TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

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

Antonín Kučera, Richard Mayr

TUM-I9830
SFB-Bericht Nr. 342/13/98 A
Dezember 98

TUM–INFO–12-I9830-80/1.–FI

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

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

Antonín Kučera*
Faculty of Informatics
Masaryk University
Botanická 68a, 60200 Brno
Czech Republic
tony@fi.muni.cz

Richard Mayr
Institut für Informatik
Technische Universität München
Arcisstr. 21, D-80290 München
Germany
mayrri@in.tum.de

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.

Keywords: 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 issues for infinite-state systems. In our paper we concentrate on decidability of weak bisimilarity between certain infinite-state processes and finite-state ones. We prove that weak bisimilarity (and also other bisimulation-like equivalences) is decidable in polynomial time between BPA and finite-state processes, and between normed BPP and finite-state ones. BPA processes can be seen as simple sequential programs (due to the binary

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

sequential composition operator), while BPP model simple parallel systems (due to the binary parallel composition operator). A process is *normed* iff at every reachable state it can terminate via a finite sequence of computational steps.

The state of the art. BPA and BPP have been studied very intensively and many decidability and complexity results about them are known today (especially in the areas of equivalence-testing and model-checking [22, 10]). Here we briefly mention some of the results which are closely related to the subject of our paper.

The first positive result about BPA processes is due to Baeten, Bergstra, and Klop [2]. They proved that *strong bisimilarity* [23] is decidable for normed BPA processes. A much simpler proof of this has later been given in [15], and there is even a polynomial-time algorithm [12]. The decidability result has later been extended to the class of all (not necessarily normed) BPA processes in [8], but the best known algorithm is doubly exponential [5]. Decidability of strong bisimilarity for BPP processes has been established in [7], but the algorithm has non-elementary complexity. However, there is also a polynomial-time algorithm for the subclass of normed BPP [13]. Strong bisimilarity between normed BPA and normed BPP processes is also decidable [6]. A more general result [17] says that strong bisimilarity remains decidable if we consider parallel compositions of normed BPA and normed BPP processes.

As for weak bisimilarity, much less is known. Semidecidability of weak bisimilarity for BPP processes is due to [9]. In [11] 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 in addition at least one of those steps must be 'visible', i.e. non-internal). The weak bisimilarity problem for general BPA and BPP is open; those problems might be decidable, but they are surely intractable (assuming $\mathcal{P} \neq \mathcal{NP}$)—in case of BPP we have $\mathcal{NP}$-hardness, and in case of BPA even *PSPACE*-hardness [24].

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 or buffers). In [19] it is shown that weak bisimilarity between BPP and finite-state processes is decidable. A more general result has recently been obtained in [16], where it is shown that many bisimulation-like equivalences (including the strong and weak ones) are decidable between PAD and finite-state processes. The

PAD class strictly subsumes not only BPA and BPP, but also PA [3] and pushdown processes. The 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 [19] and BPA [4, 20]), the algorithm is not practically usable.

Our contribution. In Section 3.1 we show that weak (and hence also strong) bisimilarity is decidable between BPA and finite-state processes in polynomial time. The proof can be divided in two parts. First we show the existence of a finite bisimulation base, which in some sense generates the greatest weak bisimulation—all pairs of bisimilar processes can be 'generated' from that base. It is interesting that we can design such a base in spite of the fact that weak bisimilarity is not a congruence w.r.t. sequential composition, and hence the possibility to derive 'new' pieces of information from 'old' ones is rather limited (actually, all what we need is the fact that weak bisimilarity is a *left* congruence). Then we take a sufficiently large relation $\mathcal{G}$ which surely subsumes the base and 'clean' it. The size of $\mathcal{G}$ is cubic in the size of problem instance, and each step of the cleaning procedure possibly deletes some of the elements of $\mathcal{G}$. If nothing is deleted, we have found the base. To be able to perform this cleaning procedure, we have to overcome the fundamental problem that the set of states which are reachable from a given BPA state in one '$\stackrel{a}{\Rightarrow}$' move (see Section 2) is infinite. We employ a 'symbolic' technique to represent those infinite sets, taking advantage of the fact that they have a simple (regular) structure which can be encoded by finite-state automata. Moreover, we can also encode the pairs of processes which can be generated from the currently computed approximation of the base by means of finite-state automata. This allows to compute the base in polynomial time, and it is actually all what we need to establish our result.

The fact that weak bisimilarity is not a congruence w.r.t. sequential composition is rather unpleasant; each equivalence called 'behavioral' should have this property. In Section 3.1.1 we propose a natural refinement of weak bisimilarity (called *termination-sensitive bisimilarity*) which preserves the nice properties of weak bisimilarity, but it also takes into account some of the main features of sequencing which are not reflected by weak bisimilarity (e.g. the distinction between termination and livelock). Consequently, termination-sensitive bisimilarity *is* a congruence w.r.t. sequential composition. Moreover, it is also decidable between BPA and finite-state processes in polynomial time (we use the same method as in case of weak bisimilarity).

