Two known algorithms for checking bisimilarity of normed BPPs

Otmar Onderek, Martin Kot

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

Abstract. This paper brings two known algorithms for deciding bisimilarity of processes in BPP with their time analysis. The first algorithm was published in [1], the second one in [2]. While those materials aim on detailed proof of correctivity of those algorithms, this paper tries to explain them in an understandable form and describe them in a pseudo-language so that they could be implemented easily. This material has only limited amount of place, so the theorems are stated without proof or with only a suggestion of proof.

Keywords: Basic Parallel Process, BPP, normed BPP, bisimilarity.

1 Introduction

Recently, big attention has been paid to the area of verification of potentially infinite-state systems. Errors in computer and other systems are often very expensive. Therefore it is suitable to use formal methods to prove that some system is correct.

An important question in verification is if two systems behave in the same way – to be able to check equivalence between specification and implementation. Bisimilarity was established as the most appropriate notion of general behavioral equivalence. In short, bisimilar systems can perform the same actions in the same situation.

There are many types of models of real systems which differ in expressive power, that is, which properties are describable in concrete model. In this paper BPP systems are used. They describe parallel systems without synchronization.

In this paper, two different polynomial time algorithms are described for deciding bisimulation equivalence.

2 The basic concepts and notation

Definition 1. Basic Parallel Process (BPP) *is given by a context-free grammar in Greibach normal form (that is, each rule is of the form $X \rightarrow aY_1 \dots Y_n$) whose rules are denoted by $X \xrightarrow{a} Y_1 \dots Y_n$ instead of $X \rightarrow aY_1 \dots Y_n$. Its* processes *are multisets of nonterminals. Nonterminals themselves are called* elementary processes*. The notation $\alpha \xrightarrow{a} \beta$ means that the process β can be reached from the process α by the letter a, that is, a nonterminal $X \in \alpha$ and a rule $X \xrightarrow{a} \gamma$ exists such that $\alpha \setminus \{X\} \cup \gamma = \beta$. If the letter does not matter, we can denote $\alpha \rightarrow \beta$ only. This notation can be naturally extended to $\alpha \xrightarrow{w} \beta$ where $w \in A^*$ is a word. The notation $|\alpha|_X$ denotes a number of occurences of the elementary process X in the process α.*

Definition 2. Bisimulation *is a relation R on a set of processes of some BPP such that for each $(r_1, r_1') \in R$:*

- $\forall a, r_2 : r_1 \xrightarrow{a} r_2 \Rightarrow (\exists r_2' : r_1' \xrightarrow{a} r_2' \wedge (r_2, r_2') \in R)$,
- $\forall a, r_2' : r_1' \xrightarrow{a} r_2' \Rightarrow (\exists r_2 : r_1 \xrightarrow{a} r_2 \wedge (r_2, r_2') \in R)$.

Definition 3. Bisimilarity $\sim$ *is a maximal bisimulation (an union of all bisimulations on that BPP).*

Definition 4. Norm *of process α of a BPP (*norm α*) is the length of the shortest word w such that $\alpha \xrightarrow{w} \varepsilon$. Norm* with respect to a set $Q \subseteq S$ *(*$\text{norm}_Q\ \alpha$*) is the length of the shortest word w such that $\alpha \xrightarrow{w} \beta$ where $\beta \cap Q = \emptyset$. If no such word exists, then* norm $\alpha = \omega$*, resp.* $\text{norm}_Q\ \alpha = \omega$*. A BPP is* normed *iff all its elementary processes have a finite norm.* Norm-reducing rule *is a rule $X \xrightarrow{a} \alpha$ such that* norm $X >$ norm α*. We denote $\alpha \rightarrow_{NR} \beta$ if $\alpha \rightarrow \beta$ and* norm $\alpha >$ norm β.[1]

3 Hirshfeld-Jerrum-Moller algorithm

Definition 5. *A* decomposition *of a BPP (Σ, Π, S, R) is a function $\mathcal{D} : \Pi^* \rightarrow \Pi^*$ such that:*

- *for each elementary process X, if there exist some elementary processes $P_1, \ldots, P_m$ and some $x_1, \ldots, x_m \in \mathbf{N}$ such that $P_1^{x_1} \ldots P_m^{x_m} \sim X$ and for each P_i is* norm $P_i <$ norm X *and $\mathcal{D}(P_i) = P_i$, then $\mathcal{D}(X) = P_1^{x_1} \ldots P_m^{x_m}$, else $\mathcal{D}(X) = X$,*
- *if $\alpha = X_1^{x_1} \ldots X_m^{x_m}$ is a non-elementary process, then*
$$\mathcal{D}(\alpha) = (\mathcal{D}(X_1))^{x_1} \ldots (\mathcal{D}(X_m))^{x_m}.$$

