Weak Bisimilarity between Finite-State Systems and BPA or Normed BPP is Decidable in Polynomial Time

Antonín Kučera [a,1], Richard Mayr [b,2]

[a] *Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic. E-mail:* tony@fi.muni.cz

[b] *LIAFA - Université Denis Diderot - Case 7014 - 2, place Jussieu, F-75251 Paris Cedex 05. France. E-mail:* mayr@liafa.jussieu.fr

Abstract

We prove that weak bisimilarity is decidable in polynomial time between finite-state systems and several classes of infinite-state systems: context-free processes (BPA) and normed Basic Parallel Processes (normed BPP). To the best of our knowledge, these are the first polynomial algorithms for weak bisimilarity problems involving infinite-state systems.

Key words: concurrency, infinite-state systems, process algebras, verification, bisimulation

1 Introduction

Recently, a lot of attention has been devoted to the study of decidability and complexity of verification problems for infinite-state systems [33,12,5]. We consider the problem of weak bisimilarity between certain infinite-state processes and finite-state ones. The motivation is that the intended behavior 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, recursion, creation of new parallel subprocesses). The aim is

[1] On leave at the Institute for Informatics, Technical University Munich. Supported by the Alexander von Humboldt Foundation and by the Grant Agency of the Czech Republic, grant No. 201/00/0400.

[2] Supported by DAAD Post-Doc grant D/98/28804.

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 (also known as context-free 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 data-flow analysis in optimizing compilers [13]. BPP [8] model simple parallel systems (due to the binary operator of parallel composition). They are equivalent to communication-free nets, the subclass of Petri nets [36] where every transition has exactly one input-place [11]. A process is *normed* iff at every reachable state it can terminate via a finite sequence of computational steps.

Although the syntax of BPA and BPP allows to define simple infinite-state systems, from the practical point of view it is also important that they can give very compact definitions of finite-state processes (i.e., the size of a BPA/BPP definition of a finite-state process F can be exponentially smaller than the number of states of F—see the next section). As our verification algorithms are polynomial in the size of the BPA/BPP definition, we can (potentially) verify *very* large processes. Thus, our results can be also seen as a way how to overcome the well-known problem of state-space explosion.

The state of the art. Baeten, Bergstra, and Klop [1] proved that *strong bisimilarity* [35] is decidable for normed BPA processes. Simpler proofs have been given later in [20,14], and there is even a polynomial-time algorithm [17]. The decidability result has later been extended to the class of all (not necessarily normed) BPA processes in [10], but the best known algorithm is doubly exponential [4]. Decidability of strong bisimilarity for BPP processes has been established in [9], but the associated complexity analysis does not yield an elementary upper bound (although some deeper examination might in principle show that the algorithm *is* elementary). Strong bisimilarity of BPP has been shown to be co-$\mathcal{NP}$-hard in [28]. However, there is a polynomial-time algorithm for the subclass of normed BPP [18]. Strong bisimilarity between normed BPA and normed BPP is also decidable [7]. This result even holds for parallel compositions of normed BPA and normed BPP processes [22]. Recently, this has even been generalized to the class of all normed PA-processes [16].

For weak bisimilarity, much less is known. Semidecidability of weak bisimilarity for BPP has been shown in [11]. In [15] 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 $\mathcal{P} \neq \mathcal{NP}$). Weak bisimilarity of (normed) BPA is *PSPACE*-hard [38]. An $\mathcal{NP}$ lower bound for weak bisimilarity of BPP has been shown by Stříbrná in [38]. This result has been improved to Π_2^p-hardness by Mayr [28] and very recently to *PSPACE*-hardness by Srba in [37]. Moreover, the *PSPACE* lower bound for weak bisimilarity of BPP in [37] holds even for normed BPP.

The situation is dramatically different if we consider weak bisimilarity between certain infinite-state processes and finite-state ones. This study is motivated by the fact that the intended behavior of a process is often easy to specify (by a finite-state system), but a 'real' implementation can contain components which are infinite-state (e.g., counters, buffers, recursion, creation of new parallel subprocesses). It has been shown in [26] that weak bisimilarity between BPP and finite-state processes is decidable. A more general result has recently been obtained in [21], 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 [31,30] strictly subsumes not only BPA and BPP, but also PA [2] and pushdown processes. The result in [21] is obtained by a general reduction to the model-checking problem for the simple branching-time temporal logic *EF*, which is decidable for PAD [30]. As the model-checking problem for *EF* is hard (for example, it is known to be *PSPACE*-complete for BPP [26] and *PSPACE*-complete for BPA [39,27]), 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. 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 [17,18] only work for the normed subclasses of BPA and BPP, respectively). 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. A recent work [29] shows that strong bisimilarity between pushdown processes (a proper superclass of BPA) and finite-state ones is already *PSPACE*-hard. Furthermore, weak bisimilarity remains computationally intractable (*DP*-hard) even between processes of one-counter nets and finite-state processes [23] (one-counter nets are computationally equivalent to the subclass of Petri nets with at most one unbounded place and can be thus also seen as very simple pushdown automata). Hence, our result for BPA is rather tight. The question whether the result for normed BPP can be extended to the class of all (not necessarily normed) BPP processes is left open. It should also be noted that *simulation equivalence* with a finite-state process is co-$\mathcal{NP}$-hard for BPA/BPP processes [25], *EXPTIME*-complete for

pushdown processes [24], but polynomial for one-counter nets [24].

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 of 'small' size 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. An interesting point about this construction is that it works although weak bisimulation is not a congruence w.r.t. sequential composition, but only a *left congruence*. In Section 4, we propose a natural refinement of weak bisimilarity called termination-sensitive bisimilarity which *is* a congruence and which is also decidable between BPA and finite-state processes in polynomial time. The result demonstrates that the technique which has been used for weak bisimilarity actually has a wider applicability—it can be adapted to many 'bisimulation-like' equivalences. Finally, we should note that our aim is just to show that the mentioned problems are in $\mathcal{P}$; although we do compute the degrees of bounding polynomials explicitly, our analysis is quite simple and rough. Moreover, both presented algorithms could be easily improved by employing standard techniques. See the final section for further comments.

2 Definitions

We use process rewrite systems [31] 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 $X \in Const$ 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* [31] is specified by a finite set of *rules* Δ which have the form $E \stackrel{a}{\rightarrow} F$, where $E, F \in \mathcal{E}$ and $a \in Act$. $Const(\Delta)$ and $Act(\Delta)$ denote the sets of process constants and actions which are used in the rules of Δ, respectively (note that these sets are finite). Each process rewrite system Δ defines a unique transition system where states are process expressions over $Const(\Delta)$, $Act(\Delta)$ is the set of labels, and transitions are determined by Δ

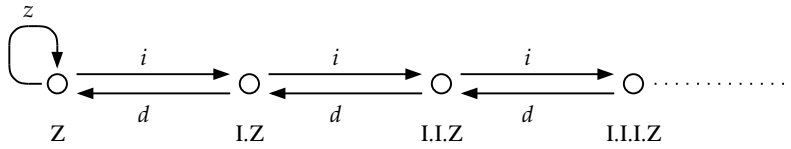
and the following inference rules (remember that '||' is commutative):

$$\frac{(E \stackrel{a}{\to} F) \in \Delta}{E \stackrel{a}{\to} F} \qquad \frac{E \stackrel{a}{\to} E'}{E.F \stackrel{a}{\to} E'.F} \qquad \frac{E \stackrel{a}{\to} E'}{E \| F \stackrel{a}{\to} E' \| F}$$

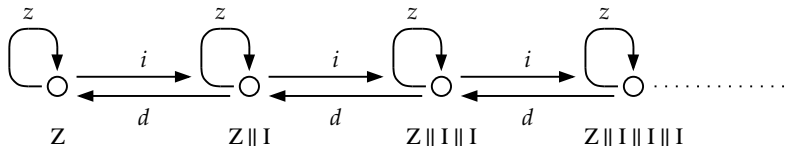
We extend the notation $E \stackrel{a}{\to} F$ to elements of Act^* in the standard way. F is *reachable* from E if $E \stackrel{w}{\to} F$ for some $w \in Act^*$.

