Strong Bisimilarity and Regularity of Basic Process Algebra Is PSPACE-Hard

Jiří Srba*

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

Abstract. Strong bisimilarity and regularity checking problems of Basic Process Algebra (BPA) are decidable, with the complexity upper bounds 2-EXPTIME. On the other hand, no lower bounds were known. In this paper we demonstrate PSPACE-hardness of these problems.

1 Introduction

Despite several positive decidability results for process algebras which generate infinite-state transition systems (see [3]), the complexity issues often remain open. Two main problems studied are *equivalence checking* and *model checking*. In this paper we investigate the first sort of problems, with a special focus on *strong bisimulation equivalence* of Basic Process Algebra (BPA), sometimes called also context-free processes.

BPA 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. While it is a well known fact that language equivalence is undecidable for context-free grammars (BPA processes), strong bisimilarity is decidable [7] and the best known upper bound is 2-EXPTIME [4] so far. Language equivalence of context-free grammars with no redundant nonterminals (unnormed variables in the terminology of process algebra) is still undecidable, whereas strong bisimilarity is decidable even in polynomial time [8]. In fact the strong bisimilarity checking problem for this subclass, called *normed* BPA, is P-complete (for P-hardness see [1]).

We prove in this paper that the complexity of strong bisimilarity checking of BPA is indeed different (unless P=PSPACE) from the case of normed BPA by giving the first nontrivial lower bound for unnormed BPA. We show that strong bisimilarity of BPA is PSPACE-hard by describing a polynomial time reduction from the problem of quantified boolean formula.

* 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.

Another interesting problem that has attracted much attention is that of *regularity checking*. The question is whether a given BPA process is strongly bisimilar to some finite-state process. Strong regularity checking is known to be decidable in 2-EXPTIME for BPA [5, 4] and in polynomial time for normed BPA [10]. As far as we know, no lower bound was given for strong regularity of BPA. We describe a polynomial time reduction from bisimilarity of strongly regular BPA processes to the strong regularity problem of BPA. This, using the fact that the processes in the PSPACE-hardness proof of strong bisimilarity of BPA are strongly regular, implies a PSPACE lower bound for strong regularity checking of BPA processes.

Finally, it is worth mentioning what is known about strong bisimilarity and regularity problems of Basic Parallel Processes (BPP). BPP are a parallel analogue of BPA, where the associative operator of sequential composition is replaced with an associative and commutative one. The strong bisimilarity checking problem for BPP is known to be decidable [6], but no primitive recursive upper bound has been given so far. Only recently Mayr proved the problem to be co-NP-hard [11]. We improved the result to PSPACE [15] by a reduction from quantified boolean formula, using similar ideas as in this paper. However, in the 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 had to be substantially modified to work for BPA as well, since we have only sequential access to the process constants contained in a 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.

Similarly, the strong regularity checking problem for BPP is decidable [9], but no primitive recursive upper bound is known. Mayr proved the problem to be co-NP-hard [11] and we recently improved the result to PSPACE [15].

Note: full and extended version of this paper appears as [16].

2 Basic Definitions

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

Let $\mathcal{A}ct$ and $\mathcal{C}onst$ be countable sets of *actions* and *process constants*, respectively, such that $\mathcal{A}ct \cap \mathcal{C}onst = \emptyset$. We define the class of BPA *process expressions* $\mathcal{E}^{\mathcal{C}onst}$ *over* $\mathcal{C}onst$ by $E ::= \epsilon \mid X \mid E.E$ where 'ϵ' is the *empty process* and X ranges over $\mathcal{C}onst$. The operator '.' stands for a *sequential 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, and 'ϵ' is a unit for '.'.

A BPA *process rewrite system* (or a (1,S)-PRS in the terminology of [12]) is a finite set $\Delta \subseteq \mathcal{C}onst \times \mathcal{A}ct \times \mathcal{E}^{\mathcal{C}onst}$ of *rewrite rules*, written $X \stackrel{a}{\longrightarrow} E$ for

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

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

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

As usual we extend the transition relation to the elements of $\mathcal{A}ct^*$. We also write $E \longrightarrow^* E'$, whenever $E \xrightarrow{w} E'$ for some $w \in \mathcal{A}ct^*$. A state E' is *reachable from a state* E iff $E \longrightarrow^* E'$. We write $E \stackrel{a}{\not\longrightarrow}$ whenever there is no F such that $E \xrightarrow{a} F$, and $E \not\longrightarrow$ whenever $E \stackrel{a}{\not\longrightarrow}$ for all $a \in \mathcal{A}ct$.

