Complexity of deciding bisimilarity of nBPP and bisimilarity of BPP with finite-state system

Martin Kot

Department of Computer Science, FEI, Technical University of Ostrava,
17. listopadu 15, 708 33, Ostrava-Poruba, Czech Republic
martin.kot@vsb.cz

Abstract. Jančar has recently shown that bisimilarity on Basic Parallel Processes (BPP) can be decided in polynomial space (presented at LiCS 2003). In this paper are summarized two results which use general techniques from Jančar's paper. First result by Kot and Jančar (presented at AVIS 2004) shows that bisimilarity on normed basic parallel processes can be decided in $O(n^3)$. The second result by Kot and Sawa (submitted to Infinity 2004) shows that bisimilarity of given BPP and finite-state system can be decided in polynomial time, concretely in $O(n^5)$.

Keywords: basic parallel process, normed basic parallel process, finite state system, bisimilarity, bisimulation equivalence, polynomial complexity.

1 Introduction

Bisimulation equivalence, also called bisimilarity, is one of fundamental behavioral equivalences studied in the area of formal verification. Up-to-date overview of results of research concerning decidability and complexity of bisimilarity for various models is in [7]. We focus on well known model—basic parallel process and its special form—normed BPP.

Jančar has shown in [3] that the problem of deciding bisimilarity on BPP is in *PSPACE* which combined with previous results ensures that the problem is *PSPACE*-complete. In the case of normed BPP, there are algorithms for deciding bisimilarity in polynomial time. First was presented in [2]. Jančar's algorithm from [3] when applied to normed BPP is in polynomial time too. This was shown in [4] and complexity bound $O(n^3)$ was suggested. Some basic ideas from this will be in section 3.

Other problem in the area of formal verification is to decide whether a process from a given class of systems is bisimilar to a given finite-state system. In the case of BPP, in [1] was shown that this problem is P-hard and in [5] that it is in *PSPACE*. The most recent paper [6] shows that this problem can be decided in polynomial time, concretely in $O(n^5)$. Basic ideas from this paper will be discussed in section 4.

In section 2 will be some definitions and notations needed for understanding both sections 3 and 4.

2 Basic definitions and notation

Bisimilarity (i.e., bisimulation equivalence) is defined for *labelled transition systems* (LTSs). An LTS is a tuple $(S, A, \{\xrightarrow{a}\}_{a\in A})$ where S is a (possibly infinite) set of *states*, A is a set of *actions* (or transition *labels*), and $\xrightarrow{a} \subseteq S \times S$ for each $a \in A$. We use infix notation $r \xrightarrow{a} r'$.

Given an LTS $(S, A, \{\xrightarrow{a}\}_{a\in A})$, *bisimulation equivalence* is the maximal symmetric relation B on S satisfying: if $(r_1, r_1') \in B$ and $r_1 \xrightarrow{a} r_2$ then there is r_2' such that $r_1' \xrightarrow{a} r_2'$ and $(r_2, r_2') \in B$. States $s_1, s_2 \in S$ are *bisimilar*, written $s_1 \sim s_2$, iff there exists a bisimulation $\mathcal{R}$ such that $(s_1, s_2) \in \mathcal{R}$.

A BPP can be defined as a tuple $(P, Tr, \text{PRE}, F, \lambda)$ where P is a finite set of *places*, Tr is a finite set of *transitions*, $\text{PRE} : Tr \to P$ is a function assigning an input place to every transition, $F : (Tr \times P) \to \mathbb{N}$ is a function assigning output places to each transition, and $\lambda : Tr \to \mathcal{A}$ is a labelling function. The set of output places of the transition t we will denote by $\sigma(t) = \{p \mid F(t, p) > 0\}$.

Let $P = \{p_1, p_2, \ldots, p_k\}$ be a set of places. A *marking* is a function $M : P \to \mathbb{N}$ which assigns number of tokens to each place. Marking M can be viewed as a vector $(x_1, x_2, \ldots, x_k)$ where $x_i \in \mathbb{N}$ and $x_i = M(p_i)$. We use $\mathcal{S}_\Sigma$ to denote the set of all markings.

A transition t is *enabled* in a marking M iff $M(\text{PRE}(t)) > 0$. Performing a transition, written $M \xrightarrow{t} M'$, means

$$M'(p) = \begin{cases} M(p) - 1 + F(t, p) & \text{if } p = \text{PRE}(t) \\ M(p) + F(t, p) & \text{otherwise} \end{cases}$$

A BPP is called *normed*, denoted nBPP, iff from each marking we can reach en empty marking (i.e., $M(p_i) = 0$ for each $p_i \in \mathcal{S}_\Sigma$) by performing a sequence of transitions. An LTS $(S, \mathcal{A}, \longrightarrow)$ corresponds to a BPP where $S = \mathcal{S}_\Sigma$ and $M \xrightarrow{a} M'$ iff there is some $t \in Tr$ such that $\lambda(t) = a$ and $M \xrightarrow{t} M'$.