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 Δ is restricted to $Const(\Delta)$, the set of all sequential expressions over $Const(\Delta)$, or the set of all parallel expressions over $Const(\Delta)$, respectively.

Example 1 *Let* $\Delta = \{Z \stackrel{z}{\to} Z, Z \stackrel{i}{\to} I.Z, I \stackrel{i}{\to} I.I, I \stackrel{d}{\to} \varepsilon\}$ *be a process rewrite system. We see that* Δ *is a BPA system; a part of the transition system associated to* Δ *which is reachable from* Z *looks as follows:*



If we replace each occurrence of the '.' operator with the '||' operator, we obtain a BPP system which generates the following transition system (again, we only draw the part reachable from Z):



A process is *normed* iff at every reachable state it can (successfully) terminate via a finite sequence of computational steps. For a BPA or BPP process, this is equivalent to the condition that for each constant $X \in Const(\Delta)$ of its underlying system Δ there is some $w \in Act^*$ such that $X \stackrel{w}{\to} \varepsilon$. We call such constants X with this property *normed*.

The semantical equivalence we are interested in here is *weak bisimilarity* [32]. This relation distinguishes between 'observable' and 'internal' moves (compu-

tational steps); the internal moves are modeled by a special action which is denoted 'τ' by convention. In what follows we consider process expressions over $Const(\Delta)$ where Δ is some fixed process rewrite system.

Definition 2 *The* extended transition relation '$\stackrel{a}{\Rightarrow}$' *is defined by* $E \stackrel{a}{\Rightarrow} F$ *iff either* $E = F$ *and* $a = \tau$, *or* $E \stackrel{\tau^i}{\rightarrow} E' \stackrel{a}{\rightarrow} E'' \stackrel{\tau^j}{\rightarrow} F$ *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 each* $a \in Act$:

- *if* $E \stackrel{a}{\rightarrow} E'$ *then there is* $F \stackrel{a}{\Rightarrow} F'$ *such that* $(E', F') \in R$, *and*
- *if* $F \stackrel{a}{\rightarrow} F'$ *then there is* $E \stackrel{a}{\Rightarrow} E'$ *such that* $(E', F') \in R$.

Processes E, F *are* weakly bisimilar, *written* $E \approx F$, *iff there is a weak bisimulation relating them.*

Weak bisimilarity can be approximated by the family of $\approx_i$ relations, which are defined as follows:

- $E \approx_0 F$ for every E, F
- $E \approx_{i+1} F$ iff $E \approx_i F$ and the following conditions hold:
  - if $E \stackrel{a}{\rightarrow} E'$ then there is $F \stackrel{a}{\Rightarrow} F'$ such that $E' \approx_i F'$
  - if $F \stackrel{a}{\rightarrow} F'$ then there is $E \stackrel{a}{\Rightarrow} E'$ such that $E' \approx_i F'$

It is worth noting that $\approx_i$ is not an equivalence for $i \geq 1$, as it is not transitive. It is possible to approximate weak bisimilarity in a different way so that the approximations *are* equivalences (see [21]). However, we do not need this for our purposes.

Let Γ be a finite-state system with n states, $f, g \in Const(\Gamma)$. It is easy to show that the problem whether $f \approx g$ is decidable in $\mathcal{O}(n^3)$ time. First we compute in $\mathcal{O}(n^3)$ time the transitive closure of the transition system w.r.t. the $\stackrel{\tau}{\rightarrow}$ transitions and thus obtain a new system in which $\stackrel{a}{\rightarrow}$ is the same as $\stackrel{a}{\Rightarrow}$ in the old system. Then it suffices to decide strong bisimilarity of f and g in the new system. This can be done in $\mathcal{O}(n^2 \log n)$ time, using partition refinement techniques from [34].

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

In this section we prove that weak bisimilarity is decidable between BPA and finite-state processes in polynomial time.

Let E be a BPA process with the underlying system Δ, F a finite-state process with the underlying system Γ such that $Const(\Delta) \cap Const(\Gamma) = \emptyset$. We assume (w.l.o.g.) that $E \in Const(\Delta)$. Moreover, we also assume that for all $f, g \in Const(\Gamma)$, $a \in Act$ such that $f \neq g$ or $a \neq \tau$ we have that $f \stackrel{a}{\Rightarrow} g$ implies $f \stackrel{a}{\rightarrow} g \in \Gamma$. If those '$\stackrel{a}{\rightarrow}$' transitions are missing in Γ, we can add them safely. Adding these transitions does not change the weak bisimilarity relation among the states. In order to do this it suffices to compute (in cubic time) the transitive closure of Γ w.r.t. the τ transitions. These extra transitions do not influence our complexity estimations, as we always consider the worst case when Γ has all possible transitions. The condition that $a \neq \tau$ is there because we do not want to add new transitions of the form $f \stackrel{\tau}{\rightarrow} f$, because then our proof for weak bisimilarity would not immediately work for termination-sensitive bisimilarity (which is defined at the end of 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 $X \stackrel{w}{\rightarrow} \varepsilon$ for some $w \in Act^*$). Note that it is decidable in $\mathcal{O}(n^2)$ time if a constant is normed. 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 [17,18]). 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 9 (we compute the *expansion* of the currently computed approximation of the base). Intuitively, a pair of processes belongs to the expansion iff for each $\stackrel{a}{\rightarrow}$ move of one component there is a $\stackrel{a}{\Rightarrow}$ move of the other component such that the

resulting pair of processes can be generated from the current approximation of $\mathcal{B}$. We have to overcome two 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 '$\stackrel{a}{\Rightarrow}$' 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 6 and 12). This allows to compute the expansion in polynomial time.

Definition 3 *A relation K is* well-formed *iff it is a subset of the relation $\mathcal{G}$ defined by*

$$\begin{aligned}\mathcal{G} = &((\mathit{Normed}(\Delta) \cdot \mathit{Const}(\Gamma)) \times \mathit{Const}(\Gamma)) \\ &\cup (\mathit{Const}(\Delta) \times \mathit{Const}(\Gamma)) \\ &\cup (\mathit{Const}(\Gamma) \times \mathit{Const}(\Gamma)) \\ &\cup (\{\varepsilon\} \times \mathit{Const}(\Gamma))\end{aligned}$$

Note that the size of any well-formed relation is $\mathcal{O}(n\,m^2)$ and that $\mathcal{G}$ is the greatest well-formed relation.

One of the well-formed relations is of special importance.

Definition 4 *The* bisimulation base *for Δ and Γ, denoted $\mathcal{B}$, is defined as follows:*

$$\begin{aligned}\mathcal{B} = &\{(Y\,f, g) \mid Y\,f \approx g,\, Y \in \mathit{Normed}(\Delta)\} \\ &\cup \{(X, g) \mid X \approx g\} \\ &\cup \{(f, g) \mid f \approx g\} \\ &\cup \{(\varepsilon, g) \mid \varepsilon \approx g\}\end{aligned}$$

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 $X \stackrel{\tau}{\to} X$, $Y \stackrel{\tau}{\to} \varepsilon$, $Z \stackrel{a}{\to} Z$. Now $X \approx Y$, but $XZ \not\approx YZ$). This generation procedure can be defined for any well-formed relation as follows:

Definition 5 *Let K be a well-formed relation. The* closure *of K, denoted*

$Cl(K)$, 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* $\beta \in Const(\Delta)^*$ *and* $h \in Const(\Gamma)$.