A BPA *process* is a pair (P, Δ), where Δ is a BPA process rewrite system and $P \in \mathcal{E}^{\mathcal{C}onst(\Delta)}$ is a BPA process expression. *States* of (P, Δ) are the states of the corresponding transition system. We say that a state E is *reachable* in (P, Δ) iff $P \longrightarrow^* E$. Whenever (P, Δ) has only finitely many reachable states, we call it a *finite-state process*. A process (P, Δ) is *normed* iff from every reachable state E in (P, Δ) there is a terminating computation, i.e., $E \longrightarrow^* \epsilon$.

Let Δ be a process rewrite system. A binary relation $R \subseteq \mathcal{E}^{\mathcal{C}onst(\Delta)} \times \mathcal{E}^{\mathcal{C}onst(\Delta)}$ over process expressions is a *strong bisimulation* iff whenever $(E, F) \in R$ then for each $a \in \mathcal{A}ct(\Delta)$: if $E \xrightarrow{a} E'$ then $F \xrightarrow{a} F'$ for some F' such that $(E', F') \in R$; and if $F \xrightarrow{a} F'$ then $E \xrightarrow{a} E'$ for some E' such that $(E', F') \in R$.

Processes (P_1, Δ) and (P_2, Δ) are *strongly bisimilar*, and we write $(P_1, \Delta) \sim (P_2, \Delta)$, iff there is a strong bisimulation R such that $(P_1, P_2) \in R$. Given a pair of processes (P_1, Δ_1) and (P_2, Δ_2) such that $\Delta_1 \neq \Delta_2$, we write $(P_1, \Delta_1) \sim (P_2, \Delta_2)$ iff $(P_1, \Delta) \sim (P_2, \Delta)$ where Δ is a disjoint union of Δ_1 and Δ_2.

We say that a process (P, Δ) is *strongly regular* iff there exists some finite-state process bisimilar to (P, Δ).

Bisimulation equivalence has an elegant characterisation in terms of *bisimulation games* [19, 17]. A bisimulation game on a pair of processes (P_1, Δ) and (P_2, Δ) is a two-player game of an 'attacker' and a 'defender'. The game is played in rounds. In each round the attacker chooses one of the processes and makes an $\xrightarrow{a}$-move for some $a \in \mathcal{A}ct(\Delta)$, and the defender must respond by making an $\xrightarrow{a}$-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. Processes (P_1, Δ) and (P_2, Δ) are strongly bisimilar iff the defender has a winning strategy (and non-bisimilar iff the attacker has a winning strategy).

3 The Main Idea

We try to explain here the main idea of the PSPACE-hardness proof given in Section 4. Our aim is to make the rewrite rules defined in Section 4 more readable

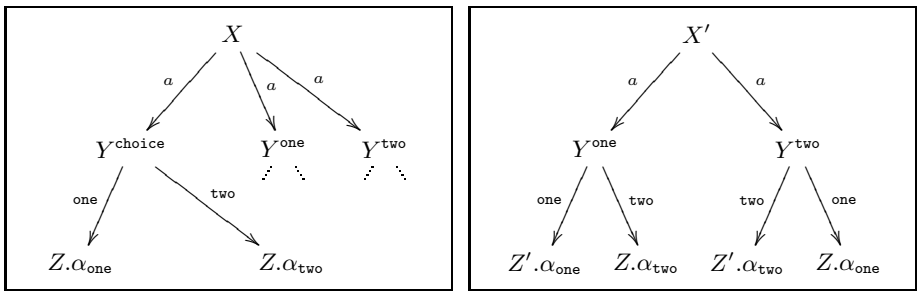

Fig. 1. Processes (X, Δ) and (X', Δ)

by demonstrating a general pattern used heavily (with small modifications) later on. Let us consider the following BPA system Δ where $\alpha_{\mathtt{one}}$ and $\alpha_{\mathtt{two}}$ are some sequences of process constants.

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

Transition systems generated by processes (X, Δ) and (X', Δ) are depicted in Figure 1. The intuition behind the construction can be nicely explained in terms of bisimulation games. Consider a bisimulation game starting from X and X'.

The attacker is forced to make the first move by playing $X \xrightarrow{a} Y^{\mathtt{choice}}$ 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 $X \xrightarrow{a} Y^{\mathtt{choice}}$ is either (i) $X' \xrightarrow{a} Y^{\mathtt{one}}$ or (ii) $X' \xrightarrow{a} Y^{\mathtt{two}}$.

