Notes on Complexity of Bisimilarity between BPA and BPP

Petr Jančar, Martin Kot, Zdeněk Sawa

Center for Applied Cybernetics, Dept of Computer Science
Technical University Ostrava, Czech Rep.

June 29, 2007

Abstract

We consider the problem of deciding bisimilarity between a BPA process and a BPP process. This problem is known to be decidable, but no complexity bound has been provided so far. Our aim is to clarify the precise complexity and some related questions. In this short submission we present some preliminary results in this direction.

In particular we present a necessary and sufficient condition for a BPP process to be bisimilar with some BPA process, which can be checked in polynomial space. In the positive case, (a modification of) the algorithm can transform the given BPP process into a special normal form (of exponential size) which can be easily converted into an equivalent BPA process.

In this way, the problem of bisimilarity between BPA and BPP can be reduced to the problem of bisimilarity over BPA, which is known to be solvable in double exponential time.

Keywords: Basic Process Algebra, Basic Parallel Processes, bisimulation equivalence, complexity

1 Introduction

Bisimilarity is one of the most studied behavioral equivalences in the concurrency theory. Decidability and complexity of deciding bisimilarity was studied on different classes of processes, see [1, 5] for a survey.

One of important classes of processes is the process algebra (PA). The process class BPA represents the subset of PA involving only sequential composition, while BPP represents the subset involving only parallel composition.

In the case of BPA, the best known algorithm for deciding bisimilarity is in 2-EXPTIME [2], and the problem is known to be PSPACE-hard [7], while in the case of BPP the problem is known to be PSPACE-complete [3, 6]. It is natural to consider the problem of deciding bisimilarity between a BPA process and a BPP process. It was shown in [8] that the problem is decidable for *normed* processes, and then in [4] that it is decidable in general case, but no complexity bound was provided; the problem was reduced to the problem of bisimilarity for pushdown processes, relying on the decidability result by Sénizergues. We try to extend the ideas used in [4] to examine the complexity of the problem and also some related questions.

In this short submission we present some preliminary results in this direction. Instead of reducing to PDA-bisimilarity, we show that bisimilarity between BPA and BPP can be reduced (just) to BPA-bisimilarity. The reduction can be done in exponential time, and thus the overall upper bound we get is triple exponential. The general strategy of the algorithm is the following: At first we specify a condition which is necessary and sufficient for a BPP to be bisimilar to *some* BPA . We show a polynomial space algorithm for checking whether a given BPP process satisfies this condition. If it is the case, (the modification of) the algorithm transforms the BPP process into a special normal form which then can be converted into a bisimilar BPA process. The constructed BPP process (and corresponding BPA process) can be of exponential size and the algorithm constructs it in exponential time. This way the original problem of deciding bisimilarity between a BPA process and a BPP process is reduced to the problem of deciding bisimilarity between two BPA processes for which a double exponential algorithm is known. Since we use this algorithm on instance of exponential size, the (straightforward) overall upper bound we get is triple exponential.

Section 2 contains basic definitions and Section 3 describes the condition for a BPP to be bisimilar to some BPA. Section 4 mentiones some planned future work. We also provide an Appendix A which describes the normal form of BPP processes for which a bisimilar BPA process exists.

2 Basic definitions and notation

A *labelled transition system* (LTS) is a triple $(S, \mathcal{A}, \longrightarrow)$, where S is a set of *states*, $\mathcal{A}$ is a finite set of *actions*, and $\longrightarrow \subseteq S \times \mathcal{A} \times S$ is a *transition relation.* We write $s \xrightarrow{a} s'$ instead of $(s, a, s') \in \longrightarrow$ and we extend this notation to elements of $\mathcal{A}^*$ in a natural way.

Let $(S, \mathcal{A}, \longrightarrow)$ be an LTS. A binary relation $\mathcal{R} \subseteq S \times S$ is a *bisimulation* iff for each $(s, t) \in \mathcal{R}$ and $a \in \mathcal{A}$ we have:

- $\forall s' \in S : s \xrightarrow{a} s' \Rightarrow (\exists t' : t \xrightarrow{a} t' \wedge (s', t') \in \mathcal{R})$, and
- $\forall t' \in S : t \xrightarrow{a} t' \Rightarrow (\exists s' : s \xrightarrow{a} s' \wedge (s', t') \in \mathcal{R})$.