Note that $Cl(K)$ contains elements of just two forms – (α, g) and $(\alpha f, g)$. Clearly $Cl(K) = \bigcup_{i=0}^{\infty} Cl(K)^i$ where $Cl(K)^0 = K$ and $Cl(K)^{i+1}$ consists of $Cl(K)^i$ and the pairs which can be immediately derived from $Cl(K)^i$ by the rules 2–6 of Definition 5.

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

Theorem 6 *Let K be a well-formed relation. For each $g \in Const(\Gamma)$ there is a finite-state automaton $\mathcal{A}_g$ of size $\mathcal{O}(n\,m^2)$ constructible in $\mathcal{O}(n\,m^2)$ time such that $L(\mathcal{A}_g) = \{\alpha \mid (\alpha, g) \in Cl(K)\} \cup \{\alpha f \mid (\alpha f, g) \in Cl(K)\}$.*

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

- $N = \{\overline{f} \mid f \in Const(\Gamma)\} \cup \{U\}$
- $\Sigma = Const(\Delta) \cup Const(\Gamma)$
- δ is defined as follows:
  - for each $(\varepsilon, h) \in K$ we add the rule $\overline{h} \to \varepsilon$.
  - for each $(f, h) \in K$ we add the rules $\overline{h} \to \overline{f}$, $\overline{h} \to f$.
  - 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 $X \in Const(\Delta)$, $f \in Const(\Gamma)$ 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., [19]). 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.

It follows immediately that for any well-formed relation K, the membership problem for $Cl(K)$ is decidable in polynomial time. Another property of $Cl(K)$ is specified in the lemma below.

Lemma 7 *Let* $(\alpha f, g) \in Cl(K)$. *If* $(\beta h, f) \in Cl(K)$, *then also* $(\alpha\beta h, g) \in Cl(K)$. *Similarly, if* $(\beta, f) \in Cl(K)$, *then also* $(\alpha\beta, g) \in Cl(K)$.

PROOF. We just give a proof for the first claim (the second one is similar). Let $(\alpha f, g) \in Cl(K)^i$. By induction on i.

- $i = 0$. Then $(\alpha f, g) \in K$ and we can immediately apply the rule 3 or 5 of Definition 5 (remember that α can be ε).
- **Induction step.** Let $(\alpha f, g) \in Cl(K)^{i+1}$. There are three possibilities (cf. Definition 5).

I. There is r such that $(\alpha f, r) \in Cl(K)^i$, $(r, g) \in K$. By induction hypothesis we know $(\alpha\beta h, r) \in Cl(K)$, hence $(\alpha\beta h, g) \in Cl(K)$ due to the rule 3 of Definition 5.

II. $\alpha = Y\gamma$ and there is r such that $(Yr, g) \in K$, $(\gamma f, r) \in Cl(K)^i$. By induction hypothesis we have $(\gamma\beta h, r) \in Cl(K)$, and hence also $(Y\gamma\beta h, r) \in Cl(K)$ by the rule 5 of Definition 5.

III. $\alpha = \gamma\delta$ where $(\gamma, g) \in Cl(K)^i$ and γ contains an unnormed constant. Then $(\gamma\delta\beta h, g) \in Cl(K)$ by the last rule of Definition 5.

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

Theorem 8 *For all* α, f, g *we have* $\alpha \approx g$ *iff* $(\alpha, g) \in Cl(\mathcal{B})$, *and* $\alpha f \approx g$ *iff* $(\alpha f, g) \in Cl(\mathcal{B})$.

PROOF. The 'if' part is obvious in both cases, as $\mathcal{B}$ contains only weakly bisimilar pairs and all the rules of Definition 5 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 $(\varepsilon, g) \in Cl(\mathcal{B})$.
- $\alpha = Y\beta$. If Y is unnormed, then $Y \approx g$ and $(Y, g) \in \mathcal{B}$. By the rule 6 of Definition 5 we obtain $(Y\beta, g) \in Cl(\mathcal{B})$. If Y is normed, then $Y\beta \stackrel{w}{\rightarrow} \beta$ for some $w \in Act^*$ and g must be able to match the sequence w by some $g \stackrel{w}{\Rightarrow} g'$ such that $\beta \approx g'$. By substitution we now obtain that $Yg' \approx g$. Clearly $(Yg', g) \in \mathcal{B}$, and $(\beta, g') \in Cl(\mathcal{B})$ by induction hypothesis. Hence $(\alpha, g) \in Cl(\mathcal{B})$ due to the rule 4 of Definition 5.

The next definition formalizes one step of the 'refinement procedure' which is applied to $\mathcal{G}$ to compute $\mathcal{B}$. The intuition is that we start with $\mathcal{G}$ as an

approximation to $\mathcal{B}$. In each refinement step some pairs are deleted from the current approximation. If in a refinement step no pairs are deleted any more then we have found $\mathcal{B}$. The next definition specifies the condition on which a given pair is *not* deleted in a refinement step from the currently computed approximation of $\mathcal{B}$.

Definition 9 *Let K be a well-formed relation. We say that a pair (X, g) of K* expands in K *iff the following two conditions hold:*

- *for each $X \stackrel{a}{\rightarrow} \alpha$ there is some $g \stackrel{a}{\Rightarrow} g'$ such that $(\alpha, g') \in Cl(K)$*
- *for each $g \stackrel{a}{\rightarrow} g'$ there is some $X \stackrel{a}{\Rightarrow} \alpha$ such that $(\alpha, g') \in Cl(K)$*

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

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

Lemma 10 *Let K be a well-formed relation such that $Exp(K) = K$. 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 '$\stackrel{a}{\rightarrow}$' move of one component there is a '$\stackrel{a}{\Rightarrow}$' move of the other component such that 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 such that $(\alpha f, h) \in Cl(K)^i$, $(h, g) \in K$.

Let $\alpha f \stackrel{a}{\rightarrow} \gamma f$ (note that α can be empty; in this case we have to consider moves of the form $f \stackrel{a}{\rightarrow} f'$. 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 $h \stackrel{a}{\Rightarrow} h'$ such that $(\gamma f, h') \in Cl(K)$. 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 7. Hence g can use the move $g \stackrel{\tau}{\Rightarrow} g$.
2) $a \neq \tau$ or $h \neq h'$. Then there is a transition $h \stackrel{a}{\rightarrow} h'$ (see the beginning of this section) and as $(h, g) \in K$, by induction hypothesis we know that there is some $g \stackrel{a}{\Rightarrow} g'$ such that $(h', g') \in Cl(K)$. Hence, $(\gamma f, g') \in Cl(K)$ due to Lemma 7.

Now let $g \stackrel{a}{\rightarrow} g'$. As $(h, g) \in K$, there is $h \stackrel{a}{\Rightarrow} h'$ such that $(h', g') \in$

$Cl(K)$. We distinguish two possibilities again:
1) $a = \tau$ and $h' = h$. Then αf can use the move $\alpha f \stackrel{\tau}{\Rightarrow} \alpha f$; 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 $h \stackrel{a}{\rightarrow} h'$ and as $(\alpha f, h) \in Cl(K)^i$, there is $\alpha f \stackrel{a}{\Rightarrow} \gamma f$ (or $\alpha f \stackrel{a}{\Rightarrow} f'$; it is handled in the same way) such that $(\gamma f, h') \in Cl(K)$. Hence also $(\gamma f, g') \in Cl(K)$ by Lemma 7.