In Section 3.2 we show that weak bisimilarity between normed BPP and finite-state processes is also decidable in polynomial time. The basic scheme

of our proof is similar as in case of BPA. The bisimulation base is simpler (due to the normedness assumption), but there are some 'new' complications caused by commutativity of the parallel operator. Moreover, the set of states which are reachable from a given BPP state in one '$\stackrel{a}{\Rightarrow}$' move is no longer regular; however, it can be in some sense represented by a context-free grammar.

In the final section we discuss further applicability of our results, and we give an informal comparison with the techniques which have been used for strong bisimilarity in [12, 13].

2 Definitions

2.1 Process Rewrite Systems

Let $Act = \{a, b, c, \ldots\}$ be a countably infinite set of *actions*. Let $Const = \{X, Y, Z, \ldots\}$ be a countably infinite set of *process constants* such that $Act \cap Const = \emptyset$. The class of *process expressions*, denoted $\mathcal{E}$, is defined by the following abstract syntax equation:

$$E \ ::= \ \epsilon \mid X \mid E \| E \mid E.E$$

Here X ranges over $Const$ and ϵ is a special constant that denotes the empty expression. Intuitively, the '.' operator corresponds to a sequential composition, while the '$\|$' operator models a simple form of parallelism.

In the rest of this paper we do not distinguish between expressions related by *structural congruence* which is the smallest congruence relation over process expressions such that the following laws hold:

- associativity for '.' and '$\|$'
- commutativity for '$\|$'
- 'ϵ' as a unit for '.' and '$\|$'.

A *process rewrite system* [18] is specified by a finite set Δ of *rules* which are of the form $E \stackrel{a}{\rightarrow} F$, where $E, F \in \mathcal{E}$ and $a \in Act$. We use $Const(\Delta)$ and $Act(\Delta)$ to denote the sets of process constants and actions which are used in rules of Δ, respectively (note that $Const(\Delta)$ and $Act(\Delta)$ are finite).

Each process rewrite system Δ determines 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):

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

We extend the notation $E \stackrel{a}{\rightarrow} F$ to elements of Act^* in an obvious way. Moreover, we say that F is *reachable* from E if $E \stackrel{w}{\rightarrow} F$ for some $w \in Act^*$. The set of *initial actions* of E, denoted $I(E)$, is defined by $I(E) = \{a \in Act \mid E \stackrel{a}{\rightarrow} F \text{ for some } F\}$.

The classes of finite-state, BPA, and BPP systems are subclasses of process rewrite systems obtained by certain restrictions on the form of the expressions which can appear on the left-hand and the right-hand side of rules. To specify those restrictions, we first define the classes of *sequential* and *parallel* expressions, composed of all process expressions which do not contain the '||' and the '.' operator, respectively. 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.

In the rest of this paper we consider *processes* as (being associated with) states in transition systems generated by process rewrite systems. A constant $X \in Const(\Delta)$ is *normed* iff $X \stackrel{w}{\rightarrow} \epsilon$ for some $w \in Act^*$. A process is normed, iff all constants of its underlying system Δ are normed.

2.2 Weak Bisimilarity

The semantical equivalence we are interested in here is *weak bisimilarity* [21]. This relation distinguishes between 'observable' and 'internal' moves (computational steps); the internal moves are modeled by a special action which is denoted 'τ' by convention. In the following we assume that all process expressions are built over $Const(\Delta)$ where Δ is some fixed process rewrite system.

Definition 2.1. The *extended transition relation* '$\stackrel{a}{\Rightarrow}$' is defined by $E \stackrel{a}{\Rightarrow} F$ iff $E \stackrel{\tau^*}{\rightarrow} E' \stackrel{a}{\rightarrow} E'' \stackrel{\tau^*}{\rightarrow} F$ for some E', E''. Moreover, we also have $E \stackrel{\tau}{\Rightarrow} E$ for every state E. A binary relation R over process expressions is a *weak bisimulation* iff whenever $(E, F) \in R$ then for every $a \in Act$

- if $E \stackrel{a}{\rightarrow} E'$ then there is $F \stackrel{a}{\Rightarrow} F'$ s.t. $(E', F') \in R$
- if $F \stackrel{a}{\rightarrow} F'$ then there is $E \stackrel{a}{\Rightarrow} E'$ s.t. $(E', F') \in R$

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

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 $F \stackrel{a}{\Rightarrow} F'$ s.t. $E' \approx_i F'$
  - if $F \stackrel{a}{\rightarrow} F'$ then there is $E \stackrel{a}{\Rightarrow} E'$ s.t. $E' \approx_i F'$

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 [16]). However, we do not need this for our purposes.

Let Γ be a finite-state system with n states, $f, g \in \mathit{Const}(\Gamma)$. It is not hard to show that the problem whether $f \approx g$ is decidable in $\mathcal{O}(n^3)$ time. We use this fact in Section 3.2.