States s and t are *bisimulation equivalent (bisimilar)*, written $s \sim t$, iff they are related by some bisimulation. We can also relate states of two different LTSs by $\sim$ by taking their disjoint union.

A *BPA process* is given by a context-free grammar in Greibach normal form. Formally it is a triple $G = (V, \mathcal{A}, \Gamma)$, where V is a finite set *variables* (nonterminals), $\mathcal{A}$ is finite set of *actions* (terminals) and $\Gamma \subseteq V \times \mathcal{A} \times V^*$ is finite set of *rewrite rules.* Again, we write $X \xrightarrow{a} \alpha$ instead of $(X, a, \alpha) \in \Gamma$. A BPA G gives rise to an LTS $\mathcal{S}_G = (V^*, \mathcal{A}, \longrightarrow)$ where $\longrightarrow$ is given by the rewrite rules extended with the *prefix rewriting rule*: if $X \xrightarrow{a} \alpha$ then $X\beta \xrightarrow{a} \alpha\beta$ for every $\beta \in V^*$.

A *BPP process* is defined in the same way, but for BPP elements of V^* are read modulo commutativity of concatenation, so they are interpreted as multisets of variables. (Any occurence of variable can be rewritten, not just the first one.)

We use $\mathbb{N} = \{0, 1, 2, \ldots\}$ to denote the set of nonnegative integers, $\mathbb{N}_\omega = \mathbb{N} \cup \{\omega\}$ where ω denotes the first infinite ordinal, and $\mathbb{N}_{\omega,-1} = \mathbb{N}_\omega \cup \{-1\}$.

Let $(S, \mathcal{A}, \longrightarrow)$ be an LTS. For a LTS we define the distance function $\mathit{dist}\colon (S \times S) \to \mathbb{N}_\omega$ as $\mathit{dist}(s, s') = \min\{\mathit{length}(w) \mid w \in \mathcal{A}^*, s \xrightarrow{w} s'\}$. (We define $\min \emptyset = \omega$.)

The set of *DD-functions* [3] is defined inductively as follows:

- For each $a \in \mathcal{A}$ the function $dd_a : S \to \mathbb{N}_\omega$, defined as $dd_a(s) = \min\{\, \mathit{dist}(s, s') \mid \neg\exists s'' : s' \xrightarrow{a} s'' \,\}$, is a DD-function.

- Let $\mathcal{F} = (d_1, \dots, d_k)$ be a tuple of DD-functions. For $s, s' \in S$, such that $s \xrightarrow{a} s'$, we define the *change* $\delta = \mathcal{F}(s') - \mathcal{F}(s)$, which is a k-tuple of values from $\mathbb{N}_{\omega,-1}$. (We put $\omega - \omega = \omega$.) A function $dd_{(a,\mathcal{F},\delta)} : S \to \mathbb{N}_\omega$, defined as

  $$dd_{(a,\mathcal{F},\delta)}(t) = \min\{\, \mathit{dist}(t,t') \mid d_i(t') < \omega \text{ for all } i, \text{ and } \mathcal{F}(t'') - \mathcal{F}(t') \neq \delta \text{ for all } t' \xrightarrow{a} t'' \,\},$$

  is a DD-function.

We use DD to denote the set of DD-functions. Informally, the value $d(s)$ of $d \in \mathsf{DD}$ represents 'distance' from s to the nearest state where certain kind of transitions is disabled, (and $d(s) = \omega$ if there is no such state reachable from s), in particular, dd_a is the distance to a state where transitions labelled with a are disabled.

It can be shown that in the case of finite-branching systems (which include BPA and BPP), $s \sim s'$ iff $d(s) = d(s')$ for each $d \in \mathsf{DD}$. It was shown in [3], that for BPP the set DD is always finite and can be effectively constructed.

For BPA or BPP, the *norm* of a state $\alpha \in V^*$, denoted $|\alpha|$, is the length of a shortest path to the empty process ε: $|\alpha| = \mathit{dist}(\alpha, \varepsilon)$. For a BPA, the transition $X\beta \xrightarrow{a} \gamma\beta$ is a *pn*-reducing step iff $|\gamma| = |X| - 1 < \omega$. Let $d \in \mathsf{DD}$. Transition $s \xrightarrow{a} s'$ is a *d-reducing step* iff $d(s) < \omega$ and $d(s') = d(s) - 1$. For a BPA G, a function $d \in \mathsf{DD}$ is *prefix-encoded* iff there is a constant $C \in \mathbb{N}$ such that for every state α of G such that $C < d(\alpha) < \omega$, the d-reducing steps are exactly the *pn*-reducing steps. It was proved in [4], that for any BPA, each DD-function is prefix encoded.