In the next round starting from (i) $Y^{\mathtt{choice}}$ and $Y^{\mathtt{one}}$ or (ii) $Y^{\mathtt{choice}}$ and $Y^{\mathtt{two}}$, the attacker can use either the action `one` or `two` — obviously it is irrelevant whether the chosen action is performed in the first or in the second process. In case (i), if the attacker chooses the action `one` then the players reach the pair $Z.\alpha_{\mathtt{one}}$ and $Z'.\alpha_{\mathtt{one}}$. If he chooses the action `two` then the players reach a pair of syntactically equal states, namely $Z.\alpha_{\mathtt{two}}$ and $Z.\alpha_{\mathtt{two}}$, 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 $Z.\alpha_{\mathtt{two}}$ and $Z'.\alpha_{\mathtt{two}}$. 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) $Z.\alpha_{\mathtt{one}}$ and $Z'.\alpha_{\mathtt{one}}$ or (ii) $Z.\alpha_{\mathtt{two}}$ and $Z'.\alpha_{\mathtt{two}}$. Note that it was the defender who had the possibility to decide between adding $\alpha_{\mathtt{one}}$ or $\alpha_{\mathtt{two}}$.

We can repeat this construction several times in a row as it is explained later.

4 Hardness of Strong Bisimilarity

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

We show that strong bisimilarity of BPA is a PSPACE-hard problem. We prove it by a reduction from QSAT[1], which is PSPACE-complete [13].

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

- α_i be a sequential composition $Q_{i_1}.Q_{i_2}.\cdots.Q_{i_\ell}$ s.t. $1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ 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 sequential composition $Q_{i_1}.Q_{i_2}.\cdots.Q_{i_\ell}$ s.t. $1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and $C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}$ are all the clauses where x_i occurs negatively,
- β_i be a sequential composition $Q_{i_1}.Q_{i_2}.\cdots.Q_{i_\ell}$ s.t. $1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and $C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}$ are all the clauses where y_i occurs positively,
- $\overline{\beta_i}$ be a sequential composition $Q_{i_1}.Q_{i_2}.\cdots.Q_{i_\ell}$ s.t. $1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$ and $C_{i_1}, C_{i_2}, \ldots, C_{i_\ell}$ are all the clauses where y_i occurs negatively.

Let $\mathcal{S}(\gamma)$ be the set of all suffixes of γ for $\gamma \in \mathcal{E}^{\{Q_1,\ldots,Q_k\}}$, i.e., $\mathcal{S}(\gamma) \stackrel{\text{def}}{=} \{\gamma' \in \mathcal{E}^{\{Q_1,\ldots,Q_k\}} \mid \exists \gamma'' \in \mathcal{E}^{\{Q_1,\ldots,Q_k\}}$ 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 BPA processes $(X_1.S, \Delta)$ and $(X_1'.S, \Delta)$, where $\mathcal{C}onst(\Delta) \stackrel{\text{def}}{=}$

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

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:

[1] This problem is known also as QBF, for *Quantified Boolean formula*.

$$A_0 \xrightarrow{c} \epsilon$$
$$A_\ell \xrightarrow{c} A_{\ell-1}.A_{\ell-2}.\cdots.A_1.A_0 \qquad \text{for all } \ell,\ 1 \le \ell \le kK-1$$

$$Q_j \xrightarrow{c} A_{jK-1}.A_{jK-2}.\cdots.A_1.A_0 \qquad \text{for all } j,\ 1 \le j \le k.$$

Remark 1. 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, $0 \le \ell \le kK-1$, which implies by using the equation $2^{jK} = M^j$ that $Q_j \xrightarrow{c^{M^j}} \epsilon$ for all j, $1 \le j \le k$. 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, $1 \le j \le k$, 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 1. *Assume that M and k are constants introduced above, i.e., $M > 1$ and $k > 0$. Let a_j, $1 \le j \le k$, be natural numbers such that $0 \le a_j \le M-1$ for all j. The following two statements are equivalent:*

