FI MU

Faculty of Informatics
Masaryk University

A Generic Framework for Checking Semantic Equivalences between Pushdown Automata and Finite-State Automata

by

Antonín Kučera
Richard Mayr

FI MU Report Series **FIMU-RS-2004-01**

 April 2004

A Generic Framework for Checking Semantic Equivalences between Pushdown Automata and Finite-State Automata

Antonín Kučera* Richard Mayr†

Abstract

We propose a generic method for deciding semantic equivalences between pushdown automata and finite-state automata. The abstract part of the method is applicable to every process equivalence which is a right PDA congruence. Practical usability of the method is demonstrated on selected equivalences which are conceptual representatives of the whole spectrum. In particular, special attention is devoted to bisimulation-like equivalences (including weak, early, delay, branching, and probabilistic bisimilarity), and it is also shown how the method applies to simulation-like and trace-like equivalences. The generality does not lead to the loss of efficiency; the algorithms obtained by applying our method are essentially time-optimal and sometimes even polynomial. The list of particular results obtained by our method includes items which are first of their kind.

*Faculty of Informatics, Masaryk University, Botanická 68a, CZ-60200 Brno, Czech Republic, tony@fi.muni.cz. On leave at the Institute for Formal Methods in Computer Science, University of Stuttgart. Supported by the Alexander von Humboldt Foundation and by the Grant Agency of the Czech Republic, grant No. 201/03/1161.

†Department of Computer Science, Albert-Ludwigs-University Freiburg Georges-Koehler-Allee 51, D-79110 Freiburg, Germany. mayrri@informatik.uni-freiburg.de. Supported by Landesstiftung Baden–Württemberg, grant No. 21–655.023.

1 Introduction

The importance of *pushdown automata (PDA)* has recently been recognized also in areas different from theory of formal languages. In particular, PDA are a natural and convenient model for sequential programs with recursive procedure calls (see, e.g., [AEM04, AEY01, EK99, ES01, EKS03]). Global data of such a program is stored in the finite control, and the stack symbols correspond to activation records of individual procedures. A procedure call is thus modeled by pushing a new symbol onto the stack, and a return from the procedure is modeled by poping the symbol from the stack. Consequently, a PDA is seen as a finite description of a "computational behavior" rather than a language acceptor in this context[1]. The behavior of a given PDA Δ is formally defined by the associated transition system $\mathcal{T}_\Delta$, where the states are configurations of Δ and $p\alpha \xrightarrow{a} q\beta$ if this move is consistent with the transition function of Δ. Hence, $\mathcal{T}_\Delta$ has infinitely many states.

One of the dominating approaches to formal verification of software systems is *equivalence-checking*. The idea is to compare the behavior of a given program with its intended behavior called the *specification*. Since the two behaviors are formalized as transition systems, the comparison means proving some kind of semantic equivalence between the initial states of the two transition systems. Since such proofs cannot be completed by humans for programs of realistic size, a natural question is whether the problem is decidable and what is its complexity. This question has been considered for many computational models and a large number of results have been

[1] From the language-theoretic point of view, the definition of PDA adopted in this area corresponds to the subclass of real-time PDA. It does not mean that the concept of ε-transitions vanished—it has only been replaced by "silent" transitions with a distinguished label τ which may (but does not have to) be taken into account by a given semantic equivalence.

achieved during the last decade (see [Mol96, Esp97, JM99, Bou01, KJ02, BCMS01, Srb02a] for surveys of some subfields).

In this paper we restrict our attention to the class of programs whose behavior is definable by pushdown automata, and to the class of specifications which are definable by finite-state systems. On the other hand, we consider a large class of equivalences which subsumes the linear / branching time spectrum of [vG99, vG93].

The state of the art: Checking semantic equivalences between two pushdown automata tends to be undecidable. Special attention has been devoted to *stateless* PDA, which are often denoted BPA[2] in this context. The first result indicating that the situation is not completely hopeless is due to Baeten, Bergstra, and Klop [BBK93] who proved that strong bisimilarity is decidable for *normed* BPA (a PDA is normed if the stack can be emptied from every reachable configuration). Simpler proofs were given later in [Cau90, Gro92, HS98], and there is even a polynomial-time algorithm [HJM96]. The decidability result has been extended to all (not necessarily normed) BPA in [CHS95], and an elementary upper complexity bound is due to [BCS95]. Recently, **PSPACE**-hardness of this problem has been established in [Srb02b]. Strong bisimilarity was shown to be decidable also for normed PDA [Sti98a]. Later, Sénizergues proved that bisimilarity is decidable for all PDA processes [Sén98]. For simulation-like and trace-like equivalences, the equivalence-checking problem is undecidable even for (normed) BPA; this follows directly from Friedman's result [Fri76]. In the presence of silent moves, the situation gets even worse. Weak bisimilarity is undecidable for PDA [Srb02c], and in fact for a very modest subclass of PDA known as one-counter nets [May03].

[2]This is because stateless PDA correspond to a natural fragment of ACP known as "BPA" (Basic Process Algebra; see [BW90]). BPA cannot model global data, but they are sufficiently powerful to model, e.g., the interprocedural data-flow [EK99]. It is worth noting that the expressive power of PDA is strictly greater than the one of BPA w.r.t. most of the considered semantic equivalences.

Comparing a PDA with a finite-state system is computationally easier. Strong and weak bisimilarity between a BPA and a finite-state system is decidable in polynomial time [KM02b]. For general pushdown automata, both problems are **PSPACE**-complete [KM02a]. Checking strong and weak simulation equivalence between a BPA and a finite-state system is **EXPTIME**-complete [KM02a], and the same holds for general PDA. Trace-like equivalences between BPA and finite-state systems are undecidable (this is a direct consequence of the undecidability of language equivalence).

Our contribution: In this paper we consider the equivalence-checking problem between PDA and finite-state systems. More precisely, we consider the problem of checking *full* equivalence between a given PDA process $p\alpha$ and a given process f of a given finite-state system $\mathcal{T}$. The processes $p\alpha$ and f are *fully equivalent* if $p\alpha$ is equivalent to f and, in addition, every reachable state of $p\alpha$ is equivalent to some state f' of $\mathcal{T}$. In other words, the specification must define the "global" behaviour of a given program. For bisimulation-like equivalences, the extra condition about reachable states is redundant. However, for simulation-like and trace-like equivalences, this condition is fully meaningful.

We propose a unified method for deciding full equivalence between PDA and finite-state systems. The method consists of two parts. The first part is generic and works for every "reasonable" semantic equivalence (an equivalence is considered "reasonable" if it is a right PDA congruence; see Definition 3.2). The authors are not aware of any semantic equivalence which is not reasonable in this sense. The second part is equivalence-specific. The difference between individual equivalences is hidden in the notion of *expansion*. There are four abstract conditions which guarantee appropriateness of the designed expansion for a given equivalence. The applicability of the method to concrete equivalences is demonstrated by defining appropriate expansions for the main conceptual representatives. Special attention is devoted to bisimulation-like equivalences (we explic-

itly consider weak, early, delay, branching, and probabilistic bisimilarity), but we also show how to handle weak simulation equivalence and weak trace equivalence. The application part is nontrivial and most of technical tricks are hidden there.

Interestingly, the generality of the method does not lead to the loss of efficiency. For bisimulation-like and simulation-like equivalences, our method results in algorithms which are *polynomial* in the size of the PDA and the finite-state system on input, and *exponential* in the number of control states of the PDA. So, the algorithm is exponential for general PDA, but polynomial for each subclass of PDA where the number of control states is bounded by a fixed constant (in particular, this applies to BPA). Since these problems are **PSPACE**-hard for general PDA processes, the obtained algorithms are essentially time-optimal. For trace-like equivalences, the algorithm requires exponential time even for BPA, but the problem is also **PSPACE**-hard for BPA.

The list of particular results obtained by applying our method includes some items which are first results of their kind. Below we explicitly mention some of them (the subclass of PDA where the number of control states is bounded by a given k is denoted PDA^k):

(a) Branching bisimilarity [vGW96] between PDA^k and finite-state systems is decidable in polynomial time. To the best of authors' knowledge, this is the first result about computational tractability of branching bisimilarity for systems with infinitely many states. Branching bisimilarity plays a distinguished role in the semantics of systems with silent moves [vG94], similarly as strong bisimilarity [Par81] for processes without silent moves. However, the "algorithmic support" for branching bisimilarity has been so far limited only to finite-state systems. A related concept of weak bisimilarity [Mil89] is substantially more developed in this sense. One reason is that weak bisimilarity admits a simple game-theoretic characterization [Sti98b, Tho93] and consequently it is "more manageable" than branching

bisimilarity. Our method treats all equivalences in the same way and consequently branching bisimilarity is equivalently manageable as weak bisimilarity in our setting (the same applies to early and delay bisimilarity; results for these equivalences are also first of their kind).

(b) Probabilistic bisimilarity [LS91, vGSST90] between PDA^k and finite-state systems is decidable in polynomial time. This result applies to (fully) probabilistic extensions of PDA and finite-state systems. Probabilistic bisimilarity has so far been considered only for finite-state systems. The obtained polynomial-time algorithm indicates that one can go beyond this limit without losing efficiency.

(c) For simulation-like equivalences (represented by weak simulation equivalence), we prove that full equivalence between PDA^k and finite-state systems is decidable in polynomial time. Since the non-full variant of the problem is **EXPTIME**-complete even for BPA [KM02a], this result shows that the extra condition about reachable states used in the definition of full equivalence actually makes the problem more tractable (rather than more complicated). The same applies to trace-like equivalences (represented by weak trace equivalence in this paper). Trace-like equivalences between BPA and finite-state systems are undecidable; this is a direct consequence of the undecidability of language equivalence. However, full trace-like equivalences between PDA and finite-state systems are decidable in exponential time (this problem is **PSPACE**-hard even for BPA).

Another generic outcome of our method is an algorithm deciding whether a given finite-state process f is the $\sim$-quotient of a given PDA process $p\alpha$ for a given semantic equivalence $\sim$. The complexity of this algorithm is essentially the same as the complexity of deciding full $\sim$-equivalence. In particular, it is polynomial for PDA^k processes when $\sim$ is simulation-like, and exponential for PDA processes when $\sim$ is trace-like. In the context of formal verification, semantic quotients are used as succinct representations of original systems. Since most (if not all) of the ex-

isting process equivalences are preserved under their respective quotients [Kuč99, KE03], the information about the state-space of a given process is faithfully preserved in its $\sim$-quotient.