Sometimes we also consider weak bisimilarity (and its approximations) 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 Weak Bisimilarity with Infinite-State Processes

3.1 BPA Processes

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 s.t. $\mathit{Const}(\Delta) \cap \mathit{Const}(\Gamma) = \emptyset$. To simplify our considerations, we assume that E is an element of $\mathit{Const}(\Delta)$; if it is not the case, i.e. if E is of the form $Y\alpha$ where $\alpha \in \mathit{Const}(\Delta)^+$, we take a new (fresh) constant X and for every $Y \stackrel{a}{\rightarrow} \beta \in \Delta$ we add to Δ the rule $X \stackrel{a}{\rightarrow} \alpha\beta$. Obviously $X \approx E$, because the transition systems generated by X and E are even isomorphic.

As for Γ, we also need one special assumption; for all $f, g \in \mathit{Const}(\Gamma)$, $a \in \mathit{Act}$ s.t. $f \neq g$ or $a \neq \tau$ we assume that whenever $f \stackrel{a}{\Rightarrow} g$, then $f \stackrel{a}{\rightarrow} g \in \Gamma$. If those '$\stackrel{a}{\rightarrow}$' transitions are missing in Γ, we can add them safely. This procedure does not change Γ 'significantly' in the sense that each state of Γ remains weakly

bisimilar to itself after the modification. The number of transitions in Γ can of course increase, but it does not influence our complexity estimations, since we always consider the worst case when for all $f, g \in \mathit{Const}(\Gamma), a \in \mathit{Act}$ there is a rule $f \stackrel{a}{\to} g$ in Γ. The reason why we do not want to add new transitions of the form $f \stackrel{\tau}{\to} f$ will become clear in Section 3.1.1.

In this section, we use upper-case letters $X, Y, \ldots$ to denote elements of $\mathit{Const}(\Delta)$, and lower-case letters $f, g, \ldots$ to denote elements of $\mathit{Const}(\Gamma)$. Greek letters $\alpha, \beta, \ldots$ are used to denote the elements of $\mathit{Const}(\Delta)^*$. 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 $\mathit{Const}(\Delta)$ can be divided into two disjoint subsets of *normed* and *unnormed* constants (remember that $X \in \mathit{Const}(\Delta)$ is normed iff $X \stackrel{w}{\to} \epsilon$ for some $w \in \mathit{Act}^*$). The set of all normed constants of Δ is denoted by $\mathit{Normed}(\Delta)$. In our constructions we also use processes of the form αf; they should be seen as BPA processes with the underlying system $\Delta \cup \Gamma$.

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 of size $\mathcal{O}(n\, m^2)$ (called *bisimulation base*) such that each pair of weakly bisimilar processes can be generated from that base. It is interesting that we can design such a relation in spite of the fact that weak bisimilarity is *not* a congruence w.r.t. sequential composition for unnormed processes—to see this, it suffices to define

$$X \stackrel{\tau}{\to} X,\ Y \stackrel{\tau}{\to} \epsilon,\ Z \stackrel{a}{\to} Z$$

Now $X \approx Y$, but $XZ \not\approx YZ$. Fortunately, weak bisimilarity *is* a left-congruence; whenever $\beta \approx \gamma$, we also have that $\alpha\beta \approx \alpha\gamma$. Another (trivial) algebraic law says that whenever X is unnormed, it holds that $\alpha X \beta \approx \alpha X$. As we shall see, these two properties of weak bisimilarity suffice for our purposes. Then we show that the bisimulation base can be computed in polynomial time. To do that, we have to overcome the fundamental problem that the set of states which are reachable from a given BPA process in one '$\stackrel{a}{\Rightarrow}$' step is generally infinite. We employ a 'symbolic' technique to represent such infinite sets, taking advantage of the fact that they have a simple (regular) structure which can be encoded by finite-state automata.

Definition 3.1. A relation K is *fundamental* iff it is a subset of

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

The greatest fundamental relation is denoted by $\mathcal{G}$.

Note that the size of any fundamental relation is $\mathcal{O}(n\,m^2)$. One of the fundamental relations is of special importance:

Definition 3.2. The *bisimulation base* for Δ and Γ, denoted $\mathcal{B}$, is defined as follows:

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

As weak bisimilarity is a left congruence w.r.t. sequential composition, we can 'derive' from $\mathcal{B}$ new pairs of weakly bisimilar processes by substitution. This derivation procedure can be defined for any fundamental relation as follows:

Definition 3.3. Let K be a fundamental 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 $\beta \in \mathit{Const}(\Delta)^*$ and $h \in \mathit{Const}(\Gamma)$.

Note that $Cl(K)$ contains elements of just two forms – (α, g) and $(\alpha f, g)$. The set $Cl(K)$ can be approximated as follows:

- $Cl(K)^0 = K$
- $Cl(K)^{i+1}$ consists of those pairs which are either contained in $Cl(K)^i$ or can be derived from pairs of $Cl(K)^i$ by an immediate application of one of the rules 2–6 of Definition 3.3.

Clearly $Cl(K) = \bigcup_{i=0}^{\infty} Cl(K)^i$. We use the $Cl(K)^i$ family in some inductive proofs.

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

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 $(\epsilon, h) \in K$ we add the rule $\overline{h} \to \epsilon$.
  - 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 $U \to XU$, $U \to X$, $U \to f$.

