Weak Bisimilarity with Infinite-State Systems can be Decided in Polynomial Time

Antonín Kučera[⋆1] and Richard Mayr[⋆⋆2]

[1] Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic, `tony@fi.muni.cz`

[2] LFCS, Dept. of Computer Science, Univ. of Edinburgh, JCMB, King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK, `mayrri@dcs.ed.ac.uk`

Abstract. We prove that weak bisimilarity is decidable in polynomial time between BPA and finite-state processes, and between normed BPP and finite-state processes. To the best of our knowledge, these are the first polynomial algorithms for weak bisimilarity with infinite-state systems.

1 Introduction

Recently, a lot of attention has been devoted to the study of decidability and complexity of verification problems for infinite-state systems [27, 11, 5]. We consider the problem of weak bisimilarity between certain infinite-state processes and finite-state ones. The motivation is that the intended behaviour of a process is often easy to specify (by a finite-state system), but a 'real' implementation can contain components which are essentially infinite-state (e.g. counters, buffers). The aim is to check if the finite-state specification and the infinite-state implementation are semantically equivalent, i.e. weakly bisimilar.

We concentrate on the classes of infinite-state processes definable by the syntax of BPA (Basic Process Algebra) and normed BPP (Basic Parallel Processes) systems. BPA processes can be seen as simple sequential programs (due to the binary operator of sequential composition). They have recently been used to solve problems of dataflow analysis in optimising compilers [12]. BPP model simple parallel systems (due to the binary operator of parallel composition). A process is *normed* iff at every reachable state it can terminate via a finite sequence of computational steps.

The state of the art. Baeten, Bergstra, and Klop [1] proved that *strong bisimilarity* [29] is decidable for normed BPA processes. Simpler proofs have been given later in [18, 13], and there is even a polynomial-time algorithm [15]. The decidability result has later been extended to the class of all (not necessarily normed) BPA processes in [9], but the best known algorithm is doubly exponential [4]. Decidability of strong bisimilarity for BPP processes has been established in [8], but the algorithm has non-elementary complexity. However, there is a polynomial-time algorithm for the subclass of normed BPP [16]. Strong bisimilarity between normed BPA and normed BPP is also decidable

⋆ Supported by a Research Fellowship granted by the Alexander von Humboldt Foundation and by a Post-Doc grant GA ČR No. 201/98/P046.

⋆⋆ This work was partly supported by a DAAD Post-Doc grant.

[7]. This result even holds for parallel compositions of normed BPA and normed BPP processes [20].

For weak bisimilarity, much less is known. Semidecidability of weak bisimilarity for BPP is due to [10]. In [14] it is shown that weak bisimilarity is decidable for those BPA and BPP processes which are 'totally normed' (a process is totally normed if it can terminate at any moment via a finite sequence of computational steps, but at least one of those steps must be 'visible', i.e. non-internal). Decidability of weak bisimilarity for general BPA and BPP is open; those problems might be decidable, but they are surely intractable (assuming $\mathcal{P} \neq \mathcal{NP}$)—for BPP we have $\mathcal{NP}$-hardness, and for BPA even $PSPACE$-hardness [30].

The situation is dramatically different if we consider weak bisimilarity between certain infinite-state processes and finite-state ones. In [24] it is shown that weak bisimilarity between BPP and finite-state processes is decidable. A more general result has recently been obtained in [19], where it is shown that many bisimulation-like equivalences (including the strong and weak ones) are decidable between PAD and finite-state processes. The class PAD strictly subsumes not only BPA and BPP, but also PA [2] and pushdown processes. This result is obtained by a general reduction to the model-checking problem for the simple branching-time temporal logic *EF*. As the model-checking problem for *EF* is hard (for example, it is known to be $PSPACE$-complete for BPP [24] and $PSPACE$-hard for BPA [25]), this does not yield an efficient algorithm.

Our contribution. We show that weak (and hence also strong) bisimilarity is decidable in polynomial time between BPA and finite-state processes, and between normed BPP and finite-state processes. Due to the aforementioned hardness results for the 'symmetric case' (when we compare two BPA or two (normed) BPP processes) we know that our results cannot be extended in this direction. To the best of our knowledge, these are the first polynomial algorithms for weak bisimilarity with infinite-state systems. Moreover, the algorithm for BPA is the first example of an efficient decision procedure for a class of *unnormed* infinite-state systems (the polynomial algorithms for strong bisimilarity of [15, 16] only work for normed subclasses of BPA and BPP, respectively). It should also be noted that *simulation equivalence* between BPA/BPP and finite-state systems is co-$\mathcal{NP}$-hard [22].

The basic scheme of our constructions for BPA and normed BPP processes is the same. The main idea is that weak bisimilarity between BPA (or normed BPP) processes and finite-state ones can be generated from a finite base and that certain infinite subsets of BPA and BPP state-space can be 'symbolically' described by finite automata and context-free grammars, respectively. A more detailed intuition is given in Section 3. As weak bisimilarity is not a congruence w.r.t. sequencing (see Section 3), we propose its natural refinement called termination-sensitive bisimilarity which *is* a congruence and which is also decidable between BPA and finite-state processes in polynomial time.

2 Definitions

We use process rewrite systems [23] as a formal model for processes. Let $Act = \{a, b, c, \ldots\}$ and $Const = \{X, Y, Z, \ldots\}$ be disjoint countably infinite sets of *actions* and *process constants*, respectively. The class of *process expressions* $\mathcal{E}$ is defined by

$E ::= \varepsilon \mid X \mid E \| E \mid E.E$, where $X \in \mathit{Const}$ and ε is a special constant that denotes the empty expression. Intuitively, '.' is sequential composition and '$\|$' is parallel composition. We do not distinguish between expressions related by *structural congruence* which is given by the following laws: '.' and '$\|$' are associative, '$\|$' is commutative, and 'ε' is a unit for '.' and '$\|$'.

A *process rewrite system* [23] is specified by a finite set of *rules* $\varDelta$ which have the form $E \xrightarrow{a} F$, where $E, F \in \mathcal{E}$ and $a \in \mathit{Act}$. $\mathit{Const}(\varDelta)$ and $\mathit{Act}(\varDelta)$ denote the sets of process constants and actions which are used in the rules of $\varDelta$, respectively (note that these sets are finite). Each process rewrite system $\varDelta$ defines a unique transition system where states are process expressions over $\mathit{Const}(\varDelta)$, $\mathit{Act}(\varDelta)$ is the set of labels, and transitions are determined by $\varDelta$ and the following inference rules (remember that '$\|$' is commutative):

$$\frac{(E \xrightarrow{a} F) \in \varDelta}{E \xrightarrow{a} F} \qquad \frac{E \xrightarrow{a} E'}{E.F \xrightarrow{a} E'.F} \qquad \frac{E \xrightarrow{a} E'}{E\|F \xrightarrow{a} E'\|F}$$

We extend the notation $E \xrightarrow{a} F$ to elements of Act^* in the standard way. F is *reachable* from E if $E \xrightarrow{w} F$ for some $w \in \mathit{Act}^*$.

