Complexity of some Bisimilarity Problems between BPP and BPA or Finite-State System

Martin Kot*

Center for Applied Cybernetics, Dept of Computer Science,
Technical University of Ostrava, Czech Republic
martin.kot@vsb.cz

Supervisor(s): Petr Jančar, Technical University of Ostrava
Keywords: Bisimulation Equivalence, Basic Parallel Processes, Basic Process Algebra, Regularity, Complexity

Abstract. I present my research concerning complexity of bisimilarity between BPP and other simple process rewrite systems, specifically BPA and finite-state systems. The results are based on some general notions introduced by Jančar in [1] where he has shown PSPACE-completeness of bisimilarity on BPP. Some of presented results are joint work with Zdeněk Sawa or my supervisor Petr Jančar.

1 Research Area

Failures of some systems may have serious consequences. Hence, designers of such systems want to ensure the correctness of the design. They perform so called verification when they check correctness of the implementation with respect to the given specification. Main approaches that allow to ensure the correctness for all possible behaviors of the systems are theorem proving, equivalence checking and model checking.

In equivalence checking we have given two descriptions of systems and the question is whether they are equivalent with respect to some equivalence. Usually we want to ensure that the behavior of the system is the same as the behavior of the specification.

Many types of models are used for the description of systems. The equivalence checking problems are undecidable for models with great expressive power. Other models do not allow to describe many aspects of the real systems. Many researchers concentrate on the question which equivalence checking problems are decidable. Moreover, the exact computational complexity of the decidable problems plays important role. Due to the complexity, many algorithms may not be used in practice for real-life instances.

* The author is supported by the Ministry of Education of the Czech Republic, Project 1M0567.

2 Directions of the work

Mayr defined Process Rewrite Systems (PRS) in [2]. PRS unify many previously known formalisms. They are defined as follows. Let $\mathcal{A} = \{a, b, c, \ldots\}$ be the (possibly infinite but countable) set of atomic actions and $Var = \{X, Y, Z, \ldots\}$ be the countable set of process variables. Process terms are defined by the following abstract grammar

$$P ::= \varepsilon \mid X \mid P_1.P_2 \mid P_1||P_2$$

where ε is the empty term, X is a process variable, '.' denotes sequential composition and '||' parallel composition. Sequential composition is associative, both compositions are commutative.

Process rewrite system is a finite set Δ of rules of the form $t_1 \xrightarrow{a} t_2$ where t_1, t_2 are process terms and $a \in \mathcal{A}$. We can define different types of subclasses of PRS based on four classes of process terms:

1 - terms consisting of a single process variable
S - terms consisting of ε, a single variable, or a sequential composition of process variables
P - terms consisting of ε, a single variable, or a parallel composition of process variables
G - terms without any restrictions

Let $\alpha, \beta \in \{1, \mathcal{S}, \mathcal{P}, \mathcal{G}\}$ be classes of terms such that $\alpha \subseteq \beta$. (α, β)-PRS is defined as a finite set Δ of rules $l \xrightarrow{a} r$ where the term l is from class α ($l \neq \varepsilon$) and r is from β. (α, β)-PRS form the whole hierarchy. We are interested in three simple classes at the bottom of the hierarchy: $(1, 1)$-PRS – finite-state systems, $(1, \mathcal{S})$-PRS – Basic Process Algebras and $(1, \mathcal{P})$-PRS – Basic Parallel Processes.

There are many possible equivalences defined which may be used for equivalence checking. The most important equivalences were organized by van Glabbek into the hierarchy called linear time – branching time spectrum. One of the most studied equivalences is bisimulation equivalence because it has found its way into many practical applications.

My research in the area of equivalence checking is focused on bisimulation problems on BPP. Deciding bisimilarity of BPP is known to be PSPACE-complete [1]. But it is worth to study also complexity of bisimilarity with some other process rewrite systems. Sometimes could be handy to use one model for specification and some other for implementation. In the section 4 are mentioned the following results:

- Bisimilarity on normed BPP can be decided in time $O(n^3)$ (joint work with Petr Jančar)
- Bisimulation equivalence of a BPP and a finite-state system can be decided in polynomial time (joint work with Zdeněk Sawa)
- Regularity (existence of bisimilar finite-state system) of BPP is PSPACE-complete
- Deciding existence of bisimilar BPA to a given BPP is PSPACE-complete (joint work with Petr Jančar and Zdeněk Sawa)

3 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 $\beta \in V^*$.