3 Condition for a BPP to be bisimilar to some BPA

Let as assume we have some BPP process Σ, and we want to know if there is some bisimilar BPA process Δ. For Σ we compute the corresponding set of DD-functions DD, and Δ must agree with Σ on each $d \in \mathsf{DD}$.

We say that $d, d' \in \mathsf{DD}$ *grow independently* in Σ if they can be arbitrarily big (but finite) with an arbitrary big difference between them, i.e., for every $C \in \mathbb{N}$ there is a reachable state α of Σ such that $C < d(\alpha), d'(\alpha) < \omega$ and $|d(\alpha) - d'(\alpha)| > C$. As follows from the fact, that for any BPA every DD-function is prefix encoded, if there are some $d, d' \in \mathsf{DD}$ that grow independently, there is no BPA equivalent with Σ. Here is a simple example of such BPP:

$S \xrightarrow{c} AS \quad S \xrightarrow{c} BS \quad A \xrightarrow{a} \varepsilon \quad B \xrightarrow{b} \varepsilon$

So the condition that there are no d, d' that grow independently is necessary for an existence of a bisimilar BPA, but it is not sufficient as the following example shows:

$S \xrightarrow{c} AB \quad B \xrightarrow{b} \varepsilon$
$A \xrightarrow{a} AA \quad A \xrightarrow{a} \varepsilon$

$$\begin{tikzcd}
B \arrow[d, "b"] & AB \arrow[l, "a"'] \arrow[r, shift left, "a"] \arrow[d, "b"] & AAB \arrow[l, shift left, "a"] \arrow[r, shift left, "a"] \arrow[d, "b"] & AAAB \arrow[l, shift left, "a"] \arrow[r, shift left, "a"] \arrow[d, "b"] & {} \arrow[l, shift left, "a"] \\
\varepsilon & A \arrow[l, "a"'] \arrow[r, shift left, "a"] & AA \arrow[l, shift left, "a"] \arrow[r, shift left, "a"] & AAA \arrow[l, shift left, "a"] \arrow[r, shift left, "a"] & {} \arrow[l, shift left, "a"]
\end{tikzcd}$$

Suppose that there is a bisimilar BPA Δ. If dd_a (number of occurrences of A) is big, Δ must store the value of this function in the prefix of the configuration. But Δ also have to remember if b is enabled. This could change in one step hence BPA must store this information in first variable of configuration. Then decreasing of dd_a leads inevitably to the loss of the information about the value of dd_b. Hence a bisimilar BPA does not exist.

We say that $d \in \mathsf{DD}$ *overgrows* $d' \in \mathsf{DD}$ in Σ if the difference between them can be arbitrary big while d' is nonzero (i.e., for every $C \in \mathbb{N}$ there is a reachable state α of Σ such that $0 < d'(\alpha) < d(\alpha) < \omega$ and $d(\alpha) - d'(\alpha) > C$), d' can be changed without setting d to ω, and d can be decreased without setting d' to ω.

We have (a little bit technical) proof of the following lemma, which we do not show here due to space limitations.

Lemma 1 *A BPP is bisimilar to some BPA iff there is no pair of DD-functions that grow independently and there is nor pair of DD-functions where one overgrows the other.*

Theorem 2 *There is a polynomial space algorithm that decides for a given BPP whether there is some equivalent BPA.*

Proof idea: The algorithm nondeterministically guesses the run of the BPP that shows that the condition specified in Lemma 1 is violated. □

It can be shown that if a BPP does not violate the condition, it can be transformed into a special normal form. See Appendix A for an example of a transformation of a BPP into the normal form.

Theorem 3 *There is an algorithm running in exponential time that for a given BPP checks if there exists a bisimilar BPA, and if it is the case, then it constructs a BPP in the normal form (of exponential size).*

A BPP in a normal form can be easily converted to a BPA of the same size.

Theorem 4 *There is an algorithm with triple exponential running time deciding bisimilarity between BPA and BPP.*

Proof idea: The algorithm tries to convert the given BPP into an equivalent BPA, and if it fails, stops with the answer NO, otherwise runs the algorithm for deciding bisimilarity on BPA on the constructed BPA and the BPA from the instance. □

4 Final remarks

We present a work in progress. The triple exponential complexity seems to be still an "overshot", which we hope to improve after clarifying a necessary and sufficient condition for a BPA to be bisimilar with some BPP.