The proof that G_g indeed generates the mentioned language is routine. Now we translate G_g to $\mathcal{A}_g$ (see e.g. [14]). Note that the size of $\mathcal{A}_g$ is 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 $Cl(K)$ for any fundamental relation K is easily decidable in polynomial time. Another property of $Cl(K)$ is specified in the lemma below.

Lemma 3.5. *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 the 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 3.3 (remember that α can be ϵ).
- **Induction step.** Let $(\alpha f, g) \in Cl(K)^{i+1}$. There are three possibilities (cf. Definition 3.3).

I. There is r s.t. $(\alpha f, r) \in Cl(K)^i$, $(r, g) \in K$. 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 3.3.

II. $\alpha = Y\gamma$ and there is r s.t. $(Yr, g) \in K$, $(\gamma f, r) \in Cl(K)^i$. 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 3.3.

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 3.3.

■

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 3.6. *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 3.3 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 give the first proof; the second one is similar).

- $\alpha = \epsilon$. Then $(\epsilon, g) \in \mathcal{B}$, hence $(\epsilon, g) \in Cl(\mathcal{B})$.
- $\alpha = Y\beta$. If Y is unnormed, then $Y \approx g$ and $(Y, g) \in \mathcal{B}$. By the rule 6 of Definition 3.3 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 $g \stackrel{w}{\Rightarrow} g'$ s.t. $\beta \approx g'$. 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 rule 4 of Definition 3.3. ■

Now we concentrate on the problem how to *construct* the bisimulation base. Intuitively, the base is computed by 'cleaning' $\mathcal{G}$ (the greatest fundamental relation) in a polynomial number of 'cleaning steps'. Each such step possibly deletes some pairs of $\mathcal{G}$ (if nothing is deleted, we have found $\mathcal{B}$). The next definition specifies the condition on which a given pair is *not* deleted in one cleaning step from the currently computed approximation of $\mathcal{B}$.

Definition 3.7. 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 $X \stackrel{a}{\rightarrow} \alpha$ there is some $g \stackrel{a}{\Rightarrow} g'$ s.t. $(\alpha, g') \in Cl(K)$
- for each $g \stackrel{a}{\rightarrow} g'$ there is some $X \stackrel{a}{\Rightarrow} \alpha$ s.t. $(\alpha, g') \in Cl(K)$

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 $\epsilon \stackrel{\tau}{\Rightarrow} \epsilon$). The set of all pairs of K which expand in K is denoted $Exp(K)$.

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

Lemma 3.8. *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 '$\stackrel{a}{\rightarrow}$' move of one component there is a '$\stackrel{a}{\Rightarrow}$' 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 $\alpha f \stackrel{a}{\rightarrow} \gamma f$ (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 induction hypothesis and conclude that there is $h \stackrel{a}{\Rightarrow} h'$ 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 3.5. 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 $g \stackrel{a}{\Rightarrow} g'$ s.t. $(h', g') \in Cl(K)$. Hence, $(\gamma f, g') \in Cl(K)$ due to Lemma 3.5.
  Now let $g \stackrel{a}{\rightarrow} g'$. As $(h, g) \in K$, there is $h \stackrel{a}{\Rightarrow} h'$ s.t. $(h', g') \in Cl(K)$. We distinguish two possibilities again:
  1) $a = \tau$ and $h' = h$. Then αf can use the move $\alpha f \stackrel{\tau}{\Rightarrow} \alpha f$; 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 $h \xrightarrow{a} h'$ 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) s.t. $(\gamma f, h') \in Cl(K)$. Hence also $(\gamma f, g') \in Cl(K)$ by Lemma 3.5.

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 $g \stackrel{a}{\Rightarrow} g'$ 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 3.5.

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) $Yh \stackrel{a}{\Rightarrow} \gamma h$ s.t. $(\gamma h, g') \in Cl(K)$. 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) $Yh \stackrel{a}{\Rightarrow} h'$ s.t. $(h', g') \in Cl(K)$. 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 $h \xrightarrow{y} h'$. 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 variable 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 $g \stackrel{a}{\Rightarrow} g'$ 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 3.3.

Let $g \xrightarrow{a} g'$. As $(\beta, g) \in Cl(K)^i$, there is $\beta \stackrel{a}{\Rightarrow} \delta$ s.t. $(\delta, g') \in Cl(K)$ 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 3.3. ■

The notion of expansion also allows to approximate $\mathcal{B}$ in the following way:

$$\begin{aligned} \mathcal{B}^0 &= \mathcal{G} \\ \mathcal{B}^{i+1} &= Exp(\mathcal{B}^i) \end{aligned}$$

Theorem 3.9. *There is a $j \in \mathbf{N}$, bounded by $\mathcal{O}(n\, m^2)$, such that $\mathcal{B}^j = \mathcal{B}^{j+1}$. Moreover, $\mathcal{B}^j = \mathcal{B}$.*

Proof: The *Exp* (viewed as a function on the complete lattice of fundamental 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 $\mathcal{B} = \mathit{Exp}(\mathcal{B})$ (it follows immediately from Definition 3.2, Definition 3.7, and Theorem 3.6). 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 3.8. Thus, processes of every pair in $\mathcal{B}^j$ are weakly bisimilar. ■