This paper is organized as follows. We start with basic definitions in Section 2. In Section 3, a suitable composition principle allowing to derive new pairs of equivalent processes from already existing ones is developed. This, in turn, allows to represent full equivalence between a given PDA and a given finite-state system by a finite relation called *base*. The method is related to the technique of bisimulation bases pioneered by Caucal [Cau90], and can also be seen as a generalization of the method used in [KM02b] to prove that weak bisimilarity between BPA and finite-state systems is decidable in polynomial time. In Section 4 we show how to compute the base. The first part of our development is again generic; we give an abstract algorithm for computing the base and identify the equivalence-specific part of the problem which is hidden in the notion of expansion. In subsequent subsections, we show how to define expansions for various concrete process equivalences.

2 Basic Definitions

Definition 2.1. *A* transition system *is a triple $\mathcal{T} = (S, \rightarrow, \mathcal{A})$ where S is a finite or countably infinite set of* states*, $\mathcal{A}$ is a finite set of* actions*, and $\rightarrow \subseteq S \times \mathcal{A} \times S$ is a* transition relation.

We write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \rightarrow$, and we extend this notation to the elements of $\mathcal{A}^*$ in the standard way. We say that a state t is *reachable* from a state s, written $s \rightarrow^* t$, if there is $w \in \mathcal{A}^*$ such that $s \xrightarrow{w} t$. Let τ be a distinguished *silent* action, and let $\mathcal{A}_\tau = \mathcal{A} \cup \{\tau\}$. For every $a \in \mathcal{A}_\tau$ we define the relation $\stackrel{a}{\Rightarrow} \subseteq S \times S$ as follows:

- $s \stackrel{\tau}{\Rightarrow} t$ iff there is a sequence of the form $s = p_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_k = t$ where $k \geq 0$;

• $s \stackrel{a}{\Rightarrow} t$ where $a \neq \tau$ iff there are p, q such that $s \stackrel{\tau}{\Rightarrow} p \stackrel{a}{\rightarrow} q \stackrel{\tau}{\Rightarrow} t$.

From now on, a *process* is formally understood as a state of (some) transition system. Intuitively, transitions from a given process s model possible computational steps, and the silent action τ is used to mark those steps which are internal (i.e., not externally observable).

Definition 2.2. *Let s be a process of a transition system $\mathcal{T} = (S, \rightarrow, \mathcal{A})$, and let $\sim$ be a process equivalence. The* $\sim$-quotient *of s is the process $[s]$ of the transition system $\mathcal{T}/\sim = (S/\sim, \mathcal{A}, \rightarrowtail)$ where $[t] \stackrel{a}{\rightarrowtail} [u]$ iff there are some $t', u' \in S$ such that $t \sim t'$, $u \sim u'$, and $t' \stackrel{a}{\rightarrow} u'$.*

Most (if not all) of the existing process equivalences are *preserved under quotients* in the sense that each process is equivalent to its corresponding $\sim$-quotient (see [Kuč99, KE03] for a more detailed discussion).

Definition 2.3. *A* pushdown automaton (PDA) *is a tuple $\Delta = (Q, \Gamma, \mathcal{A}, \delta)$ where Q is a finite set of* control states, *Γ is a finite* stack alphabet, *$\mathcal{A}$ is a finite* input alphabet, *and $\delta : (Q \times \Gamma) \rightarrow 2^{\mathcal{A} \times Q \times \Gamma^{\leq 2}}$ is a* transition function *where $\Gamma^{\leq 2} = \{\varepsilon\} \cup \Gamma \cup (\Gamma \times \Gamma)$*

In the rest of this paper we adopt a more intuitive notation, writing $pX \stackrel{a}{\rightarrow} q\beta \in \delta$ instead of $(a, (q, \beta)) \in \delta(p, X)$. To Δ we associate the transition system $\mathcal{T}_\Delta$ where $Q \times \Gamma^*$ is the set of states (we write $p\alpha$ instead of (p, α)), $\mathcal{A}$ is the set of actions, and the transition relation is determined by $pX\alpha \stackrel{a}{\rightarrow} q\beta\alpha$ iff $pX \stackrel{a}{\rightarrow} q\beta \in \delta$.

3 A Finite Semantic Base for PDA

For the rest of this section, let us fix a pushdown automaton $\Delta = (Q, \Gamma, \mathcal{A}, \delta)$ and a finite state system $\mathcal{T} = (F, \mathcal{A}, \rightarrow)$. The symbol $F_\perp$ denotes the set $F \cup \{\perp\}$, where $\perp \notin F$ stands for "undefined".

Definition 3.1. *For every process* $p\alpha$ *of* Δ *we define the set* $M_{p\alpha} = \{q \in Q \mid p\alpha \rightarrow^* q\varepsilon\}$. *A function* $\mathcal{F} : Q \rightarrow F_\perp$ *is* compatible *with* $p\alpha$ *iff for every* $q \in M_{p\alpha}$ *we have that* $\mathcal{F}(q) \neq \perp$. *The class of all functions that are compatible with* $p\alpha$ *is denoted* $\mathcal{C}(p\alpha)$.

For every process $p\alpha$ of Δ and every $\mathcal{F} \in \mathcal{C}(p\alpha)$ we define the process $p\alpha\mathcal{F}$ whose transitions are determined by the following rules:

$$\frac{p\alpha \xrightarrow{a} q\beta}{p\alpha\mathcal{F} \xrightarrow{a} q\beta\mathcal{F}}\mathcal{F} \in \mathcal{C}(p\alpha) \qquad \frac{\mathcal{F}(p) \xrightarrow{a} f}{p\mathcal{F} \xrightarrow{a} p\mathcal{F}[f/p]}\mathcal{F} \in \mathcal{C}(p\varepsilon)$$

Here $\mathcal{F}[f/p] : Q \rightarrow F_\perp$ is a function which returns the same result as $\mathcal{F}$ for every argument except for p where $\mathcal{F}[f/p](p) = f$. In other words, $p\alpha\mathcal{F}$ behaves like $p\alpha$ until the point when the stack is emptied and a configuration of the form $q\varepsilon$ is entered; from that point on, $p\alpha\mathcal{F}$ behaves like $\mathcal{F}(q)$. Note that if $\mathcal{F} \in \mathcal{C}(p\alpha)$ and $p\alpha \rightarrow^* q\beta$, then $\mathcal{F} \in \mathcal{C}(q\beta)$. We put $\mathit{Stack}(\Delta, F) = \Gamma^* \cup \{p\alpha\mathcal{F} \mid p \in Q, \alpha \in \Gamma^*, \mathcal{F} \in (F_\perp)^Q\}$, and $\mathcal{P}(\Delta, F) = \{p\alpha \mid p \in Q, \alpha \in \Gamma^*\} \cup \{p\alpha\mathcal{F} \mid p \in Q, \alpha \in \Gamma^*, \mathcal{F} \in \mathcal{C}(p\alpha)\}$.

Definition 3.2. *We say that an equivalence* $\sim$ *over* $\mathcal{P}(\Delta, F) \cup F$ *is a* right PDA congruence *iff the following conditions are satisfied:*

- *For every process* $p\alpha$ *of* Δ *and all* $w, v \in \mathit{Stack}(\Delta, F)$ *we have that if* $qw \sim qv$ *for all* $q \in M_{p\alpha}$, *then also* $p\alpha w \sim p\alpha v$.
- $p\mathcal{F} \sim \mathcal{F}(p)$ *for every* $p\mathcal{F} \in \mathcal{P}(\Delta, F)$. *(This condition is satisfied by all "behavioral" equivalences which do not distinguish between isomorphic processes. However,* $\sim$ *can be an arbitrary equivalence, and therefore this condition is not redundant.)*

One intuitively expects that every "reasonable" semantic equivalence should be a right PDA congruence. In particular, bisimulation-like, simulation-like, and trace-like equivalences (even in their "weak" forms) are right PDA congruences. For the rest of this section, we fix a right PDA congruence $\sim$.

In this paper we consider the problem of full equivalence checking between PDA and finite-state processes. The notion of full equivalence is introduced in our next definition.

Definition 3.3. *Let* $p\alpha$ *be a process of* Δ *and* $f \in F$. *We say that* $p\alpha$ *is* fully *equivalent to* f *(with respect to* $\sim$*), written* $p\alpha \precsim f$, *iff* $p\alpha \sim f$ *and for every* $p\alpha \rightarrow^* q\beta$ *there is some* $f' \in F$ *such that* $q\beta \sim f'$. *(Note that* f' *does not have to be reachable from* f*.)*

Now we formulate a composition lemma for pushdown processes.

Lemma 3.4. *Let* $p\alpha\mathcal{G} \precsim f$, *where* $\mathcal{G} \in \mathcal{C}(p\alpha)$ *and* $f \in F$. *Further, let* $\beta, \gamma \in \Gamma^*$ *and* $\mathcal{H} : Q \rightarrow F_\perp$. *Then the following holds:*

(1) If $q\beta \precsim \mathcal{G}(q)$ *for all* $q \in M_{p\alpha}$, *then* $p\alpha\beta \precsim f$.

(2) If $\mathcal{H} \in \mathcal{C}(q\gamma)$ *and* $q\gamma\mathcal{H} \precsim \mathcal{G}(q)$ *for all* $q \in M_{p\alpha}$, *then* $\mathcal{H} \in \mathcal{C}(p\alpha\gamma)$ *and* $p\alpha\gamma\mathcal{H} \precsim f$.

Proof.

(1) First we show that for every $p\alpha \rightarrow^* p'\alpha'$ we have that $p'\alpha'\beta \sim p'\alpha'\mathcal{G}$. Clearly $M_{p'\alpha'} \subseteq M_{p\alpha}$. Since $q\beta \precsim \mathcal{G}(q)$ and $\mathcal{G}(q) \sim q\mathcal{G}$ for all $q \in M_{p'\alpha'}$, we have that $q\beta \sim q\mathcal{G}$ for each $q \in M_{p'\alpha'}$. Hence, $p'\alpha'\beta \sim p'\alpha'\mathcal{G}$ because $\sim$ is a right PDA congruence.

Now we can conclude that $p\alpha\beta \sim p\alpha\mathcal{G} \sim f$. It remains to show that for every $p\alpha\beta \rightarrow^* r\gamma$ there is some $f' \in F$ such that $r\gamma \sim f'$. There are two possibilities:

(a) $p\alpha\beta \rightarrow^* p'\alpha'\beta = r\gamma$ where $p\alpha \rightarrow^* p'\alpha'$. Then $r\gamma = p'\alpha'\beta \sim p'\alpha'\mathcal{G}$. Since $p\alpha\mathcal{G} \precsim f$ and $p\alpha\mathcal{G} \rightarrow^* p'\alpha'\mathcal{G}$, there is some $f' \in \mathcal{F}$ such that $p'\alpha'\mathcal{G} \sim f'$, hence also $r\gamma = p'\alpha'\beta \sim f'$ as needed.

(b) $p\alpha\beta \rightarrow^* q\beta \rightarrow^* r\gamma$ where $p\alpha \rightarrow^* q\varepsilon$. Since $q\beta \precsim \mathcal{G}(q)$ and $q\beta \rightarrow^* r\gamma$, there must be some $f' \in F$ such that $r\gamma \sim f'$.

