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

Petr Jančar [1], Martin Kot [2,3]

Department of Computer Science
Technical University of Ostrava
Ostrava-Poruba, Czech Republic

Abstract

A recent paper by Jančar (presented at LiCS 2003) demonstrated that bisimilarity on Basic Parallel Processes (BPP) can be decided in polynomial space. Here we explore a (more detailed) version of the respective algorithm when applied to the subclass called normed BPP (nBPP), and show that in this case the algorithm runs in polynomial time; we provide a complexity analysis yielding the upper bound $O(n^3)$. This strenghtens a result by Hirshfeld, Jerrum and Moller (1996) who showed a different polynomial-time algorithm for nBPP; they did not analyse the complexity of their algorithm more precisely but it does not seem to allow the above mentioned upper bound.

Key words: bisimulation equivalence, normed basic parallel processes, polynomial complexity

1 Introduction

Bisimilarity is a fundamental behavioural equivalence on (reactive) concurrent systems, and its computational complexity is a natural subject of research in the area of verification. Surveys of this research focusing on infinite-state systems can be found, e.g., in [1] and [5].

Bisimilarity (i.e., bisimulation equivalence) is defined for *labelled transition systems* (LTSs). An LTS can be viewed as a tuple $(S, A, \{\stackrel{a}{\longrightarrow}\}_{a\in A})$ where S is a (possibly infinite) set of *states*, A is a set of *actions* (or transition *labels*), and $\stackrel{a}{\longrightarrow} \subseteq S \times S$ for each $a \in A$. We use infix notation $r \stackrel{a}{\longrightarrow} r'$ and read, e.g., "state r allows to perform action a resulting in state r'" or "r' is an a-successor of r" (or similarly).

[1] Email: petr.jancar@vsb.cz
[2] Email: martin.kot@vsb.cz
[3] Both authors supported by the Grant Agency of the Czech Republic, No. 201/03/1161

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

Given an LTS $(S, A, \{\stackrel{a}{\longrightarrow}\}_{a\in A})$, *bisimulation equivalence* is the maximal symmetric relation B on S satisfying: if $(r_1, r_1') \in B$ and $r_1 \stackrel{a}{\longrightarrow} r_2$ then there is r_2' such that $r_1' \stackrel{a}{\longrightarrow} r_2'$ and $(r_2, r_2') \in B$. (Informally: Two states are bisimilar iff any action from one of them can be matched by the same action from the other so that the resulting states are again bisimilar.)

Basic Parallel Processes (BPP) constitute one of the basic classes of infinite-state systems; they arise as a natural extension of (nondeterministic) finite automata with a parallel operator, or, equivalently, as processes generated by commutative context-free grammars (in Greibach normal form). The respective LTS, corresponding to a given context-free grammar in Greibach normal form, has finite sequences of variables (also called nonterminals) as states; a (grammar) rule

$$X \stackrel{a}{\longrightarrow} Y_1 Y_2 \dots Y_n$$

allows to perform action a in any state α containing X, which results in the state β arising from α by replacing one occurrence of X with $Y_1 Y_2 \dots Y_n$.

We easily note that two states (sequences) with the same numbers of (occurrences of) each variable (i.e., with the same Parikh images) are bisimilar and can be identified; so the variables in a 'sequence' can be viewed as composed by a (commutative) parallel operator (therefore we used the notion of *commutative* context-free grammars).

We now look at the following decision problem, called *BPP-problem*:

Instance: a BPP-system, i.e., a finite set of Greibach normal form rules and two states (i.e., sequences of variables) α, β.
Question: are α and β bisimilar (i.e., related by the bisimulation equivalence) ?

A natural subclass of BPP-systems is the class of normed BPP-systems, denoted nBPP: a *BPP-system* is *normed* iff each variable can derive a terminal word, i.e., iff any state can reach the empty state (i.e., the empty sequence ε) by performing a sequence of actions. (E.g., the system $\{X \stackrel{a}{\longrightarrow} X\}$ is not normed but the system $\{X \stackrel{a}{\longrightarrow} X,\ X \stackrel{b}{\longrightarrow} \varepsilon\}$ is.)

The BPP-problem restricted to nBPP will be called the *nBPP-problem*.

Christensen, Hirshfeld and Moller [2] showed that the BPP-problem is decidable but only a nonprimitive recursive upper bound was deduced from the respective algorithm. Hirshfeld, Jerrum and Moller [3] then developed a specific algorithm for nBPP which works in polynomial time; they did not analyse the degree of the polynomial but it does not seem to fit in $O(n^3)$.