We define a *finite state system* (FS) as a special type of a BPP where is exactly one token which can not be duplicated or removed. We use $\mathcal{S}_\Delta = \{s_1, s_2, \ldots, s_k\}$ to denote set of markings (states) of FS.

Let us have an LTS $(S, \mathcal{A}, \longrightarrow)$ and some set $R \subseteq S$. We can define a *distance function* $D_R : S \to \mathbb{N}_\omega$ as follows: Let $W = \{w \in \mathcal{A}^* \mid s \xrightarrow{w} s' \wedge s' \in R\}$. Then $D_R(s) = \min(\{|w| \mid w \in W\} \cup \{\omega\})$.

A crucial notion, introduced in [3] and used in both following sections, is the notion of *DD-functions*. They are defined inductively. For every transition label a a function dd_a which, for every place s, gives the "distance to disabling" transitions with label a is a DD-function. Formally, dd_a is defined as $dd_a(s) = \min\left\{ \mathit{dist}(s, s') \mid \neg\exists s'' : s' \xrightarrow{a} s'' \right\}$. Given a tuple of DD-functions $\mathcal{F} = (d_1, d_2, \ldots, d_k)$, each transition $s \xrightarrow{a} s'$ determines a *change* $\mathcal{F}(s') - \mathcal{F}(s)$, denoted δ, which is a k-tuple of values from $\{-1\} \cup \mathbb{N}_\omega \cup \{\omega\}$. For each triple $(a, \mathcal{F}, \delta)$, the function $dd_{(a,\mathcal{F},\delta)}$ (distance to disabling the action a causing the change δ of $\mathcal{F}$) is also a DD-function, defined by

$$dd_{(a,\mathcal{F},\delta)}(s) = \min\left\{ \mathit{dist}(s, s') \mid \forall s'' : \text{ if } s' \xrightarrow{a} s'' \text{ then } \mathcal{F}(s'') - \mathcal{F}(s') \neq \delta \right\}.$$

All DD-functions are *bisimulation invariant*, i.e., if s and s' are bisimilar then $d(s) = d(s')$ for all DD-functions d. So equality of the values of all DD-functions is a necessary condition for two places being bisimilar. In the case of BPP this condition is also sufficient. In [3] was shown that, for any BPP, DD-functions coincide with so called 'norms':

Given $Q \subseteq \mathcal{S}_\Sigma$, we define function $\textsc{norm}_Q$ by

$$\textsc{norm}_Q(M) = \min \left\{ \mathit{dist}(M, M') \mid M'(p) = 0 \text{ for each } p \in Q \right\}.$$

Each $\textsc{norm}_Q$ is a linear function, i.e, for each $p \in P_\Sigma$ there is $c_p \in \mathbb{N}_\omega$ such that

$$\textsc{norm}_Q(M) = \sum_p c_p \cdot M(p)$$

3 Bisimilarity on normed Basic Parallel Processes can be decided in time $O(n^3)$

In this section will be shown that so called NBPP-BISIM problem can be solved in $O(n^3)$. This result was published in [4] and presented on AVIS 2004—Third International Workshop on Automated Verification of Infinite-state Systems which took place in Barcelona as affiliated workshop of ETAPS 2004 (European Joint Conferences of Theory And Practice of Software). The proceedings will be published electronically on ENTCS.

Problem NBPP-BISIM can be formulated as follows: Given a normed BPP system and two markings M, M', is $M \sim M'$?

Our algorithm which solves NBPP-BISIM problem performs a *stepwise decomposition* of the set *Tr* of transitions, i.e., it constructs a sequence of decompositions of *Tr*, where each new one refines the old one. We start with the (initial) decomposition according to the transition labels. Each step of our algorithm refines current decomposition of *Tr* according to the changes which the rules cause on the functions $\textsc{norm}_{\mathrm{PRE}(T')}$, for all current decomposition classes T'.

This algorithm surely finishes, with a decomposition denoted *decomp*(*Tr*). Then M and M' are bisimilar iff $\textsc{norm}_{\mathrm{PRE}(T')}(M) = \textsc{norm}_{\mathrm{PRE}(T')}(M')$ for each class T' in $\mathit{decomp}(T)$. The decomposition problem is crucial for us because NBPP-BISIM problem can be easily reduced to decomposition.

