Strong Bisimilarity and Regularity of Basic Parallel Processes 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. We show that the problem of checking whether two processes definable in the syntax of Basic Parallel Processes (BPP) are strongly bisimilar is PSPACE-hard.
We also demonstrate that there is a polynomial time reduction from the strong bisimilarity checking problem of regular BPP to the strong regularity (finiteness) checking of BPP. This implies that strong regularity of BPP is also PSPACE-hard.

1 Introduction

In verification of infinite-state systems (see e.g. [2]), there are two central problems concerned with decidability and complexity of (i) checking whether two processes are equivalent with regard to some behavioural equivalence and (ii) model checking process properties expressed in a suitable logic. In this paper we study the first sort of problems.

Strong bisimilarity [15, 13] is a well accepted notion of behavioural equivalence for concurrent processes. Unlike other equivalences in van Glabbeek's spectrum [20, 21], strong bisimilarity remains decidable for many process algebras that generate infinite-state systems. There are e.g. positive results for bisimilarity checking of context-free process algebra [4] (also called BPA for Basic Process Algebra) and its parallel analogue Basic Parallel Processes [3] (BPP). We study here the BPP process algebra, a fragment of CCS [13] without restriction, relabelling and communication.

Decidability of strong bisimilarity for BPP was demonstrated by Christensen et al. [3], however, the algorithm relies on Dickson's Lemma [5] and its exact complexity is unknown. To the best of our knowledge, no algorithm with elementary complexity (bounded number of exponentiations) has been found yet. On the other hand, if we consider only *normed* BPP processes (from every reachable state there is a terminating computation), strong bisimilarity is decidable in polynomial time [6]. The conjecture that the unnormed case is also decidable

* 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.

in polynomial time was only recently proved to be wrong (unless P = NP) by Mayr. He showed that strong bisimilarity of BPP is co-NP-hard [11].

We improve Mayr's co-NP lower bound to PSPACE by demonstrating a reduction from QSAT [14]. The main idea of our proof is, however, different from Mayr's approach. Speaking in terms of bisimulation games of an attacker and a defender: in our proof we exploit the possibility that the defender can force the attacker to switch sides in the bisimulation game an arbitrary number of times. Moreover, we introduce a technique in which the defender can decide on the successive configurations in the bisimulation game independently of the attacker's will. This enables to encode quantified instances of SAT problems as BPP processes. Similar ideas appeared in [7] in connection with high undecidability of weak bisimilarity for Petri nets, but the other lower bound proofs (see e.g. [18, 11, 16]) use only a constant number of switching sides when generating an instance of a problem.

Another issue that is often studied is strong regularity (finiteness) checking. The question is whether a given BPP process is strongly bisimilar to some finite-state process. Strong regularity is decidable even for Petri nets [8]. As BPP systems are a communication free subclass of Petri nets, strong regularity is decidable for them as well. Again, restricting ourself to the normed case only, the problem is solvable in polynomial time [10]. The complexity of the unnormed case is open. It is known that the problem is co-NP-hard [11] but no elementary upper bound has been established.

In this paper, we provide a polynomial time reduction from strong bisimilarity checking of regular BPP to strong regularity checking. Hence using our PSPACE-hardness of strong bisimilarity for BPP (processes used in the proof are finite-state and thus trivially regular), we get PSPACE lower bound for strong regularity checking of BPP.

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 BPP *process expressions* $\mathcal{E}^{\mathcal{C}onst}$ *over* $\mathcal{C}onst$ by the following abstract syntax

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

where 'ϵ' is the *empty process* and X ranges over $\mathcal{C}onst$. The operator '$\|$' stands for a *parallel composition*. We do not distinguish between process expressions related by a *structural congruence*, which is the smallest congruence over process expressions such that '$\|$' is associative and commutative, and 'ϵ' is a unit for '$\|$'.

A BPP *process rewrite system* (or a (1,P)-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 $\mathcal{A}ct(\Delta)$ and $\mathcal{C}onst(\Delta)$, respectively. 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 (recall that '$\|$' is commutative).