Jančar [4] developed another algorithm for the BPP-problem which he showed to work in polynomial space. (Combined with a result by Srba [6], PSPACE-completeness of the BPP-problem has thus been established.)

Here we present (a more detailed version of) Jančar's algorithm in the case of nBPP, and show that it works in time $O(n^3)$. The result holds (even) for the

case when the numbers of occurrences of variables in the right-hand sides of rules and in the input states are given in binary.

2 Basic notions and ideas

Here we introduce some necessary notions and sketch the ideas on which the algorithm is based.

Relations $\stackrel{a}{\longrightarrow}$ are naturally extended to relations $\stackrel{w}{\longrightarrow}$ for sequences of actions w. Given an LTS $(S, A, \{\stackrel{a}{\longrightarrow}\}_{a\in A})$, we define the *distance from state s to state t* by

$$dist(s,t) = \min\{\, length(w) \mid w \in A^* \text{ and } s \stackrel{w}{\longrightarrow} t \,\}.$$

We stipulate $\min\emptyset = \omega$, using ω as a symbol for infinite amount; we put $\omega - n = \omega$ for each $n \in \mathbb{N} \cup \{\omega\}$ (where $\mathbb{N}$ denotes the set of natural numbers $\{0, 1, 2, \ldots\}$).

A crucial notion (introduced in [4]) is the notion of *DD-functions*; they are defined inductively. First, for every action a we define a function dd_a which, for every state s, gives the "distance to disabling" the action a:

$$dd_a(s) = \min\{\, dist(s,t) \mid \neg\exists t' : t \stackrel{a}{\longrightarrow} t' \,\}.$$

Given a tuple of (so far defined) DD-functions $\mathcal{F} = (d_1, d_2, \ldots, d_k)$, we observe that each transition $s \stackrel{a}{\longrightarrow} t$ determines a *change* of $\mathcal{F}$, denoted $\mathcal{F}(t) - \mathcal{F}(s)$, which is a k-tuple of values from $\{-1\} \cup \mathbb{N} \cup \{\omega\}$:

$$\mathcal{F}(t) - \mathcal{F}(s) = (d_1(t) - d_1(s), d_2(t) - d_2(s) \ldots, d_k(t) - d_k(s)) .$$

For each triple $(a, \mathcal{F}, \delta)$, where a is an action, $\mathcal{F}$ is a k-tuple of DD-functions, and δ is a k-tuple of values from $\{-1\}\cup\mathbb{N}\cup\{\omega\}$, 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

$$dd_{(a,\mathcal{F},\delta)}(s) = \min\{\, dist(s,t) \mid \forall r: \text{ if } t \stackrel{a}{\longrightarrow} r \text{ then } \mathcal{F}(r) - \mathcal{F}(t) \neq \delta \,\}.$$

It can be easily confirmed that all DD-functions are *bisimulation invariant*, i.e., if s and t are bisimilar then $d(s) = d(t)$ for all DD-functions d. So equality of the values of all DD-functions is a necessary condition for two states being bisimilar; it is not hard to see that in the case of image-finite LTSs this condition is also sufficient. (An LTS is image-finite iff each state has only finitely many a-successors for each a; LTSs corresponding to BPP-systems are clearly image-finite.)

A crucial point in [4] shows that, for any BPP-system (with VAR as the set of variables), DD-functions coincide with certain special functions called (relative) 'norms':

Given $Q \subseteq$ VAR, we define function NORM_Q by

$$\text{NORM}_Q(\alpha) = \min\{\, dist(\alpha,\beta) \mid \beta \text{ does not contain any variable from } Q \,\}.$$

Moreover, each norm_Q is a linear function, i.e, for each $X \in \text{var}$ there is $c_X \in \mathbb{N}$ such that

$$\text{norm}_Q(\alpha) = \sum_X c_X \cdot |\alpha|_X$$

where $|\alpha|_X$ denotes the number of occurrences of X in α. (Surely, $c_X = 0$ iff $X \notin Q$.)

Remark. For general BPP-systems, some c_X can be ω; but this is not the case for nBPP.

We note that the change of (the value of) such a linear function caused by using a rule (transition) $r : X \xrightarrow{a} \gamma$ does not depend on the state in which the rule r is used; such a change is always

$$-c_X + \sum_Y c_Y \cdot |\alpha|_Y$$

Our main algorithm performs a *stepwise decomposition* of the set T of rules (also called transitions), i.e., it constructs a sequence of decompositions of T, where each new decomposition refines the old one.

For each subset (a decomposition class) $T' \subseteq T$, notation $\text{Pre}(T')$ is used for the set $\{X \mid X \text{ is lhs of a rule in } T'\}$ (*lhs* abbreviates *left-hand side*).