Sequential and *parallel* expressions are those process expressions which do not contain the '$\|$' and the '.' operator, respectively. Finite-state, BPA, and BPP systems are subclasses of process rewrite systems obtained by putting certain restrictions on the form of the rules. Finite-state, BPA, and BPP allow only a single constant on the left-hand side of rules, and a single constant, sequential expression, and parallel expression on the right-hand side, respectively. The set of states of a transition system which is generated by a finite-state, BPA, or BPP process $\varDelta$ is restricted to $\mathit{Const}(\varDelta)$, the set of all sequential expressions over $\mathit{Const}(\varDelta)$, or the set of all parallel expressions over $\mathit{Const}(\varDelta)$, respectively. A constant $X \in \mathit{Const}(\varDelta)$ is *normed* iff $X \xrightarrow{w} \varepsilon$ for some $w \in \mathit{Act}^*$. A process is normed, iff all constants of its underlying system $\varDelta$ are normed.

The semantical equivalence we are interested in here is *weak bisimilarity* [26]. This relation distinguishes between 'observable' and 'internal' moves (computational steps); the internal moves are modelled by a special action which is denoted 'τ' by convention. In what follows we consider process expressions over $\mathit{Const}(\varDelta)$ where $\varDelta$ is some fixed process rewrite system.

Definition 1. *The* extended transition relation *'$\overset{a}{\Rightarrow}$' is defined by $E \overset{a}{\Rightarrow} F$ iff either $E = F$ and $a = \tau$, or $E \xrightarrow{\tau^i} E' \xrightarrow{a} E'' \xrightarrow{\tau^j} F$ for some $i, j \in \mathbb{N}_0$, $E', E'' \in \mathcal{E}$. A binary relation R over process expressions is a* weak bisimulation *iff whenever $(E, F) \in R$ then for every $a \in \mathit{Act}$: if $E \xrightarrow{a} E'$ then there is $F \overset{a}{\Rightarrow} F'$ s.t. $(E', F') \in R$ and if $F \xrightarrow{a} F'$ then there is $E \overset{a}{\Rightarrow} E'$ s.t. $(E', F') \in R$. Processes E, F are* weakly bisimilar, *written $E \approx F$, iff there is a weak bisimulation relating them.*

Let $\varGamma$ be a finite-state system with n states, $f, g \in \mathit{Const}(\varGamma)$. It is easy to show that the problem whether $f \approx g$ is decidable in $\mathcal{O}(n^3)$ time. For example, we can compute the '$\overset{a}{\Rightarrow}$' relation of $\varGamma$ and then start to refine $\mathit{Const}(\varGamma) \times \mathit{Const}(\varGamma)$ in a number of steps until it 'stabilises' w.r.t. $\overset{a}{\Rightarrow}$. We note that the use of some advanced techniques (see e.g.

[28]) could probably decrease the mentioned upper bound; however, the complexity of the algorithms which are designed in this paper is a bit worse (even if we could decide the problem $f \approx g$ in constant time), hence we do not try to improve this bound.

Sometimes we also consider weak bisimilarity between processes of *different* process rewrite systems, say Δ and Γ. Formally, Δ and Γ can be considered as a *single* system by taking their disjoint union.

3 BPA Processes

Let E be a BPA process with the underlying system Δ, F a finite-state process with the underlying system Γ s.t. $Const(\Delta) \cap Const(\Gamma) = \emptyset$. We assume (w.l.o.g.) that $E \in Const(\Delta)$; moreover, we also assume that for all $f, g \in Const(\Gamma)$, $a \in Act$ s.t. $f \neq g$ or $a \neq \tau$ we have that $f \overset{a}{\Rightarrow} g$ implies $f \xrightarrow{a} g \in \Gamma$. If those '$\xrightarrow{a}$' transitions are missing in Γ, we can add them safely—it does not influence our complexity estimations, as we always consider the worst case when Γ has all possible transitions (we do not want to add new transitions of the form $f \xrightarrow{\tau} f$, because then our proof for weak bisimilarity would not immediately work for termination-sensitive bisimilarity which is designed at the end of this section).

In this section, we use upper-case letters $X, Y, \ldots$ to denote elements of $Const(\Delta)$, and lower-case letters $f, g, \ldots$ to denote elements of $Const(\Gamma)$. Greek letters $\alpha, \beta, \ldots$ are used to denote elements of $Const(\Delta)^*$. The size of Δ is denoted by n, and the size of Γ by m (we measure the complexity of our algorithm in (n, m)).

The set $Const(\Delta)$ can be divided into two disjoint subsets of *normed* and *unnormed* constants (remember that $X \in Const(\Delta)$ is normed iff $X \xrightarrow{w} \varepsilon$ for some $w \in Act^*$). The set of all normed constants of Δ is denoted $Normed(\Delta)$. In our constructions we also use processes of the form αf; they should be seen as BPA processes with the underlying system $\Delta \cup \Gamma$.

Intuition: Our proof can be divided into two parts: first we show that the greatest weak bisimulation between processes of Δ and Γ is finitely representable. There is a finite relation $\mathcal{B}$ of size $\mathcal{O}(n\,m^2)$ (called *bisimulation base*) such that each pair of weakly bisimilar processes can be generated from that base (a technique first used by Caucal [6]). Then we show that the bisimulation base can be computed in polynomial time. To do that, we take a sufficiently large relation $\mathcal{G}$ which surely subsumes the base and 'refine' it (this refinement technique has been used in [15, 16]). The size of $\mathcal{G}$ is still $\mathcal{O}(n\,m^2)$, and each step of the refinement procedure possibly deletes some of the elements of $\mathcal{G}$. If nothing is deleted, we have found the base (hence we need at most $\mathcal{O}(n\,m^2)$ steps). The refinement step is formally introduced in Definition 4 (we compute the *expansion* of the currently computed approximation of the base). Intuitively, a pair of processes belongs to the expansion iff for each $\xrightarrow{a}$ move of one component there is a $\overset{a}{\Rightarrow}$ move of the other component s.t. the resulting pair of processes can be generated from the current approximation of $\mathcal{B}$. We have to overcome two fundamental problems:

1. The set of pairs which can be generated from $\mathcal{B}$ (and its approximations) is infinite.
2. The set of states which are reachable from a given BPA state in one '$\overset{a}{\Rightarrow}$' move is infinite.

We employ a 'symbolic' technique to represent those infinite sets (similar to the one used in [3]), taking advantage of the fact that they have a simple (regular) structure which can be encoded by finite-state automata (see Theorem 1 and 4). This allows to compute the expansion in polynomial time.