(2) Similarly. □

Definition 3.5. *Let* $\alpha \in \Gamma^*$, $\mathcal{F}, \mathcal{G} : Q \to F_\perp$. *We write*

- $\alpha \simeq \mathcal{F}$ *iff* $\forall p \in Q : \mathcal{F}(p) \neq \perp \implies p\alpha \precsim \mathcal{F}(p)$;
- $\alpha\mathcal{G} \simeq \mathcal{F}$ *iff* $\forall p \in Q : \mathcal{F}(p) \neq \perp \implies \mathcal{G} \in \mathcal{C}(p\alpha) \wedge p\alpha\mathcal{G} \precsim \mathcal{F}(p)$.

Definition 3.6. *Let*

$$K = \{(\varepsilon, \mathcal{F}) \mid \varepsilon \simeq \mathcal{F}\} \cup \{(\mathcal{G}, \mathcal{F}) \mid \mathcal{G} \simeq \mathcal{F}\} \cup K'$$

where $K' \subseteq \Gamma \times (F_\perp)^Q \cup (\Gamma \times (F_\perp)^Q)) \times (F_\perp)^Q$. *(That is,* K' *consists of (some) pairs of the form* $(X, \mathcal{F})$ *and* $(X\mathcal{G}, \mathcal{F})$*).*

We say that K *is* well-formed *iff* K *satisfies the following conditions:*

- *if* $(X\mathcal{G}, \mathcal{F}) \in K$ *and* $\mathcal{F}(p) \neq \perp$*, then* $\mathcal{G} \in \mathcal{C}(pX)$;
- *if* $(X, \mathcal{F}) \in K$ *(or* $(X\mathcal{G}, \mathcal{F}) \in K$*) and* $(\mathcal{F}, \mathcal{H}) \in K$*, then also* $(X, \mathcal{H}) \in K$ *(or* $(X\mathcal{G}, \mathcal{H}) \in K$*, resp.).*

It is clear that there are only finitely many well-formed sets, and that there exists the greatest well-formed set G whose size is $\mathcal{O}(|\Gamma| \cdot |F|^{2 \cdot |Q|})$. Further, observe that if $\sim$ is decidable for finite-state processes, then G is effectively constructible.

Definition 3.7. *Let* K *be a well-formed set. The* closure of K*, denoted* $Cl(K)$*, is the least set* L *satisfying the following conditions:*

(1) $K \subseteq L$;
(2) if $(\alpha\mathcal{G}, \mathcal{F}) \in L$, $(\varepsilon, \mathcal{G}) \in K$*, and* $\alpha \neq \varepsilon$*, then* $(\alpha, \mathcal{F}) \in L$;
(3) if $(\alpha\mathcal{G}, \mathcal{F}) \in L$, $(\mathcal{H}, \mathcal{G}) \in K$*, and* $\alpha \neq \varepsilon$*, then* $(\alpha\mathcal{H}, \mathcal{F}) \in L$;
(4) if $(\alpha\mathcal{G}, \mathcal{F}) \in L$, $(X, \mathcal{G}) \in K$*, and* $\alpha \neq \varepsilon$*, then* $(\alpha X, \mathcal{F}) \in L$;
(5) if $(\alpha\mathcal{G}, \mathcal{F}) \in L$, $(X\mathcal{H}, \mathcal{G}) \in K$*, and* $\alpha \neq \varepsilon$*, then* $(\alpha X\mathcal{H}, \mathcal{F}) \in L$.

Note that $Cl(K) = \bigcup_{i=0}^{\infty} Cl^i(K)$ where $Cl^0(K) = K$ and $Cl^{i+1}(K)$ consists of exactly those pairs which are either in $Cl^i(K)$ or can be derived from K and $Cl^i(K)$ by applying one of the rules (1)–(5) of Definition 3.7. Another simple observation (which will be useful later) is the following:

Lemma 3.8. *Let* K *be a well-formed set, and let* $(\mathcal{F},\mathcal{H}) \in \mathsf{K}$. *If* $(\alpha,\mathcal{F}) \in \dot{Cl}(\mathsf{K})$, *then also* $(\alpha,\mathcal{H}) \in \dot{Cl}(\mathsf{K})$. *Similarly, if* $(\alpha\mathcal{G},\mathcal{F}) \in \dot{Cl}(\mathsf{K})$, *then also* $(\alpha\mathcal{G},\mathcal{H}) \in \dot{Cl}(\mathsf{K})$.

For our purposes, the following well-formed set is particularly important:

Definition 3.9. *The* base $\mathcal{B}$ *is defined as follows:*

$$\begin{aligned}\mathcal{B} &= \{(\varepsilon,\mathcal{F}) \mid \varepsilon \simeq \mathcal{F}\} \cup \{(\mathcal{G},\mathcal{F}) \mid \mathcal{G} \simeq \mathcal{F}\} \cup \{(\mathsf{X},\mathcal{F}) \mid \mathsf{X} \simeq \mathcal{F}\} \\ &\cup \{(\mathsf{X}\mathcal{G},\mathcal{F}) \mid \mathsf{X}\mathcal{G} \simeq \mathcal{F}\}\end{aligned}$$

Theorem 3.10. *Let* $\alpha \in \Gamma^*$ *and* $\mathcal{F},\mathcal{G} : \mathsf{Q} \to \mathsf{F}_\perp$. *We have*

- $\alpha \simeq \mathcal{F}$ *iff* $(\alpha,\mathcal{F}) \in Cl(\mathcal{B})$;
- $\alpha\mathcal{G} \simeq \mathcal{F}$ *iff* $(\alpha\mathcal{G},\mathcal{F}) \in Cl(\mathcal{B})$.

Proof. For the "$\Leftarrow$" direction, it suffices to show that all of the rules introduced in Definition 3.7 preserve the relation $\simeq$. We give an explicit proof just for (5) (the other cases follow similarly). Let $\alpha\mathcal{G} \simeq \mathcal{F}$ and $\mathsf{X}\mathcal{H} \simeq \mathcal{G}$. We show that $\alpha\mathsf{X}\mathcal{H} \simeq \mathcal{F}$. So, let $\mathsf{p} \in \mathsf{Q}$ such that $\mathcal{F}(\mathsf{p}) \neq \perp$. Since $\alpha\mathcal{G} \simeq \mathcal{F}$, we have that $\mathsf{p}\alpha\mathcal{G} \precsim \mathcal{F}(\mathsf{p})$. For every $\mathsf{q} \in \mathsf{M}_{\mathsf{p}\alpha}$ we have that $\mathcal{G}(\mathsf{q}) \neq \perp$. Hence, $\mathcal{H} \in \mathcal{C}(\mathsf{q}\mathsf{X})$ and $\mathsf{q}\mathsf{X}\mathcal{H} \precsim \mathcal{G}(\mathsf{q})$ because $\mathsf{X}\mathcal{H} \simeq \mathcal{G}$. Now we can apply Lemma 3.4 to conclude that $\mathcal{H} \in \mathcal{C}(\mathsf{p}\alpha\mathsf{X})$ and $\mathsf{p}\alpha\mathsf{X}\mathcal{H} \precsim \mathcal{F}(\mathsf{p})$. Thus, we obtain $\alpha\mathsf{X}\mathcal{H} \simeq \mathcal{F}$.

The "$\Rightarrow$" direction will be shown by induction on the length of α. If $\alpha = \varepsilon$, we are done immediately because for all $\varepsilon \simeq \mathcal{F}$ and $\mathcal{G} \simeq \mathcal{F}$ we have that $(\varepsilon,\mathcal{F})$ and $(\mathcal{G},\mathcal{F})$ are in $\mathcal{B}$. Now assume that $\alpha = \beta\mathsf{X}$, and let $\beta\mathsf{X} \simeq \mathcal{F}$ (the case when $\beta\mathsf{X}\mathcal{G} \simeq \mathcal{F}$ follows in the same way and therefore it is not considered explicitly). Let us define the function $\mathcal{G} : \mathsf{Q} \to \mathsf{F}_\perp$ as follows (for purposes of this definition, fix an arbitrary linear ordering over F):

$$\mathcal{G}(\mathsf{q}) = \begin{cases} \text{the least } \mathsf{f} \text{ s.t. } \mathsf{q}\mathsf{X} \precsim \mathsf{f} & \text{if } \exists \mathsf{p} \in \mathsf{Q} \text{ s.t. } \mathcal{F}(\mathsf{p}) \neq \perp \\ & \text{and } \mathsf{p}\beta \to^* \mathsf{q}\varepsilon; \\ \perp & \text{otherwise.} \end{cases}$$

First, let us verify that $\mathcal{G}$ is correctly defined, i.e., if $q \in Q$ for which there is $p \in Q$ where $\mathcal{F}(p) \neq \bot$ and $p\beta \rightarrow^* q\varepsilon$, then there is *at least one* $f \in F$ such that $qX \precsim f$. Since $\mathcal{F}(p) \neq \bot$ and $\beta X \simeq \mathcal{F}$, we have that $p\beta X \precsim \mathcal{F}(p)$. As $p\beta \rightarrow^* q\varepsilon$, we also have that $p\beta X \rightarrow^* qX$ and by definition of $\precsim$ there must be some $f \in F$ such that $qX \sim f$. Moreover, $qX \precsim f$ because each state reachable from qX is also reachable from $p\beta X$ and therefore it must be equivalent to some state of F.

Now we can readily confirm that $\beta\mathcal{G} \simeq \mathcal{F}$ and $X \simeq \mathcal{G}$ just by applying the definition of $\mathcal{G}$ above. This means that $(\beta\mathcal{G}, \mathcal{F}) \in Cl(\mathcal{B})$ (by induction hypothesis), $(X, \mathcal{G}) \in \mathcal{B}$ (by definition of $\mathcal{B}$), and hence also $(\beta X, \mathcal{F}) \in Cl(\mathcal{B})$ by applying the rule (4) of Definition 3.7. □

4 Computing the Base

In this section we present algorithms for computing the base $\mathcal{B}$ for various process equivalences. We start by describing the generic part of the method together with some auxiliary technical results which are also valid for every process equivalence which is a right PDA congruence. The applicability of the method to concrete process equivalences is demonstrated in subsequent subsections (due to the lack of space, we could include only a subsection devoted to bisimulation equivalences with silent moves; the other parts can be found in [KM04]). For the rest of this section, let us fix

- a pushdown automaton $\Delta = (Q, \Gamma, \mathcal{A}, \delta)$ of size n;
- a finite state system $\mathcal{T} = (F, \mathcal{A}, \rightarrow)$ of size m.
- a right PDA congruence $\sim$ over $\mathcal{P}(\Delta, F) \cup F$ which is decidable for finite-state processes.