In other words, $\mathcal{B}$ can be obtained from $\mathcal{G}$ in $\mathcal{O}(n\,m^2)$ cleaning 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 3.10. *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 s.t. $L(\mathcal{A}_{(X,a)}) = \{\alpha \mid X \stackrel{a}{\Rightarrow} \alpha\}$*

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. [14]). Note that the size of $\mathcal{A}_{(X,a)}$ is 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 $Y \in \mathit{Const}(\Delta)$ s.t. $Y \stackrel{\tau}{\Rightarrow} \epsilon$ and $Y \stackrel{a}{\Rightarrow} \epsilon$, 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 indicate whether the action 'a' has been already emitted.
- $\Sigma = \mathit{Const}(\Delta)$
- δ is defined as follows:
  - we add the rule $S \to X^a$ to δ, and if $X \stackrel{a}{\Rightarrow} \epsilon$ then we also add the rule $S \to \epsilon$.
  - for every transition $Y \stackrel{a}{\to} Z_1.\cdots.Z_k$ of Δ and every i s.t. $1 \leq i \leq k$ we test whether $Z_j \stackrel{\tau}{\Rightarrow} \epsilon$ for every $0 \leq j < i$. 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. $1 \leq i \leq k$ we do the following:
  * we test whether $Z_j \stackrel{\tau}{\Rightarrow} \epsilon$ for every $0 \leq j < i$. If this is the case, we add to δ the rules
$$Y^a \rightarrow Z_i^a Z_{i+1} \cdots Z_k,\ Y^\tau \rightarrow Z_i^\tau Z_{i+1} \cdots Z_k,\ Y^\tau \rightarrow Z_i \cdots Z_k$$
  * we test whether there is a $t < i$ such that $Z_t \stackrel{a}{\Rightarrow} \epsilon$ and $Z_j \stackrel{\tau}{\Rightarrow} \epsilon$ 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 'cleaning procedure') 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 3.11. *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 3.4 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 3.10 for all X, a. This takes $\mathcal{O}(n^4)$ time. Furthermore, we also compute the set of all pairs of the form $(f, g), (\epsilon, g)$ 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. $f \stackrel{a}{\rightarrow} f'$, 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 3.7. Each such pair can be checked in constant time, because the set of all pairs $(f, g), (\epsilon, g)$ 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 $Y \stackrel{a}{\rightarrow} \alpha$ there is some $g \stackrel{a}{\Rightarrow} h$ 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 3.7, we need to find out whether for each $g \stackrel{a}{\rightarrow} h$ there is some $X \stackrel{a}{\Rightarrow} \alpha$ 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 $(Y\,f, g)$ are handled in a similar way; the first condition of Definition 3.7 is again no problem, as we are interested only in the '$\stackrel{a}{\rightarrow}$' moves of the left component. Now let $g \stackrel{a}{\rightarrow} g'$. 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.
- $Y \stackrel{a}{\Rightarrow} \epsilon$ and there is some $f \stackrel{\tau}{\Rightarrow} f'$ s.t. $(f', g') \in Cl(K)$.
- $Y \stackrel{\tau}{\Rightarrow} \epsilon$ and there is some $f \stackrel{a}{\Rightarrow} f'$ 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 3.9). This gives us the following main theorem:

Theorem 3.12. *Weak bisimilarity is decidable between BPA and finite-state processes in $\mathcal{O}(n^5\,m^7)$ time.*

3.1.1 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.

In our opinion, any 'reasonable' model of sequential behaviors should be able to express (and distinguish) 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).

The difference between successful and unsuccessful termination is certainly significant. The need to distinguish between termination and livelock 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 definition of BPA systems there can be $X \in \mathit{Const}(\Delta)$ s.t. 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 [3]. Note that δ is *unnormed* by definition. States of the form $\delta\alpha$ are called *deadlocked*.

In 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 $\delta \approx \epsilon$ 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 livelock; 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 3.13. A process E is *livelocked* 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 livelocked; in case of finite-state systems we only need linear time.

Definition 3.14. A binary relation R over process expressions is a *termination-sensitive bisimulation* iff whenever $(E, F) \in R$ then the following conditions hold:

- E is deadlocked iff F is deadlocked
- E is livelocked iff F is livelocked
- if $E \stackrel{a}{\rightarrow} E'$, then there is $F \stackrel{a}{\Rightarrow} F'$ s.t. $(E', F') \in R$
- if $F \stackrel{a}{\rightarrow} F'$, then there is $E \stackrel{a}{\Rightarrow} E'$ s.t. $(E', F') \in R$

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

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 two requirements of Definition 3.14. Now it is straightforward to prove the following theorem:

Theorem 3.15. *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 3.16. *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 3.1 are also safe w.r.t. termination-sensitive bisimilarity; note that it would not be true if we also assumed an existence of a τ-loop $f \stackrel{\tau}{\rightarrow} f$ for every $f \in \mathit{Const}(\Gamma)$. This explains why the assumptions about Γ are formulated so carefully. The only thing which has to be modified is the notion of fundamental relation; it is defined in the same way, but in addition we require that processes of every pair which is contained in a fundamental 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 3.14 in our constructions anymore; everything works without a single change. ■

The previous proof indicates that the 'method' of Section 3.1 can be adapted to other bisimulation-like equivalences. See the final section for further comments.