II. $\alpha = Y\beta$ and there is h such that $(Yh, g) \in K$, $(\beta f, h) \in Cl(K)^i$.

Let $Y\beta f \stackrel{a}{\rightarrow} \gamma\beta f$. As $(Yh, g) \in K$, we can use induction hypothesis and conclude that there is $g \stackrel{a}{\Rightarrow} g'$ such that $(\gamma h, g') \in Cl(K)$. As $(\beta f, h) \in Cl(K)$, we obtain $(\gamma\beta f, g') \in Cl(K)$ by Lemma 7.

Let $g \stackrel{a}{\rightarrow} g'$. As $(Yh, g) \in K$, by induction hypothesis we know that Yh can match the move $g \stackrel{a}{\rightarrow} g'$; there are two possibilities:
1) $Yh \stackrel{a}{\Rightarrow} \gamma h$ such that $(\gamma h, g') \in Cl(K)$. Then also $Y\beta f \stackrel{a}{\Rightarrow} \gamma\beta f$. As $(\beta f, h) \in Cl(K)$, we immediately have $(\gamma\beta f, g') \in Cl(K)$ as required.
2) $Yh \stackrel{a}{\Rightarrow} h'$ such that $(h', g') \in Cl(K)$. The transition $Yh \stackrel{a}{\Rightarrow} h'$ can be 'decomposed' into $Yh \stackrel{x}{\Rightarrow} h$, $h \stackrel{y}{\Rightarrow} h'$ where $x = a \wedge y = \tau$ or $x = \tau \wedge y = a$. If $y = \tau$ and $h' = h$, we are done immediately because then $Y\beta \stackrel{a}{\Rightarrow} \beta$ 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 $h \stackrel{y}{\rightarrow} h'$. As $(\beta f, h) \in Cl(K)^i$, due to induction hypothesis we know that there is some $\beta f \stackrel{y}{\Rightarrow} \gamma f$ (or $\beta f \stackrel{y}{\Rightarrow} f'$; this is handled in the same way) with $(\gamma f, h') \in Cl(K)$. Clearly $Y\beta f \stackrel{a}{\Rightarrow} \gamma f$. 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 $\alpha \stackrel{a}{\rightarrow} \alpha'$. Then $\alpha' = \delta\gamma$ and $\beta \stackrel{a}{\rightarrow} \delta$. As $(\beta, g) \in Cl(K)^i$, there is $g \stackrel{a}{\Rightarrow} g'$ such that $(\delta, g') \in Cl(K)$ due to the induction hypothesis. Clearly δ contains an unnormed constant, hence $(\delta\gamma, g') \in Cl(K)$ by the last rule of Definition 5.

Let $g \stackrel{a}{\rightarrow} g'$. As $(\beta, g) \in Cl(K)^i$, there is $\beta \stackrel{a}{\Rightarrow} \delta$ such that $(\delta, g') \in Cl(K)$ and δ contains an unnormed constant. Hence $\alpha \stackrel{a}{\Rightarrow} \delta\gamma$ and $(\delta\gamma, g') \in Cl(K)$ due to the last rule of Definition 5.

The notion of expansion allows to approximate $\mathcal{B}$ in the following way: $\mathcal{B}^0 = \mathcal{G}$, $\mathcal{B}^{i+1} = Exp(\mathcal{B}^i)$.

Theorem 11 *There is a $j \in \mathbb{N}$, bounded by $\mathcal{O}(n\,m^2)$, such that $\mathcal{B}^j = \mathcal{B}^{j+1}$. Moreover, $\mathcal{B}^j = \mathcal{B}$.*

PROOF. *Exp* (viewed as a function on the complete lattice of well-formed relations) is monotonic, hence the greatest fixed-point exists and must be reached after $\mathcal{O}(n\,m^2)$ steps, as the size of $\mathcal{G}$ is $\mathcal{O}(n\,m^2)$. We prove that $\mathcal{B}^j = \mathcal{B}$. '$\supseteq$:' First, let us realize that $\mathcal{B} = Exp(\mathcal{B})$ (it follows immediately from Definition 4, Definition 9, and Theorem 8). The inclusion $\mathcal{B} \subseteq \mathcal{B}^j$ can be proved

by a simple inductive argument; clearly $\mathcal{B} \subseteq \mathcal{B}^0$, and if $\mathcal{B} \subseteq \mathcal{B}^i$, we also have $\mathcal{B} \subseteq \mathcal{B}^{i+1}$ by definition of the expansion and the fact $\mathcal{B} = \mathit{Exp}(\mathcal{B})$.

'$\subseteq$:' As $\mathit{Exp}(\mathcal{B}^j) = \mathcal{B}^j$, we know that $\mathit{Cl}(\mathcal{B}^j)$ is a weak bisimulation due to Lemma 10. Thus, processes of every pair in $\mathcal{B}^j$ are weakly bisimilar.

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 $\mathit{Exp}(K)$ 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 12 *For all $X \in \mathit{Const}(\Delta)$, $a \in \mathit{Act}(\Delta)$ there is a finite-state automaton $\mathcal{A}_{(X,a)}$ of size $\mathcal{O}(n^2)$ constructible in $\mathcal{O}(n^2)$ time such that $L(\mathcal{A}_{(X,a)}) = \{\alpha \mid X \stackrel{a}{\Rightarrow} \alpha\}$*

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., [19]). 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 $Y \in \mathit{Const}(\Delta)$ such that $Y \stackrel{\tau}{\Rightarrow} \varepsilon$ and $Y \stackrel{a}{\Rightarrow} \varepsilon$, respectively. Let $G_{(X,a)} = (N, \Sigma, \delta, S)$ where

- $N = \{Y^a, Y^\tau \mid Y \in \mathit{Const}(\Delta)\} \cup \{S\}$. Intuitively, the index indicates whether the action 'a' has already been emitted.
- $\Sigma = \mathit{Const}(\Delta)$
- δ is defined as follows:
  - We add the production $S \to X^a$ to δ, and if $X \stackrel{a}{\Rightarrow} \varepsilon$ then we also add the production $S \to \varepsilon$.
  - For every transition $Y \stackrel{a}{\to} Z_1 \cdots Z_k$ of Δ and every i such that $1 \leq i \leq k$ we test whether $Z_j \stackrel{\tau}{\Rightarrow} \varepsilon$ for every $0 \leq j < i$. If this is the case, we add to δ the productions
    $$Y^a \to Z_i Z_{i+1} \cdots Z_k \text{ and } Y^a \to Z_i^\tau Z_{i+1} \cdots Z_k$$
  - For every transition $Y \stackrel{\tau}{\to} Z_1 \cdots Z_k$ of Δ and every i such that $1 \leq i \leq k$ we do the following:
    We test whether $Z_j \stackrel{\tau}{\Rightarrow} \varepsilon$ for every $0 \leq j < i$. If this is the case, we add to δ the productions
    $$Y^a \to Z_i^a Z_{i+1} \cdots Z_k,\ Y^\tau \to Z_i^\tau Z_{i+1} \cdots Z_k \text{ and } Y^\tau \to Z_i Z_{i+1} \cdots Z_k$$
    We test whether there is a $t < i$ such that $Z_t \stackrel{a}{\Rightarrow} \varepsilon$ and $Z_j \stackrel{\tau}{\Rightarrow} \varepsilon$ for every $0 \leq j < i$, $j \neq t$. If this is the case, we add to δ the productions
    $$Y^a \to Z_i^\tau Z_{i+1} \cdots Z_k \text{ and } Y^a \to Z_i Z_{i+1} \cdots Z_k$$

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 13 *Let K be a well-formed 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 6 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 12 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' such that $f \xrightarrow{a} f'$, and at most m states g' with $g \stackrel{a}{\Rightarrow} g'$, hence we need to check only $\mathcal{O}(m^2)$ pairs to verify the first (and consequently also the second) condition of Definition 9. 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 already been 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 $g \stackrel{a}{\Rightarrow} h$ such that $(\alpha, h) \in Cl(K)$. 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 9, we need to find out whether for each $g \xrightarrow{a} h$ there is some $X \stackrel{a}{\Rightarrow} \alpha$ such that $(\alpha, h) \in Cl(K)$. 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 $\mathcal{O}(n^3\, m^3)$ time suffices.