The process starts with the (initial) decomposition according to action labels: transitions $X_1 \xrightarrow{a_1} \gamma_1$, $X_2 \xrightarrow{a_2} \gamma_2$ are in the same class iff $a_1 = a_2$.

The iterated step of the main algorithm refines a 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'. (For the initial decomposition we can observe that $\text{norm}_{\text{Pre}(T')}$ is dd_a for the respective action a.)

This surely finishes, with a decomposition denoted $\mathit{decomp}(T)$. Results of [4] guarantee that α and β are bisimilar iff $\text{norm}_{\text{Pre}(T')}(\alpha) = \text{norm}_{\text{Pre}(T')}(\beta)$ for each class T' in $\mathit{decomp}(T)$.

It is useful to realize that the *decomposition problem* is the crucial problem for us; the bisimilarity problem can be easily reduced to it:

Having a BPP system and two states α, β we can add two fresh variables A, B and rules $r_A = A \xrightarrow{a} \alpha$, $r_B = B \xrightarrow{a} \beta$ (for any chosen action a). It is clear that $\alpha \sim \beta$ ($\sim$ denoting the bisimulation equivalence) in the original system iff $A \sim B$ in the new system. Moreover, it can be readily verified that $A \sim B$ iff r_A, r_B are in the same class of $\mathit{decomp}(T)$.

In later analysis, we use the following general fact, which puts a limit on the number of decomposition classes which can appear during the process of stepwise decomposition.

Proposition 2.1 *In a stepwise decomposition of (nonempty) T, at most $|T|-1$ subsets with more than one element can appear.*

Proof. We proceed by induction on the cardinality of T.

For $|T| = 1$, the claim is obvious (a singleton has no subsets with more than one element).

In the induction step we assume $|T| \geq 2$. It is obvious that for achieving the maximal number of subsets appearing in a stepwise decomposition we start with the one-class decomposition $\{T\}$ and then continue with some ('best') decomposition $\{T_1, T_2, \ldots, T_k\}$; we note $k \geq 2$ and $|T_1| + |T_2| + \ldots + |T_k| = |T|$. In this way we can get, using the inductive hypothesis, at most

$$1 + |T_1| - 1 + |T_2| - 1 + \ldots + |T_k| - 1 = 1 + |T| - k$$

subsets. Since $k \geq 2$, we thus get at most $|T| - 1$ subsets as required. □

For complexity analysis, we have to make precise the way of presenting (normed) BPP-systems and determining their size w.r.t. which the complexity is measured. We will generally assume that a (normed) BPP-system has m variables $\text{VAR} = \{A_1, A_2, \ldots, A_m\}$ and is presented by ℓ (ordered) rules

$$\begin{aligned} r_1 &: X_1 \xrightarrow{a_1} \alpha_1 \\ r_2 &: X_2 \xrightarrow{a_2} \alpha_2 \\ &\ldots \\ r_\ell &: X_\ell \xrightarrow{a_\ell} \alpha_\ell \end{aligned}$$

We further assume that each sequence α on the right-hand sides of rules is presented as a sequence of pairs of positive numbers $(j_1, e_1), (j_2, e_2), \ldots, (j_p, e_p)$ in binary where $1 \leq j_1 < j_2 < \ldots < j_p \leq m$. It means to represent the sequence $(A_{j_1})^{e_1}(A_{j_2})^{e_2} \ldots (A_{j_p})^{e_p}$ where A^e stands for e occurrences of A. We note that the set $\{A_{j_1}, A_{j_2}, \ldots, A_{j_p}\}$ equals to the *carrier* of α where we define $\text{carrier}(\alpha) = \{X \mid |\alpha|_X \geq 1\}$.

By the *size n of a given BPP-system* we mean the number of bits in which the above presentation can be written.

Convention. In what follows, symbol e_{ij} $(1 \leq i \leq \ell,\ 1 \leq j \leq m)$ means $|\alpha_i|_{A_j}$ ($e_{ij} = 0$ iff $A_j \notin \text{carrier}(\alpha_i)$). We also use numbers k_{ij}: for $e_{ij} = 0$ we put $k_{ij} = 0$, and for $e_{ij} > 0$ we take k_{ij} to be the size (number of bits) of the binary presentation of e_{ij}; for technical reasons, we assume *the first bit* in this presentation of e_{ij} *to be 0*. Imposing this assumption increases the size n of a BPP-system to less than $2n$, and is harmless for our results.

3 Computing norms is in $O(n^2)$

Based on the scheme sketched in [4], we show an algorithm which, given an nBPP-system with a set of variables VAR and a subset $Q \subseteq \text{VAR}$, computes the coefficients of the (linear) function NORM_Q.