An elementary process X is called a prime *iff $\mathcal{D}(X) = X$. The set of all primes in the decomposition $\mathcal{D}$ is denoted by $\Pi(\mathcal{D})$.*

Theorem 1. *For each BPP:*

1. *There is one and only one decomposition $\mathcal{D}$.*
2. *$\mathcal{D}(\alpha \cup \beta) = \mathcal{D}(\alpha) \cup \mathcal{D}(\beta)$.*
3. *$\mathcal{D}(\alpha) \sim \alpha$.*

[1] The algorithm for computing norm X is intuitive and has no influence on the complexity of the whole algorithm, so it is not mentioned here.

4. $\alpha \sim \beta$ *iff* $\mathcal{D}(\alpha) = \mathcal{D}(\beta)$.

In the following text, $\mathcal{D}$ denotes the decomposition defined above and D denotes a temporary decomposition found by the algorithm. In time analysis, we denote the number of variables of the grammar by $\mathbf{m}$ and the number of rules of the grammar by $\mathbf{n}$ (we assume that for each variable there is at least one rule with that variable on its left side, so $\mathbf{m} \leq \mathbf{n}$). We assume that the right side of each rule $X \to \alpha$ is given by a binary vector $(x_1, \ldots, x_{\mathbf{m}})$ such that $x_i = |\alpha|_X$. The number of bits necessary to express each x_i is denoted by $\mathbf{k}$, so the greatest possible number of the same variable in one right side is $2^{\mathbf{k}}$ and the greatest possible size of the grammar is $\mathbf{mnk}$.

3.1 Checking bisimilarity of two processes α, β in a BPP

```
find norms for elementary processes
sort the variables according to their norms (that is, $Y < X$ iff norm $Y \leq$ norm $X$)
for each variable $X$, create the first decomposition $D$ by algorithm 3.2
repeat: {
    let $b := false$
    denote the current decomposition by $D'$
    for each variable $X \notin \Pi(X)$: {
        create a new decomposition $D(X)$ by algorithm 3.3
        if $D(X) \neq D'(X)$, let $b := true$
    }
} while $b = true$
if $D(\alpha) = D(\beta)$, then return true, else return false
```

3.2 Creating the first decomposition $D(X)$ for a variable X if $D(Y)$ for each $Y < X$ is known

```
choose a rule $X \xrightarrow{a}_{NR} \alpha$
for each $P \xrightarrow{a}_{NR} \beta$ such that $P \in \Pi(D)$, $P < X$, $D(\beta) \subseteq D(\alpha)$ {
    let $D(X) := D(\alpha) + D(P) - D(\beta)$
    if test of $D(X)$ with algorithm 3.4 is successful, return $D(X)$
}
declare $X$ as a prime and return $D(X) := X$
```

(If X is the first variable, there is no variable Y such that $Y < X$, so the cycle “for” will be never performed and X will be declared as a prime.)

3.3 Creating the next decomposition $D(X)$ if a previous decomposition D' is known

```
if test of $D'(X)$ with algorithm 3.4 is successful, return $D'(X)$
choose a rule $X \xrightarrow{a}_{NR} \alpha$
for each $P \xrightarrow{a}_{NR} \beta$ such that $P \in \Pi(D)$, $P < X$, $D(\beta) \subseteq D(\alpha)$ where:
```

```
either $P \in \Pi(D) \setminus \Pi(D')$, or $(D(\alpha) - D(\beta)) \cap (\Pi(D) \setminus \Pi(D')) \neq \emptyset$:
{
    let $D(X) := D(\alpha) + D(P) - D(\beta)$
    if test of $D(X)$ with algorithm 3.4 is successful, return $D(X)$
}
declare $X$ as a prime and return $D(X) := X$
```