Definition 2. *A relation K is* fundamental *iff it is a subset of*

$$((\mathit{Normed}(\Delta) \cdot \mathit{Const}(\Gamma)) \times \mathit{Const}(\Gamma)) \cup (\mathit{Const}(\Delta) \times \mathit{Const}(\Gamma)) \cup$$
$$((\{\varepsilon\} \cup \mathit{Const}(\Gamma)) \times \mathit{Const}(\Gamma))$$

Note that the size of any fundamental relation is $\mathcal{O}(n\,m^2)$. The greatest fundamental relation is denoted by $\mathcal{G}$. The bisimulation base *for Δ and Γ, denoted $\mathcal{B}$, is defined as follows: $\mathcal{B} = \{(Yf, g) \mid Yf \approx g, Y \in \mathit{Normed}(\Delta)\} \cup \{(X, g) \mid X \approx g\} \cup \{(f, g) \mid f \approx g\} \cup \{(\varepsilon, g) \mid \varepsilon \approx g\}$.*

As weak bisimilarity is a left congruence w.r.t. sequential composition, we can 'generate' from $\mathcal{B}$ new pairs of weakly bisimilar processes by substitution (it is worth noting that weak bisimilarity is *not* a right congruence w.r.t. sequencing—to see this, it suffices to define $X \xrightarrow{\tau} X,\ Y \xrightarrow{\tau} \varepsilon,\ Z \xrightarrow{a} Z$. Now $X \approx Y$, but $XZ \not\approx YZ$). This generation procedure can be defined for any fundamental relation as follows:

Definition 3. *Let K be a fundamental relation. The* closure *of K, denoted $\mathit{Cl}(K)$, is the least relation M which satisfies the following conditions:*

1. *$K \subseteq M$*
2. *if $(f, g) \in K$ and $(\alpha, f) \in M$, then $(\alpha, g) \in M$*
3. *if $(f, g) \in K$ and $(\alpha h, f) \in M$, then $(\alpha h, g) \in M$*
4. *if $(Yf, g) \in K$ and $(\alpha, f) \in M$, then $(Y\alpha, g) \in M$*
5. *if $(Yf, g) \in K$ and $(\alpha h, f) \in M$, then $(Y\alpha h, g) \in M$*
6. *if $(\alpha, g) \in M$ and α contains an unnormed constant, then $(\alpha\beta, g), (\alpha\beta h, g) \in M$ for every $\beta \in \mathit{Const}(\Delta)^*$ and $h \in \mathit{Const}(\Gamma)$.*

Note that $\mathit{Cl}(K)$ contains elements of just two forms – (α, g) and $(\alpha f, g)$. Clearly $\mathit{Cl}(K) = \bigcup_{i=0}^{\infty} \mathit{Cl}(K)^i$ where $\mathit{Cl}(K)^0 = K$ and $\mathit{Cl}(K)^{i+1}$ consists of $\mathit{Cl}(K)^i$ and the pairs which can be immediately derived from $\mathit{Cl}(K)^i$ by the rules 2–6 of Definition 3.

Although the closure of a fundamental relation can be infinite, its structure is in some sense regular. This fact is precisely formulated in the following theorem:

Theorem 1. *Let K be a fundamental relation. For each $g \in \mathit{Const}(\Gamma)$ there is a finite-state automaton $\mathcal{A}_g$ of size $\mathcal{O}(n\,m^2)$ constructible in $\mathcal{O}(n\,m^2)$ time s.t. $L(\mathcal{A}_g) = \{\alpha \mid (\alpha, g) \in \mathit{Cl}(K)\} \cup \{\alpha f \mid (\alpha f, g) \in \mathit{Cl}(K)\}$*

Proof. We construct a regular grammar of size $\mathcal{O}(n\,m^2)$ which generates the mentioned language. Let $G_g = (N, \Sigma, \delta, \overline{g})$ where

- $N = \{\overline{f} \mid f \in \mathit{Const}(\Gamma)\} \cup \{U\}$
- $\Sigma = \mathit{Const}(\Delta) \cup \mathit{Const}(\Gamma)$
- δ is defined as follows:
  - for each $(\varepsilon, h) \in K$ we add the rule $\overline{h} \rightarrow \varepsilon$.
  - for each $(f, h) \in K$ we add the rules $\overline{h} \rightarrow \overline{f},\ \overline{h} \rightarrow f$.

- for each $(Yf, h) \in K$ we add the rules $\overline{h} \to Yf$, $\overline{h} \to Y\overline{f}$.
- for each $(X, h) \in K$ we add the rule $\overline{h} \to X$ and if X is unnormed, then we also add the rule $\overline{h} \to XU$.
- for each $X \in \mathit{Const}(\Delta)$, $f \in \mathit{Const}(\Gamma)$ we add the rules $U \to XU$, $U \to X$, $U \to f$.

A proof that G_g indeed generates the mentioned language is routine. Now we translate G_g to $\mathcal{A}_g$ (see e.g. [17]). Note that the size of $\mathcal{A}_g$ is essentially the same as the size of G_g; $\mathcal{A}_g$ is non-deterministic and can contain ε-rules. □

As an immediate consequence of the previous theorem we obtain that the membership to $\mathit{Cl}(K)$ for any fundamental relation K is easily decidable in polynomial time. Another property of $\mathit{Cl}(K)$ is specified in the lemma below.

Lemma 1. *Let $(\alpha f, g) \in \mathit{Cl}(K)$. If $(\beta h, f) \in \mathit{Cl}(K)$, then also $(\alpha\beta h, g) \in \mathit{Cl}(K)$. Similarly, if $(\beta, f) \in \mathit{Cl}(K)$, then also $(\alpha\beta, g) \in \mathit{Cl}(K)$.*

The importance of the bisimulation base is clarified by the following theorem. It says that $\mathit{Cl}(\mathcal{B})$ subsumes the greatest weak bisimulation between processes of Δ and Γ.

Theorem 2. *For all α, f, g we have $\alpha \approx g$ iff $(\alpha, g) \in \mathit{Cl}(\mathcal{B})$, and $\alpha f \approx g$ iff $(\alpha f, g) \in \mathit{Cl}(\mathcal{B})$.*

Proof. The 'if' part is obvious in both cases, as $\mathcal{B}$ contains only weakly bisimilar pairs and all the rules of Definition 3 produce pairs which are again weakly bisimilar. The 'only if' part can, in both cases, be easily proved by induction on the length of α (we just show the first proof; the second one is similar).

– $\alpha = \varepsilon$. Then $(\varepsilon, g) \in \mathcal{B}$, hence $(\varepsilon, g) \in \mathit{Cl}(\mathcal{B})$.
– $\alpha = Y\beta$. If Y is unnormed, then $Y \approx g$ and $(Y, g) \in \mathcal{B}$. By the rule 6 of Definition 3 we obtain $(Y\beta, g) \in \mathit{Cl}(\mathcal{B})$. If Y is normed, then $Y\beta \xrightarrow{w} \beta$ for some $w \in \mathit{Act}^*$ and g must be able to match the sequence w by some $g \overset{w}{\Rightarrow} g'$ s.t. $\beta \approx g'$. By substitution we now obtain that $Yg' \approx g$. Clearly $(Yg', g) \in \mathcal{B}$, and $(\beta, g') \in \mathit{Cl}(\mathcal{B})$ by induction hypothesis. Hence $(\alpha, g) \in \mathit{Cl}(\mathcal{B})$ due to the rule 4 of Definition 3. □

The next definition formalises one step of the 'refinement procedure' which is applied to $\mathcal{G}$ to compute $\mathcal{B}$.

Definition 4. *Let K be a fundamental relation. We say that a pair (X, g) of K* expands *in K iff the following two conditions hold:*

– *for each $X \xrightarrow{a} \alpha$ there is some $g \overset{a}{\Rightarrow} g'$ s.t. $(\alpha, g') \in \mathit{Cl}(K)$*
– *for each $g \xrightarrow{a} g'$ there is some $X \overset{a}{\Rightarrow} \alpha$ s.t. $(\alpha, g') \in \mathit{Cl}(K)$*

The expansion of a pair of the form (Yf, g), (f, g), (ε, g) in K is defined in the same way—for each '$\xrightarrow{a}$' move of the left component there must be some '$\overset{a}{\Rightarrow}$' move of the right component such that the resulting pair of processes belongs to $\mathit{Cl}(K)$, and vice versa (note that $\varepsilon \overset{\tau}{\Rightarrow} \varepsilon$). The set of all pairs of K which expand in K is denoted by $\mathit{Exp}(K)$.