In our complexity estimations we also use the parameter $z = |F|^{|Q|}$.

Let $\mathcal{W}$ be the (finite) set of all well-formed sets. Note that $(\mathcal{W}, \subseteq)$ is a complete lattice. Let $Exp : \mathcal{W} \rightarrow \mathcal{W}$ be a function satisfying the following four conditions:

(1) $Exp(\mathcal{B}) = \mathcal{B}$.

(2) Exp is monotonic, i.e. $K \subseteq L$ implies $Exp(K) \subseteq Exp(L)$.

(3) If $K = Exp(K)$, then $K \subseteq \mathcal{B}$.

(4) For every well formed set K, the membership to $Exp(K)$ is decidable.

The conditions (1) and (3) together say that $\mathcal{B}$ is the greatest fixed-point of Exp. Since Exp is monotonic and $\mathcal{W}$ is finite, we further have $\mathcal{B} = \bigcap_{i=0}^{\infty} Exp^i(G)$ where G is the greatest well-formed set. In other words, the base $\mathcal{B}$ can be computed by the algorithm of Figure 1. Observe that G is effectively computable because $\sim$ is decidable over finite-state processes.

Input: A PDA Δ, a finite-state system $\mathcal{T}$

Output: The base $\mathcal{B}$

1: $\mathcal{B} :=$ the greatest well-formed set;
2: **repeat**
3: $\quad K := \mathcal{B}; \mathcal{B} := \emptyset$
4: $\quad$ **for all** $(w, \mathcal{F}) \in K$ **do**
5: $\quad\quad$ **if** $(w, \mathcal{F}) \in Exp(K)$ **then** $\mathcal{B} := \mathcal{B} \cup \{(w, \mathcal{F})\}$ **fi**
6: $\quad$ **od**;
7: **until** $\mathcal{B} = K$

Figure 1: An algorithm for computing $\mathcal{B}$

As we shall see, an appropriate Exp satisfying the conditions (1)–(4) can be designed for almost every process equivalence of the linear / branching time spectrum [vG99, vG93]. Now we introduce further notions and results which underpin our technical constructions.

For every set of processes $\mathcal{P}$ and every action a we define the sets

- $Post_a(\mathcal{P}) = \{t \mid \exists s \in \mathcal{P} : s \xrightarrow{a} t\}$
- $Post^*(\mathcal{P}) = \{t \mid \exists s \in \mathcal{P} : s \rightarrow^* t\}$
- $Post^*_\tau(\mathcal{P}) = \{t \mid \exists s \in \mathcal{P} : s \stackrel{\tau}{\Rightarrow} t\}$

Note that if $\mathcal{P}$ is a subset of $\mathcal{P}(\Delta, F)$, then so are $Post_a(\mathcal{P})$, $Post^*(\mathcal{P})$, and $Post^*_\tau(\mathcal{P})$.

To be able to represent infinite subsets of $\mathcal{P}(\Delta, F)$ in a finite and compact way, we borrow the following concept from [BEM97]:

Definition 4.1. *A* multi-automaton *is a tuple* $\mathcal{M} = (S, \Sigma, \delta, Acc)$ *where*

- S *is a finite set of* states *such that* $Q \subseteq S$ *(i.e, the control states of* Δ *are among the states of* $\mathcal{M}$*);*
- $\Sigma = \Gamma \cup \{\mathcal{F} \mid \mathcal{F} : Q \to F_\perp\}$ *is the* input alphabet *(the alphabet has a special symbol for each* $\mathcal{F} : Q \to F_\perp$*);*
- $\delta \subseteq S \times \Sigma \times S$ *is a transition relation;*
- $Acc \subseteq S$ *is a set of* accepting states.

Every multi-automaton $\mathcal{M}$ *determines a unique set*

$$\mathcal{L}(\mathcal{M}) = \{pw \mid p \in Q, w \in \Sigma^*, \delta(p, w) \cap Acc \neq \emptyset\}$$

A set $\mathcal{P} \subseteq \mathcal{P}(\Delta, F)$ *is* recognized *by a multi-automaton* $\mathcal{M}$ *iff* $\mathcal{P} = \mathcal{L}(\mathcal{M})$.

A proof of the following lemma can be found, e.g., in [EHRS00].

Lemma 4.2. *Let* $\mathcal{P} \subseteq \mathcal{P}(\Delta, F)$ *be a set of processes recognized by a multi-automaton* $\mathcal{M}$*. Then one can compute multi-automata recognizing the sets* $Post_a(\mathcal{P})$*,* $Post^*(\mathcal{P})$*, and* $Post^*_\tau(\mathcal{P})$ *in time which is polynomial in* m, n, z *and the size of* $\mathcal{M}$*.*

Definition 4.3. *Let* K *be a well-formed set. For all* $f \in F$ *and* $i \in \mathbb{N}_0$ *we define the set* $Gen^i_f(K) =$

$$\begin{aligned} & \{p\alpha \mid \exists \mathcal{F} \text{ s.t. } \mathcal{F}(p) = f \text{ and } (\alpha, \mathcal{F}) \in Cl^i(K)\} \\ \cup\ & \{p\alpha\mathcal{G} \mid \exists \mathcal{F} \text{ s.t. } \mathcal{F}(p) = f \text{ and } (\alpha\mathcal{G}, \mathcal{F}) \in Cl^i(K)\} \end{aligned}$$

Further, we put $Gen_f(K) = \bigcup_{i=0}^{\infty} Gen^i_f(K)$.

Lemma 4.4. *The relation* $\precsim$ *over* $\mathcal{P}(\Delta, F) \times F$ *is exactly* $\bigcup_{f \in F} Gen_f(\mathcal{B}) \times \{f\}$.

Proof. It suffices to apply Theorem 3.10 and Definition 4.3. □

Lemma 4.5. *Let* K *be a well-formed set and* $\mathsf{f} \in \mathsf{F}$. *The set* $Gen_{\mathsf{f}}(\mathsf{K})$ *is recognized by a multi-automaton* $\mathcal{M}_{\mathsf{K},\mathsf{f}}$ *which is constructible in time polynomial in* m, n, z.

Proof. We refer to [KM02b] where a similar result is proven explicitly; the construction required for Lemma 4.5 differs from the one presented in [KM02b] only in minor details. □

As we shall see in the next subsections, Lemma 4.2 and Lemma 4.5 are heavily used in algorithms which decide the membership to $Exp(\mathsf{K})$ for a given well-formed set K. With their help it is also possible to decide whether a given finite-state system $\mathcal{T}$ is the $\sim$-quotient of a given PDA process $p\alpha$ (see the algorithm of Fig. 2). Observe that the **if** statements in the lines 5–6 can be implemented by computing a multi-automaton $\mathcal{M}$ recognizing the set $Post_a(Post^*(\{p\alpha\}) \cap Gen_{\mathsf{f}}(\mathcal{B})) \cap Gen_{\mathsf{f}'}(\mathcal{B})$ (this is possible due to Lemma 4.2 and Lemma 4.5) and checking whether $\mathcal{L}(\mathcal{M}) = \emptyset$. Moreover, if the base $\mathcal{B}$ is computable in time polynomial in m, n, z, then the algorithm of Fig. 2 terminates in time which is also polynomial in m, n, z.

We finish this part by an auxiliary technical lemma whose proof is also independent of a concrete choice of $\sim$.

Lemma 4.6. *Let* K *be a well-formed set. The following conditions hold:*

(1) If $q\beta\mathcal{G} \in Gen_g(\mathsf{K})$ *and* $(\varepsilon, \mathcal{G}) \in \mathsf{K}$, *then* $q\beta \in Gen_g(\mathsf{K})$.
(2) If $q\beta\mathcal{G} \in Gen_g(\mathsf{K})$, $(X, \mathcal{G}) \in \mathsf{K}$, *then* $q\beta X \in Gen_g(\mathsf{K})$.
(3) $p\mathcal{G} \in Gen_g(\mathsf{K})$ *iff* $\mathcal{G}(p) \precsim g$.
(4) Let $g \precsim g'$. *Then* $pw \in Gen^i_g(\mathsf{K})$ *implies* $pw \in Gen^i_{g'}(\mathsf{K})$.

Proof. (1) and (2) follow directly from Definition 3.7 and Definition 4.3.

(3), "$\Rightarrow$": As $p\mathcal{G} \in Gen_g(\mathsf{K})$, by Definition 4.3 there is $\mathcal{F}$ such that $\mathcal{F}(p) = g$ and $(\mathcal{G}, \mathcal{F}) \in Cl(\mathsf{K})$. However, this means that $(\mathcal{G}, \mathcal{F}) \in \mathsf{K}$ by Definition 3.7. Further, $\mathcal{G} \simeq \mathcal{F}$ by Definition 3.6. Thus, we obtain $p\mathcal{G} \precsim \mathcal{F}(p) = g$ and as $\mathcal{G}(p) \sim p\mathcal{G}$, we are done.

Input: A PDA $\Delta = (Q, \Gamma, \mathcal{A}, \delta)$, a process $p\alpha$ of Δ,
and a finite-state system $\mathcal{T} = (F, \mathcal{A}, \rightarrow)$.

Output: YES if $\mathcal{T}$ is the $\sim$-quotient of $p\alpha$, NO otherwise.

1: **if** there are $f, f' \in F$ such that $f \neq f'$ and $f \sim f'$ **then return** NO; **fi**
2: compute the base $\mathcal{B}$;
3: **if** there is no $f \in F$ such that $p\alpha \in Gen_f(\mathcal{B})$ **then return** NO; **fi**
4: **for all** $f, f' \in F$, $a \in \mathcal{A}$ **do**
5: **if** $f \xrightarrow{a} f'$ **and** $Post_a(Post^*(\{p\alpha\}) \cap Gen_f(\mathcal{B})) \cap Gen_{f'}(\mathcal{B}) = \emptyset$
then return NO; **fi**
6: **if** $f \not\xrightarrow{a} f'$ **and** $Post_a(Post^*(\{p\alpha\}) \cap Gen_f(\mathcal{B})) \cap Gen_{f'}(\mathcal{B}) \neq \emptyset$
then return NO; **fi**
7: **od**
8: **return** YES;

Figure 2: Deciding if $\mathcal{T}$ is the $\sim$-quotient of $p\alpha$.

(3), "$\Leftarrow$": Let $\mathcal{G}(p) \precsim g$. This means that $\mathcal{G} \simeq \mathcal{G}[g/p]$, hence $(\mathcal{G}, \mathcal{G}[g/p]) \in K$ by Definition 3.6. Thus, $p\mathcal{G} \in Gen_{\mathcal{G}[g/p](p)}K = Gen_g(K)$ as required.