3.4 Testing a possible decomposition $D(X)$

```
if we have a previous decomposition $D'$: {
    for each rule $X \xrightarrow{a} \alpha$: {
        for each rule $P \xrightarrow{a} \beta$ such that $P \in D(X)$: {
            if $D'(D(X)) - D'(P) + D'(\beta) = D'(\alpha)$: {
                mark the rule $P \xrightarrow{a} \beta$
                go to (1)
            }
        }
        return "fail"
        (1):
    }
    for each unmarked rule $P \xrightarrow{a} \beta$ such that $P \in D(X)$: {
        for each rule $X \xrightarrow{a} \alpha$: {
            if $D'(D(X)) - D'(P) + D'(\beta) = D'(\alpha)$, go to (2)
        }
        return "fail"
        (2):
    }
    unmark all marked rules
}
else: {
    if norm $X \neq$ norm $D(X)$, return "fail"
}
for each rule $X \xrightarrow{a}_{NR} \alpha$: {
    for each rule $P \xrightarrow{a}_{NR} \beta$ such that $P \in D(X)$: {
        if $D(X) - D(P) + D(\beta) = D(\alpha)$, mark $P \xrightarrow{a}_{NR} \beta$ and go to (3)
    }
    return "fail"
    (3):
}
for each unmarked rule $P \xrightarrow{a}_{NR} \beta$ such that $P \in D(X)$: {
    for each rule $X \xrightarrow{a}_{NR} \alpha$: {
        if $D'(D(X)) - D'(P) + D'(\beta) = D'(\alpha)$, go to (4)
    }
    return "fail"
    (4):
}
```

return “pass”

Theorem 2. *The algorithm 3.4 is in $O(\mathbf{m}^2\mathbf{n}^2\mathbf{k})$. The algorithms 3.2 and 3.3 are both in $O(\mathbf{m}^2\mathbf{n}^3\mathbf{k})$. The algorithm 3.1 is in $O(\mathbf{m}^4\mathbf{n}^3\mathbf{k})$.*
Proof. Let us denote the variables $A_1, \ldots, A_\mathbf{m}$ according to their ordering. The variable A_1 is surely a prime, that is, $|D(A_1)|=1$. For each $i > 1$, A_i and for each rule $A_i \rightarrow_{NR} \alpha$, $|D(A_i)| \leq |D(\alpha)|+1$. Because α consists only from $A_1, \ldots, A_{i-1}$ and $\forall j : |\alpha|_{A_j} \leq 2^\mathbf{k}$, $|D(\alpha)| \leq \sum_{j=1}^{i-1} 2^\mathbf{k}|D(A_j)|$. It follows that for each i, $|D(A_i)| \leq (2^\mathbf{k}+1)^{i-1}$. For each $X \rightarrow \beta$ and Y, $|D(\beta)|_Y \leq \sum_{j=1}^{\mathbf{m}} 2^\mathbf{k}|D(A_j)|_Y = 2^\mathbf{k}\sum_{j=1}^{\mathbf{m}}(2^\mathbf{k}+1)^{j-1} < 2^\mathbf{k}(2^\mathbf{k}+1)^\mathbf{m}$, so $|D(\beta)|_Y$ for each β can be written in $\mathbf{k}+\mathbf{mk}$ bits. The whole decomposition $D(\beta)$ can be written as a binary vector consisting of $\mathbf{m}$ elements (one for each Y), and so in $\mathbf{mk}(\mathbf{m}+1)$ bits. Thus, $D(X) - D(P) + D(\beta)$ can be computed in $O(\mathbf{m}^2\mathbf{k})$. It easily follows that the algorithm 3.4 is in $O(\mathbf{m}^2\mathbf{n}^2\mathbf{k})$, the algorithms 3.2 and 3.3 are both in $O(\mathbf{m}^2\mathbf{n}^3\mathbf{k})$, and the algorithm 3.1 is in $O(\mathbf{m}^4\mathbf{n}^3\mathbf{k})$. □

4 Jančar algorithm

Definition 6. *For a given BPP (Σ, Π, S, R) where $\Pi = \{X_1, \ldots, X_\mathbf{m}\}$, a* linear function *is a function $c : \Pi^* \rightarrow \mathbf{N}_\omega$ such that there exist coefficients $c_{X_1}, \ldots, c_{X_\mathbf{m}}$ such that for each process α, $c(\alpha) = \sum_{i=1}^{\mathbf{m}} c_{X_i} \cdot |\alpha|_{X_i}$.*

Theorem 3. *For each set of elementary processes $Q \subseteq \Pi$, the function* norm_Q *: $\Pi^* \rightarrow \mathbf{N}_\omega$ which assigns to each process its norm with respect to Q is a linear function.*

4.1 Computing a norm with respect to a set of variables Q

```
for each variable $X \in Q$, let $c_X := \omega$
for each variable $X \notin Q$, let $c_X := 0$
repeat: {
    for each variable $X \in Q$, let $d_X := \omega$
    for each rule $X \rightarrow \alpha$ such that $X \in Q$: {
        let $d := \sum_Y c_Y \cdot |\alpha|_Y$
        if $d < d_X$, let $d_X := d$
    }
    let $d := \min\{d_X \mid X \in Q\}$
    for each variable $X \in Q$ such that $d_X = d$: {
        let $c_X := d + 1$
        remove $X$ from the set $Q$
    }
} while $Q \neq \emptyset$
```