For each $X \in \text{VAR}$, the algorithm will compute the respective coefficient c_X. In fact, to each $X \in Q$, the algorithm will also attach an (optimal) rule r_X with lhs X.

The algorithm uses the following data structures, with the intended meanings

DV ... a set of determined variables (all X with determined c_X)

UV ... a set of undetermined variables, i.e., of all X for which c_X has not yet been determined; only a temporary (candidate) value d_X is attached

MIN ... contains an element X of UV with minimal d_X

RUL ... a set of unprocessed rules (with lhs in UV but with d_r not determined)

Algorithm 1 *Computing coefficients of* NORM_Q

$UV := Q$; *for each* $X \in UV$ *do* $d_X := \omega$
$DV := \text{VAR} - Q$; *for each* $X \in DV$ *do* $c_X := 0$
{ *for each rule* r *with lhs not in* Q, *we can deem* $d_r = 0$}
$RUL :=$ { *all rules with lhs in* Q }
while $UV \neq \emptyset$ *do*
    let MIN *refer to some* $Z \in UV$ *with minimal* d_Z
    for each rule $r = X \to \alpha$ *in* RUL *with* $\text{carrier}(\alpha) \subseteq DV$ *do*
        $d_r := 1 + \sum_{Y \in \text{carrier}(\alpha)} c_Y \cdot |\alpha|_Y$
        if $d_r < d_X$ *then* $d_X := d_r$
        if $d_r < d_{MIN}$ *then* $MIN := X$
        remove r *from* RUL
    remove MIN *from* UV *and add it to* DV
    $c_{MIN} := d_{MIN}$
    remove all rules with lhs MIN *from* RUL

Computing all

$$d_r := 1 + \sum_{Y \in \text{carrier}(\alpha)} c_Y \cdot |\alpha|_Y \tag{1}$$

takes a crucial portion in the running of the algorithm. Below we shall show that the sizes of presentations of numbers d_r (and hence also c_X) are in $O(n)$. Assuming this, it is a technical routine to check that the time of the computation *without counting the time of performing (1)* is in $O(n^2)$.

So now we show that d_r can be written in $O(n)$ bits and that the aggregated complexity of performing all instructions (1) is in $O(n^2)$.

For the aim of notationally convenient analysis (and without any loss of generality), we assume that the ordering of variables $A_1, A_2, \ldots, A_m$ coincides with the order in which the above algorithm adds the variables to DV. Moreover, we consider the rules in the order

$$r_1 : A_1 \xrightarrow{a_1} \alpha_1,\ r_2 : A_2 \xrightarrow{a_1} \alpha_1,\ \ldots,\ r_m : A_m \xrightarrow{a_m} \alpha_m,\ r_{m+1},\ r_{m+2},\ \ldots,\ r_\ell$$

where for $i = 1, 2, \ldots, m$ each r_i is optimal for A_i (hence $d_{r_i} = c_{A_i}$).

Further we abridge d_{r_i} to d_i and c_{A_i} to c_i. We note that $c_1 \leq c_2 \leq \ldots \leq c_m$.

We also recall our conventions about numbers e_{ij} and k_{ij}.

Proposition 3.1 *The size (in bits) of each d_i (and hence each c_i) which is greater than 1 is at most*

$$\sum_{i \geq p > q \geq 1} k_{pq}$$

Hence this size is in $O(n)$.

Proof.

It is clear that d_1 (and c_1) is at most 1.

Suppose now $d_{i+1} > 1$. When we are performing (1) for computing d_{i+1}, only A_j with $j < i+1$ can be in the carrier of the right-hand side (of r_{i+1}).

So

$$d_{i+1} \leq 1 + \sum_{j=1}^{min\{i,m\}} e_{i+1,j} c_j \tag{2}$$

We note that each nonzero summand $e_{i+1,j}c_j$ can be written either in $k_{i+1,j}-1$ bits (when $c_j = 1$) or in $k_{i+1,j}-1+\sum_{j \geq p > q \geq 1} k_{pq}$ bits (by induction hypothesis).

From this observation we can readily verify that the whole sum surely can be written in

$$\sum_{i+1 > q \geq 1} k_{i+1,q} + \sum_{i \geq p > q \geq 1} k_{pq} = \sum_{i+1 \geq p > q \geq 1} k_{pq}$$

bits. □

Proposition 3.2 *The aggregated complexity of all multiplications in computing coefficients performed in 1) is in $O(n^2)$.*

Proof. Complexity of the product of two numbers x and y is in $O(size(x) \cdot size(y))$ ($size$ referring to the number of bits).