The notion of expansion is in some sense 'compatible' with the definition of weak bisimulation. This intuition is formalised in the following lemma.

Lemma 2. *Let K be a fundamental relation s.t. $Exp(K) = K$. Then $Cl(K)$ is a weak bisimulation.*

Proof. We prove that every pair (α, g), $(\alpha f, g)$ of $Cl(K)^i$ has the property that for each '$\xrightarrow{a}$' move of one component there is a '$\overset{a}{\Rightarrow}$' move of the other component s.t. the resulting pair of processes belongs to $Cl(K)$ (we consider just pairs of the form $(\alpha f, g)$; the other case is similar). By induction on i.

- $i = 0$. Then $(\alpha f, g) \in K$; as $K = Exp(K)$, the claim follows directly from the definitions.
- **Induction step.** Let $(\alpha f, g) \in Cl(K)^{i+1}$. There are three possibilities:
  I. There is an h s.t. $(\alpha f, h) \in Cl(K)^i$, $(h, g) \in K$.
  Let $\alpha f \xrightarrow{a} \gamma f$ (note that α can be empty; in this case we have to consider moves of the form $f \xrightarrow{a} f'$. It is done in a similar way as below). As $(\alpha f, h) \in Cl(K)^i$, we can use the induction hypothesis and conclude that there is $h \overset{a}{\Rightarrow} h'$ s.t. $(\gamma f, h') \in Cl(K)$. We distinguish two cases:
  1) $a = \tau$ and $h' = h$. Then $(\gamma f, h) \in Cl(K)$ and as $(h, g) \in K$, we obtain $(\gamma f, g) \in Cl(K)$ due to Lemma 1. Hence g can use the move $g \overset{\tau}{\Rightarrow} g$.
  2) $a \neq \tau$ or $h \neq h'$. Then there is a transition $h \xrightarrow{a} h'$ (see the beginning of this section) and as $(h, g) \in K$, by induction hypothesis we know that there is some $g \overset{a}{\Rightarrow} g'$ s.t. $(h', g') \in Cl(K)$. Hence, $(\gamma f, g') \in Cl(K)$ due to Lemma 1.
  Now let $g \xrightarrow{a} g'$. As $(h, g) \in K$, there is $h \overset{a}{\Rightarrow} h'$ s.t. $(h', g') \in Cl(K)$. We distinguish two possibilities again:
  1) $a = \tau$ and $h' = h$. Then αf can use the move $\alpha f \overset{\tau}{\Rightarrow} \alpha f$; we have $(h, g') \in Cl(K)$ and $(\alpha f, h) \in Cl(K)$, hence also $(\alpha f, g') \in Cl(K)$.
  2) $a \neq \tau$ or $h \neq h'$. Then $h \xrightarrow{a} h'$ and as $(\alpha f, h) \in Cl(K)^i$, there is $\alpha f \overset{a}{\Rightarrow} \gamma f$ (or $\alpha f \overset{a}{\Rightarrow} f'$; it is handled in the same way) s.t. $(\gamma f, h') \in Cl(K)$. Hence also $(\gamma f, g') \in Cl(K)$ by Lemma 1.

  II. $\alpha = Y\beta$ and there is h s.t. $(Yh, g) \in K$, $(\beta f, h) \in Cl(K)^i$.
  Let $Y\beta f \xrightarrow{a} \gamma\beta f$. As $(Yh, g) \in K$, we can use induction hypothesis and conclude that there is $g \overset{a}{\Rightarrow} g'$ s.t. $(\gamma h, g') \in Cl(K)$. As $(\beta f, h) \in Cl(K)$, we obtain $(\gamma\beta f, g') \in Cl(K)$ by Lemma 1.
  Let $g \xrightarrow{a} g'$. As $(Yh, g) \in K$, by induction hypothesis we know that Yh can match the move $g \xrightarrow{a} g'$; there are two possibilities:
  1) $Yh \overset{a}{\Rightarrow} \gamma h$ s.t. $(\gamma h, g') \in Cl(K)$. Then also $Y\beta f \overset{a}{\Rightarrow} \gamma\beta f$. As $(\beta f, h) \in Cl(K)$, we immediately have $(\gamma\beta f, g') \in Cl(K)$ as required.

  2) $Yh \overset{a}{\Rightarrow} h'$ s.t. $(h', g') \in Cl(K)$. The transition $Yh \overset{a}{\Rightarrow} h'$ can be 'decomposed' into $Yh \overset{x}{\Rightarrow} h$, $h \overset{y}{\Rightarrow} h'$ where $x = a \wedge y = \tau$ or $x = \tau \wedge y = a$. If $y = \tau$ and $h' = h$, we are done immediately because then $Y\beta \overset{a}{\Rightarrow} \beta$ and as (h, g'), $(\beta, h) \in Cl(K)$, we also have $(\beta, g') \in Cl(K)$ as needed. If $y \neq \tau$ or $h' \neq h$, there is a transition $h \xrightarrow{y} h'$. As $(\beta f, h) \in Cl(K)^i$, due to induction hypothesis we know that there is some $\beta f \overset{y}{\Rightarrow} \gamma f$ (or $\beta f \overset{y}{\Rightarrow} f'$; this is

handled in the same way) with $(\gamma f, h') \in Cl(K)$. Clearly $Y\beta f \overset{a}{\Rightarrow} \gamma f$. As (h', g'), $(\gamma f, h') \in Cl(K)$, we also have $(\gamma f, g') \in Cl(K)$.

III. $\alpha = \beta\gamma$ where β contains an unnormed constant and $(\beta, g) \in Cl(K)^i$.
Let $\alpha \xrightarrow{a} \alpha'$. Then $\alpha' = \delta\gamma$ and $\beta \xrightarrow{a} \delta$. As $(\beta, g) \in Cl(K)^i$, there is $g \overset{a}{\Rightarrow} g'$ s.t. $(\delta, g') \in Cl(K)$ due to the induction hypothesis. Clearly δ contains an unnormed constant, hence $(\delta\gamma, g') \in Cl(K)$ by the last rule of Definition 3.
Let $g \xrightarrow{a} g'$. As $(\beta, g) \in Cl(K)^i$, there is $\beta \overset{a}{\Rightarrow} \delta$ s.t. $(\delta, g') \in Cl(K)$ and δ contains an unnormed constant. Hence $\alpha \overset{a}{\Rightarrow} \delta\gamma$ and $(\delta\gamma, g') \in Cl(K)$ due to the last rule of Definition 3. □

The notion of expansion allows to approximate $\mathcal{B}$ in the following way: $\mathcal{B}^0 = \mathcal{G}$, $\mathcal{B}^{i+1} = Exp(\mathcal{B}^i)$. A proof of the next theorem is now easy to complete.