Pairs of the form $(Y f, g)$ are handled in a similar way; the first condition of Definition 9 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' $\stackrel{a}{\Rightarrow}$ move of

$Y f$ 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.
- $Y \stackrel{a}{\Rightarrow} \varepsilon$ and there is some $f \stackrel{\tau}{\Rightarrow} f'$ such that $(f', g') \in Cl(K)$.
- $Y \stackrel{\tau}{\Rightarrow} \varepsilon$ and there is some $f \stackrel{a}{\Rightarrow} f'$ such that $(f', g') \in Cl(K)$.

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 11). This gives us the following main theorem:

Theorem 14 *Weak bisimilarity is decidable between BPA and finite-state processes in $\mathcal{O}(n^5\, m^7)$ time.*

4 Termination-Sensitive Bisimilarity

As we already mentioned in the previous section, weak bisimilarity is not a congruence w.r.t. sequential composition. This is a major drawback, as any equivalence which is to be considered as 'behavioral' 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 respects some of the main features of sequencing which are 'overlooked' by weak bisimilarity; consequently, it is a congruence w.r.t. sequential composition. We also show that termination-sensitive bisimilarity is decidable between BPA and finite-state processes in polynomial time by adapting the method of the previous section. It should be noted right at the beginning that we do *not* aim to design any new 'fundamental' notion of the theory of sequential processes (that is why the properties of termination-sensitive bisimilarity are not studied in detail). We just want to demonstrate that our method is applicable to a larger class of bisimulation-like equivalences and the relation of termination-sensitive bisimilarity provides a (hopefully) convincing evidence that some of them might be interesting and useful.

In our opinion, any 'reasonable' model of sequential behaviors should be able to express (and distinguish) the following 'basic phenomena' 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* (cycling).

The difference between successful and unsuccessful termination is certainly significant. The need to distinguish between termination and cycling has also been recognized in practice; major examples come, e.g., from the theory of operating systems.

BPA processes are a very natural model of recursive sequential behaviors. Successful termination is modeled by reaching 'ε'. There is also a 'hidden' syntactical tool to model deadlock—note that by the definition of BPA systems there can be an $X \in Const(\Delta)$ such that Δ does not contain any rule of the form $X \stackrel{a}{\rightarrow} \alpha$ (let us call such constants *undefined*). A state $X\beta$ models the situation when the executed process reaches a deadlock—there is no transition (no computational step) from $X\beta$, the process is 'stuck'. It is easy to see that we can safely assume that Δ contains at most *one* undefined constant (the other ones can be simply renamed to X), which is denoted δ by convention [2]. Note that δ is *unnormed* by definition. States of the form $\delta\alpha$ are called *deadlocked*.

In the case of finite-state systems, we can distinguish between successful and unsuccessful termination in a similar way. Deadlock is modeled by a distinguished undefined constant δ, and the other undefined constants model successful termination.

Note that $\delta \approx \varepsilon$ by definition of weak bisimilarity. As 'ε' represents a successful termination, this is definitely not what we want. Before we define the promised relation of termination-sensitive bisimilarity, we need to clarify what is meant by cycling; intuitively, it is the situation when a process enters an infinite internal loop. In other words, it can do 'τ' forever without a possibility to do anything else or to terminate (either successfully or unsuccessfully).

Definition 15 *The set of* initial actions *of a process E, denoted $I(E)$, is defined by $I(E) = \{a \in Act \mid E \stackrel{a}{\rightarrow} F \text{ for some } F\}$. A process E is* cycling *iff every state F which is reachable from E satisfies $I(F) = \{\tau\}$.*

Note that it is easily decidable in quadratic time whether a given BPA process is cycling; in the case of finite-state systems we only need linear time.

Definition 16 *We say that an expression E is* normal *iff E is not cycling, deadlocked, or successfully terminated.*

A binary relation R over process expressions is a termination-sensitive bisi-

mulation *iff whenever* $(E, F) \in R$ *then the following conditions hold:*

- *if one of the expressions* E, F *is cycling then the other is also cycling;*
- *if one of the expressions* E, F *is deadlocked then the other is either normal or it is also deadlocked;*
- *if one of the expressions* E, F *is successfully terminated then the other is either normal or it is also successfully terminated;*
- *if* $E \stackrel{a}{\rightarrow} E'$ *then there is* $F \stackrel{a}{\Rightarrow} F'$ *such that* $(E', F') \in R$*;*
- *if* $F \stackrel{a}{\rightarrow} F'$ *then there is* $E \stackrel{a}{\Rightarrow} E'$ *such that* $(E', F') \in R$*.*

Processes E, F *are* termination-sensitive bisimilar*, written* $E \simeq F$*, iff there is a termination-sensitive bisimulation relating them.*

Termination-sensitive bisimilarity seems to be a natural refinement of weak bisimilarity which better captures an intuitive understanding of 'sameness' of sequential processes. It distinguishes among the phenomena mentioned at the beginning of this section, but it still allows to ignore internal computational steps to a large extent. For example, a deadlocked process is still equivalent to a process which is not deadlocked yet but which *necessarily* deadlocks after a finite number of τ transitions (this example also explains why the first three conditions of Definition 16 are stated so carefully).

The family of $\simeq_i$ approximations is defined in the same way as in case of weak bisimilarity; the only difference is that $\simeq_0$ relates exactly those processes which satisfy the first three conditions of Definition 16. The following theorem follows immediately from this definition.

Theorem 17 *Termination-sensitive bisimilarity is a congruence w.r.t. sequential composition.*

The technique which has been used in the previous section also works for termination-sensitive bisimilarity.

Theorem 18 *Termination-sensitive bisimilarity is decidable between BPA and finite-state processes in* $\mathcal{O}(n^5\, m^7)$ *time.*

PROOF. First, all assumptions about Δ and Γ which were mentioned at the beginning of Section 3 are also safe w.r.t. termination-sensitive bisimilarity; note that it would not be true if we also assumed the existence of a τ-loop $f \stackrel{\tau}{\rightarrow} f$ for every $f \in \mathit{Const}(\Gamma)$. Now we see why the assumptions about Γ are formulated so carefully. The only thing which has to be modified is the notion of well-formed relation; it is defined in the same way, but in addition we require that processes of every pair which is contained in a well-formed relation K are related by $\simeq_0$. It can be easily shown that processes of pairs contained in $\mathit{Cl}(K)$ are then also related by $\simeq_0$. In other words, we do not

have to take care about the first two requirements of Definition 16 in our constructions anymore; everything works without a single change.

The previous proof indicates that the 'method' of Section 3 can be adapted to other bisimulation-like equivalences. See the final section for further comments.

5 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.