(4): Let $g \sim g'$. We show that if $pw \in Gen^i_g(K)$, then also $pw \in Gen^i_{g'}(K)$. Since $pw \in Gen^i_g(K)$, there is $\mathcal{F}$ such that $\mathcal{F}(p) = g$ and $(w, \mathcal{F}) \in Cl^i(K)$. Since $g \precsim g'$, we have that $\mathcal{F} \simeq \mathcal{F}[g'/p]$. Hence, $(\mathcal{F}, \mathcal{F}[g'/p]) \in K$ by Definition 3.6 and thus $(w, \mathcal{F}[g'/p]) \in Cl^i(K)$ by Lemma 3.8. This means that $pw \in Gen^i_{\mathcal{F}[g'/p](p)}(K) = Gen^i_{g'}(K)$ and we are done. □

4.1 Bisimulation Equivalences with Silent Moves

In this subsection we show how to compute the base $\mathcal{B}$ for bisimulation-like equivalences which take into account silent moves. We explicitly consider the main four representatives which are *weak, early*, *delay*, and *branching bisimilarity*. We prove that for all these equivalences, the base $\mathcal{B}$ is computable in time polynomial in m, n, z.

Definition 4.7. *Let* R *be a binary relation over processes, and let* $(s,t) \in R$. *We say that a move* $t \stackrel{a}{\Rightarrow} t'$ *is* R-consistent with a move $s \stackrel{a}{\rightarrow} s'$ in a weak, early, delay, or branching style, respectively, *if one of the following conditions is satisfied:*

- $a = \tau$, $t = t'$, *and* $(s', t) \in R$*;*
- *the move* $t \stackrel{a}{\Rightarrow} t'$ *is of the form* $t{=}u_0 \stackrel{\tau}{\rightarrow} \cdots \stackrel{\tau}{\rightarrow} u_i \stackrel{a}{\rightarrow} v_0 \stackrel{\tau}{\rightarrow} \cdots \stackrel{\tau}{\rightarrow} v_j{=}t'$, *where* $i, j \geq 0$, *such that* $(s', t') \in R$ *and*

  (i) if the style is early or branching, then also $(s, u_i) \in R$*;*

  (ii) if the style is delay or branching, then also $(s', v_0) \in R$.

We say that $(s,t) \in R$ expands in R *(in the respective style) iff for all* $a \in Act_\tau$ *and* $s \stackrel{a}{\rightarrow} s'$ *there is a move* $t \stackrel{a}{\Rightarrow} t'$ *which is* R*-consistent with* $s \stackrel{a}{\rightarrow} s'$. *Furthermore, we say that* $(s,t) \in R$ b-expands in R *(in the respective style) if* (s,t) *expands in* R *and* (t,s) *expands in* R^{-1} *in the respective style.*

A binary relation R *over processes is a* weak, early, delay, or branching bisimulation *if for every* $(s,t) \in R$ *we have that* (s,t) *b-expands in* R *in the respective style. Processes* s,t *are weakly, early, delayed, or branching bisimilar if they are related by some weak, early, delay, or branching bisimulation, respectively.*

Remark 4.8. *An important fact (which will be used in the proof of Lemma 4.11) is that the* same *notion of weak, early, delay, and branching bisimilarity is obtained when the conditions* (i) *and* (ii) *of Definition 4.7 are reformulated as follows:*

(i) *if the style is early or branching, then* $(s,u_\ell) \in R$ *for* all $0 \leq \ell \leq i$;
(ii) *if the style is delay or branching, then* $(s',v_\ell) \in R$ *for* all $0 \leq \ell \leq j$,

Since our constructions are to a large extent independent of the chosen style of bisimilarity, from now on we refer just to "bisimilarity" which is denoted by $\sim$ in the rest of this subsection. It follows directly from Definition 4.7 that $\sim = \precsim$ over $\mathcal{P}(\Delta,F) \times F$ and therefore we do not distinguish between these two relations.

For technical reasons which become clear in (the proof of) Theorem 4.7, we need to assume that the transition relation of $\mathcal{T}$ is "complete" in the following sense:

Definition 4.9. *Let* $\sim_F$ *be the relation of bisimilarity restricted to* $F \times F$. *We say that* $\mathcal{T}$ *is* complete *if for all* $f,f' \in F$ *and* $a \in \mathcal{A}_\tau$ *the following condition is satisfied: If there is a sequence of transitions forming a* $f \stackrel{a}{\Rightarrow} f'$ *move which is* $\sim_F$*-consistent with a* hypothetical *transition* $f \stackrel{a}{\rightarrow} f'$ *(note that the condition of* $\sim_F$*-consistency with* $f \stackrel{a}{\rightarrow} f'$ *makes a clear sense even if* $f \stackrel{a}{\rightarrow} f'$ *is not a transition of* $\mathcal{T}$*), then* $f \stackrel{a}{\rightarrow} f'$ *is a* real *transition of* $\mathcal{T}$.

From now on in this subsection, we assume that $\mathcal{T}$ is complete. This assumption is not restrictive because if we add the missing transitions to $\mathcal{T}$ (which can be done in polynomial time because $\sim_F$ is computable in polynomial time), each state f of $\mathcal{T}$ stays bisimilar to itself. A pleasant consequence of this assumption is that we do not have to deal with the "$\stackrel{a}{\Rightarrow}$" moves of f; it suffices to consider the "$\stackrel{a}{\rightarrow}$" ones.

Definition 4.10. *Let* $R \subseteq \mathcal{P}(\Delta, F) \times F$ *be a relation. We say that a pair* $(pw, f) \in R$ quasi-expands *in* R *iff it satisfies the following conditions:*

- *for all* $a \in \mathcal{A}$ *and* $pw \xrightarrow{a} qv$*, there is* $f \xrightarrow{a} g$ *such that* $(qv, g) \in R$*;*
- *for all* $a \in \mathcal{A}$ *and* $f \xrightarrow{a} g$*, one of the following conditions is satisfied:*
  - $a = \tau$ *and* $(pw, g) \in R$*;*
  - *there is an* R*-consistent move* $pw \stackrel{a}{\Rightarrow} qv$ *such that* $(qv, g) \in R$*. Moreover, we require that if* pw *is of the form* $p\alpha\mathcal{G}$*, then the move* $p\alpha\mathcal{G} \stackrel{a}{\Rightarrow} qv$ *contains at most one transition of the form* $r\mathcal{G} \xrightarrow{x} r\mathcal{H}$ *(which can appear only at the end of the whole move).*

We say that R *is a* quasi-bisimulation *iff every pair of* R *quasi-expands in* R*. Processes* pw *and* f *are quasi-bisimilar iff they are related by some quasi-bisimulation.*

Every quasi-bisimulation is clearly a bisimulation. The opposite is not necessarily true, but we can prove the following (here we need the fact formulated in Remark 4.8 and the assumption that $\mathcal{T}$ is complete):

Lemma 4.11. *The relation* $\sim$ *restricted to* $\mathcal{P}(\Delta, F) \times F$ *is a quasi-bisimulation.*

Proof. Let $pw \in \mathcal{P}(\Delta, F)$ and $f \in F$ such that $pw \sim f$. We show that (pw, f) quasi-expands in $\sim$.

- Let $pw \xrightarrow{a} q\beta$. We need to prove that there is $f \xrightarrow{a} g$ such that $q\beta \sim g$. Since (pw, f) b-expands in $\sim$, there are two possibilities:
  - $a = \tau$ and $q\beta \sim f$. Since $\mathcal{T}$ is complete, there is a move $f \xrightarrow{\tau} f$ and we are done;
  - there is a $\sim$-consistent move $f \stackrel{a}{\Rightarrow} g$ where $q\beta \sim g$. Since $\mathcal{T}$ is complete, there is a move $f \xrightarrow{a} g$ as needed.
- Let $p \xrightarrow{a} g$. As (pw, f) b-expands in $\sim$, we either have that $a = \tau$ and $pw \sim g$ (in which case we are done), or there is a $\sim$-consistent move $pw \stackrel{a}{\Rightarrow} qv$ such that $qv \sim g$. Note that here we can assume that this move is $\sim$-consistent even in the "stronger" sense of Remark 4.8. If pw

is of the form $p\alpha\mathcal{G}$ and the move $pw \stackrel{a}{\Rightarrow} qv$ is of the form $p\alpha\mathcal{G} \stackrel{x}{\Rightarrow} q\mathcal{G} = q\mathcal{G}_0 \stackrel{y_1}{\rightarrow} \cdots \stackrel{y_k}{\rightarrow} q\mathcal{G}_k$ where $k \geq 2$, we can argue that $\mathcal{G}_0(q) \stackrel{y_1}{\rightarrow} \cdots \stackrel{y_k}{\rightarrow} \mathcal{G}_k(q)$ is a sequence of transitions of $\mathcal{T}$ satisfying the condition given in Definition 4.9 . Hence, $\mathcal{G}_0(q) \stackrel{y}{\rightarrow} \mathcal{G}_k(q)$ because $\mathcal{T}$ is complete. This means that the process $p\alpha\mathcal{G}$ can also perform $p\alpha\mathcal{G} \stackrel{x}{\Rightarrow} q\mathcal{G} \stackrel{y}{\rightarrow} q\mathcal{G}_k$ which is admissible by Definition 4.10. □

Definition 4.12. *Let* K *be a well-formed set, and let* $R = \bigcup_{f \in F} \mathit{Gen}_f(K) \times \{f\}$. *The set* $\mathit{BExp}(K)$ *consists of all pairs* $(w, \mathcal{F}) \in K$ *such that for each* $p \in Q$ *we have that if* $\mathcal{F}(p) \neq \bot$, *then the pair* $(pw, \mathcal{F}(p))$ *quasi-expands in* R.

Now we prove that *BExp* satisfies the conditions (1)–(4) formulated at the beginning of Section 4. It follows immediatelly from the definition of *BExp* that *BExp* is monotonic. Due to Lemma 4.4 and Lemma 4.11 we obtain $\mathit{BExp}(\mathcal{B}) = \mathcal{B}$. Now we prove that if $K = \mathit{BExp}(K)$ then $K \subseteq \mathcal{B}$. This is where we need the above introduced technicalities (completeness of $\mathcal{T}$, quasi-expansion, etc.). If the definition of *BExp* was based "directly" on the notion of b-expansion, which seems to be the most natural possibility, the following theorem would *not* hold.

Theorem 4.13. *Let* K *be a well-formed set. If* $K = \mathit{BExp}(K)$, *then* $K \subseteq \mathcal{B}$.

Proof. We show that if $K = \mathit{BExp}(K)$, then the relation $R = \bigcup_{f \in F} \mathit{Gen}_f(K) \times \{f\}$ is a quasi-bisimulation. This means to show that for all $h \in F$ and $pw \in \mathit{Gen}_h^i(K)$ the pair (pw, h) quasi-expands in R. We proceed by induction on i. (We present a detailed proof just for the case when pw is of the form $p\alpha$ where $\alpha \in \Gamma^*$; the other case when pw is of the form $p\alpha\mathcal{F}$ follows similarly).