Theorem 3. *There is a $j \in \mathbb{N}$, bounded by $\mathcal{O}(n\,m^2)$, such that $\mathcal{B}^j = \mathcal{B}^{j+1}$. Moreover, $\mathcal{B}^j = \mathcal{B}$.*

In other words, $\mathcal{B}$ can be obtained from $\mathcal{G}$ in $\mathcal{O}(n\,m^2)$ refinement steps which correspond to the construction of the expansion. The only thing which remains to be shown is that $Exp(K)$ is effectively constructible in polynomial time. To do that, we employ a 'symbolic' technique which allows to represent infinite subsets of BPA state-space in an elegant and succinct way.

Theorem 4. *For all $X \in Const(\Delta)$, $a \in Act(\Delta)$ there is a finite-state automaton $\mathcal{A}_{(X,a)}$ of size $\mathcal{O}(n^2)$ constructible in $\mathcal{O}(n^2)$ time s.t. $L(\mathcal{A}_{(X,a)}) = \{\alpha \mid X \overset{a}{\Rightarrow} \alpha\}$*

Proof. We define a left-linear grammar $G_{(X,a)}$ of size $\mathcal{O}(n^2)$ which generates the mentioned language. This grammar can be converted to $\mathcal{A}_{(X,a)}$ by a standard algorithm known from automata theory (see e.g. [17]). Note that the size of $\mathcal{A}_{(X,a)}$ is essentially the same as the size of $G_{(X,a)}$. First, let us realize that we can compute in $\mathcal{O}(n^2)$ time the sets M_τ and M_a consisting of all $Y \in Const(\Delta)$ s.t. $Y \overset{\tau}{\Rightarrow} \varepsilon$ and $Y \overset{a}{\Rightarrow} \varepsilon$, respectively. Let $G_{(X,a)} = (N, \Sigma, \delta, S)$ where

- $N = \{Y^a, Y^\tau \mid Y \in Const(\Delta)\} \cup \{S\}$. Intuitively, the index indicates whether the action 'a' has already been emitted.
- $\Sigma = Const(\Delta)$
- δ is defined as follows:
  - we add the rule $S \to X^a$ to δ, and if $X \overset{a}{\Rightarrow} \varepsilon$ then we also add the rule $S \to \varepsilon$.
  - for every transition $Y \xrightarrow{a} Z_1.\cdots.Z_k$ of Δ and every i s.t. $1 \le i \le k$ we test whether $Z_j \overset{\tau}{\Rightarrow} \varepsilon$ for every $0 \le j < i$. If this is the case, we add to δ the rules
    $$Y^a \to Z_i \cdots Z_k,\; Y^a \to Z_i^\tau Z_{i+1} \cdots Z_k$$
  - for every transition $Y \xrightarrow{\tau} Z_1.\cdots.Z_k$ of Δ and every i s.t. $1 \le i \le k$ we do the following:
    - ∗ we test whether $Z_j \overset{\tau}{\Rightarrow} \varepsilon$ for every $0 \le j < i$. If this is the case, we add to δ the rules
      $$Y^a \to Z_i^a Z_{i+1} \cdots Z_k,\; Y^\tau \to Z_i^\tau Z_{i+1} \cdots Z_k,\; Y^\tau \to Z_i \cdots Z_k$$

- $*$ we test whether there is a $t < i$ such that $Z_t \overset{a}{\Rightarrow} \varepsilon$ and $Z_j \overset{\tau}{\Rightarrow} \varepsilon$ for every $0 \leq j < i$, $j \neq t$. If this is the case, we add to δ the rules

$$Y^a \rightarrow Z_i^\tau Z_{i+1} \cdots Z_k, \; Y^a \rightarrow Z_i \cdots Z_k$$

The fact that $G_{(X,a)}$ generates the mentioned language is intuitively clear and a formal proof of that is easy. The size of $G_{(X,a)}$ is $\mathcal{O}(n^2)$, as Δ contains $\mathcal{O}(n)$ basic transitions of length $\mathcal{O}(n)$. □

The crucial part of our algorithm (the 'refinement step') is presented in the proof of the next theorem. Our complexity analysis is based on the following facts: Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a non-deterministic automaton with ε-rules, and let t be the total number of states and transitions of $\mathcal{A}$.

- The problem whether a given $w \in \Sigma^*$ belongs to $L(\mathcal{A})$ is decidable in $\mathcal{O}(|w| \cdot t)$ time.
- The problem whether $L(\mathcal{A}) = \emptyset$ is decidable in $\mathcal{O}(t)$ time.

Theorem 5. *Let K be a fundamental relation. The relation $Exp(K)$ can be effectively constructed in $\mathcal{O}(n^4\, m^5)$ time.*

Proof. First we construct the automata $\mathcal{A}_g$ of Theorem 1 for every $g \in Const(\Gamma)$. This takes $\mathcal{O}(n\, m^3)$ time. Then we construct the automata $\mathcal{A}_{(X,a)}$ of Theorem 4 for all X, a. This takes $\mathcal{O}(n^4)$ time. Furthermore, we also compute the set of all pairs of the form $(f, g), (\varepsilon, g)$ which belong to $Cl(K)$. It can be done in $\mathcal{O}(m^2)$ time. Now we show that for each pair of K we can decide in $\mathcal{O}(n^3\, m^3)$ time whether this pair expands in K.

The pairs of the form (f, g) and (ε, g) are easy to handle; there are at most m states f' s.t. $f \xrightarrow{a} f'$, and at most m states g' with $g \overset{a}{\Rightarrow} g'$, hence we need to check only $\mathcal{O}(m^2)$ pairs to verify the first (and consequently also the second) condition of Definition 4. Each such pair can be checked in constant time, because the set of all pairs $(f, g), (\varepsilon, g)$ which belong to $Cl(K)$ has been already computed at the beginning.

Now let us consider a pair of the form (Y, g). First we need to verify that for each $Y \xrightarrow{a} \alpha$ there is some $g \overset{a}{\Rightarrow} h$ s.t. $(\alpha, h) \in Cl(K)$. This requires $\mathcal{O}(n\, m)$ tests whether $\alpha \in L(\mathcal{A}_h)$. As the length of α is $\mathcal{O}(n)$ and the size of $\mathcal{A}_h$ is $\mathcal{O}(n\, m^2)$, each such test can be done in $\mathcal{O}(n^2\, m^2)$ time, hence we need $\mathcal{O}(n^3\, m^3)$ time in total. As for the second condition of Definition 4, we need to find out whether for each $g \xrightarrow{a} h$ there is some $X \overset{a}{\Rightarrow} \alpha$ s.t. $(\alpha, h) \in Cl(K)$. To do that, we simply test the emptiness of $L(\mathcal{A}_{(X,a)}) \cap L(\mathcal{A}_h)$. The size of the product automaton is $\mathcal{O}(n^3\, m^2)$ and we need to perform only $\mathcal{O}(m)$ such tests, hence the time $\mathcal{O}(n^3\, m^3)$ suffices.

Pairs of the form (Yf, g) are handled in a similar way; the first condition of Definition 4 is again no problem, as we are interested only in the '$\xrightarrow{a}$' moves of the left component. Now let $g \xrightarrow{a} g'$. An existence of a 'good' $\overset{a}{\Rightarrow}$ move of Yf can be verified by testing whether one of the following conditions holds:

- $L(\mathcal{A}_{(Y,a)}) \cdot \{f\} \cap L(\mathcal{A}_{g'})$ is nonempty.
- $Y \overset{a}{\Rightarrow} \varepsilon$ and there is some $f \overset{\tau}{\Rightarrow} f'$ s.t. $(f', g') \in Cl(K)$.
- $Y \overset{\tau}{\Rightarrow} \varepsilon$ and there is some $f \overset{a}{\Rightarrow} f'$ s.t. $(f', g') \in Cl(K)$.