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

Proof. By uniqueness of M-ary representations. Details are in [16]. □

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

$$V_i^{Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\mathtt{choice}}$$
$$V_i^{Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\mathtt{yes}} \qquad V_i^{'Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\mathtt{yes}}$$
$$V_i^{Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\mathtt{no}} \qquad V_i^{'Q.\gamma} \xrightarrow{a} V_i^{Q.\gamma,\mathtt{no}}$$

$$V_i^{Q.\gamma,\mathtt{choice}} \xrightarrow{\mathtt{yes}} V_i^{\gamma}.Q \qquad V_i^{Q.\gamma,\mathtt{yes}} \xrightarrow{\mathtt{yes}} V_i^{'\gamma}.Q$$
$$V_i^{Q.\gamma,\mathtt{choice}} \xrightarrow{\mathtt{no}} V_i^{\gamma} \qquad V_i^{Q.\gamma,\mathtt{no}} \xrightarrow{\mathtt{no}} V_i^{'\gamma}$$
$$V_i^{Q.\gamma,\mathtt{yes}} \xrightarrow{\mathtt{no}} V_i^{\gamma} \qquad V_i^{Q.\gamma,\mathtt{no}} \xrightarrow{\mathtt{yes}} V_i^{\gamma}.Q$$

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

$$W_i^{Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\mathtt{choice}}$$
$$W_i^{Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\mathtt{yes}} \qquad W_i^{'Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\mathtt{yes}}$$
$$W_i^{Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\mathtt{no}} \qquad W_i^{'Q.\gamma} \xrightarrow{a} W_i^{Q.\gamma,\mathtt{no}}$$

$$W_i^{Q.\gamma,\mathtt{choice}} \xrightarrow{\mathtt{yes}} W_i^{\gamma}.Q \qquad W_i^{Q.\gamma,\mathtt{yes}} \xrightarrow{\mathtt{yes}} W_i^{'\gamma}.Q$$
$$W_i^{Q.\gamma,\mathtt{choice}} \xrightarrow{\mathtt{no}} W_i^{\gamma} \qquad W_i^{Q.\gamma,\mathtt{no}} \xrightarrow{\mathtt{no}} W_i^{'\gamma}$$
$$W_i^{Q.\gamma,\mathtt{yes}} \xrightarrow{\mathtt{no}} W_i^{\gamma} \qquad W_i^{Q.\gamma,\mathtt{no}} \xrightarrow{\mathtt{yes}} W_i^{\gamma}.Q$$

Assume now a bisimulation game starting from $(V_i^{\alpha_i}, \Delta)$ and $(V_i^{'\alpha_i}, \Delta)$. As shown in Section 3, 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) $V_i^{\alpha_i,\mathtt{choice}}$ and $V_i^{\alpha_i,\mathtt{yes}}$ or (ii) $V_i^{\alpha_i,\mathtt{choice}}$ and $V_i^{\alpha_i,\mathtt{no}}$. 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 a 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 the pairs $(V_i^{\overline{\alpha_i}}, \Delta)$ 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, $1 \leq i \leq n$,

$$X_i \xrightarrow{a} Y_i^{\mathtt{choice}}$$
$$X_i \xrightarrow{a} Y_i^{\mathtt{tt}} \qquad X_i' \xrightarrow{a} Y_i^{\mathtt{tt}}$$
$$X_i \xrightarrow{a} Y_i^{\mathtt{ff}} \qquad X_i' \xrightarrow{a} Y_i^{\mathtt{ff}}$$

$$Y_i^{\mathtt{choice}} \xrightarrow{\mathtt{tt}} V_i^{\alpha_i} \qquad Y_i^{\mathtt{tt}} \xrightarrow{\mathtt{tt}} V_i^{'\alpha_i}$$
$$Y_i^{\mathtt{choice}} \xrightarrow{\mathtt{ff}} V_i^{\overline{\alpha_i}} \qquad Y_i^{\mathtt{ff}} \xrightarrow{\mathtt{ff}} V_i^{'\overline{\alpha_i}}$$
$$Y_i^{\mathtt{tt}} \xrightarrow{\mathtt{ff}} V_i^{\overline{\alpha_i}} \qquad Y_i^{\mathtt{ff}} \xrightarrow{\mathtt{tt}} V_i^{\alpha_i}$$

$$V_i^{\epsilon} \xrightarrow{a} Z_i \qquad V_i^{'\epsilon} \xrightarrow{a} Z_i'$$

$$Z_i \xrightarrow{\mathtt{tt}} W_i^{\beta_i} \qquad Z_i' \xrightarrow{\mathtt{tt}} W_i^{'\beta_i}$$
$$Z_i \xrightarrow{\mathtt{ff}} W_i^{\overline{\beta_i}} \qquad Z_i' \xrightarrow{\mathtt{ff}} W_i^{'\overline{\beta_i}}$$

$$W_i^{\epsilon} \xrightarrow{a} X_{i+1} \qquad W_i^{'\epsilon} \xrightarrow{a} X_{i+1}'$$

