_c$ or $A \xrightarrow{a} \epsilon$.

After j rounds the resulting states are $P_1'\|A^{i''}\|A_c\|B_c$ for $i'' \leq i - i' + 1$, and $X\|B_c$. The attacker wins by performing the move $X\|B_c \xrightarrow{a} P_2\|A_c\|B_c$. Again, since we assume that $P_2 \not\sim \epsilon$ the defender has to answer with $P_1'\|A^{i''}\|A_c\|B_c \xrightarrow{a} P_1\|A^{i''}\|A_c\|B_c$. The attacker has now a winning strategy from $P_1\|A^{i''}\|A_c\|B_c$ and $P_2\|A_c\|B_c$: the fact that $P_1 \not\sim P_2$ and that the actions a and b are fresh ones implies that $P_1\|A^{i''}\|A_c\|B_c \not\sim P_2\|A_c\|B_c$. □

Lemma 7. *If* $(P_1, \Delta) \sim (P_2, \Delta)$ *then* (P, Δ') *is strongly regular.*

Proof. Assume that $(P_1, \Delta) \sim (P_2, \Delta)$ which implies that $(P, \Delta') \sim (P, \Delta'')$ where $\Delta'' \overset{\text{def}}{=} \Delta' \smallsetminus \Delta^2$ (strong bisimilarity is a congruence w.r.t. the parallel operator). It is enough to show that (P, Δ'') is strongly regular. Observe that $(A^i\|A_c, \Delta'') \sim (A_c, \Delta'')$ for any i such that $0 \leq i$. Then also $(P_1\|A^i\|A_c\|B_c, \Delta'') \sim (P_1\|A_c\|B_c, \Delta'')$ and $(P_1'\|A^i\|A_c\|B_c, \Delta'') \sim (P_1'\|A_c\|B_c, \Delta'')$ for any i such that $0 \leq i$. This implies that

$$(X\|A^i\|B_c, \Delta'') \sim (P_1'\|A_c\|B_c, \Delta'') \tag{1}$$

for any i such that $0 \leq i$. Since (P_1, Δ) is a strongly regular process then $(P_1' \| A_c \| B_c, \Delta'')$ is also strongly regular. This by using (1) in particular gives that $(X \| A^0 \| B_c, \Delta'') = (X \| B_c, \Delta'') = (P, \Delta'')$ is strongly regular. □

Theorem 3 follows from Lemma 6 and Lemma 7. □

Theorem 4. *Strong regularity of BPP is PSPACE-hard.*

Proof. By Theorem 1, Remark 5 and Theorem 3. □

4.2 Strong Regularity of BPA

Theorem 5 (Reduction from bisimilarity to regularity). *Let (P_1, Δ) and (P_2, Δ) be strongly regular BPA processes. We can construct in polynomial time a BPA process (P, Δ') such that*

$$(P_1, \Delta) \sim (P_2, \Delta) \quad \textit{if and only if} \quad (P, \Delta') \textit{ is strongly regular.}$$

Proof. Assume that (P_1, Δ) and (P_2, Δ) are strongly regular BPA processes. We construct a BPA process (P, Δ') with

$$\mathcal{C}onst(\Delta') \overset{\text{def}}{=} \mathcal{C}onst(\Delta) \cup \{X, A, C, S, P_1', P_2'\}$$

and

$$\mathcal{A}ct(\Delta') \overset{\text{def}}{=} \mathcal{A}ct(\Delta) \cup \{a, s\}$$

where X, A, C, S, P_1', P_2' are new process constants and a, s are new actions. We define $\Delta' \overset{\text{def}}{=} \Delta \cup \Delta^1 \cup \Delta^2$ where the set of transition rules Δ^1 is given by

$$\begin{array}{llll}
X \xrightarrow{a} X.A & X \xrightarrow{a} \epsilon & A \xrightarrow{a} \epsilon & S \xrightarrow{s} S \\
 & & & \\
A \xrightarrow{a} P_1'.S & A \xrightarrow{a} P_1.S & P_1' \xrightarrow{a} P_1' & P_1' \xrightarrow{a} P_1 \\
X \xrightarrow{a} P_1'.S & X \xrightarrow{a} P_1.S & & \\
C \xrightarrow{a} P_1'.S & C \xrightarrow{a} P_1.S & &
\end{array}$$

and Δ^2 is given by

$$\begin{array}{llll}
A \xrightarrow{a} P_2'.S & A \xrightarrow{a} P_2.S & P_2' \xrightarrow{a} P_2' & P_2' \xrightarrow{a} P_2 \\
X \xrightarrow{a} P_2'.S & X \xrightarrow{a} P_2.S & & \\
C \xrightarrow{a} P_2'.S & C \xrightarrow{a} P_2.S. & &
\end{array}$$

Let $P \overset{\text{def}}{=} X.C$.