We assume that instance of our problem is encoded as lists of places and transitions where encoding of each transition t contains a list of all pairs of the form $(p, F(t, p))$ where where $F(t, p) > 0$. We assume that numbers are encoded in binary. By n we will denote the size in bits of an instance in such encoding.

Now we will show time complexity of our algorithm. At first, we need know how many times our decomposition can be refined. Using mathematical induction, it can be easily shown that 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 $\textsc{norm}_{\mathrm{PRE}(T)}$, then we compute changes on this function caused by transitions and we refine our decomposition. All mentioned steps are done sequentially $O(n)$ times. In following paragraphs we show that computing coefficients, changes, as well as refining decomposition can be done in $O(n^2)$. It follows that the whole algorithm is in $O(n^3)$.

Norm of the set Q means how many transitions are at least needed to remove all tokens from the set Q. Coefficient of linear function for each place means the length of some sequence of transitions which removes from set Q one token located in this place. In fact, for each place we look for 'optimal' transition which is first in such sequence.

Places out of the set Q have obviously coefficients 0. In the case of normed BPP, all coefficients are finite. There must be at least one transition which does not have any output place at all or has each output place out of Q. Input places of all such transitions have coefficients 1.

For each transition t which removes a token from a place with unknown coefficient and adds tokens only to places with known coefficients we can compute a candidate d_t for coefficient $c_{\text{PRE}(t)}$. The minimal among already computed candidates d_{t_i} is declared as final coefficient $c_{\text{PRE}(t_i)}$. We repeat this process and stop it when we know coefficients for all places.

Now we show that number of bits needed for representation of each coefficient of NORM_Q is in $O(n)$. Without loss of generality we can suppose that coefficients are indexed according to order in which are computed and corresponding places and 'optimal' transitions have the same indexes. Other transitions are indexed in arbitrary order. Hence $c_1 = 1$ and the size of c_1 in bits is obviously less than the size of used 'optimal' transition t_1.

Now we use mathematical induction. Suppose that the size of each d_{t_i} in bits is less then the size of representation of transitions t_1 to t_i. Using this we will show, that the size of $d_{t_{i+1}}$ in bits is less then the size of the representation of transitions t_1 to t_{i+1}. Let $\text{SIZE}(x)$ be the size in bits of representation of x.

From our algorithm follows that each c_j has to be known and finite when we compute

$$d_{t_{i+1}} = 1 + \sum_{p_j \in \sigma(t_{i+1})} c_j \cdot F(t_{i+1}, p_j)$$

Hence each c_j is one from c_1 to c_i. Each $F(t_{i+1}, p_j) \cdot c_j$ could be written in $\text{SIZE}(F(t_{i+1}, p_j)) + \text{SIZE}(c_j)$ bits. The sum of all such products can be written in the size of maximal of them plus some number less then their count (overflow caused by addition). This size is less then

$$\text{SIZE}(\max\{c_j | 1 \leq j \leq \min\{i, |\mathcal{S}_\Sigma|\}\}) + \sum_{j=1}^{\min\{i, |\mathcal{S}_\Sigma|\}} \text{SIZE}(F(t_{i+1}, p_j))$$

bits. The second summand (the sum) is less then count of bits of representation of $(i+1)$-th transition. Using induction hypothesis, maximal c_j can be written in the count of bits needed for first i transitions. Therefore d_{i+1} (and hence c_{i+1} too) could be written in the space needed for representations of transitions 1 to $i+1$, thus in $O(n)$ bits.

Now we can show that all coefficients of $\textsc{norm}_Q$ are computed together in $O(n^2)$. In our algorithm doing this, the most time-consuming step is computing all d_i. In computation of this, multiplications are more time-consuming than additions. Hence, it suffices to show that aggregated complexity of all multiplications is in $O(n^2)$. Note that computing of $x \cdot y$ is in $O(\textsc{size}(x) \cdot \textsc{size}(y))$.

In our algorithm, each d_i is computed only once. During computation of d_i we need to determine all products $F(t_i, p_j) \cdot c_j$ where j goes over places to which t_i gives at least one token. We know that each c_j is in $O(n)$ bits. Hence one product is computed in $O(n \cdot \textsc{size}(F(t_i, p_j)))$. If we sum complexities of such products for all transitions and places to which transitions give tokens, we get the aggregated complexity of all multiplications

$$O(\sum_{i,j}(n \cdot \textsc{size}(F(t_i, p_j)))) = O(n \cdot \sum_{i,j} \textsc{size}(F(t_i, p_j))) = O(n^2)$$

.

Changes δ_t are computed using very similar expression as computing coefficients of norm. Hence, it can be similarly shown that computing all δ_t for one norm can be done in time $O(n^2)$. The overall size of all δ_t have to be in $O(n^2)$ too. It follows that the decomposition according to δ_t can be done in $O(n^2)$.