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

Lemma 2. *If $(X_1.S, \Delta) \sim (X_1'.S, \Delta)$ then the quantified 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

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

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 Section 3, in the first round the attacker is forced to perform the move $X_1.S \xrightarrow{a} Y_1^{\mathtt{choice}}.S$. The defender can respond either by (i) $X_1'.S \xrightarrow{a} Y_1^{\mathtt{tt}}.S$ (which corresponds to setting the variable x_1 to true) or by (ii) $X_1'.S \xrightarrow{a} Y_1^{\mathtt{ff}}.S$ (which corresponds to setting the variable x_1 to false). In the second round the attacker performs the action (i) $\mathtt{tt}$ or (ii) $\mathtt{ff}$, 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 (Remark 1) 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 1 — 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 1. Hence the attacker has a winning strategy and $(X_1.S, \Delta) \not\sim (X'_1.S, \Delta)$. □

Lemma 3. *If the quantified formula C is true then $(X_1.S, \Delta) \sim (X'_1.S, \Delta)$.*

Proof. Assume a bisimulation game starting from $(X_1.S, \Delta)$ and $(X'_1.S, \Delta)$. We show that the defender has a winning strategy. As mentioned in Section 3 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 a 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. Hence $(X_1.S, \Delta) \sim (X'_1.S, \Delta)$. □

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

Proof. By Lemma 2 and Lemma 3. □

Remark 2. Notice that there are only finitely many reachable states from both $(X_1.S, \Delta)$ and $(X'_1.S, \Delta)$. Hence $(X_1.S, \Delta)$ and $(X'_1.S, \Delta)$ are strongly regular.

5 Hardness of Strong Regularity

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

The idea to reduce bisimilarity to regularity first appeared in the literature due to Mayr [11]. He showed a technique for reducing weak bisimilarity of regular BPP to weak regularity of BPP. However, in his reduction τ actions are used. Developing this approach, we provide a polynomial time reduction from strong bisimilarity of regular BPA to strong regularity of BPA.

Theorem 2 (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 processes. We construct a BPA process (P, Δ') with $\mathcal{C}onst(\Delta') \stackrel{\text{def}}{=} \mathcal{C}onst(\Delta) \cup \{X, A, C, S, P_1', P_2'\}$ and $\mathcal{A}ct(\Delta') \stackrel{\text{def}}{=} \mathcal{A}ct(\Delta) \cup \{a, s\}$ where X, A, C, S, P_1', P_2' are new process constants and a, s are new actions. Let Δ' contain all the rules from Δ plus

$$X \stackrel{a}{\longrightarrow} X.A \qquad X \stackrel{a}{\longrightarrow} \epsilon \qquad A \stackrel{a}{\longrightarrow} \epsilon \qquad S \stackrel{s}{\longrightarrow} S$$

$$\begin{array}{llll}
A \stackrel{a}{\longrightarrow} P_1'.S & A \stackrel{a}{\longrightarrow} P_1.S & P_1' \stackrel{a}{\longrightarrow} P_1' & P_1' \stackrel{a}{\longrightarrow} P_1 \\
X \stackrel{a}{\longrightarrow} P_1'.S & X \stackrel{a}{\longrightarrow} P_1.S & & \\
C \stackrel{a}{\longrightarrow} P_1'.S & C \stackrel{a}{\longrightarrow} P_1.S & & \\
A \stackrel{a}{\longrightarrow} P_2'.S & A \stackrel{a}{\longrightarrow} P_2.S & P_2' \stackrel{a}{\longrightarrow} P_2' & P_2' \stackrel{a}{\longrightarrow} P_2 \\
X \stackrel{a}{\longrightarrow} P_2'.S & X \stackrel{a}{\longrightarrow} P_2.S & & \\
C \stackrel{a}{\longrightarrow} P_2'.S & C \stackrel{a}{\longrightarrow} P_2.S. & &
\end{array}$$

Let $P \stackrel{\text{def}}{=} X.C$. It remains to establish that (P, Δ') is strongly regular if and only if $(P_1, \Delta) \sim (P_2, \Delta)$. Details can be found in [16]. □

Theorem 3. *Strong regularity of BPA is PSPACE-hard.*

Proof. By Theorem 1, Remark 2 and Theorem 2. □

6 Conclusion

The main results of this paper are the first nontrivial lower bounds for strong bisimilarity and strong regularity of BPA. We proved both problems to be PSPACE-hard. Another contribution of this paper is with regard to the normedness notion. Our results show a substantial difference (unless P=PSPACE) between *normed* and *unnormed* processes — strong bisimilarity and regularity checking is PSPACE-hard for unnormed BPA, whereas it is decidable in polynomial time for normed BPA [8, 10].