We can define a BPP as a labelled Petri Net in which each transition has a unique input place. Formally, it is a tuple $\mathcal{N} = (P, T, F, A, l)$ where P is finite set of *places*, T is finite set of *transitions*, $F : (P \times T \cup T \times P) \to \mathbb{N}$ is a *flow function*, A is a finite set of labels, and $l : T \to A$ is a *labelling function*. A *marking* is a function $M : P \to \mathbb{N}$ which associates to each place a finite number of *tokens*. If $M(p) \geq F(p, t)$ for each place p then a transition t is *enabled* at marking M. An enabled transition t may be fired from M, producing a marking M' defined by $M'(p) = M(p) - F(p, t) + F(t, p)$. This is denoted by $M \xrightarrow{t} M'$. We can associate a LTS to a Petri net $\mathcal{N}$ where the set of states is the set of all markings, A is the set of labels, and $M \xrightarrow{a} M'$ iff there is a transition t such that $l(t) = a$ and $M \xrightarrow{t} M'$.

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

The set of *DD-functions* [1] for LTS $\mathcal{S} = (S, \mathcal{A}, \longrightarrow)$ is defined inductively as follows:

- For each $a \in \mathcal{A}$ the function $dd_a : S \to \mathbb{N}_\omega$, defined as $dd_a(s) = \min\left\{ dist(s, s') \mid \neg\exists s'' : s' \xrightarrow{a} s'' \right\}$, is a DD-function.
- Let $\mathcal{F} = (d_1, \ldots, d_k)$ 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 $dd_{(a,\mathcal{F},\delta)}(t) = \min\left\{ dist(t, t') \mid d_i(t') < \omega \text{ for all } i, \text{ and } \mathcal{F}(t'') - \mathcal{F}(t') \neq \delta \text{ for all } t' \xrightarrow{a} t'' \right\}$, is a DD-function.

Informally, the value $d(s)$ of DD-function d 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 DD-function d. It was shown in [1], that for BPP the set of DD-functions is always finite and can be effectively constructed.

4 Achieved results

In 2003 Jančar presented a PSPACE algorithm deciding bisimilarity on BPP [1]. He has shown, that the set of all DD-functions for BPP may be computed in PSPACE and that two BPP processes are bisimilar if and only if all DD-functions give the same values on those processes.

4.1 Bisimilarity on normed Basic Parallel processes

In the case of normed BPP, a polynomial time algorithm was presented in [3] but authors did not analyze the complexity of their algorithm more precisely. When applied to the normed BPP, Jančar's algorithm from [1] also works in polynomial time.

The problem *nBPP-bisim* can be formulated as follows: Given a normed BPP system and two markings M, M', is $M \sim M'$?

In [4] we have expressed Jančar's algorithm more precisely in the version only for normed BPP. Then we provided a complexity analysis yielding the upper bound $O(n^3)$.

Our algorithm performs a *stepwise decomposition* of the set T of transitions. It starts with the (initial) decomposition according to the transition labels. Each step of the algorithm refines current decomposition of T according to the changes which the rules cause on the functions $\text{NORM}_{\text{PRE}(T')}$, for all current decomposition classes T'.

This algorithm surely finishes, with a decomposition denoted *decomp*(T). Then M and M' are bisimilar iff $\text{NORM}_{\text{PRE}(T')}(M) = \text{NORM}_{\text{PRE}(T')}(M')$ for each class T' in *decomp*(T). The decomposition problem is crucial for us because *nBPP-bisim* problem can be easily reduced to the decomposition.

At most $2l - 1$ different decomposition classes could appear in a stepwise decomposition of a set with the size l. Hence in our case the number of decomposition classes is in $O(n)$. According each decomposition class T we compute coefficients of linear function $\text{NORM}_{\text{PRE}(T)}$, then we compute changes on this function caused by transitions and we refine our decomposition. All mentioned steps are done sequentially $O(n)$ times. Computing coefficients, changes, as well as refining decomposition can be done in $O(n^2)$. It follows that the whole algorithm is in $O(n^3)$.

4.2 Bisimilarity of a BPP and a finite state system

In this section we will mention the result from [5] which shows that so called *BPP-FS-bisim* problem can be solved in polynomial time, concretely in $O(n^5)$.

The problem *BPP-FS-bisim* is formulated as follows: Given a BPP Σ together with an initial marking M and a FS Δ with an initial state s, is $M \sim s$?

The algorithm which solves *BPP-FS-bisim* problem works on the disjoint union of Σ and Δ. The DD-functions $D_1, D_2, \ldots$ are computed. To those functions correspond equivalences $\equiv_0, \equiv_1, \equiv_2, \ldots$ on the set of states S of corresponding LTS where $s \equiv_i s'$ iff $D_j(s) = D_j(s')$ for each $1 \leq j \leq i$. There is only a finite number of different functions we can add during the computation. Hence we can find a fixpoint $\equiv_i$ on sequence of equivalences satisfying $\equiv_i = \equiv_j$ for every $j > i$. It could be shown that $M \sim s$ iff $M \equiv_i s$.

Analyzing the complexity we get that the algorithm runs in $O(n^5)$.