All those conditions can be checked in $\mathcal{O}(n^3\, m^3)$ time (the required analysis has been in fact done above). As K contains $\mathcal{O}(n\, m^2)$ pairs, the total time which is needed to compute $Exp(K)$ is $\mathcal{O}(n^4\, m^5)$. □

As the BPA process E (introduced at the beginning of this section) is an element of $Const(\Delta)$, we have that $E \approx F$ iff $(E, F) \in \mathcal{B}$. To compute $\mathcal{B}$, we have to perform the computation of the expansion $\mathcal{O}(n\, m^2)$ times (see Theorem 3). This gives us the following main theorem:

Theorem 6. *Weak bisimilarity is decidable between BPA and finite-state processes in $\mathcal{O}(n^5\, m^7)$ time.*

The fact that weak bisimilarity is not a congruence w.r.t. sequential composition is rather unpleasant; any equivalence which is to be considered as 'behavioural' should have this property. We propose a solution to this problem by designing a natural refinement of weak bisimilarity called *termination-sensitive bisimilarity*. This relation distinguishes between the following 'basic phenomenons' of sequencing:

– *successful termination* of the process which is currently being executed. The system can then continue to execute the next process in the queue.
– *unsuccessful termination* of the executed process (deadlock). This models a severe error which causes the whole system to 'get stuck'.
– *entering an infinite internal loop* (livelock).

Termination-sensitive bisimilarity is a congruence w.r.t. sequencing, and it is also decidable between BPA and finite state processes in polynomial time. It can be proved by adapting the proof for weak bisimilarity. Formal definitions and proofs are omitted due to the lack of space—see [21] for details.

4 Normed BPP Processes

In this section we prove that weak bisimilarity is decidable in polynomial time between normed BPP and finite-state processes. The basic structure of our proof is similar to the one for BPA. The key is that the weak bisimulation problem can be decomposed into problems about the single constants and their interaction with each other. In particular, a normed BPP process is finite w.r.t. weak bisimilarity iff every single reachable process constant is finite w.r.t. weak bisimilarity. This does not hold for general BPP and thus our construction does not carry over to general BPP.

Even for normed BPP, we have to solve some additional problems. The bisimulation base and its closure are simpler due to the normedness assumption, but the 'symbolic' representation of BPP state-space is more problematic (see below). The set of states which are reachable from a given BPP state in one '$\overset{a}{\Rightarrow}$' move is no longer regular, but it can be in some sense represented by a CF-grammar. In our algorithm we use the facts that emptiness of a CF language is decidable in polynomial time, and that CF languages are closed under intersection with regular languages. Most proofs in this section are omitted due to the lack of space. See [21] for details.

Let E be a BPP process and F a finite-state process with the underlying systems Δ and Γ, respectively. We can assume w.l.o.g. that $E \in \mathit{Const}(\Delta)$. Elements of $\mathit{Const}(\Delta)$ are denoted by $X, Y, Z, \ldots$, elements of $\mathit{Const}(\Gamma)$ by $f, g, h, \ldots$ The set of all parallel expressions over $\mathit{Const}(\Delta)$ is denoted by $\mathit{Const}(\Delta)^{\otimes}$ and its elements by Greek letters $\alpha, \beta, \ldots$ The size of Δ is denoted by n, and the size of Γ by m.

In our constructions we represent certain subsets of $\mathit{Const}(\Delta)^{\otimes}$ by finite automata and CF grammars. The problem is that elements of $\mathit{Const}(\Delta)^{\otimes}$ are considered modulo commutativity; however, finite automata and CF grammars of course distinguish between different 'permutations' of the same word. As the classes of regular and CF languages are not closed under permutation, this problem is important. As we want to clarify the distinction between α and its possible 'linear representations', we define for each α the set $\mathit{Lin}(\alpha)$ as follows:

$$\mathit{Lin}(X_1 \| \cdots \| X_k) = \{X_{p(1)} \cdots X_{p(k)} \mid p \text{ is a permutation of the set } \{1, \cdots, k\}\}$$

For example, $\mathit{Lin}(X \| Y \| Z) = \{XYZ,\ XZY,\ YXZ,\ YZX,\ ZXY,\ ZYX\}$. We also assume that each $\mathit{Lin}(\alpha)$ contains some (unique) element called *canonical form* of $\mathit{Lin}(\alpha)$. It is not important how the canonical form is chosen; we need it just to make some constructions deterministic (for example, we can fix some linear order on process constants and let the canonical form of $\mathit{Lin}(\alpha)$ be the sorted order of constants of α).

Definition 5. *A relation K is* fundamental *iff it is a subset of $(\mathit{Const}(\Delta) \cup \{\varepsilon\}) \times \mathit{Const}(\Gamma)$. The greatest fundamental relation is denoted by $\mathcal{G}$. The* bisimulation base *for Δ and Γ, denoted $\mathcal{B}$, is defined as follows:*

$$\mathcal{B} = \{(X, f) \mid X \approx f\} \cup \{(\varepsilon, f) \mid \varepsilon \approx f\}$$

Definition 6. *Let K be a fundamental relation. The* closure *of K, denoted $\mathit{Cl}(K)$, is the least relation M which satisfies*

1. $K \subseteq M$
2. *if $(X, g) \in K$, $(\beta, h) \in M$, and $f \approx g \| h$, then $(\beta \| X, f) \in M$*
3. *if $(\varepsilon, g) \in K$, $(\beta, h) \in M$, and $f \approx g \| h$, then $(\beta, f) \in M$*

The family of $\mathit{Cl}(K)^i$ approximations is defined in the same way as in the previous section.

Lemma 3. *Let $(\alpha, f) \in \mathit{Cl}(K)$, $(\beta, g) \in \mathit{Cl}(K)$, $f \| g \approx h$. Then $(\alpha \| \beta, h) \in \mathit{Cl}(K)$.*

Again, the closure of the bisimulation base is the greatest weak bisimulation between processes of Δ and Γ.

Theorem 7. *Let $\alpha \in \mathit{Const}(\Delta)^{\otimes}$, $f \in \mathit{Const}(\Gamma)$. We have that $\alpha \approx f$ iff $(\alpha, f) \in \mathit{Cl}(\mathcal{B})$.*

The closure of any fundamental relation can in some sense be represented by a finite-state automaton, as stated in the next theorem.

Theorem 8. *Let K be a fundamental relation. For each $g \in Const(\Gamma)$ there is a finite-state automaton $\mathcal{A}_g$ of size $\mathcal{O}(n\,m)$ constructible in $\mathcal{O}(n\,m)$ time s.t. the following conditions hold:*

– *whenever $\mathcal{A}_g$ accepts an element of $Lin(\alpha)$, then $(\alpha, g) \in Cl(K)$*
– *if $(\alpha, g) \in Cl(K)$, then $\mathcal{A}_g$ accepts at least one element of $Lin(\alpha)$*

It is important to realize that if $(\alpha, g) \in Cl(K)$, then $\mathcal{A}_g$ does not necessarily accept *all* elements of $Lin(\alpha)$. Generally, $\mathcal{A}_g$ cannot be 'repaired' to do so (see the beginning of this section); however, there is actually no need for such 'repairs', because $\mathcal{A}_g$ has the following nice property:

Lemma 4. *Let K be a fundamental relation s.t. $\mathcal{B} \subseteq K$. If $\alpha \approx g$, then the automaton $\mathcal{A}_g$ of (the proof of) Theorem 8 constructed for K accepts all elements of $Lin(\alpha)$.*

The set of states which are reachable from a given $X \in Const(\Delta)$ in one '$\overset{a}{\Rightarrow}$' move is no longer regular, but it can, in some sense, be represented by a CF grammar.