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

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 BPP *process* is a pair (P, Δ), where Δ is a BPP process rewrite system and $P \in \mathcal{E}^{\mathcal{C}onst(\Delta)}$ is a BPP 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., there is some E' such that $E \longrightarrow^* E' \not\longrightarrow$.

Remark 1. Let i be a natural number and $A \in \mathcal{C}onst$. We use the notation A^i for a parallel composition of i occurrences of A, i.e., $A^0 \stackrel{\text{def}}{=} \epsilon$ and $A^{i+1} \stackrel{\text{def}}{=} A^i\|A$.

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 Hardness of strong bisimilarity

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

We show that strong bisimilarity of BPP is a PSPACE-hard problem. We prove it by a reduction from QSAT[1], which is PSPACE-complete [14]. We use a version where the prefix of quantifiers starts with the existential one.

Problem:	QSAT
Instance:	A natural number n 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, $1 \le j \le k$, is a disjunction of literals. We define the following BPP processes (X_1, Δ) and (X'_1, Δ), where

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

and

$$\mathcal{A}ct(\Delta) \stackrel{\text{def}}{=} \{q_1, \ldots, q_k, a, \mathtt{tt}, \mathtt{ff}\}.$$

For each i, $1 \le i \le n$, let

- α_i be a parallel composition of process constants from $\{Q_1, \ldots, Q_k\}$ such that Q_j appears in α_i iff the literal x_i occurs in C_j (i.e. if x_i is set to true then C_j is satisfied),
- $\overline{\alpha_i}$ be a parallel composition of process constants from $\{Q_1, \ldots, Q_k\}$ such that Q_j appears in $\overline{\alpha_i}$ iff the literal $\neg x_i$ occurs in C_j (i.e. if x_i is set to false then C_j is satisfied),
- β_i be a parallel composition of process constants from $\{Q_1, \ldots, Q_k\}$ such that Q_j appears in β_i iff the literal y_i occurs in C_j,
- $\overline{\beta_i}$ be a parallel composition of process constants from $\{Q_1, \ldots, Q_k\}$ such that Q_j appears in $\overline{\beta_i}$ iff the literal $\neg y_i$ occurs in C_j.

Example 1. 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 $\alpha_1 = Q_1 \| Q_3$, $\overline{\alpha_1} = Q_2$, $\beta_1 = Q_2 \| Q_3$, $\overline{\beta_1} = Q_1$, $\alpha_2 = \epsilon$, $\overline{\alpha_2} = \epsilon$, $\beta_2 = Q_1 \| Q_2 \| Q_3$, and $\overline{\beta_2} = Q_3$.

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

The set Δ is given by the following rewrite rules:

– for all j such that $1 \leq j \leq k$: $\quad Q_j \xrightarrow{q_j} Q_j$

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

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

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

We can see the processes (X_1, Δ) and (X_1', Δ) in Figure 1 if we set $i = 1$ and $\gamma_1 = \epsilon$. The intuition behind the construction can be nicely explained in terms of bisimulation games. Consider a bisimulation game starting from X_1 and X_1'.

The attacker is forced to make the first move by playing $X_1 \xrightarrow{a} Y_1$ 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 $X_1 \xrightarrow{a} Y_1$ 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 switch processes and play either (i) $Y_1^{\mathtt{tt}} \xrightarrow{\mathtt{tt}} Z_1' \| \alpha_1$ or (ii) $Y_1^{\mathtt{ff}} \xrightarrow{\mathtt{ff}} Z_1' \| \overline{\alpha_1}$, according to the defender's choice in the first round. If the attacker chooses any other move, the defender has the possibility to make the resulting processes syntactically equal. Now, the defender has only one possible answer by playing in the first process — in the case (i) he plays $Y_1 \xrightarrow{\mathtt{tt}} Z_1 \| \alpha_1$ and in the case (ii) he plays $Y_1 \xrightarrow{\mathtt{ff}} Z_1 \| \overline{\alpha_1}$. 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