3.2 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; however, some of the arguments become more subtle because of commutativity of the parallel operator. Moreover, the set of states which are reachable from a given BPP state in one '$\stackrel{a}{\Rightarrow}$' move is no longer regular. As we shall see, it can, in some sense, be represented by a context-free 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 $E \in Const(\Delta)$. Elements of $Const(\Delta)$ are denoted by $X, Y, Z, \ldots$, elements of $Const(\Gamma)$ by $f, g, h, \ldots$ The set of all parallel expressions over $Const(\Delta)$ (i.e. the set of all potentially reachable states of Δ) is denoted by $Const(\Delta)^{\otimes}$ 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 (to see that the mentioned closure properties fail, consider the grammar $S \to abcS \mid abc$. The 'permutation' of $L(S)$ consists of exactly those words in which a, b, c appear equally many times). 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 for each α there is some (unique) element of $Lin(\alpha)$ called *canonical form* of α (it is not important how the canonical form is chosen; we need it just to make some constructions deterministic).

We also need one special assumption on Γ—for each $f \in Const(\Gamma)$ there is some $f \stackrel{w}{\to} f'$ s.t. $f' \approx \epsilon$. If $E \approx f$, then each state which is reachable from f

must have this property due to the normedness of Δ (if this condition is not satisfied, we can conclude $E \not\approx f$). Those elements of $Const(\Gamma)$ which are not reachable from f can safely be deleted. An immediate consequence of this assumption is the fact that whenever $f_1 \| \cdots \| f_k \approx g$, then there is an h s.t. $f_2 \| \cdots \| f_k \approx h$ (remember that '$\|$' is commutative). To see this, realize that $f_1 \| \cdots \| f_k \stackrel{v}{\rightarrow} f_1' \| f_2 \| \cdots \| f_k$ for some $v \in Act^*$ s.t. $f_1' \approx \epsilon$. Hence there is $g \stackrel{v}{\Rightarrow} h$ s.t. $f_1' \| f_2 \| \cdots \| f_k \approx h$. Clearly $f_1' \| f_2 \| \cdots \| f_k \approx f_2 \| \cdots \| f_k$.

Definition 3.17. A relation K is *fundamental* iff it is a subset of

$$(Const(\Delta) \cup \{\epsilon\}) \times Const(\Gamma)$$

The greatest fundamental relation is denoted by $\mathcal{G}$. The *bisimulation base* for Δ and Γ, denoted $\mathcal{B}$, is defined as follows:

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

Definition 3.18. Let K be a fundamental 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 $(\epsilon, g) \in K$, $(\beta, h) \in M$, 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 the previous section.

Lemma 3.19. *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 3.18.
- **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 $r \| s \approx f$. 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 3.18 (note that $r \| t \approx h$).

II. $(\alpha, r) \in Cl(K)^i$ and there is some s s.t. $(\epsilon, s) \in K$ and $r \| s \approx f$. As $r \| s \| g \approx h$, there is some t s.t. $r \| g \approx t$. 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 3.18. ■

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

Theorem 3.20. *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 $length(\alpha)$.

- $\alpha = \epsilon$. Then $(\epsilon, f) \in \mathcal{B}$.
- $\alpha = X \| \beta$. As Δ is normed and $X \| \beta \approx f$, there are $w, v \in Act^*$ s.t. $X \| \beta \stackrel{w}{\rightarrow} \beta$, $X \| \beta \stackrel{v}{\rightarrow} X$. The process f must be able to match the sequences w, v by entering weakly bisimilar states—there are $g, h \in Const(\Delta)$ s.t. $\beta \approx g$, $X \approx h$, 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 3.18. ■

The closure of any fundamental relation can in some sense be represented by a finite-state automaton, as stated in the next theorem. In its construction we assume that the set $\{(f \| g, h) \mid f \| g \approx h\}$ has already been computed; as weak bisimilarity is decidable in cubic time for finite-state processes (see Section 2.2) and the size of $f \| g$ is $\mathcal{O}(m^2)$, we need $\mathcal{O}(m^6)$ time to check whether $f \| g \approx h$. As there are $\mathcal{O}(m^3)$ pairs of this form, the total needed time is $\mathcal{O}(m^9)$. As we shall see, the complexity of our algorithm which decides weak bisimilarity between normed BPP and finite-state processes is $\mathcal{O}(n^{12}\, m^9)$ even if we were given this set for free. Hence we do not try to improve the $\mathcal{O}(m^9)$ bound (although it seems to be possible).

Theorem 3.21. *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 $Lin(\alpha)$*

Proof: We design a regular grammar of size $\mathcal{O}(n\,m)$ s.t. $L(G_g)$ 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 $(\epsilon, f) \in K$ we add the rule $S \to f$.
  - for all $f, r, s \in \mathit{Const}(\Gamma)$, $X \in \mathit{Const}(\Delta)$ s.t. $(X, r) \in K$, $f \approx r \| s$ we add the rule $s \to Xf$.
  - for all $f, r, s \in \mathit{Const}(\Gamma)$ s.t. $(\epsilon, r) \in K$, $f \approx r \| s$ we add the rule $s \to f$.
  - we add the rule $g \to \epsilon$.