Another related question is the possibility of extending these results to bisimilarity between PDA and BPP.

References

[1] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In *Handbook of Process Algebra*, pages 545–623, 2001.

[2] O. Burkart, D. Caucal, and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *LNCS*, pages 423–433. Springer-Verlag, 1995.

[3] P. Jančar. Strong bisimilarity on basic parallel processes is PSPACE-complete. In *Proc. 18th LiCS*, pages 218–227. IEEE Computer Society, 2003.

[4] P. Jančar, A. Kučera, and F. Moller. Deciding bisimilarity between BPA and BPP processes. In *Proceedings of CONCUR 2003*, volume 2761 of *LNCS*, pages 159–173. Springer-Verlag, 2003.

[5] J. Srba. Roadmap of infinite results. In *Bulletin of EATCS*, volume 78, pages 163–175, October 2002. See updated version at `http://www.brics.dk/~srba/roadmap/`.

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

[7] J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP'02)*, volume 2380 of *LNCS*, pages 716–727. Springer, 2002.

[8] I. Černá, M. Křetínský, and A. Kučera. Comparing expressibility of normed BPA and normed BPP processes. *Acta Informatica*, 36:233–256, 1999.

A Normal form of a BPP

In this section we describe normal form of a BPP bisimilar to original BPP.

Suppose that a BPP system Σ is bisimilar to some BPA system Δ. The set of states of Σ can be divided into *stages*, where one stage represents the subset of states with the same subset of DD-functions which are set to ω.

We will illustrate the transformation on the following example:

$$S \xrightarrow{a} C \quad C \xrightarrow{a} D \quad C \xrightarrow{a} EG \quad D \xrightarrow{d} DD \quad D \xrightarrow{d} \varepsilon \quad E \xrightarrow{e} F$$
$$F \xrightarrow{f} E \quad G \xrightarrow{g} GG \quad G \xrightarrow{g} H \quad H \xrightarrow{e} H \quad H \xrightarrow{f} H$$

In the normal form, for each stage we have a special set of variables. Variables A_i represent the beginning of the stage when BPP acts as a finite-state system. In our example $A_1 \xrightarrow{a} A_2$ where A_1 resp. A_2 represents configurations S resp. C.

Then BPP system can choose which DD-function (or more DD-functions with the same values on reachable configurations) will be able to grow unlimitedly. For each such possible set Q of DD-functions, in the normal form are: variable Z_i and possibly empty set of variables $X_{1,i}, X_{2,i}, \ldots, X_{k_i,i}$. For each $d \in Q$ it holds $d(Z_i) = 1$. Hence the value of d we represent by the number of occurrences of Z_i.

In the example we need Z_1 for the function dd_d and Z_2 for dd_g. We get $A_2 \xrightarrow{a} Z_1$, $Z_1 \xrightarrow{d} Z_1Z_1$ and $Z_1 \xrightarrow{d} \varepsilon$ as a representation of rules $C \xrightarrow{a} D$, $D \xrightarrow{d} DD$, $D \xrightarrow{d} \varepsilon$.

As long as the value of some DD-function d grows, there could be more nonbisimilar states with the same value of d. This we represent using variables X. In the example $G^iE \nsim G^iF$ for all i. After first decrease of d, only one state for each value of d can exist. This is done by setting all DD-functions different on such nonbisimilar states to ω. Hence the set R has changed and the system is in the next stage.

In the example after first use of $G \xrightarrow{g} H$ (decreasing dd_g), it holds $G^iE \sim G^iF$ for all i because dd_e and dd_f are set to ω. Hence the normal form for the BPP from the example must contain furthermore following rules: $A_2 \xrightarrow{a} X_{1,2}Z_2$, $X_{1,2} \xrightarrow{e} X_{2,2}$, $X_{2,2} \xrightarrow{f} X_{1,2}$, $Z_2 \xrightarrow{g} Z_2Z_2$, $Z_2 \xrightarrow{g} X'_{1,1}$, $X'_{1,1} \xrightarrow{e} X'_{1,1}$, $X'_{1,1} \xrightarrow{f} X'_{1,1}$.

When a DD-function, which can grow unlimitedly, is set to ω, the BPP enters the next stage. In configurations, there can be variables from previous stages which are unimportant with respect to bisimilarity because they increase only DD-functions already set to ω.