Theorem 4. *The algorithm 4.1 is in $O(\mathbf{m}^3\mathbf{nk})$.*
Proof. Let us denote the variables $A_1, \ldots, A_\mathbf{m}$ in the same order as the algorithm

4.1 determines their norms. Then $c_{A_1} = 1$ and $c_{A_i} \le \sum_{j=1}^{i-1} 2^{\mathbf{k}} c_{A_j} + 1$. It follows that $c_{A_i} \le (2^{\mathbf{k}}+1)^{i-1} \le (2^{\mathbf{k}}+1)^{\mathbf{m}-1}$. Because $|\alpha|_Y \le 2^{\mathbf{k}}$ for each Y, $c_Y \cdot |\alpha|_Y \le 2^{\mathbf{k}}(2^{\mathbf{k}}+1)^{\mathbf{m}-1}$, and thus $c_Y \cdot |\alpha|_Y$ can be written in $(\mathbf{m}-1)(\mathbf{k}+1)+\mathbf{k}$ bits, so it is in $O(\mathbf{mk})$. Then the sum $\sum_Y c_Y \cdot |\alpha|_Y$ is in $O(\mathbf{m}^2\mathbf{k})$. The outer cycle "repeat" is repeated at most $\mathbf{m}$ times because at least one element is removed from the set Q in each repeating. The inner cycle "for each rule" is repeated $\mathbf{n}$ times. It follows that the whole algorithm is in $O(\mathbf{m}^3\mathbf{nk})$. □

4.2 Checking bisimilarity of processes α and β

```
decompose the set of all rules to subsets $T_1, \dots, T_n$ according to $\equiv$:
$X \xrightarrow{a} \alpha \equiv Y \xrightarrow{b} \beta \iff a = b$
repeat: {
    for each set $T_i$: {
        compute the coefficients of the function $\mathrm{norm}_{\mathrm{Pre}\ T_i}$
        for each rule $X \to \alpha$:
            let $\delta_i(X \to \alpha) := \mathrm{norm}_{\mathrm{Pre}\ T_i}\ \alpha - \mathrm{norm}_{\mathrm{Pre}\ T_i}\ X$
        decompose $T_i$ to subsets $T'_{i,1}, \dots, T'_{i,m_i}$ according to $\equiv$:
        $X \to \alpha \equiv Y \to \beta \iff \forall i : \delta_i(X \to \alpha) = \delta_i(Y \to \beta)$
    }
    denote all sets $T'_{1,1}, \dots, T'_{n,m_n}$ as $T_1, \dots, T_m$
} while $m > n$
for each decomposition set $T_i$: {
    compute the coefficients of the function $\mathrm{norm}_{\mathrm{Pre}\ T_i}$
    if $\mathrm{norm}_{\mathrm{Pre}\ T_i}\ \alpha \neq \mathrm{norm}_{\mathrm{Pre}\ T_i}\ \beta$, return false
}
return true
```

Theorem 5. *The algorithm 4.2 is in $O(\mathbf{m}^3\mathbf{n}^3\mathbf{k})$.*
Proof. The cycles "repeat" and "for each set T_i" are repeated at most $\mathbf{n}$ times because the decomposition can have at most $\mathbf{n}$ subsets and at least one new subset is added in each repeating. Computation of the coefficients of function norm is in $O(\mathbf{m}^3\mathbf{nk})$. The cycle "for each rule $X \to \alpha$" is repeated $\mathbf{n}$ times and the computation of $\delta_i(X \to \alpha)$ from $\mathrm{norm}_{\mathrm{Pre}\ T_i}$ is in $O(\mathbf{m}^2\mathbf{k})$, so the computation of norm is strictly longer. Decomposing and the final step after reaching the finest decomposition are also strictly shorter than the first step, so the whole algorithm is in $O(\mathbf{m}^3\mathbf{n}^3\mathbf{k})$. □

References

1. Yoram Hirshfeld, Mark Jerrum, Faron Moller. A polynomial time algorithm for deciding bisimulation equivalence of normed Basic Parallel Processes. *Journal of Mathematical Structures in Computer Science* **6**: 251–259, 1996.
2. Petr Jančar. Strong Bisimilarity on Basic Parallel Processes is PSPACE-complete. *Proc. 18th LiCS*, pages 218–227. IEEE Computer Society, 2003.