The first claim follows from an observation that whenever we have $\overline{\alpha} \in \mathit{Lin}(\alpha)$ s.t. $\overline{\alpha} f$ 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 $\overline{\alpha} \in \mathit{Lin}(\alpha)$ s.t. $\overline{\alpha} f$ 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 accept all elements of $\mathit{Lin}(\alpha)$ (see the beginning of this section). However, there is actually no need to do that, because $\mathcal{A}_g$ has the following nice property:

Lemma 3.22. *Let K be a fundamental relation s.t. $\mathcal{B} \subseteq K$. If $\alpha \approx g$, then $\mathcal{A}_g$ accepts all elements of $\mathit{Lin}(\alpha)$.*

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

- $\gamma = \epsilon$. As $\epsilon \approx r$, the pair (ϵ, r) belongs to $\mathcal{B}$. Hence $s \to f$ by definition of G_g.

- Let $length(\gamma) = i + 1$ and let $X\overline{\beta} \in Lin(\gamma)$. Then γ is of the form $X\|\beta$ where $\overline{\beta} \in Lin(\beta)$. As $X\|\beta \approx r$ and Δ is normed, there are $u, v \in Const(\Gamma)$ s.t. $X \approx u$, $\beta \approx v$, and $u\|v \approx r$. Hence we also have $s\|u\|v \approx f$, thus $s\|u \approx t$ for some $t \in Const(\Gamma)$. 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 $r \in Const(\Gamma)$ s.t. $\epsilon \approx r$. Hence $(\epsilon, r) \in \mathcal{B}$ 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 $\overline{\alpha} \in Lin(\alpha)$. As $g \to \epsilon$ 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 $X \in Const(\Delta)$ in one '$\stackrel{a}{\Rightarrow}$' move is no longer regular. The next theorem says that this set can, in some sense, be represented by a CF grammar.

Theorem 3.23. *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 $X \stackrel{a}{\Rightarrow} \alpha$*
- *if $X \stackrel{a}{\Rightarrow} \alpha$, then $G_{(X,a)}$ generates at least one element of $Lin(\alpha)$*

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

- $N = \{Y^a, Y^\tau \mid Y \in Const(\Delta)\} \cup \{S\}$
- $\Sigma = Const(\Delta)$
- 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 $Y^a \to \epsilon$).
  - 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 $Y^\tau \to \epsilon$). 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 $Y \in Const(\Delta)$ 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 [14]). 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 3.24. 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 $X \stackrel{a}{\to} \alpha$ there is some $f \stackrel{a}{\Rightarrow} g$ s.t. $\overline{\alpha} \in L(\mathcal{A}_g)$, where $\overline{\alpha}$ is the canonical form of α.
- for each $f \stackrel{a}{\to} g$ the language $L(\mathcal{A}_g) \cap L(G_{(X,a)})$ is non-empty.

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

Theorem 3.25. *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 3.21 for all $g \in Const(\Gamma)$. This takes $\mathcal{O}(n\, m^2)$ time. Then we compute the grammars $G_{(X,a)}$ of Theorem 3.23 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 3.24 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 s.t. $f \stackrel{a}{\Rightarrow} g$, 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 α; 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 3.24 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 [14]. 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. [14]). 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. ■

It remains to show that *Exp* really does what we need.

Theorem 3.26. *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 $\alpha \stackrel{a}{\rightarrow} \beta$ there is some $f \stackrel{a}{\Rightarrow} g$ 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 $X \stackrel{a}{\rightarrow} \beta$. By Definition 3.24 there is $f \stackrel{a}{\Rightarrow} g$ 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 3.4.

     Let $f \stackrel{a}{\rightarrow} g$. By Definition 3.24 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 3.23, and $(\beta, g) \in Cl(K)$ due to Theorem 3.4.

  2. $\alpha = \epsilon$

     Let $f \stackrel{a}{\rightarrow} g$. Then $a = \tau$ and $\epsilon \in L(\mathcal{A}_g)$ by Definition 3.24. Hence $(\epsilon, g) \in Cl(K)$ due to Theorem 3.4.

- **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 $X \| \alpha \stackrel{a}{\rightarrow} \beta$. 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 $r \stackrel{a}{\Rightarrow} r'$ s.t. $(\delta, r') \in Cl(K)$. As $r \| s \approx f$ and $r \stackrel{a}{\Rightarrow} r'$, there is some $f \stackrel{a}{\Rightarrow} g$ 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 3.19.