Theorem 9. *For all $X \in Const(\Delta)$, $a \in Act(\Delta)$ there is a context-free grammar $G_{(X,a)}$ in 3-GNF of size $\mathcal{O}(n^4)$ constructible in $\mathcal{O}(n^4)$ time s.t. the following two conditions hold:*

– *if $G_{(X,a)}$ generates an element of $Lin(\alpha)$, then $X \overset{a}{\Rightarrow} \alpha$*
– *if $X \overset{a}{\Rightarrow} \alpha$, then $G_{(X,a)}$ generates at least one element of $Lin(\alpha)$*

The notion of expansion is defined in a different way (when compared to the one of the previous section).

Definition 7. *Let K be a fundamental relation. We say that a pair $(X, f) \in K$* expands *in K iff the following two conditions hold:*

– *for each $X \xrightarrow{a} \alpha$ there is some $f \overset{a}{\Rightarrow} g$ s.t. $\overline{\alpha} \in L(\mathcal{A}_g)$, where $\overline{\alpha}$ is the canonical form of $Lin(\alpha)$.*
– *for each $f \xrightarrow{a} g$ the language $L(\mathcal{A}_g) \cap L(G_{(X,a)})$ is non-empty.*

A pair $(\varepsilon, f) \in K$ expands in K iff $f \xrightarrow{a} g$ implies $a = \tau$, and for each $f \xrightarrow{\tau} g$ we have that $\varepsilon \in L(\mathcal{A}_g)$. The set of all pairs of K which expand in K is denoted by $Exp(K)$.

Theorem 10. *Let K be a fundamental relation. The set $Exp(K)$ can be computed in $\mathcal{O}(n^{11}\,m^8)$ time.*

Proof. First we compute the automata $\mathcal{A}_g$ of Theorem 8 for all $g \in Const(\Gamma)$. This takes $\mathcal{O}(n\,m^2)$ time. Then we compute the grammars $G_{(X,a)}$ of Theorem 9 for all $X \in Const(\Delta)$, $a \in Act$. This takes $\mathcal{O}(n^6)$ time. Now we show that it is decidable in $\mathcal{O}(n^{10}\,m^7)$ time whether a pair (X, f) of K expands in K.

The first condition of Definition 7 can be checked in $\mathcal{O}(n^3\,m^2)$ time, as there are $\mathcal{O}(n)$ transitions $X \xrightarrow{a} \alpha$, $\mathcal{O}(m)$ states g s.t. $f \overset{a}{\Rightarrow} g$, and for each such pair (α, g) we verify whether $\overline{\alpha} \in L(\mathcal{A}_g)$ where $\overline{\alpha}$ is the canonical form of $Lin(\alpha)$; this membership test can be done in $\mathcal{O}(n^2\,m)$ time, as the size of $\overline{\alpha}$ is $\mathcal{O}(n)$ and the size of $\mathcal{A}_g$ is $\mathcal{O}(n\,m)$.

The second condition of Definition 7 is more expensive. To test the emptiness of $L(\mathcal{A}_g) \cap L(G_{(X,a)})$, we first construct a pushdown automaton $\mathcal{P}$ which recognises this language. $\mathcal{P}$ has $\mathcal{O}(m)$ control states and its total size is $\mathcal{O}(n^5\, m)$. Furthermore, each rule $pX \xrightarrow{a} q\alpha$ of $\mathcal{P}$ has the property that $length(\alpha) \leq 2$, because $G_{(X,a)}$ is in 3-GNF. Now we transform this automaton to an equivalent CF grammar by a well-known procedure described e.g. in [17]. The size of the resulting grammar is $\mathcal{O}(n^5\, m^3)$, and its emptiness can be thus checked in $\mathcal{O}(n^{10}\, m^6)$ time (cf. [17]). This construction has to be performed $\mathcal{O}(m)$ times, hence we need $\mathcal{O}(n^{10}\, m^7)$ time in total.

Pairs of the form (ε, f) are handled in a similar (but less expensive) way. As K contains $\mathcal{O}(n\, m)$ pairs, the computation of $Exp(K)$ takes $\mathcal{O}(n^{11}\, m^8)$ time. □

The previous theorem is actually a straightforward consequence of Definition 7. The next theorem says that *Exp* really does what we need.

Theorem 11. *Let K be a fundamental relation s.t. $Exp(K) = K$. Then $Cl(K)$ is a weak bisimulation.*

Proof. Let $(\alpha, f) \in Cl(K)^i$. We prove that for each $\alpha \xrightarrow{a} \beta$ there is some $f \overset{a}{\Rightarrow} g$ s.t. $(\beta, g) \in Cl(K)$ and vice versa. By induction on i.

- $i = 0$. Then $(\alpha, f) \in K$, and we can distinguish the following two possibilities:
  1. $\alpha = X$
     Let $X \xrightarrow{a} \beta$. By Definition 7 there is $f \overset{a}{\Rightarrow} g$ s.t. $\overline{\beta} \in L(\mathcal{A}_g)$ for some $\overline{\beta} \in Lin(\beta)$. Hence $(\beta, g) \in Cl(K)$ due to the first part of Theorem 8.
     Let $f \xrightarrow{a} g$. By Definition 7 there is some string $w \in L(\mathcal{A}_g) \cap L(G_{(X,a)})$. Let $w \in Lin(\beta)$. We have $X \overset{a}{\Rightarrow} \beta$ due to the first part of Theorem 9, and $(\beta, g) \in Cl(K)$ due to Theorem 8.
  2. $\alpha = \varepsilon$
     Let $f \xrightarrow{a} g$. Then $a = \tau$ and $\varepsilon \in L(\mathcal{A}_g)$ by Definition 7. Hence $(\varepsilon, g) \in Cl(K)$ due to Theorem 8.
- **Induction step.** Let $(\alpha, f) \in Cl(K)^{i+1}$. There are two possibilities.
  I. $\alpha = X \| \gamma$ and there are r, s s.t. $(X, r) \in K$, $(\gamma, s) \in Cl(K)^i$, and $r \| s \approx f$.
     Let $X \| \alpha \xrightarrow{a} \beta$. The action '$a$' can be emitted either by X or by α. We distinguish the two cases.
     1) $X \| \gamma \xrightarrow{a} \delta \| \gamma$. As $(X, r) \in K$ and $X \xrightarrow{a} \delta$, there is some $r \overset{a}{\Rightarrow} r'$ s.t. $(\delta, r') \in Cl(K)$. As $r \| s \approx f$ and $r \overset{a}{\Rightarrow} r'$, there is some $f \overset{a}{\Rightarrow} g$ s.t. $r' \| s \approx g$. To sum up, we have $(\delta, r') \in Cl(K)$, $(\gamma, s) \in Cl(K)$, $r' \| s \approx g$, hence $(\delta \| \gamma, g) \in Cl(K)$ due to Lemma 3.
     2) $X \| \gamma \xrightarrow{a} X \| \rho$. As $(\gamma, s) \in Cl(K)^i$ and $\gamma \xrightarrow{a} \rho$, there is $s \overset{a}{\Rightarrow} s'$ s.t. $(\rho, s') \in Cl(K)$. As $r \| s \approx f$ and $s \overset{a}{\Rightarrow} s'$, there is $f \overset{a}{\Rightarrow} g$ s.t. $(r \| s') \approx g$. Due to Lemma 3 we obtain $(X \| \rho, g) \in Cl(K)$.
     Let $f \xrightarrow{a} g$. As $r \| s \approx f$, there are $r \overset{x}{\Rightarrow} r'$, $s \overset{y}{\Rightarrow} s'$ where $x = a \wedge y = \tau$ or $x = \tau \wedge y = a$ s.t. $r' \| s' \approx g$. As $(X, r) \in K$, $(\gamma, s) \in Cl(K)^i$, there are $X \overset{x}{\Rightarrow} \delta$, $\gamma \overset{y}{\Rightarrow} \rho$ s.t. (δ, r'), $(\rho, s') \in Cl(K)$. Clearly $X \| \gamma \overset{a}{\Rightarrow} \delta \| \rho$ and $(\delta \| \rho, g) \in Cl(K)$ due to Lemma 3.
  II. $(\alpha, r) \in Cl(K)^i$ and there is some s s.t. $(\varepsilon, s) \in K$ and $r \| s \approx f$.
     The proof can be completed along the same lines as above. □