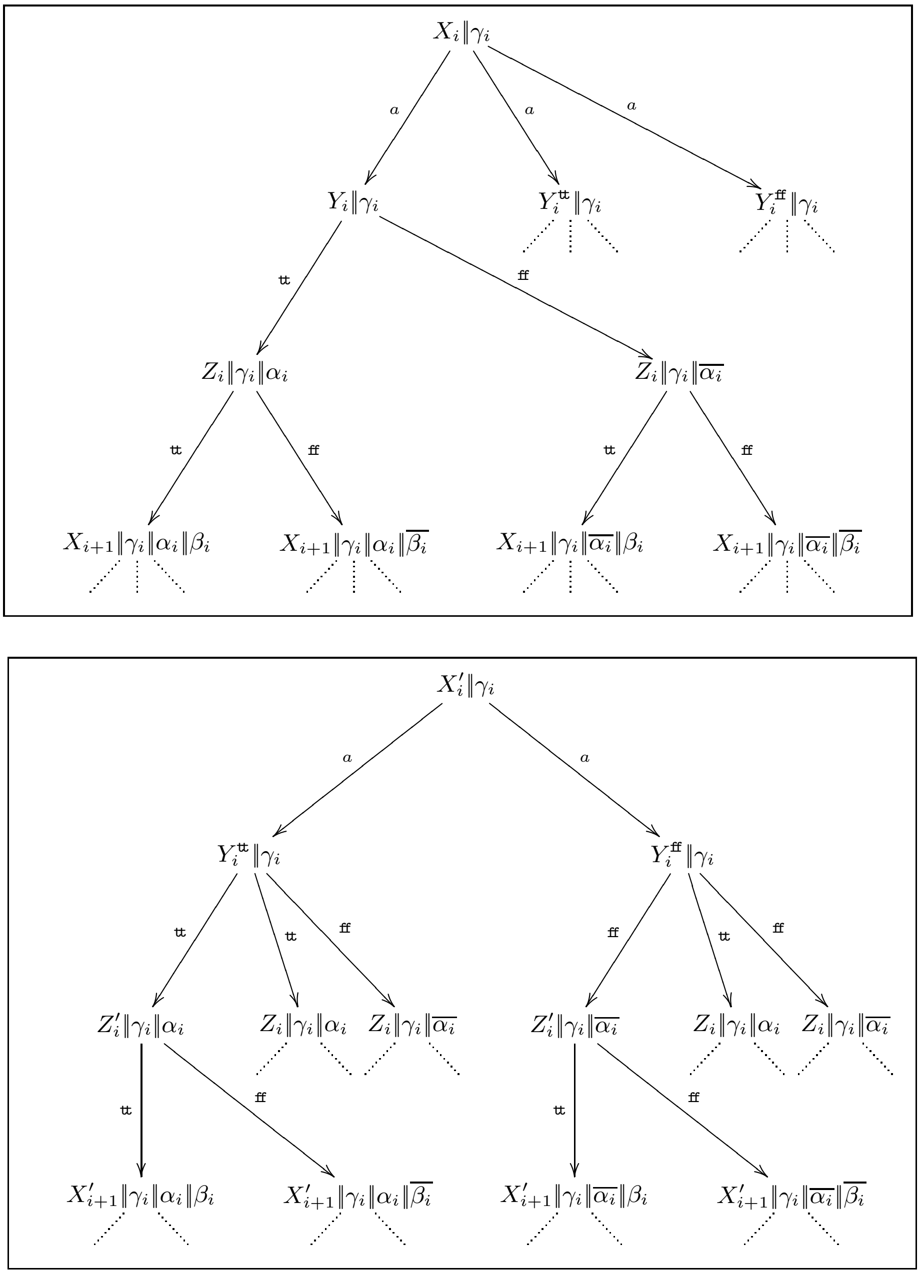

Fig. 1. Processes $(X_i \| \gamma_i, \Delta)$ and $(X'_i \| \gamma_i, \Delta)$

of the actions $q_1, \ldots, q_k$ 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 $X_{n+1} \| \gamma_{n+1} \xrightarrow{a} Q_1 \| Q_2 \| \ldots \| Q_{k-1} \| Q_k \| \gamma_{n+1}$ to which the defender has only one answer, namely $X'_{n+1} \| \gamma_{n+1} \xrightarrow{a} \gamma_{n+1}$. From the states $Q_1 \| Q_2 \| \ldots \| Q_{k-1} \| Q_k \| \gamma_{n+1}$ 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 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 $Q_1 \| Q_2 \| \ldots \| Q_{k-1} \| Q_k \| \gamma_{n+1} \sim \gamma_{n+1}$ 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 C is true.*

Proof. We show that $(X_1, \Delta) \not\sim (X'_1, \Delta)$ under the assumption that 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, Δ) and (X'_1, Δ). The attacker's strategy starts with performing a sequence of actions $a, \widetilde{x}_1, \widetilde{y}_1, \ldots, a, \widetilde{x}_i, \widetilde{y}_i, \ldots, a, \widetilde{x}_n, \widetilde{y}_n, a$ where $\widetilde{x}_i, \widetilde{y}_i \in \{\mathtt{tt}, \mathtt{ff}\}$ for all i, $1 \leq i \leq n$. The attacker is switching regularly the sides of processes as described above, i.e., actions a are played on the side of the first process (X_1, Δ), $\widetilde{x}_i$ are played on the side of the second process (X'_1, Δ), and $\widetilde{y}_i$ are played, let us say, in the first process again. The choice of $\widetilde{x}_i$ is done by the defender and of $\widetilde{y}_i$ by the attacker — again see the discussion above. This means that whatever values for $\widetilde{x}_1, \ldots, \widetilde{x}_n$ are chosen by the defender, the attacker can still decide on values for $\widetilde{y}_1, \ldots, \widetilde{y}_n$ such that the generated assignment satisfies the formula $\neg(C_1 \wedge C_2 \wedge \ldots \wedge C_k)$. Hence there must be some j, $1 \leq j \leq k$, 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 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, $1 \leq i \leq n$, such that 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 still true. Sets $\mathcal{AS}_i$ are defined by