Lemma 8. *If* $(P_1, \Delta) \not\sim (P_2, \Delta)$ *then* (P, Δ') *is not strongly regular.*

Proof. Let $(P_1, \Delta) \not\sim (P_2, \Delta)$. Without loss of generality assume that $P_1 \not\sim \epsilon$ and $P_2 \not\sim \epsilon$. We show that there are infinitely many strongly nonbisimilar states reachable from (P, Δ'). Consider the states of the form $A^i.C$ for any natural number i. Of course, $P \longrightarrow^* A^i.C$. In order to prove that (P, Δ') is not strongly regular, it is enough to show that $(A^i.C, \Delta') \not\sim (A^j.C, \Delta')$ for any $i < j$. Next paragraph describes attacker's winning strategy from the states $A^i.C$ and $A^j.C$.

The attacker is playing only in the second process $A^j.C$. He performs a sequence of actions a of the length j by using the rule $A \xrightarrow{a} \epsilon$ and reaches the state C. By examining all possible moves of the defender from the process $A^i.C$, we get that a different rule from $A \xrightarrow{a} \epsilon$ must be used within the first j rounds since $i < j$. Using the assumption that $P_1 \not\sim \epsilon$ and $P_2 \not\sim \epsilon$ we derive that only four types of states (reachable by the defender from $A^i.C$ after j rounds) must be considered. Namely

- $P'_1.S.A^{i'}.C$ for some i', $0 \leq i' < i$
- $P'_2.S.A^{i'}.C$ for some i', $0 \leq i' < i$
- $P'_1.S$ or
- $P'_2.S$.

Notice that $S.\alpha \sim S$ for any process expression α, which in particular means that $P'_1.S.A^{i'}.C \sim P'_1.S$ and $P'_2.S.A^{i'}.C \sim P'_2.S$. Hence it is enough to examine only the states $P'_1.S$ and $P'_2.S$. Let us consider the bisimulation game continuing from the pair of states $P'_1.S$ and C — the other case (from states $P'_2.S$ and C) is symmetric. The attacker wins by performing the move $C \xrightarrow{a} P_2.S$. The defender has to answer by $P'_1.S \xrightarrow{a} P_1.S$ since the move $P'_1.S \xrightarrow{a} P'_1.S$ means an immediate loss for the defender (we assume that $P_1 \not\sim \epsilon$ and $P_2 \not\sim \epsilon$). Now the resulting states after $j + 1$ rounds are $P_1.S$ and $P_2.S$. The attacker has a winning strategy from this pair by our assumption that $P_1 \not\sim P_2$ and by the fact that $S \xrightarrow{s} S$ is the only rewrite rule for S and the action s is a fresh one. □

Lemma 9. *If* $(P_1, \Delta) \sim (P_2, \Delta)$ *then* (P, Δ') *is strongly regular.*

Proof. Assume that $(P_1, \Delta) \sim (P_2, \Delta)$ which implies that $(P, \Delta') \sim (P, \Delta'')$ where $\Delta'' \stackrel{\text{def}}{=} \Delta' \smallsetminus \Delta^2$ (strong bisimilarity is a congruence w.r.t. the sequential operator[2]). It is enough to show that (P, Δ'')

[2] Under the usual assumption that Δ' contains no deadlocks — see Subsection 2.3. This is obviously the case for processes from Theorem 2.

is strongly regular. In what follows we often use (without explicitly mentioning it) the fact that $S.\alpha \sim S$ for any process expression α.

Let us first observe that $(C, \Delta'') \sim (P_1'.S, \Delta'')$ which implies that $(A^i.C, \Delta'') \sim (P_1'.S, \Delta'')$ for all i, $i \geq 0$. Using this fact we get that

$$(X.A^i.C, \Delta'') \sim (P_1'.S, \Delta'') \tag{2}$$

for all i, $i \geq 0$. Recall that (P_1, Δ) is a strongly regular process. It is easily seen now that $(P_1'.S, \Delta'')$ is also a strongly regular process. Hence (2) in particular gives that $(X.A^0.C, \Delta'') = (X.C, \Delta'') = (P, \Delta'')$ is strongly regular. □

Theorem 5 follows from Lemma 8 and Lemma 9. □

Theorem 6. *Strong regularity of BPA is PSPACE-hard.*

Proof. By Theorem 2, Remark 8 and Theorem 5. □

4.3 Strong Regularity of Normed BPA and BPP

This subsection aims to show that under the condition of normedness, strong regularity of BPA and BPP are complete problems for nondeterministic logarithmic space (NL). Kučera in [18] argues that strong regularity of BPA and BPP is decidable in polynomial time but it is easy to see that a test whether a BPA (BPP) process contains an *accessible* and *growing* process constant (a condition equivalent to regularity) can be performed even in nondeterministic logarithmic space. In [19] the previous results are extended to normed PA processes, and again it can be shown that the decision algorithm for strong regularity of normed PA can be implemented in NL.