- $i = 0$. Since $p\alpha \in \mathit{Gen}_h^0(K)$, there is $\mathcal{F}$ such that $\mathcal{F}(p) = h$ and $(\alpha, \mathcal{F}) \in K$. Since $K = \mathit{BExp}(K)$, we have $(\alpha, \mathcal{F}) \in \mathit{BExp}(K)$, which means that $(p\alpha X, h)$ quasi-expands in R by Definition 4.12.
- *Induction step:* Let $p\alpha \in \mathit{Gen}_h^{i+1}(K)$. Then there must be some $\mathcal{F}$ such that $\mathcal{F}(p) \neq \bot$ and $(\alpha, \mathcal{F}) \in \mathit{Cl}^{i+1}(K)$ (see Definition 4.3). Hence, we either

have that $(\alpha, \mathcal{F}) \in Cl^i(K)$ (in which case it suffices to apply induction hypothesis), or the pair $(\alpha, \mathcal{F})$ can be derived from a pair of K and a pair of $Cl^i(K)$ using the rule (2) or (4) of Definition 3.7. We consider these two cases separately.

"**rule (2)**:" Then there is $\mathcal{G}$ such that $(\alpha\mathcal{G}, \mathcal{F}) \in Cl^i(K)$ and $(\varepsilon, \mathcal{G}) \in K$. We show that $(p\alpha, \mathcal{F}(p))$ quasi-expands in R.

(A) Let $p\alpha \xrightarrow{a} q\beta$. Then $p\alpha\mathcal{G} \xrightarrow{a} q\beta\mathcal{G}$ and since $(p\alpha\mathcal{G}, \mathcal{F}(p))$ quasi-expands in R by induction hypothesis, there is a move $\mathcal{F}(p) \xrightarrow{a} g$ which is R-consistent with $p\alpha\mathcal{G} \xrightarrow{a} q\beta\mathcal{G}$, and hence also with $p\alpha \xrightarrow{a} q\beta$ by applying Lemma 4.6 (1).

(B) Let $\mathcal{F}(p) \xrightarrow{a} g$. Since $(p\alpha\mathcal{G}, \mathcal{F}(p))$ quasi-expands in R by induction hypothesis, there is an "appropriate" responding move of $p\alpha\mathcal{G}$ which is R-consistent with $\mathcal{F}(p) \xrightarrow{a} g$. We need to distinguish several cases.

(1) $a = \tau$ and $p\alpha\mathcal{G} \in Gen_{\mathcal{F}(p)}(K)$. Then also $p\alpha \in Gen_{\mathcal{F}(p)}(K)$ by Lemma 4.6 (1) and we are done.

(2) The responding move of $p\alpha\mathcal{G}$ is of the form $p\alpha\mathcal{G} \stackrel{\tau}{\Rightarrow} q\beta\mathcal{G} \xrightarrow{a} r\gamma\mathcal{G} \stackrel{\tau}{\Rightarrow} s\delta\mathcal{G}$, where the sequence $p\alpha \stackrel{\tau}{\Rightarrow} q\beta \xrightarrow{a} r\gamma \stackrel{\tau}{\Rightarrow} s\delta$ is performable, $s\delta\mathcal{G} \in Gen_g(K)$, and

- if the style is early or branching, then also $q\beta\mathcal{G} \in Gen_{\mathcal{F}(p)}(K)$,
- if the style is delay or branching, then also $r\gamma\mathcal{G} \in Gen_g(K)$.

Hence, the sequence $p\alpha \stackrel{\tau}{\Rightarrow} q\beta \xrightarrow{a} r\gamma \stackrel{\tau}{\Rightarrow} s\delta$ is a move which is R-consistent with $\mathcal{F}(p) \xrightarrow{a} g$ by Lemma 4.6 (1) (this holds for all styles).

(3) The responding move of $p\alpha\mathcal{G}$ is of the form $p\alpha\mathcal{G} \stackrel{\tau}{\Rightarrow} q\beta\mathcal{G} \xrightarrow{a} r\gamma\mathcal{G} \stackrel{\tau}{\Rightarrow} s\mathcal{G} \xrightarrow{\tau} s\mathcal{H}$, where the sequence $p\alpha \stackrel{\tau}{\Rightarrow} q\beta \xrightarrow{a} r\gamma \stackrel{\tau}{\Rightarrow} s\varepsilon$ is performable, $s\mathcal{H} \in Gen_g(K)$, and

- if the style is early or branching, then also $q\beta\mathcal{G} \in Gen_{\mathcal{F}(p)}(K)$
- if the style is delay or branching, then also $r\gamma\mathcal{G} \in Gen_g(K)$.

We prove that the sequence $p\alpha \stackrel{\tau}{\Rightarrow} q\beta \stackrel{a}{\rightarrow} r\gamma \stackrel{\tau}{\Rightarrow} s\varepsilon$ is R-consistent with the move $\mathcal{F}(p) \stackrel{a}{\rightarrow} g$. To do that, it suffices to show that $s\varepsilon \in \mathit{Gen}_g(K)$ (the other states of this sequence are "handled" by Lemma 4.6 (1)). As $(\varepsilon, \mathcal{G}) \in K$, by induction hypothesis we know that $(s\varepsilon, \mathcal{G}(s))$ quasi-expands in R. Hence, there is a "$\stackrel{\tau}{\Rightarrow}$" move of $s\varepsilon$ which is R-consistent with $\mathcal{G}(s) \stackrel{\tau}{\rightarrow} \mathcal{H}(s)$. The only available move is $s\varepsilon \stackrel{\tau}{\Rightarrow} s\varepsilon$, hence $s\varepsilon \in \mathit{Gen}_{\mathcal{H}(s)}(K)$. Since $s\mathcal{H} \in \mathit{Gen}_g(K)$, we have $\mathcal{H}(s) \sim g$ by Lemma 4.6 (3). As $s\varepsilon \in \mathit{Gen}_{\mathcal{H}(s)}(K)$ and $\mathcal{H}(s) \sim g$, we obtain $s\varepsilon \in \mathit{Gen}_g(K)$ by Lemma 4.6 (4).

(4) The responding move of $p\alpha\mathcal{G}$ is of the form $p\alpha\mathcal{G} \stackrel{\tau}{\Rightarrow} q\mathcal{G} \stackrel{a}{\rightarrow} q\mathcal{H}$, where the sequence $p\alpha \stackrel{\tau}{\Rightarrow} q\varepsilon$ is performable, $q\mathcal{H} \in \mathit{Gen}_g(K)$, and if the style is early or branching, then also $q\mathcal{G} \in \mathit{Gen}_{\mathcal{F}(p)}(K)$. Since $q\mathcal{H} \in \mathit{Gen}_g(K)$, we have that $\mathcal{H}(q) \sim g$ by Lemma 4.6 (3). As $(\varepsilon, \mathcal{G}) \in K$, the pair $(q\varepsilon, \mathcal{G}(q))$ quasi-expands in R. As $\mathcal{G}(q) \stackrel{a}{\rightarrow} \mathcal{H}(q)$, the process $q\varepsilon$ must be able to respond by an appropriate "$\stackrel{a}{\Rightarrow}$" move. This is possible only if $a = \tau$ and $q\varepsilon \in \mathit{Gen}_{\mathcal{H}(q)}(K)$. Since $g \sim \mathcal{H}(q)$, we also have that $q\varepsilon \in \mathit{Gen}_g(K)$ by Lemma 4.6 (4). If the style is early or branching, we have that $q\mathcal{G} \in \mathit{Gen}_{\mathcal{F}(p)}(K)$ and thus also $q\varepsilon \in \mathit{Gen}_{\mathcal{F}(p)}(K)$ by Lemma 4.6 (1). To sum up, the move $p\alpha \stackrel{\tau}{\Rightarrow} q\varepsilon$ is R-consistent with $\mathcal{F}(p) \stackrel{a}{\rightarrow} g$.

"rule (4):" Then $\alpha = \beta X$ and there are $\mathcal{G}, \mathcal{H}$ such that $(\beta\mathcal{H}, \mathcal{F}) \in \mathit{Cl}^i(K)$ and $(X, \mathcal{H}) \in K$. The proof can be completed along the same lines as above, using Lemma 4.6(2) instead of Lemma 4.6(1). □

Now we show how to decide the membership to $\mathit{BExp}(K)$. At the same time, we perform a (rough) complexity analysis. Pairs of the form $(\mathcal{G}, \mathcal{F})$ and $(\varepsilon, \mathcal{F})$ belong to $\mathit{BExp}(K)$ if and only if they belong to K. Hence, they do not require any special attention. As for pairs of the form $(X, \mathcal{F})$, by Definition 4.12 we have that $(X, \mathcal{F}) \in \mathit{BExp}(K)$ iff for all $p \in Q$ such that

$\mathcal{F}(p) \neq \bot$ we have that the pair $(pX, \mathcal{F}(p))$ quasi-expands in $\bigcup_{f \in F} Gen_f(K) \times \{f\}$. This means to check if

- for all $pX \xrightarrow{a} q\beta$ there is some $\mathcal{F}(p) \xrightarrow{a} g$ such that $q\beta \in Gen_g(K)$. In other words, we are interested if there is some $g \in F$ such that $\mathcal{F}(p) \xrightarrow{a} g$ and $q\beta \in \mathcal{L}(\mathcal{M}_{K,g})$. Since the multi-automaton $\mathcal{M}_{K,g}$ is effectively constructible in time which is polynomial in m, n, z (see Lemma 4.5), this condition can be also checked in time which is polynomial in m, n, z.
- for all $\mathcal{F}(p) \xrightarrow{a} g$, one of the following two conditions is satisfied:
  - $a = \tau$ and $pX \in Gen_g(K)$. In other words, we check whether $pX \in \mathcal{L}(\mathcal{M}_{K,g})$ which can be done in time polynomial in m, n, z due to Lemma 4.5.
  - there is a sequence $pX \stackrel{\tau}{\Rightarrow} q\alpha \xrightarrow{a} r\beta \stackrel{\tau}{\Rightarrow} s\gamma$ such that $s\gamma \in Gen_g(K)$ and
    - if the style is early or branching, then $q\alpha \in Gen_{\mathcal{F}(p)}(K)$;
    - if the style is delay or branching, then $r\beta \in Gen_g(K)$.

    Depending on whether the style is weak, early, delay, or branching, this condition can be reformulated as follows:

    - $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}))) \cap Gen_g(K) \neq \emptyset$
    - $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}) \cap Gen_{\mathcal{F}(p)}(K))) \cap Gen_g(K) \neq \emptyset$
    - $Post^*_\tau(Post_a(Post^*_\tau(\{pX\})) \cap Gen_g(K)) \cap Gen_g(K) \neq \emptyset$
    - $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}) \cap Gen_{\mathcal{F}(p)}(K)) \cap Gen_g(K)) \cap Gen_g(K) \neq \emptyset$

    Due to Lemma 4.5 and Lemma 4.2, each of these four conditions can be checked in a purely "symbolic" way by performing the required operations directly on the underlying multi-automata. Obviously, the whole procedure takes time which is still polynomial in m, n, z.