Example 19 *Consider the unnormed BPP that is defined by the following rules.*

$$X_i \stackrel{a_{i+1}}{\longrightarrow} X_i \| Y_i, \quad Y_i \stackrel{a_i}{\longrightarrow} \varepsilon \ \textit{for}\ 1 \leq i \leq n-1$$
$$X_n \stackrel{a_1}{\longrightarrow} X_n \| Y_n,\ Y_n \stackrel{a_n}{\longrightarrow} \varepsilon$$

Then the process $X_1 \| X_2 \| \ldots \| X_n$ is finite w.r.t. bisimilarity, but every subprocess (e.g. $X_3 \| X_4 \| X_7$ or every single constant X_i) is infinite w.r.t. bisimilarity.

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 '$\stackrel{a}{\Rightarrow}$' 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.

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 $Const(\Delta)^{\otimes}$ by finite automata and CF grammars. The problem is that elements of $Const(\Delta)^{\otimes}$ 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 $Lin(\alpha)$ as follows:

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

For example, $Lin(X\|Y\|Z) = \{XYZ, XZY, YXZ, YZX, ZXY, ZYX\}$. We also assume that each $Lin(\alpha)$ contains some (unique) element called *canonical form* of $Lin(\alpha)$. 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 $Lin(\alpha)$ be the sorted order of constants of α).

Definition 20 *A relation K is* well-formed *iff it is a subset of $\mathcal{G} = (Const(\Delta) \cup \{\varepsilon\}) \times Const(\Gamma)$. 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 21 *Let K be a well-formed relation. The* closure *of K, denoted $Cl(K)$, 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 $Cl(K)^i$ approximations is defined in the same way as in Section 3.

Lemma 22 *Let* $(\alpha, f) \in Cl(K)$, $(\beta, g) \in Cl(K)$, $f\|g \approx h$. *Then* $(\alpha\|\beta, h) \in Cl(K)$.

PROOF. Let $(\alpha, f) \in Cl(K)^i$. By induction on i.

- $i = 0$. Then $(\alpha, f) \in K$ and we can immediately apply the rule 2 or 3 of Definition 21.
- **Induction step.** Let $(\alpha, f) \in Cl(K)^{i+1}$. There are two possibilities.

  I. $\alpha = X\|\gamma$ and there are r, s such that $(X, r) \in K$, $(\gamma, s) \in Cl(K)^i$, and $r\|s \approx f$. Clearly $r\|s\|g \approx h$, hence also $s\|g \approx t$ for some t. By induction

hypothesis we have $(\gamma\|\beta, t) \in Cl(K)$. Now $(X\|\gamma\|\beta, h) \in Cl(K)$ due to the second rule of Definition 21 (note that $r\|t \approx h$).

II. $(\alpha, r) \in Cl(K)^i$ and there is some s such that $(\varepsilon, s) \in K$ and $r\|s \approx f$. As $r\|s\|g \approx h$, there is some t such that $r\|g \approx t$. By induction hypothesis we obtain $(\alpha\|\beta, t) \in Cl(K)$, and hence $(\alpha\|\beta, h) \in Cl(K)$ due to the third rule of Definition 21.

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

Theorem 23 *Let $\alpha \in Const(\Delta)^{\otimes}$, $f \in Const(\Gamma)$. We have that $\alpha \approx f$ iff $(\alpha, f) \in Cl(\mathcal{B})$.*

PROOF. The 'if' part is obvious. The 'only if' part can be proved by induction on $length(\alpha)$.

- $\alpha = \varepsilon$. Then $(\varepsilon, f) \in \mathcal{B}$.
- $\alpha = X\|\beta$. As Δ is normed and $X\|\beta \approx f$, there are $w, v \in Act^*$ such that $X\|\beta \stackrel{w}{\rightarrow} \beta$, $X\|\beta \stackrel{v}{\rightarrow} X$. The process f must be able to match the sequences w, v by entering weakly bisimilar states—there are $g, h \in Const(\Delta)$ such that $\beta \approx g$, $X \approx h$, and consequently also $f \approx g\|h$ (here we need the fact that weak bisimilarity is a congruence w.r.t. the parallel operator). Clearly $(X, h) \in \mathcal{B}$ and $(\beta, g) \in Cl(\mathcal{B})$ by induction hypothesis, hence $(X\|\beta, f) \in Cl(\mathcal{B})$ by Definition 21.

The closure of any well-formed relation can in some sense be represented by a finite-state automaton, as stated in the next theorem. For this construction we first need to compute the set $\{(f\|g, h) \mid f\|g \approx h\}$. We consider the parallel composition of the finite-state system with itself, i.e., the states of this system are of the form $f\|g$. Let our new system be the union of this system with the old system. The new system has size $\mathcal{O}(m^2)$ and its states are of the form $f\|g$ or h. Then we apply the usual cubic-time partition refinement algorithm to decide bisimilarity on the new system (see Section 2). This gives us the set $\{(f\|g, h) \mid f\|g \approx h\}$ in $\mathcal{O}(m^6)$ time.

Theorem 24 *Let K be a well-formed 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 such that 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 $Lin(\alpha)$*

PROOF. We design a regular grammar of size $\mathcal{O}(n\,m)$ such that $L(G_g)$ has the mentioned properties. Let $G_g = (N, \Sigma, \delta, S)$ where

- $N = \mathit{Const}(\Gamma) \cup \{S\}$
- $\Sigma = \mathit{Const}(\Delta)$
- δ is defined as follows:
  - for each $(X, f) \in K$ we add the rule $S \to Xf$.
  - for each $(\varepsilon, f) \in K$ we add the rule $S \to f$.
  - for all $f, r, s \in \mathit{Const}(\Gamma)$, $X \in \mathit{Const}(\Delta)$ such that $(X, r) \in K$, $f \approx r \| s$ we add the rule $s \to Xf$.
  - for all $f, r, s \in \mathit{Const}(\Gamma)$ such that $(\varepsilon, r) \in K$, $f \approx r \| s$ we add the rule $s \to f$.
  - we add the rule $g \to \varepsilon$.

The first claim follows from an observation that whenever we have $\overline{\alpha} \in \mathit{Lin}(\alpha)$ such that $\overline{\alpha} f$ is a sentence of G_g, then $(\alpha, f) \in \mathit{Cl}(K)$. This can be easily proved by induction on the length of the derivation of $\overline{\alpha} f$. For the second part, it suffices to prove that if $(\alpha, f) \in \mathit{Cl}(K)^i$, then there is $\overline{\alpha} \in \mathit{Lin}(\alpha)$ such that $\overline{\alpha} f$ is a sentence of G_g. It can be done by a straightforward induction on i.

It is important to realize that if $(\alpha, g) \in \mathit{Cl}(K)$, then $\mathcal{A}_g$ does not necessarily accept *all* elements of $\mathit{Lin}(\alpha)$. For example, if $K = \{(X, f), (Y, r), (Z, h)\}$, $\mathit{Const}(\Gamma) = \{f, g, h, r, s\}$ with $f \| r \approx s$, $s \| h \approx g$, and $f \| h \not\approx p$ for any $p \in \mathit{Const}(\Gamma)$, then $\mathcal{A}_g$ accepts the string XYZ but not the string XZY. 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 25 *Let K be a well-formed relation such that $\mathcal{B} \subseteq K$. If $\alpha \approx g$, then the automaton $\mathcal{A}_g$ of (the proof of) Theorem 24 constructed for K accepts all elements of $\mathit{Lin}(\alpha)$.*

PROOF. Let G_g be the grammar of the previous proof. First we prove that for all $s, r, f \in \mathit{Const}(\Gamma)$, $\gamma \in \mathit{Const}(\Delta)^{\otimes}$ such that $\gamma \approx r$, $s \| r \approx f$ there is a derivation $s \to^* \overline{\gamma} f$ in G_g for every $\overline{\gamma} \in \mathit{Lin}(\gamma)$. By induction on $\mathit{length}(\gamma)$.