Theorem 7. *Strong regularity of normed BPA and normed BPP is NL-hard.*

Proof. In order to prove NL-hardness, we reduce the reachability problem for acyclic directed graphs (for NL-completeness see e.g. [24]) to strong regularity checking of normed BPA (BPP).

Problem: Reachability for acyclic directed graphs

Instance: An acyclic directed graph $G = (V, E)$ such that $V = \{v_1, \ldots, v_n\}$, $1 \leq n$, and $E \subseteq V \times V$.

Question: Is it the case that $(v_1, v_n) \in E^*$ where E^* is a reflexive and transitive closure of E?

Let $G = (V, E)$ be an instance of the reachability problem for acyclic directed graphs. For $u \in V$ we define its out-degree by

$$u^+ \overset{\text{def}}{=} |\{v \in V \mid (u,v) \in E\}|$$

and without loss of generality assume that $v_1^+, v_n^+ > 0$. Let Δ be a finite-state system where $\mathcal{C}onst(\Delta) \overset{\text{def}}{=} \{X_u \mid u \in V \wedge u^+ > 0\} \cup \{X\}$, $\mathcal{A}ct(\Delta) \overset{\text{def}}{=} \{a\}$ and

$$\begin{aligned}\Delta \overset{\text{def}}{=} &\{X_u \xrightarrow{a} X_v \mid (u,v) \in E \ \wedge \ v^+ > 0\} \cup \\ &\{X_u \xrightarrow{a} \epsilon \mid (u,v) \in E \ \wedge \ v^+ = 0\} \cup \\ &\{X_{v_n} \xrightarrow{a} X\}.\end{aligned}$$

Obviously, $(v_1, v_n) \in E^*$ if and only if $X_{v_1} \longrightarrow^* X$. Let us define a BPA system

$$\Delta_1 \overset{\text{def}}{=} \Delta \ \cup \ \{X \xrightarrow{a} X.X, \ X \xrightarrow{a} \epsilon\}$$

and a BPP system

$$\Delta_2 \overset{\text{def}}{=} \Delta \ \cup \ \{X \xrightarrow{a} X \| X, \ X \xrightarrow{a} \epsilon\}.$$

It is an easy observation that (X, Δ_1) and (X, Δ_2) are normed and nonregular processes. This implies that (X_{v_1}, Δ_1) and (X_{v_1}, Δ_2) are also normed processes (G is acyclic) such that $(v_1, v_n) \in E^*$ iff (X_{v_1}, Δ_1) is not strongly regular, and $(v_1, v_n) \in E^*$ iff (X_{v_1}, Δ_2) is not strongly regular. Recall that NL=co-NL (see e.g. [24]). Hence the problems of strong regularity for normed BPA and BPP are NL-hard (our reductions are obviously in logarithmic space). □

5 Conclusion

We proved that strong bisimilarity and regularity problems for BPA and BPP are PSPACE-hard. Our proofs are by reduction from the problem of quantified satisfiability (QSAT). The general idea (Subsection 3.1) for generating quantified instances of QSAT applies to both BPA and BPP. However, the proofs for BPA and BPP differ in checking that all clauses are indeed satisfied. This is due to the fact that BPP enables parallel access to all process constants contained in the current state whereas BPA does not.

We expect that the technique for generation of QSAT instances can be used in similar contexts, e.g. for showing lower bounds of weak bisimilarity.

	strong bisimilarity	strong regularity
BPA	$\in$ 2-EXPTIME [4] **PSPACE-hard**	$\in$ 2-EXPTIME [5,4] **PSPACE-hard**
normed BPA	$\in$ P [11] P-hard [1]	$\in$ NL [18] **NL-hard**
BPP	decidable [8] **PSPACE-hard**	decidable [16] **PSPACE-hard**
normed BPP	$\in$ P [12] P-hard [1]	$\in$ NL [18] **NL-hard**
PA	? **PSPACE-hard**	? **PSPACE-hard**
normed PA	2-NEXPTIME [10] P-hard [1]	$\in$ NL [19] **NL-hard**

Fig. 7 Summary of results for BPA, BPP and PA

An interesting observation is that only one unnormed process constant (namely S) is used in the hardness proofs for BPA. In contrast, the hardness proofs for strong bisimilarity of BPP (see [20] and Subsection 3.2) require a polynomial number of unnormed process constants.