4.3 Regularity of BPP

The problem *BPP-regularity* is formulated as follows: Given a BPP Σ together with an initial marking M, is there some FS Δ with an initial state s such that $M \sim s$?

In [6] I have shown an algorithm deciding *BPP-regularity* problem in polynomial space. This problem was known to be PSPACE-hard hence PSPACE-completeness was achieved.

The algorithm looks for a cycle in the BPP-net such that it generates tokens to a place with finite positive coefficient of some DD-function d. Moreover, the cycle must be markable without setting d to infinity and transition in the cycle may not set d to infinity.

4.4 Bisimilarity between BPA and BPP

Currently, I'm working on some problems related to bisimilarity between BPA and BPP. It is a joint work with Jančar and Sawa. The results mentioned here were not published yet.

Bisimilarity between BPA and BPP is known to be decidable. But no complexity bound was provided. Actually we have an algorithm working probably in 3-EXPTIME. But we are trying to improve this complexity bound.

We can also define a problem called *BPA-ity* as follows: Given a BPP Σ together with an initial marking M, is there some BPA Δ with an initial configuration α such that $M \sim \alpha$?

We have shown that the problem is PSPACE-complete. The PSPACE-hardness can be proved by reduction from the problem of regularity of a BPP which is known to be PSPACE-hard. The algorithm deciding *BPA-ity* problem checks in PSPACE two conditions that are necessary and sufficient for a BPP to have a bisimilar BPA.

Suppose an LTS $\mathcal{S} = (S, \mathcal{A}, \longrightarrow)$ with an initial state s. We say that DD-functions d, d' are *independently large* iff for every $C \in \mathbb{N}$ there is a reachable

state $r \in S$ such that $C < d(r) < \omega, C < d'(r) < \omega$ and there is a transition $r \xrightarrow{a} q$ into some state q such that $d(q) - d(r) \neq d'(q) - d'(r)$.

The first condition for existence of bisimilar BPA is the following: Given a LTS $\mathcal{S}$ corresponding to a BPP, there is no pair of independently large DD-functions.

We say that DD-function d *overgrows* DD-function d' if $\forall C_1, C_2 \in \mathbb{N}$ there is a reachable state q such that all the following conditions holds: $C_1 < d(q) < \omega$, there is a transition leading from q to some state p and changing d and not setting d' to infinity, from q and p can be done C_2 times d-reducing step not changing d'.

The second condition for existence of bisimilar BPA is the following: Given a LTS $\mathcal{S}$ corresponding to a BPP, there is not any pair of DD-functions d, d' such that d overgrows d'.

Our 3-EXPTIME algorithm deciding bisimilarity between BPA and BPP uses the algorithm for *BPA-ity* problem as a subroutine. We check if there is a bisimilar BPA to the BPP. In the positive case we can construct a BPA in a normal form bisimilar to the given BPP. Then we can use a standard algorithm for checking bisimilarity on BPA. Our algorithm is better than previously known one from [7] which used as subroutine more complex algorithm checking bisimilarity on PDA. But we would like to avoid using general algorithm for checking bisimilarity on BPA and in this way to improve the complexity bound.

In the similar way as *BPA-ity* we can define *BPP-ity*. Given a BPA we ask if there is a bisimilar BPP. We have shown that this problem is also PSPACE-hard by reduction from the problem of regularity of BPA. But we do not have any algorithm deciding the *BPP-ity* problem yet.

References

1. Jančar, P.: Strong bisimilarity on basic parallel processes is PSPACE-complete. In: Proc. 18th LiCS, IEEE Computer Society (2003) 218–227
2. Mayr, R.: Process rewrite systems. Information and Computation **156**(1) (2000) 264–286
3. Hirsfeld, Y., Jerrum, M., Moller, F.: A polynomial-time algorithm for deciding bisimulation equivalence of normed basic parallel processes. Mathematical Structures in Computer Science **6** (1996) 251–259
4. Jančar, P., Kot, M.: Bisimilarity on normed basic parallel processes can be decided in time $O(n^3)$. In Bharadwaj, R., ed.: Proceedings of the Third International Workshop on Automated Verification of Infinite-State Systems – AVIS 2004. (2004)
5. Kot, M., Sawa, Z.: Bisimulation equivalence of a BPP and a finite-state system can be decided in polynomial time. ENTCS **138** (2005) 49–60 Proceedings of INFINITY'04.
6. Kot, M.: Regularity of BPP is PSPACE-complete. In: Proceedings of the 3rd annual workshop WOFEX 2005, VŠB-TUO FEECS (2005) 393–398
7. Jančar, P., Kučera, A., Moller, F.: Deciding bisimilarity between BPA and BPP processes. In: Proceedings of CONCUR 2003. Volume 2761 of LNCS., Springer-Verlag (2003) 159–173