- $\gamma = \varepsilon$. As $\varepsilon \approx r$, the pair (ε, r) belongs to $\mathcal{B}$. Hence $s \to f$ by definition of G_g.
- Let $\mathit{length}(\gamma) = i + 1$ and let $X\overline{\beta} \in \mathit{Lin}(\gamma)$. Then γ is of the form $X \| \beta$ where $\overline{\beta} \in \mathit{Lin}(\beta)$. As $X \| \beta \approx r$ and Δ is normed, there are $u, v \in \mathit{Const}(\Gamma)$ such that $X \approx u$, $\beta \approx v$, and $u \| v \approx r$. Hence we also have $s \| u \| v \approx f$, thus $s \| u \approx t$ for some $t \in \mathit{Const}(\Gamma)$. As $X \approx u$, the pair (X, u) belongs to $\mathcal{B}$.

Clearly $s \to Xt$ by definition of G_g. As $\beta \approx v$ and $v\|t \approx f$, we can use the induction hypothesis and conclude $t \to^* \overline{\beta} f$. Hence $s \to^* X\overline{\beta} f$ as required.

Now let $\alpha \approx g$. As Δ is normed, there is some $r \in \mathit{Const}(\Gamma)$ such that $\varepsilon \approx r$. Hence $(\varepsilon, r) \in \mathcal{B}$ and $S \to r$ by definition of G_g. Clearly $r\|g \approx g$ and due to the above proved property we have $r \to^* \overline{\alpha} g$ for every $\overline{\alpha} \in \mathit{Lin}(\alpha)$. As $g \to \varepsilon$ is a rule of G_g, we obtain $S \to r \to^* \overline{\alpha} g \to \overline{\alpha}$.

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

Theorem 26 *For all $X \in \mathit{Const}(\Delta)$, $a \in \mathit{Act}(\Delta)$ there is a context-free grammar $G_{(X,a)}$ in 3-GNF (Greibach normal form, i.e., with at most 2 variables at the right hand side of every production) of size $\mathcal{O}(n^4)$ constructible in $\mathcal{O}(n^4)$ time such that the following two conditions hold:*

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

PROOF. Let $G_{(X,a)} = (N, \Sigma, \delta, X^a)$ where

- $N = \{Y^a, Y^\tau \mid Y \in \mathit{Const}(\Delta)\} \cup \{S\}$
- $\Sigma = \mathit{Const}(\Delta)$
- δ is defined as follows:
  - the rule $S \to X^a$ is added to δ.
  - for each transition $Y \stackrel{a}{\to} Z_1\| \cdots \|Z_k$ of Δ we add the rule
    $$Y^a \to Z_1^\tau \cdots Z_k^\tau$$
    (if $k = 0$, we add the rule $Y^a \to \varepsilon$).
  - for each transition $Y \stackrel{\tau}{\to} Z_1\| \cdots \|Z_k$ of Δ we add the rule
    $$Y^\tau \to Z_1^\tau \cdots Z_k^\tau$$
    (if $k = 0$, we add $Y^\tau \to \varepsilon$). Moreover, if $k \geq 1$ then for each $1 \leq i \leq k$ we also add the rule
    $$Y^a \to Z_1^\tau \cdots Z_i^a \cdots Z_k^\tau$$
  - for each $Y \in \mathit{Const}(\Delta)$ we add the rule
    $$Y^\tau \to Y.$$

The fact that $G_{(X,a)}$ satisfies the above mentioned conditions follows directly from its construction. Note that the size of $G_{(X,a)}$ is $\mathcal{O}(n^2)$ at the moment. Now we transform $G_{(X,a)}$ to 3-GNF by a standard procedure of automata theory (see [19]). It can be done in $\mathcal{O}(n^4)$ time and the size of resulting grammar is $\mathcal{O}(n^4)$.

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

Definition 27 *Let K be a well-formed relation. We say that a pair $(X, f) \in K$* expands *in K iff the following two conditions hold:*

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

A pair $(\varepsilon, f) \in K$ expands in K iff $f \stackrel{a}{\to} g$ implies $a = \tau$, and for each $f \stackrel{\tau}{\to} g$ 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 28 *Let K be a well-formed 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 24 for all $g \in Const(\Gamma)$. This takes $\mathcal{O}(n\, m^2)$ time. Then we compute the grammars $G_{(X,a)}$ of Theorem 26 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 27 can be checked in $\mathcal{O}(n^3\, m^2)$ time, as there are $\mathcal{O}(n)$ transitions $X \stackrel{a}{\to} \alpha$, $\mathcal{O}(m)$ states g such that $f \stackrel{a}{\Rightarrow} g$, 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 27 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 recognizes this language. $\mathcal{P}$ has $\mathcal{O}(m)$ control states and its total size is $\mathcal{O}(n^5\, m)$. Furthermore, each rule $pX \stackrel{a}{\to} q\alpha$ 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 [19]. 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. [19]). 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 27. The next theorem says that *Exp* really does what we need.

Theorem 29 *Let K be a well-formed relation such that $Exp(K) = K$. Then $Cl(K)$ is a weak bisimulation.*

PROOF. Let $(\alpha, f) \in Cl(K)^i$. We prove that for each $\alpha \stackrel{a}{\rightarrow} \beta$ there is some $f \stackrel{a}{\Rightarrow} g$ such that $(\beta, g) \in Cl(K)$ 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 $X \stackrel{a}{\rightarrow} \beta$. By Definition 27 there is $f \stackrel{a}{\Rightarrow} g$ such that $\overline{\beta} \in L(\mathcal{A}_g)$ for some $\overline{\beta} \in Lin(\beta)$. Hence $(\beta, g) \in Cl(K)$ due to the first part of Theorem 24.

Let $f \stackrel{a}{\rightarrow} g$. By Definition 27 there is some string $w \in L(\mathcal{A}_g) \cap L(G_{(X,a)})$. Let $w \in Lin(\beta)$. We have $X \stackrel{a}{\Rightarrow} \beta$ due to the first part of Theorem 26, and $(\beta, g) \in Cl(K)$ due to Theorem 24.

(2) $\alpha = \varepsilon$

Let $f \stackrel{a}{\rightarrow} g$. Then $a = \tau$ and $\varepsilon \in L(\mathcal{A}_g)$ by Definition 27. Hence $(\varepsilon, g) \in Cl(K)$ due to Theorem 24.

- **Induction step.** Let $(\alpha, f) \in Cl(K)^{i+1}$. There are two possibilities.

I. $\alpha = X \| \gamma$ and there are r, s such that $(X, r) \in K$, $(\gamma, s) \in Cl(K)^i$, and $r \| s \approx f$.

Let $X \| \alpha \stackrel{a}{\rightarrow} \beta$. The action '$a$' can be emitted either by X or by α. We distinguish the two cases.

1) $X \| \gamma \stackrel{a}{\rightarrow} \delta \| \gamma$. As $(X, r) \in K$ and $X \stackrel{a}{\rightarrow} \delta$, there is some $r \stackrel{a}{\Rightarrow} r'$ such that $(\delta, r') \in Cl(K)$. As $r \| s \approx f$ and $r \stackrel{a}{\Rightarrow} r'$, there is some $f \stackrel{a}{\Rightarrow} g$ such that $r' \| s \approx g$. 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 22.