4 Bisimulation equivalence of a BPP and a finite state system can be decided in polynomial time

In this section we show that so called BPP-FS-BISIM problem can be solved in polynomial time, concretely in $O(n^5)$. This result is described in paper [6] which is submitted to Infinity 2004—affiliated workshop of Concur 2004. Hence it was neither presented nor published yet.

The problem BPP-FS-BISIM is formulated as follows: Given a BPP Σ together with a marking $M \in \mathcal{S}_\Sigma$ and a FS Δ with a state $s \in \mathcal{S}_\Delta$, is $M \sim s$?

Let a BPP $\Sigma = (P_\Sigma, Tr_\Sigma, \textsc{pre}_\Sigma, F_\Sigma, \lambda_\Sigma)$, a FS $\Delta = (P_\Delta, Tr_\Delta, \textsc{pre}_\Delta, F_\Delta, \lambda_\Delta)$ and a pair of their markings from $\mathcal{S}_\Sigma$ and $\mathcal{S}_\Delta$ be an instance of BPP-FS-BISIM. A BPP $(P, Tr, \textsc{pre}, F, \lambda)$ will be a disjoint union of Σ and Δ and $\mathcal{S} = \mathcal{S}_\Sigma \cup \mathcal{S}_\Delta$.

Our algorithm deciding BPP-FS-BISIM works in following way. Assume that norm functions $D_1, D_2, \ldots, D_{i-1}$ are constructed (at the beginning we have an empty list of functions). We choose $s \in \mathcal{S}_\Delta$ and $t \in Tr$. For the chosen s we define I as maximal set of functions D_i which are finite on s. We can find the set Q of markings of Σ which give infinite values on functions from I. Then we partition the set of transitions Tr according to 'change' values on functions from I. The partition we will denote by $\mathcal{T}$.

Let $T \in \mathcal{T}$ be the equivalence class containing the transition t chosen at the beginning of the step. We define the function D_i as a distance to the set of markings where every $t' \in T$ is disabled and where every $D_j \in I$ has a finite value. This can be computed as norm of $Q \cup \textsc{pre}(T)$.

In this way we create functions $D_1, D_2, \ldots$. To those functions correspond equivalences $\equiv_0, \equiv_1, \equiv_2, \ldots$ on the set S where $s \equiv_i s'$ iff $D_j(s) = D_j(s')$ for each $1 \leq j \leq i$. When computing D_i, we have given T and Q. We don't add two functions with the same T and Q because both refine our partition in the same way. Since the T and Q are subsets of finite sets, there is only a finite number of different functions we can add. 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 $M \equiv_i s$ and we have solution of BPP-FS-BISIM problem.

Now we show the complexity of presented algorithm. For each $s \in \mathcal{S}_\Delta$ we can compute the sequence of distance functions. In every step of our algorithm norm is computed for some Q and T. The set Q can only grow and the partition of transitions can be only refined during run of algorithm. At least one of them must be different when we add new norm function. We can do at most $|Tr|$ steps when the value of Q remain unchanged. Since $Q \subseteq P$ and every partition has at most $|Tr|$ classes, we obtain the result the complexity $O(|P_\Delta| \cdot |Tr| \cdot (|P|))$. This is surely in $O(n^3)$. So we $O(n^3)$ times compute norm function and do the decomposition. Similarly as in section 3, it could be shown that each norm function and changes on norm function caused by transitions can be computed in $O(n^2)$. It follows that our algorithm runs in $O(n^5)$.

5 Conclusion

We have shown that complexity of deciding bisimilarity for normed BPP is in $O(n^3)$ and that bisimilarity of BPP and finite-state system can be decided in polynomial time (in $O(n^5)$). In future work we want to prove that regularity of BPP (given a BPP, is there some finite-state system bisimilar with BPP?) is *PSPACE*-complete. Now it is only known to be decidable and *PSPACE*-hard.

References

1. J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is p-complete. *Formal Aspects of Computing*, 4(6A):638–648, 1992.
2. Y. Hirsfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimulation equivalence of normed basic parallel processes. *Mathematical Structures in Computer Science*, 6:251–259, 1996.
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 and M. Kot. Bisimilarity on normed basic parallel processes can be decided in time $O(n^3)$. In *Proceedings of the Third International Workshop on Automated Verification of Infinite-State Systems – AVIS 2004*.
5. P. Jančar, A. Kučera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. *Theoretical Computer Science*, 258(1-2):409–433, 2001.
6. M. Kot and Z. Sawa. Bisimulation equivalence of a BPP and a finite state system can be decided in polynomial time. Submited to Infinity 2004.
7. 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/.