Pairs of the form $(X\mathcal{G}, \mathcal{F})$ are handled in a similar way. So, the membership to $BExp(K)$ for a given K is decidable in time polynomial in m, n, z. This means that the algorithm of Fig. 1 terminates in time which is polynomial in m, n, z. So, we obtain the following theorem:

Theorem 4.14. *The problem of weak, early, delay, and branching bisimilarity between PDA and finite-state processes is decidable in time polynomial in* m, n, z. *For PDA*^{k} *processes, the same problem is decidable in time polynomial in* m, n *(for each fixed* k*).*

4.2 Probabilistic Bisimulation Equivalence

A (fully) probabilistic transition system is a tuple $\mathcal{T} = (S, \mathcal{A}, \rightarrow, \mathit{Prob})$, where S, $\mathcal{A}$, and $\rightarrow$ are defined as for (non-probabilistic) transition systems, and *Prob* is a function which to each transition $s \xrightarrow{a} t$ of $\mathcal{T}$ assigns its probability $\mathit{Prob}(s \xrightarrow{a} t) \in (0, 1]$ so that for every $s \in S$ we have $\sum_{s \xrightarrow{a} t} \mathit{Prob}(s \xrightarrow{a} t) \in \{0, 1\}$. In the rest of this section we write $s \xrightarrow{a,x} t$ instead of $\mathit{Prob}(s \xrightarrow{a} t) = x$.

Definition 4.15. *Let* $\mathcal{T} = (S, \mathcal{A}, \rightarrow)$ *be a probabilistic transition system, and let* R *be an equivalence over* S. *For all* $s \in S$, $a \in \mathcal{A}$, *and* $C \in S/R$ *we define*

- $\mathit{Succ}(s, a, C) = \{t \in C \mid s \xrightarrow{a} t\}$,
- $\mathit{Prob}(s, a, C) = \sum_{t \in \mathit{Succ}(s,a,C)} \mathit{Prob}(s \xrightarrow{a} t)$.

We say that $(s, t) \in S \times S$ p-expands *in* R *iff for all* $a \in \mathcal{A}$ *and* $C \in S/R$ *we have that* $\mathit{Prob}(s, a, C) = \mathit{Prob}(t, a, C)$. *An equivalence* R *over* S *is a* probabilistic bisimulation *iff each pair of* R *p-expands in* R. *Probabilistic processes* $s, t \in S$ *are* probabilistic bisimilar, *written* $s \sim t$, *iff they are related by some probabilistic bisimulation.*

A *probabilistic PDA* is a tuple $\Delta = (Q, \Gamma, \mathcal{A}, \delta, \mathit{Prob})$ where all elements except for *Prob* are defined as in the non-probabilistic case, and *Prob* is a a function which to each transition $pX \xrightarrow{a} q\alpha \in \delta$ assigns its probability $\mathit{Prob}(pX \xrightarrow{a} q\alpha) \in (0, 1]$ so that for all $p \in Q$ and $X \in \Gamma$ we have that $\sum_{pX \xrightarrow{a} q\alpha} \mathit{Prob}(pX \xrightarrow{a} q\alpha) \in \{0, 1\}$.

Each probabilistic PDA determines a unique probabilistic transition system in the very same way as in the non-probabilistic case. In fact, the

only difference is that transitions are now labeled by pairs of the form a, x rather than by single actions. So, the previously introduced notions make a clear sense also in the probabilistic setting. In particular, it is easy to check that probabilistic bisimilarity is a right PDA congruence.

For the rest of this subsection, we fix a probabilistic PDA $\Delta = (Q, \Gamma, \mathcal{A}, \delta, \mathit{Prob})$ of size n, and a probabilistic finite state system $\mathcal{T} = (F, \mathcal{A}, \rightarrow, \mathit{Prob}')$ of size m. As before, we use z to denote $|F|^{|Q|}$.

Definition 4.16. *Let K be a well-formed set. Let $\equiv_K$ be the least equivalence over $\mathcal{P}(\Delta, F) \cup F$ subsuming the relation $\bigcup_{f \in F} \mathit{Gen}_f(K) \times \{f\}$. The set $\mathit{PExp}(K)$ consists of all pairs $(w, \mathcal{F}) \in K$ such that for each $p \in Q$ we have that if $\mathcal{F}(p) \neq \bot$, then the pair $(pw, \mathcal{F}(p))$ p-expands in $\equiv_K$.*

The function *PExp* is clearly monotonic and one can easily check that $\mathit{PExp}(\mathcal{B}) = \mathcal{B}$. Before verifying the condition (3) of Section 4 we need to state one auxiliary lemma.

Lemma 4.17. *Let $pw \in \mathcal{P}(\Delta, F)$ and $g \in F$. If $pw \equiv_K g$ then there is $h \in F$ such that $g \equiv_K h$ and $pw \in \mathit{Gen}_h(K)$.*

Theorem 4.18. *Let K be a well-formed set. If $K = \mathit{PExp}(K)$, then $K \subseteq \mathcal{B}$.*

Proof. We show that if $K = \mathit{PExp}(K)$, then the relation $\equiv_K$ is a probabilistic bisimulation. This means to prove that every $(s, t) \in \equiv_K$ p-expands in $\equiv_K$. Since $\equiv_K$ is the reflexive, symmetric, and transitive closure of the relation $R = \bigcup_{f \in F} \mathit{Gen}_f(K) \times \{f\}$, we can distinguish two cases:

- $(s, t) \in R$. That is, $s = pw$ and $t = f$ where $pw \in \mathit{Gen}_f^i(K)$ for some $i \in \mathbb{N}_0$. We proceed by induction on i (considering only the case when $pw = p\alpha$ for some $\alpha \in \Gamma^*$; the case when $pw = p\alpha\mathcal{F}$ is handled similarly). We proceed by induction on i.
  - $i = 0$. Then there is $\mathcal{F}$ such that $\mathcal{F}(p) = f$ and $(\alpha, \mathcal{F}) \in K$. As $K = \mathit{PExp}(K)$, we immediatelly obtain that $(p\alpha, f)$ p-expands in $\equiv_K$ just by applying the definitions.

– *Induction step:* If $p\alpha \in Gen_f^{i+1}(K)$, there is some $\mathcal{F}$ such that $\mathcal{F}(p) \neq \bot$ and $(\alpha, \mathcal{F}) \in Cl^{i+1}(K)$. By definition of $Cl^{i+1}(K)$, we either have that $(\alpha, \mathcal{F}) \in Cl^{i}(K)$ (in which case we just apply induction hypothesis), or the pair $(\alpha, \mathcal{F})$ has been derived using the rule (2) or (4) of Definition 3.7.

"rule (2):" Then $\alpha \neq \varepsilon$ and there is $\mathcal{G}$ such that $(\alpha\mathcal{G}, \mathcal{F}) \in Cl^{i}(K)$ and $(\varepsilon, \mathcal{G}) \in K$. By induction hypothesis we know that for all $a \in \mathcal{A}$ and $C \in (\mathcal{P}(\Delta, F) \cup F)/\equiv_K$ we have that $Prob(p\alpha\mathcal{G}, a, C) = Prob(f, a, C)$. Since $\alpha \neq \varepsilon$, we have that $p\alpha\mathcal{G} \xrightarrow{a,x} q\beta\mathcal{G}$ iff $p\alpha \xrightarrow{a,x} q\beta$. Hence, it suffices to prove that if $p\alpha\mathcal{G} \xrightarrow{a,x} q\beta\mathcal{G}$ and $q\beta\mathcal{G} \in C$ for a given $C \in (\mathcal{P}(\Delta, F) \cup F)/\equiv_K$, then also $q\beta \in C$.

So, let $p\alpha\mathcal{G} \xrightarrow{a,x} q\beta\mathcal{G}$ where $q\beta\mathcal{G} \in C$. As $(p\alpha\mathcal{G}, f)$ p-expands in $\equiv_K$, there must be some $f \xrightarrow{a} g$ where $q\beta\mathcal{G} \equiv_K g$. By Lemma 4.17 there is some $h \in F$ such that $g \equiv_K h$ and $q\beta\mathcal{G} \in Gen_h(K)$. Since $(\varepsilon, \mathcal{G}) \in K$, we obtain $q\beta \in Gen_h(K)$ by Lemma 4.6 (1). Hence, $q\beta \equiv_K h \equiv_K g \equiv_K q\beta\mathcal{G}$.

"rule (4):" Then $\alpha = \beta X$ where $\beta \neq \varepsilon$ and there are $\mathcal{G}, \mathcal{H}$ such that $(\beta\mathcal{H}, \mathcal{F}) \in Cl^{i}(K)$ and $(X, \mathcal{H}) \in K$. The proof can be completed similarly as above.

- $(s, t) \notin R$. Since (s, t) belongs to the reflexive, symmetric, and transitive closure of R, there is a sequence $s = s_0, \cdots, s_k = t$, where $k \in \mathbb{N}_0$ and for each $0 \leq i < k$ we have that either $(s_i, s_{i+1}) \in R$ or $(s_{i+1}, s_i) \in R$. This means that (s_i, s_{i+1}) p-expands in $\equiv_K$ (see above), and hence we can readily confirm that also (s, t) p-expands in $\equiv_K$. □

Deciding the membership to $PExp(K)$ is easy—for example, to find out whether $(X, \mathcal{F}) \in PExp(K)$, it suffices to compute the $\equiv_K$ relation between the successors of pX and $\mathcal{F}(p)$ (for all $p \in Q$ such that $\mathcal{F}(p) \neq \bot$). Obviously, this can be done in time polynomial in m, n, z. Thus, we obtain the following:

Theorem 4.19. *The problem of probabilistic bisimilarity between PDA and finite-state processes is decidable in time polynomial in* m, n, z. *For PDA*^{k} *processes, the same problem is decidable in time polynomial in* m, n.

4.3 Simulation-Like Equivalences

Definition 4.20. *A binary relation* R *over processes is a* weak simulation *iff every pair* $(s,t) \in R$ *expands in* R *in the weak style (see Definition 4.7).*

Let s, t *be processes. We say that* t weakly simulates s*, written* $s \sqsubseteq t$*, if there is a weak simulation* R *such that* $(s,t) \in R$. *Further,* s, t *are* weakly simulation equivalent*, written* $s \sim t$*, if they simulate each other.*

Definition 4.21. *Let* $R \subseteq \mathcal{P}(\Delta, F) \times F$ *be a relation. We say that a pair* $(pw, f) \in R$ s-expands *in* R *iff the following two conditions are satisfied:*

- *for all* $a \in \mathcal{A}$ *and* $pw \xrightarrow{a} qv$ *there are* $\bar{g} \in F$ *and* $f \stackrel{a}{\Rightarrow} g$ *such that* $(qv, \bar{g}) \in R$ *and* $\bar{g} \sqsubseteq g$*;*
- *for all* $a \in \mathcal{A}$ *and* $f \xrightarrow{a} g$ *there are* $\bar{g} \in F$ *and* $pw \stackrel{a}{\Rightarrow} qv$ *such that* $(qv, \bar{g}) \in R$ *and* $g \sqsubseteq \bar{g}$.