Now we can approximate (and compute) the bisimulation base in the same way as in the previous section.

Theorem 12. *There is a $j \in \mathbb{N}$, bounded by $\mathcal{O}(n\,m)$, such that $\mathcal{B}^j = \mathcal{B}^{j+1}$. Moreover, $\mathcal{B}^j = \mathcal{B}$.*

Theorem 13. *Weak bisimilarity between normed BPP and finite-state processes is decidable in $\mathcal{O}(n^{12}\,m^9)$ time.*

5 Conclusions

We have proved that weak bisimilarity is decidable between BPA processes and finite-state processes in $\mathcal{O}(n^5\,m^7)$ time, and between normed BPP and finite-state processes in $\mathcal{O}(n^{12}\,m^9)$ time. It may be possible to improve the algorithm by re-using previously computed information, for example about sets of reachable states, but the exponents would still be very high. This is because the whole bisimulation basis is constructed. To get a more efficient algorithm, one could try to avoid this. Note however, that once we construct $\mathcal{B}$ (for a BPA/nBPP system Δ and a finite-state system Γ) and the automaton $\mathcal{A}_g$ of Theorem 1/Theorem 8 (for $K = \mathcal{B}$ and some $g \in Const(\Gamma)$), we can decide weak bisimilatity between a BPA/nBPP process α over Δ and a process $f \in Const(\Gamma)$ in time $\mathcal{O}(|\alpha|)$—it suffices to test whether $\mathcal{A}_f$ accepts α (observe that there is no substantial difference between $\mathcal{A}_f$ and $\mathcal{A}_g$ except for the initial state).

The technique of bisimulation bases has also been used for strong bisimilarity in [15, 16]. However, those bases are different from ours; their design and the way how they generate 'new' bisimilar pairs of processes rely on additional algebraic properties of strong bisimilarity (which is a full congruence w.r.t. sequencing, allows for unique decompositions of normed processes w.r.t. sequencing and parallelism, etc.). The main difficulty of those proofs is to show that the membership in the 'closure' of the defined bases is decidable in polynomial time. The main point of our proofs is the use of 'symbolic' representation of infinite subsets of BPA and BPP state-space.

We would also like to mention that our proofs can be easily adapted to other bisimulation-like equivalences, where the notion of 'bisimulation-like' equivalence is the one of [19]. A concrete example is termination-sensitive bisimilarity of Section 3. Intuitively, almost every bisimulation-like equivalence has the algebraic properties which are needed for the construction of the bisimulation base, and the 'symbolic' technique for state-space representation can also be adapted. See [19] for details.

References

1. J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *JACM*, 40:653–682, 1993.
2. J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.
3. A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model checking. In *Proceedings of CONCUR'97*, volume 1243 of *LNCS*, pages 135–150. Springer, 1997.

4. O. Burkart, D. Caucal, and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proceedings of MFCS'95*, volume 969 of *LNCS*, pages 423–433. Springer, 1995.
5. O. Burkart and J. Esparza. More infinite results. *ENTCS*, 5, 1997.
6. D. Caucal. Graphes canoniques des graphes algébriques. *RAIRO*, 24(4):339–352, 1990.
7. I. Černá, M. Křetínský, and A. Kučera. Comparing expressibility of normed BPA and normed BPP processes. *Acta Informatica*, 36(3):233–256, 1999.
8. S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for all basic parallel processes. In *Proceedings of CONCUR'93*, volume 715 of *LNCS*, pages 143–157. Springer, 1993.
9. S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.
10. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. In *Proceedings of FCT'95*, volume 965 of *LNCS*, pages 221–232. Springer, 1995.
11. J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.
12. J. Esparza and J. Knop. An automata-theoretic approach to interprocedural data-flow analysis. In *Proceedings of FoSSaCS'99*, volume 1578 of *LNCS*, pages 14–30. Springer, 1999.
13. J.F. Groote. A short proof of the decidability of bisimulation for normed BPA processes. *Information Processing Letters*, 42:167–171, 1992.
14. Y. Hirshfeld. Bisimulation trees and the decidability of weak bisimulations. *ENTCS*, 5, 1996.
15. Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *TCS*, 158:143–159, 1996.
16. Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimulation equivalence of normed basic parallel processes. *MSCS*, 6:251–259, 1996.
17. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
18. H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. In *Proceedings of LICS'91*, pages 376–386. IEEE Computer Society Press, 1991.
19. P. Jančar, A. Kučera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. In *Proceedings of ICALP'98*, volume 1443 of *LNCS*, pages 200–211. Springer, 1998.
20. A. Kučera. On effective decomposability of sequential behaviours. *TCS*. To appear.
21. A. Kučera and R. Mayr. Weak bisimilarity with infinite-state systems can be decided in polynomial time. Technical report TUM-I9830, Institut für Informatik, TU-München, 1998.
22. A. Kučera and R. Mayr. Simulation preorder on simple process algebras. In *Proceedings of ICALP'99*, LNCS. Springer, 1999. To apper.
23. R. Mayr. Process rewrite systems. *Information and Computation*. To appear.
24. R. Mayr. Weak bisimulation and model checking for basic parallel processes. In *Proceedings of FST&TCS'96*, volume 1180 of *LNCS*, pages 88–99. Springer, 1996.
25. R. Mayr. Strict lower bounds for model checking BPA. *ENTCS*, 18, 1998.
26. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
27. F. Moller. Infinite results. In *Proceedings of CONCUR'96*, volume 1119 of *LNCS*, pages 195–216. Springer, 1996.
28. R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, 1987.
29. D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings* 5^{th} *GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.
30. J. Stříbrná. Hardness results for weak bisimilarity of simple process algebras. *ENTCS*, 18, 1998.