$\mathcal{AS}_i \stackrel{\text{def}}{=} \{\ \widetilde{\alpha}_1 \| \widetilde{\beta}_1 \| \widetilde{\alpha}_2 \| \widetilde{\beta}_2 \| \ldots \| \widetilde{\alpha}_i \| \widetilde{\beta}_i\ \mid$

such that for all j, $1 \leq j \leq i$, 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 $\}$.

By definition $\mathcal{AS}_0 = \{\epsilon\}$. 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.

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

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

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

Proof. By Lemma 1 and Lemma 2. $\square$

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

Remark 3. Theorem 1 can be easily extended to 1-safe Petri nets where each transition has exactly one input place (for the definition of 1-safe Petri nets see e.g. [9]). It is enough to introduce for each $\alpha_i/\overline{\alpha_i}$ and $\beta_i/\overline{\beta_i}$, $1 \le i \le n$, 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. Related results about 1-safe Petri nets can be found in [9].

4 Hardness of strong regularity

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

The idea of the next theorem's proof appeared first in [11] in the context of weak bisimilarity. We adapt this idea to the case of strong bisimilarity (in the proof from [11] τ actions are needed).

Theorem 2 (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

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

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

Fig. 2. Winning strategy for the attacker ($i < j$)

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 3. *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 2).