2) $X \| \gamma \stackrel{a}{\rightarrow} X \| \rho$. As $(\gamma, s) \in Cl(K)^i$ and $\gamma \stackrel{a}{\rightarrow} \rho$, there is $s \stackrel{a}{\Rightarrow} s'$ such that $(\rho, s') \in Cl(K)$. As $r \| s \approx f$ and $s \stackrel{a}{\Rightarrow} s'$, there is $f \stackrel{a}{\Rightarrow} g$ such that $(r \| s') \approx g$. Due to Lemma 22 we obtain $(X \| \rho, g) \in Cl(K)$.

Let $f \stackrel{a}{\rightarrow} g$. As $r \| s \approx f$, there are $r \stackrel{x}{\Rightarrow} r'$, $s \stackrel{y}{\Rightarrow} s'$ where $x = a \wedge y = \tau$ or $x = \tau \wedge y = a$ such that $r' \| s' \approx g$. As $(X, r) \in K$, $(\gamma, s) \in Cl(K)^i$, there are $X \stackrel{x}{\Rightarrow} \delta$, $\gamma \stackrel{y}{\Rightarrow} \rho$ such that (δ, r'), $(\rho, s') \in Cl(K)$. Clearly $X \| \gamma \stackrel{a}{\Rightarrow} \delta \| \rho$ and $(\delta \| \rho, g) \in Cl(K)$ due to Lemma 22.

II. $(\alpha, r) \in Cl(K)^i$ and there is some s such that $(\varepsilon, s) \in K$ 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 Section 3.

Theorem 30 *There is a $j \in \mathbb{N}$, bounded by $\mathcal{O}(n\,m)$, such that $\mathcal{B}^j = \mathcal{B}^{j+1}$. Moreover, $\mathcal{B}^j = \mathcal{B}$.*

PROOF. '$\supseteq$:' It suffices to show that $Exp(\mathcal{B}) = \mathcal{B}$. Let $(\alpha, f) \in \mathcal{B}$. Then $\alpha \approx f$, and $\alpha = X$ for some $X \in Const(\Delta)$ or $\alpha = \varepsilon$. We show that (X, f) expands in $\mathcal{B}$ (a proof for the pair (ε, f) is similar).

Let $X \stackrel{a}{\rightarrow} \beta$. As $X \approx f$, there is $f \stackrel{a}{\Rightarrow} g$ such that $\beta \approx g$. Let $\overline{\beta}$ be the canonical form of $Lin(\beta)$. Due to Lemma 25 we have $\overline{\beta} \in L(\mathcal{A}_g)$.

Let $f \stackrel{a}{\rightarrow} g$. As $X \approx f$, there is $X \stackrel{a}{\Rightarrow} \beta$ such that $\beta \approx g$. Due to Theorem 26 there is $\overline{\beta} \in Lin(\beta)$ such that $\overline{\beta} \in L(G_{(X,a)})$. Moreover, $\overline{\beta} \in L(\mathcal{A}_g)$ due to Lemma 25. Hence, $L(\mathcal{A}_g) \cap L(G_{(X,a)})$ is nonempty.

'$\subseteq$:' It follows directly from Theorem 29.

Theorem 31 *Weak bisimilarity between normed BPP and finite-state processes is decidable in $\mathcal{O}(n^{12}\, m^9)$ time.*

PROOF. By Theorem 30 the computation of the expansion of Theorem 28 (which costs $\mathcal{O}(n^{11}\, m^8)$ time) has to be done $\mathcal{O}(n\, m)$ times.

6 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 have constructed $\mathcal{B}$ (for a BPA/nBPP system Δ and a finite-state system Γ) and the automaton $\mathcal{A}_g$ of Theorem 6/Theorem 24 (for $K = \mathcal{B}$ and some $g \in Const(\Gamma)$), we can decide weak bisimilarity 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 [17,18]. 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 polyno-

mial 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 [21]. A concrete example is termination-sensitive bisimilarity of Section 4. 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 [21] 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. *Journal of the Association for Computing Machinery*, 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 *Lecture Notes in Computer Science*, 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 *Lecture Notes in Computer Science*, pages 423–433. Springer, 1995.

[5] O. Burkart and J. Esparza. More infinite results. *Electronic Notes in Theoretical Computer Science*, 5, 1997.

[6] D. Caucal. Graphes canoniques des graphes algébriques. *Informatique Théorique et Applications (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. *Decidability and Decomposition in Process Algebras*. PhD thesis, The University of Edinburgh, 1993.

[9] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for all basic parallel processes. In *Proceedings of CONCUR'93*, volume 715 of *Lecture Notes in Computer Science*, pages 143–157. Springer, 1993.

[10] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.

[11] J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. In *Proceedings of FCT'95*, volume 965 of *Lecture Notes in Computer Science*, pages 221–232. Springer, 1995.

[12] J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.

[13] J. Esparza and J. Knoop. An automata-theoretic approach to interprocedural data-flow analysis. In *Proceedings of FoSSaCS'99*, volume 1578 of *Lecture Notes in Computer Science*, pages 14–30. Springer, 1999.

[14] J.F. Groote. A short proof of the decidability of bisimulation for normed BPA processes. *Information Processing Letters*, 42:167–171, 1992.

[15] Y. Hirshfeld. Bisimulation trees and the decidability of weak bisimulations. *Electronic Notes in Theoretical Computer Science*, 5, 1996.

[16] Y. Hirshfeld and M. Jerrum. Bisimulation equivalence is decidable for normed process algebra. In *Proceedings of ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 412–421. Springer, 1999.

[17] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science*, 158:143–159, 1996.

[18] Y. Hirshfeld, 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.

[19] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[20] 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.

[21] P. Jančar, A. Kučera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. In *Proceedings of ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, pages 200–211. Springer, 1998.

[22] A. Kučera. Effective decomposability of sequential behaviours. *Theoretical Computer Science*, 242(1–2):71–89, 2000.

[23] A. Kučera. Efficient verification algorithms for one-counter processes. In *Proceedings of ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 317–328. Springer, 2000.

[24] A. Kučera. On simulation-checking with sequential systems. In *Proceedings of ASIAN 2000*, Lecture Notes in Computer Science. Springer, 2000. To Appear.

[25] A. Kučera and R. Mayr. Simulation preorder on simple process algebras. In *Proceedings of ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 503–512. Springer, 1999.

[26] R. Mayr. Weak bisimulation and model checking for basic parallel processes. In *Proceedings of FST&TCS'96*, volume 1180 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 1996.

[27] R. Mayr. Strict lower bounds for model checking BPA. *Electronic Notes in Theoretical Computer Science*, 18, 1998.

[28] R. Mayr. On the complexity of bisimulation problems for basic parallel processes. In *Proceedings of ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 329–341. Springer, 2000.

[29] R. Mayr. On the complexity of bisimulation problems for pushdown automata. In *Proceedings of IFIP TCS 2000*, volume 1872 of *Lecture Notes in Computer Science*. Springer, 2000.

[30] R. Mayr. Decidability of model checking with the temporal logic EF. *Theoretical Computer Science*, 2000, To appear.

[31] R. Mayr. Process rewrite systems. *Information and Computation*, 156(1):264–286, 2000.

[32] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[33] F. Moller. Infinite results. In *Proceedings of CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 195–216. Springer, 1996.

[34] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, 1987.

[35] D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings* 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.

[36] J.L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981.

[37] J. Srba. Complexity of weak bisimilarity and regularity for BPA and BPP. In *Proceedings of EXPRESS 2000*, *Electronic Notes in Theoretical Computer Science*. 2000.

[38] J. Stříbrná. Hardness results for weak bisimilarity of simple process algebras. *Electronic Notes in Theoretical Computer Science*, 18, 1998.

[39] I. Walukiewicz. Model checking CTL properties of pushdown systems. In *Proceedings of FST&TCS 2000*, Lecture Notes in Computer Science. Springer, 2000. To appear.