In Figure 7 we present the state of the art of strong bisimilarity and regularity checking for BPA, BPP and PA and their normed subclasses. Results proved in this paper are in boldface. Obviously, all the lower bounds for BPA and BPP apply also to PA.

Acknowledgements. I would like to thank my advisor Mogens Nielsen for his kind supervision and useful suggestions, and Pawel Sobocinski and Jan Strejček for their remarks on some earlier drafts. I also thank the anonymous referees of STACS'02 and ICALP'02 for useful comments and suggestions.

References

1. J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4(6A):638–648, 1992.
2. J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77–121, 1985.
3. 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.

4. O. Burkart, D. Caucal, and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proceedings of MFCS'95*, volume 969 of *LNCS*, pages 423–433. Springer-Verlag, 1995.
5. O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proceedings of CONCUR'96*, volume 1119 of *LNCS*, pages 247–262. Springer-Verlag, 1996.
6. D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106(1):61–86, 1992.
7. S. Christensen. *Decidability and Decomposition in Process Algebras.* PhD thesis, The University of Edinburgh, 1993.
8. S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for basic parallel processes. In *Proceedings of CONCUR'93*, volume 715 of *LNCS*, pages 143–157. Springer-Verlag, 1993.
9. S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.
10. Y. Hirshfeld and M. Jerrum. Bisimulation equivalence is decidable for normed process algebra. In *Automata, Languages and Programming, 26th International Colloquium (ICALP'99)*, volume 1644 of *LNCS*, pages 412–421. Springer-Verlag, 1999.
11. 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.
12. 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.
13. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.
14. H. Hüttel. Undecidable equivalences for basic parallel processes. In *Proceedings of TACS'94*, volume 789 of *LNCS*, pages 454–464. Springer-Verlag, 1994.
15. P. Jančar. High undecidability of weak bisimilarity for Petri nets. In *Proceedings of Colloquium on Trees in Algebra and Programming (CAAP'95)*, volume 915 of *LNCS*, pages 349–363. Springer-Verlag, 1995.
16. P. Jančar and J. Esparza. Deciding finiteness of Petri nets up to bisimulation. In *Proceedings of ICALP'96*, volume 1099 of *LNCS*, pages 478–489. Springer-Verlag, 1996.
17. Lalita Jategaonkar and Albert R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, 1996.
18. A. Kučera. Regularity is decidable for normed BPA and normed BPP processes in polynomial time. In *Proceedings of SOFSEM'96*, volume 1175 of *LNCS*, pages 377–384. Springer-Verlag, 1996.
19. A. Kučera. Regularity is decidable for normed PA processes in polynomial time. In *Proceedings of FST&TCS'96*, volume 1180 of *LNCS*, pages 111–122. Springer-Verlag, 1996.
20. R. Mayr. On the complexity of bisimulation problems for basic parallel processes. In *Proceedings of 27st International Colloquium on Automata, Languages and Programming (ICALP'00)*, volume 1853 of *LNCS*, pages 329–341. Springer-Verlag, 2000.

21. R. Mayr. Process rewrite systems. *Information and Computation*, 156(1):264–286, 2000.
22. R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
23. F. Moller. Infinite results. In *Proceedings of CONCUR'96*, volume 1119 of *LNCS*, pages 195–216. Springer-Verlag, 1996.
24. Ch.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, New York, 1994.
25. D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings* 5^{th} *GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.
26. G. Plotkin. A structural approach to operational semantics. Technical Report Daimi FN-19, Department of Computer Science, University of Aarhus, 1981.
27. J. Srba. Basic process algebra with deadlocking states. *Theoretical Computer Science*, 266(1–2):605–630, 2001.
28. J. Srba. Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard. In *Proceedings of 19th International Symposium on Theoretical Aspects of Computer Science (STACS'02)*, volume 2285 of *LNCS*, pages 535–546. Springer-Verlag, 2002.
29. J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proceedings of 29th International Colloquium on Automata, Languages and Programming (ICALP'02)*, LNCS. Springer-Verlag, 2002. To appear.
30. 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.
31. 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.
32. R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions.* PhD thesis, CWI/Vrije Universiteit, 1990.
33. R.J. van Glabbeek. The linear time—branching time spectrum. In *Proceedings of CONCUR'90*, volume 458 of *LNCS*, pages 278–297. Springer-Verlag, 1990.