The products we are interested in are $e_{ij} \cdot c_j$.

From Proposition 3.1 we know that c_j (> 1) can be written in

$$\sum_{j \geq p > q \geq 1} k_{pq}$$

bits. Thus the complexity of the product $e_{ij} \cdot c_j$, denoted $complex(i,j)$, is in

$$O(k_{ij} \cdot \sum_{i \geq p > q \geq 1} k_{pq}) = O(\sum_{i \geq p > q \geq 1} k_{ij} \cdot k_{pq})$$

The aggregated complexity of multiplications is

$$\sum_{1 \leq i \leq \ell, 1 \leq j \leq m} complex(i,j)$$

which is surely dominated by the sum of the products $k_{pq} \cdot k_{p'q'}$ for all $1 \leq p \leq \ell$, $1 \leq p' \leq \ell$, $1 \leq q \leq m$, $1 \leq q' \leq m$.

So the aggregated complexity is in

$$O(\,(\sum_{1\leq i\leq \ell, 1\leq j\leq m} k_{ij})^2\,)$$

which is obviously in $O(n^2)$. □

Lemma 3.3 *Given a normed BBP system and a set Q of variables, the coefficients of the (linear) function* $\textsc{norm}_Q$ *are computable (by the above algorithm) in $O(n^2)$.*

Proof. In view of the previous propositions, it remains to show that the amount of time needed for additions in (1) is in $O(n^2)$. But it is obvious that the algorithm $O(n)$ times adds two numbers with $O(n)$ bits. □

4 Decomposition problem is in $O(n^3)$

We already sketched (in Section 2) the ideas of the main algorithm decomposing the set T of rules (transitions). The algorithm uses the following data structures, with the intended meanings

$\mathcal{T}$ ... a decomposition of the set T of all rules

UIS ... a set of unprocessed (important) sets of variables (the norms of such sets correspond to DD-functions)

PIS ... a set of processed (important) sets of variables (for each Q here, the current decomposition $\mathcal{T}$ already separates each two rules which cause different changes on $\textsc{norm}_Q$)

Algorithm 2 *The decomposition algorithm*

Compute $\mathcal{T} = \{T_1, T_2, \ldots T_p\}$ *as the decomposition of the set T*
according to the action labels.
Let UIS contain all (different) sets $\mathrm{Pre}(T_1)$, $\mathrm{Pre}(T_2)$, ... , $\mathrm{Pre}(T_p)$
$PIS := \emptyset$
while $UIS \neq \emptyset$ *do*
  For each $Q \in UIS$ *do*
    compute all coefficients c_Y *of* $\textsc{norm}_Q$
    for each rule $r = X \to \alpha$ *appearing in a nonsingleton class of* $\mathcal{T}$ *do*
      $\delta(r) := -c_X + \sum_{Y\in \mathrm{carrier}(\alpha)} c_Y |\alpha|_Y$
    decompose each (nonsingleton) class in $\mathcal{T}$
      according to the computed values $\delta(r)$
    let $\mathcal{T}$ *refer to the newly arisen decomposition*
  $PIS := PIS \cup UIS$
  $UIS := \emptyset$
  for each (newly arisen) class T' *of* $\mathcal{T}$ *do*
    if $\mathrm{Pre}(T') \notin PIS$ *then* $UIS := UIS \cup \{\mathrm{Pre}(T')\}$

Theorem 4.1 *The decomposition algorithm for normed BBPs runs in $O(n^3)$. (Hence bisimilarity for nBPP is decidable in $O(n^3)$.)*

Proof. Proposition 2.1 implies that at most $2\ell - 1$ subsets of T can appear in the stepwise decomposition performed by the algorithm. This means that only $O(n)$ subsets Q of variables are processed (and put in PIS).

By Lemma 3.3, coefficients of each norm_Q can be computed in $O(n^2)$. For each norm_Q, the aggregated complexity of computing the changes $\delta(r)$ and the respective refinement of decomposition $\mathcal{T}$ can be also shown to be in $O(n^2)$ (similarly as in Section 3).

We can thus readily derive that the decomposition algorithm runs in time $O(n^3)$. □

References

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

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

[3] Hirshfeld, Y., M. Jerrum, and F. Moller, *A polynomial algorithm for deciding bisimulation equivalence of normed basic parallel processes*, Mathematical Structures in Computer Science **6** (1996), 251–259.

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

[5] Srba, J., "Roadmap of infinite results", Bull. of EATCS, 78:163–175, October 2002. See also an updated online version at http://www.brics.dk/srba/roadmap/.

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