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 $B_c \stackrel{a}{\not\longrightarrow}$ 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 nor 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 $P_2 \not\sim \epsilon$ 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 a winning strategy from $P_1 \| A^{i''} \| A_c \| B_c$ and $P_2 \| A_c \| B_c$ now: 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 4. *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 $\Delta'' \stackrel{\text{def}}{=} \Delta' \setminus \Delta^2$ (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 strongly regular 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 2 follows from Lemma 3 and Lemma 4. □

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

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

5 Conclusion

We proved PSPACE-hardness of strong bisimilarity and regularity of BPP by using a technique in which the defender has the possibility to force the attacker to perform a certain move and switch sides of processes arbitrarily many times. We belive that this technique can be used in other hardness results concerning bisimilarity. Another contribution of this paper is with regard to the normedness notion. Our results show a substantial difference (unless P = NP) between *normed* and *unnormed* processes — strong bisimilarity and regularity checking is PSPACE-hard for unnormed BPP, whereas it is decidable in polynomial time for normed BPP [6, 10].

Recently some lower bounds appeared for *weak* bisimilarity of BPA and BPP [18, 11, 16], 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 apply also to weak bisimilarity and regularity. Hence we have another proof of PSPACE-hardness of weak bisimilarity and regularity for BPP. The difference is that in the proof from [16] only a constant number of switching sides during the bisimulation game is needed for the attacker to win, and even more importantly, the proof is valid also for the normed case. On the other hand the usage of τ actions is essential for the reductions from [16].

	strong bisimilarity	weak bisimilarity
BPP	decidable [3] **PSPACE-hard**	? **PSPACE-hard** [16]
normed BPP	decidable in P [6] P-hard [1]	? PSPACE-hard [16]

	strong regularity	weak regularity
BPP	decidable [8] **PSPACE-hard**	? **PSPACE-hard** [16]
normed BPP	decidable in NL [10] **NL-hard**	? PSPACE-hard [16]

Remark 4. Complexity of strong regularity of normed BPP 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 BPP process contains an *accessible* and *growing* process constant (a condition equivalent to regularity checking) 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 [14]) to strong regularity checking of normed BPP. Given an acyclic directed graph G with a pair of nodes v_1 and v_2, we naturally construct a BPP system Δ by introducing a new process constant for each node of G, with transitions respecting the edges of G and labelled by a. 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 $A \xrightarrow{a} A\|A$ and $A \xrightarrow{b} \epsilon$. 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 BPP 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. [14]). Hence strong regularity of normed BPP is NL-complete.

Acknowledgement. I would like to thank my advisor Mogens Nielsen for his kind supervision. I also thank Jan Strejcek and the referees for useful remarks.

References

[1] J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4(6A):638–648, 1992.

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

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

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

[5] L.E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with distinct factors. *American Journal of Mathematics*, 35:413–422, 1913.

[6] 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.

[7] P. Jancar. High undecidability of weak bisimilarity for Petri nets. In *Proc. of CAAP'95*, volume 915 of *LNCS*, pages 349–363. Springer-Verlag, 1995.

[8] P. Jancar and J. Esparza. Deciding finiteness of Petri nets up to bisimilarity. In *Proc. of ICALP'96*, volume 1099 of *LNCS*, pages 478–489. Springer-Verlag, 1996.

[9] L. Jategaonkar and A.R. Meyer. Deciding true concurrency equivalences on safe, finite nets. *Theoretical Computer Science*, 154(1):107–143, 1996.

[10] A. Kucera. Regularity is decidable for normed BPA and normed BPP processes in polynomial time. In *Proc. 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 *Proc. ICALP'00*, volume 1853 of *LNCS*, pages 329–341. Springer-Verlag, 2000.

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

[13] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

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

[15] D.M.R. Park. Concurrency and automata on infinite sequences. In *Proc. of 5th GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.

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

[17] C. Stirling. Local model checking games. In *Proc. 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 *Proc. 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 *Proc. 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.

[20] R.J. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions.* PhD thesis, CWI/Vrije Universiteit, 1990.

[21] R.J. van Glabbeek. The linear time—branching time spectrum. In *Proc. of CONCUR'90*, volume 458 of *LNCS*, pages 278–297. Springer-Verlag, 1990.