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 [11] and [15]) require a polynomial number of unnormed process constants.

Recently some lower bounds appeared for *weak* bisimilarity of BPA and BPP [18, 11, 14], even though the problems are not known to be decidable. In the following tables we compare the results for strong/weak bisimilarity and regularity. New results achieved in this paper are in boldface. Obviously, the lower bounds for strong bisimilarity and regularity checking apply also to weak bisimilarity and regularity, thus improving the DP lower bound for weak regularity of BPA from [14] to PSPACE.

	strong bisimilarity	weak bisimilarity
BPA	decidable in 2-EXPTIME [4] **PSPACE-hard**	? PSPACE-hard [18]
normed BPA	decidable in P [8] P-hard [1]	? NP-hard [18]

	strong regularity	weak regularity
BPA	decidable in 2-EXPTIME [5, 4] **PSPACE-hard**	? **PSPACE-hard**
normed BPA	decidable in NL [10] **NL-hard**	? NP-hard [14]

Remark 3. Complexity of strong regularity of normed BPA needs more explanation. Kucera in [10] argues that the problem is decidable in polynomial time but it is easy to see that a test whether a BPA process contains an *accessible* and *growing* process constant (a condition equivalent to regularity) can be performed even in nondeterministic logarithmic space (NL).

In order to prove NL-hardness, we reduce the reachability problem for acyclic directed graphs (NL-complete problem, see [13]) to strong regularity checking of normed BPA. Given an acyclic directed graph G with a pair of nodes v_1 and v_2 (w.l.o.g. assume that both v_1 and v_2 have out-degree at least one), we naturally construct a BPA system Δ by introducing a new process constant for each node of G with out-degree at least one, and with a-labelled transitions respecting the edges of G. All nodes with out-degree 0 are represented by the empty process ϵ to ensure that the system is deadlock-free. Moreover, a process constant representing the node v_2 has a transition to a new process constant A such that Δ contains also the rewrite rules $A \stackrel{a}{\longrightarrow} A.A$ and $A \stackrel{b}{\longrightarrow} \epsilon$. It is easy to see that (A, Δ) is a normed and non-regular process. Let X be a process constant representing the node v_1. Since G is acyclic, (X, Δ) is a normed BPA process. Obviously, there is a directed path from v_1 to v_2 in G if and only if (X, Δ) is not a strongly regular process. Recall that NL = co-NL (see e.g. [13]). Hence strong regularity of normed BPA is NL-complete.

Acknowledgement. I would like to thank my advisor Mogens Nielsen for his kind supervision. I also thank Pawel Sobocinski and the anonymous referees 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] 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.

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

[8] 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.

[9] P. Jancar 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.

[10] A. Kucera. 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.

[11] R. Mayr. On the complexity of bisimulation problems for basic parallel processes. In *Proceedings of ICALP'00*, volume 1853 of *LNCS*, pages 329–341. Springer-Verlag, 2000.

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

[13] Ch.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[14] J. Srba. Complexity of weak bisimilarity and regularity for BPA and BPP. In *Proceedings of EXPRESS'00*, volume 39 of *ENTCS*. Elsevier Science Publishers, 2000. To appear.

[15] J. Srba. Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard. In *Proceedings of STACS'02*, volume 2285 of *LNCS*, pages 535–546. Springer-Verlag, 2002.

[16] J. Srba. Strong bisimilarity of simple process algebras: Complexity lower bounds. Technical Report RS-02-16, BRICS Research Series, 2002.

[17] C. Stirling. Local model checking games. In *Proceedings of CONCUR'95*, volume 962 of *LNCS*, pages 1–11. Springer-Verlag, 1995.

[18] J. Stribrna. Hardness results for weak bisimilarity of simple process algebras. In *Proceedings of the MFCS'98 Workshop on Concurrency*, volume 18 of *ENTCS*. Springer-Verlag, 1998.

[19] W. Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science (extended abstract). In *Proceedings of TAPSOFT'93*, volume 668 of *LNCS*, pages 559–568. Springer-Verlag, 1993.