2) $X\|\gamma \stackrel{a}{\rightarrow} X\|\rho$. As $(\gamma, s) \in Cl(K)^i$ and $\gamma \stackrel{a}{\rightarrow} \rho$, there is $s \stackrel{a}{\Rightarrow} s'$ s.t. $(\rho, s') \in Cl(K)$. As $r\|s \approx f$ and $s \stackrel{a}{\Rightarrow} s'$, there is $f \stackrel{a}{\Rightarrow} g$ s.t. $(r\|s') \approx g$. Due to Lemma 3.19 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 $x = \tau \wedge y = a$ s.t. $r'\|s' \approx g$. As $(X, r) \in K$, $(\gamma, s) \in Cl(K)^i$, there are $X \stackrel{x}{\Rightarrow} \delta$, $\gamma \stackrel{y}{\Rightarrow} \rho$ s.t. (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 3.19.

II. $(\alpha, r) \in Cl(K)^i$ and there is some s s.t. $(\epsilon, s) \in K$ 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 3.27. *There is a $j \in \mathbf{N}$, 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 $Exp(\mathcal{B}) = \mathcal{B}$ (cf. the proof of Theorem 3.9). Let $(\alpha, f) \in \mathcal{B}$. Then $\alpha \approx f$, and $\alpha = X$ for some $X \in Const(\Delta)$ or $\alpha = \epsilon$. 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 $f \stackrel{a}{\Rightarrow} g$ s.t. $\beta \approx g$. Let $\overline{\beta}$ be the canonical form of β. Due to Lemma 3.22 we have $\overline{\beta} \in L(\mathcal{A}_g)$.

Let $f \stackrel{a}{\rightarrow} g$. As $X \approx f$, there is $X \stackrel{a}{\Rightarrow} \beta$ s.t. $\beta \approx g$. Due to Theorem 3.23 there is $\overline{\beta} \in Lin(\beta)$ s.t. $\overline{\beta} \in L(G_{(X,a)})$. Moreover, $\overline{\beta} \in L(\mathcal{A}_g)$ due to Lemma 3.22. Hence, $L(\mathcal{A}_g) \cap L(G_{(X,a)})$ is nonempty.

'$\subseteq$:' It follows directly from Theorem 3.26. ■

We finish this section with the following (main) theorem:

Theorem 3.28. *Weak bisimilarity between normed BPP and finite-state processes is decidable in $\mathcal{O}(n^{12}\,m^9)$ time.*

Proof: By Theorem 3.27 the computation of the expansion of Theorem 3.25 (which costs $\mathcal{O}(n^{11}\,m^8)$ time) has to be done $\mathcal{O}(n\,m)$ times. ■

4 Conclusions

We have proved that weak bisimilarity is decidable between BPA 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. Although the degrees are rather high, it does

not necessarily mean that our algorithms are inefficient; the complexity of the worst case is not a reliable measure of practical usability. It follows from the employed techniques that the algorithms are easy to implement and might be thus evaluated in existing software tools. There are many possibilities how to improve their performance, and we argue that further development based on practical experience should bring highly satisfactory results. The algorithms might be especially useful is those situations when the intended behavior of a process is easy to specify (by a finite-state system), but its actual implementation contains components which are infinite-state (e.g. counters or buffers).

The technique of bisimulation bases has also been used in [12, 13]. However, those bases are different from ours; the way how they generate 'new' bisimilar pairs of processes is more complicated, and additional algebraic properties of strong bisimilarity are exhausted. The main difficulty of those proofs is to show that the membership to the 'closure' of the defined bases is decidable in polynomial time. Our bases are simple, and the main point of the proofs is the '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 [16]. A concrete example is termination-sensitive bisimilarity of Section 3.1.1. 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 [16] for details.

References

[1] *Proceedings of MFCS'98 Workshop on Concurrency*, FIMU-RS-98-06. Faculty of Informatics MU, Brno, 1998.

[2] 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.

[3] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[4] 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-Verlag, 1997.

[5] 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-Verlag, 1995.

[6] I. Černá, M. Křetínský, and A. Kučera. Bisimilarity is decidable in the union of normed BPA and normed BPP processes. *Electronic Notes in Theoretical Computer Science*, 5, 1997.

[7] S. Christensen, Y. Hirshfeld, and F. Moller. Decomposability, decidability and axiomatisability for bisimulation equivalence on basic parallel processes. In *Proceedings of LICS'93*. IEEE Computer Society Press, 1993.

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

[9] 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-Verlag, 1995.

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

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

[12] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial-time algorithm for deciding equivalence of normed context-free processes. In *Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 623–631. IEEE Computer Society Press, 1994.

[13] 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.

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

[15] 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.

[16] 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-Verlag, 1998.

[17] A. Kučera. On effective decomposability of sequential behaviours. *To appear in Theoretical Computer Science.*

[18] R. Mayr. Process rewrite systems. *To appear in Information and Computation.*

[19] 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-Verlag, 1996.

[20] R. Mayr. Strict lower bounds for model checking BPA. In *Proceedings of MFCS'98 Workshop on Concurrency* [1], pages 141–148.

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

[22] F. Moller. Infinite results. In *Proceedings of CONCUR'96*, volume 1119 of *LNCS*, pages 195–216. Springer-Verlag, 1996.

[23] D.M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings* 5^{th} *GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer-Verlag, 1981.

[24] J. Stříbrná. Hardness results for weak bisimilarity of simple process algebras. In *Proceedings of MFCS'98 Workshop on Concurrency* [1], pages 175–183.