We say that R *is a* $\prec$-simulation *iff every pair of* R *s-expands in* R.

The next lemma is a simple consequence of Definition 4.21.

Lemma 4.22. *The relation* $\precsim$ *over* $\mathcal{P}(\Delta, F) \times F$ *is a* $\prec$*-simulation.*

We can also prove the following:

Lemma 4.23. *Let* $pw \in \mathcal{P}(\Delta, F)$ *and* $f \in F$. *If* $(pw, f) \in R$ *for some* $\prec$*-simulation* R*, then* $pw \precsim f$.

Proof. Let R be a $\prec$-simulation. We show that for all $(pw, f) \in R$ we have that $pw \sim f$. From this and Definition 4.21 we obtain that $pw \precsim f$ as required. Let

$$R_1 = \{(pw, g) \mid \exists \bar{g} \in F \text{ such that } (pw, \bar{g}) \in R \text{ and } \bar{g} \sqsubseteq g\}$$

One can readily check that $R \subseteq R_1$ and that every pair of R_1 expands in R_1 in the weak style (see Definition 4.7). Similarly, we define

$$R_2 = \{(g, pw) \mid \exists \bar{g} \in F \text{ such that } (pw, \bar{g}) \in R \text{ and } g \sqsubseteq \bar{g}\}$$

Then $R^{-1} \subseteq R_2$ and every pair of R_2 expands in R_2 in the weak style. To sum up, if $(pw, f) \in R$, then $pw \sim f$. □

Definition 4.24. *Let* K *be a well-formed set, and let* $R = \bigcup_{f \in F} Gen_f(K) \times \{f\}$. *The set* $SExp(K)$ *consists of all pairs* $(w, \mathcal{F}) \in K$ *such that for each* $p \in Q$ *we have that if* $\mathcal{F}(p) \neq \bot$, *then the pair* $(pw, \mathcal{F}(p))$ *s-expands in* R.

SExp is clearly monotonic, and $SExp(\mathcal{B}) = \mathcal{B}$ due to Lemma 4.4 and Lemma 4.22. Hence, for every well-formed set K we have that if $\mathcal{B} \subseteq K$, then also $\mathcal{B} \subseteq SExp(K)$. Now we prove that if $K = SExp(K)$, then $K \subseteq \mathcal{B}$. From this we obtain the correctness of the algorithm of Fig. 1 (when we replace *Exp* with *SExp*).

Remark 4.25. *Similarly as in Section 4.1, we need to assume that* $\mathcal{T}$ *is* complete *in the following sense: For all* $f, g \in F$ *and* $a \in \mathcal{A}_\tau$ *we have that if* $f \stackrel{a}{\Rightarrow} g$, *then also* $f \stackrel{a}{\rightarrow} g$. *Again, this assumption is not restrictive because the missing transition can be added in polynomial time and each state of* F *stays equivalent to itself.*

Theorem 4.26. *Let* K *be a well-formed set. If* $K = SExp(K)$, *then* $K \subseteq \mathcal{B}$.

Proof. We show that if $K = SExp(K)$, then the relation $R = \bigcup_{f \in F} Gen_f(K) \times \{f\}$ is a $\prec$-simulation. That is, we prove that for all $h \in F$ and $pw \in Gen_h^i(K)$ the pair (pw, h) s-expands in R. We proceed by induction on i. (We present a detailed proof just for pairs of the form $p\alpha$; the other case follows similarly).

- $i = 0$. Since $p\alpha \in Gen_h^0(K)$, there is $\mathcal{F}$ such that $\mathcal{F}(p) = h$ and $(\alpha, \mathcal{F}) \in K$. Since $K = SExp(K)$, we have $(\alpha, \mathcal{F}) \in SExp(K)$, which means that $(p\alpha X, h)$ s-expands in R by Definition 4.24.
- *Induction step:* Let $p\alpha \in Gen_h^{i+1}(K)$. Then there must be some $\mathcal{F}$ such that $\mathcal{F}(p) \neq \bot$ and $(\alpha, \mathcal{F}) \in Cl^{i+1}(K)$ (see Definition 4.3). Hence, we either

have that $(\alpha, \mathcal{F}) \in Cl^{i}(K)$ (in which case it suffices to apply induction hypothesis), or the pair $(\alpha, \mathcal{F})$ can be derived from a pair of K and a pair of $Cl^{i}(K)$ using the rule (2) or (4) of Definition 3.7. We consider these two cases separately.

"**rule (2)**:" Then there is $\mathcal{G}$ such that $(\alpha\mathcal{G}, \mathcal{F}) \in Cl^{i}(K)$ and $(\varepsilon, \mathcal{G}) \in K$. We show that $(p\alpha, \mathcal{F}(p))$ s-expands in R.

(A) Let $p\alpha \xrightarrow{a} q\beta$. Then $p\alpha\mathcal{G} \xrightarrow{a} q\beta\mathcal{G}$ and since $(p\alpha\mathcal{G}, \mathcal{F}(p))$ s-expands in R by induction hypothesis, there are $\bar{g} \in F$ and $\mathcal{F}(p) \stackrel{a}{\Rightarrow} g$ such that $q\beta\mathcal{G} \in Gen_{\bar{g}}(K)$ and $\bar{g} \sqsubseteq g$. Since $q\beta \in Gen_{\bar{g}}(K)$ by Lemma 4.6 (1), we are done.

(B) Let $\mathcal{F}(p) \xrightarrow{a} g$. Since $(p\alpha\mathcal{G}, \mathcal{F}(p))$ s-expands in R by induction hypothesis, there are $\bar{g} \in F$ and $p\alpha\mathcal{G} \stackrel{a}{\Rightarrow} qw$ such that $qw \in Gen_{\bar{g}}(K)$ and $g \sqsubseteq \bar{g}$. We need to distinguish several cases.

(1) The responding move of $p\alpha\mathcal{G}$ is of the form $p\alpha\mathcal{G} \stackrel{a}{\Rightarrow} q\beta\mathcal{G}$ where the sequence $p\alpha \stackrel{a}{\Rightarrow} q\beta$ is performable. Since $q\beta\mathcal{G} \in Gen_{\bar{g}}(K)$, we obtain $q\beta \in Gen_{\bar{g}}(K)$ by Lemma 4.6 (1).

(2) The responding move of $p\alpha\mathcal{G}$ is of the form $p\alpha\mathcal{G} \stackrel{a}{\Rightarrow} q\mathcal{G} \stackrel{\tau}{\Rightarrow} q\mathcal{H}$ where the sequence $p\alpha \stackrel{a}{\Rightarrow} q\varepsilon$ is performable. Since $q\mathcal{H} \in Gen_{\bar{g}}(K)$, we have that $\mathcal{H}(q) \precsim \bar{g}$ by Lemma 4.6 (3), which means that $g \sqsubseteq \bar{g} \sqsubseteq \mathcal{H}(q)$. Further, as $q\mathcal{G} \stackrel{\tau}{\Rightarrow} q\mathcal{H}$, we also have $\mathcal{G}(q) \stackrel{\tau}{\Rightarrow} \mathcal{H}(q)$ and hence $\mathcal{G}(q) \xrightarrow{\tau} \mathcal{H}(q)$ because $\mathcal{T}$ is complete (see Remark 4.25). As $(\varepsilon, \mathcal{G}) \in K$, by induction hypothesis we know that $(q\varepsilon, \mathcal{G}(q))$ s-expands in R. The only possibility how $q\varepsilon$ can respond to $\mathcal{G}(q) \xrightarrow{\tau} \mathcal{H}(q)$ is to use the move $q\varepsilon \stackrel{\tau}{\Rightarrow} q\varepsilon$. Hence, there is $\bar{\bar{g}} \in F$ such that $q\varepsilon \in Gen_{\bar{\bar{g}}}(K)$ and $\mathcal{H}(q) \sqsubseteq \bar{\bar{g}}$. Since $g \sqsubseteq \bar{g} \sqsubseteq \mathcal{H}(q)$ (see above) and $\mathcal{H}(q) \sqsubseteq \bar{\bar{g}}$, we obtain $g \sqsubseteq \bar{\bar{g}}$. Hence, $p\alpha \stackrel{a}{\Rightarrow} q\varepsilon$ is an "appropriate" response to $\mathcal{F}(p) \xrightarrow{a} g$.

(3) The responding move of $p\alpha\mathcal{G}$ is of the form $p\alpha\mathcal{G} \stackrel{\tau}{\Rightarrow} q\mathcal{G} \stackrel{a}{\Rightarrow} q\mathcal{H}$ where the sequence $p\alpha \stackrel{\tau}{\Rightarrow} q\varepsilon$ is performable. We argue similarly as in (2), showing that $a = \tau$ in this case.

"rule (4):" Then there are $\mathcal{G}, \mathcal{H}$ such that $(\alpha\mathcal{H}, \mathcal{F}) \in Cl^{i}(K)$ and $(X, \mathcal{H}) \in K$. The proof can be completed along the same lines as above, using Lemma 4.6(2) instead of Lemma 4.6(1). □

The set $SExp(K)$ for a given K can be computed in time polynomial in m, n, z by using the same kind of technique as in Section 4.1. That is, we use Lemma 4.5 and Lemma 4.2 to check the required conditions symbolically. Thus, we obtain the following:

Theorem 4.27. *The problem of full simulation equivalence between PDA and finite-state processes is decidable in time polynomial in m, n, z. For PDA^k processes, the same problem is decidable in time polynomial in m, n.*

4.4 Trace-Like Equivalences

Let $w \in Act_\tau^*$ and s, t be processes. We write $s \stackrel{w}{\Rightarrow} t$ iff there is a sequence $s{=}s_0 \stackrel{w(1)}{\Longrightarrow} s_1 \stackrel{w(2)}{\Longrightarrow} \cdots \stackrel{w(n)}{\Longrightarrow} s_n{=}t$, where $w = w(1)w(2)\cdots w(n)$.

Definition 4.28. *For every process s we define the set*

$$Tr(s) = \{w \in Act^* \mid \textit{there is a } t \textit{ such that } s \stackrel{w}{\Rightarrow} t\}.$$

Processes s, t are (weakly) trace equivalent, *written $s \sim t$, iff $Tr(s) = Tr(t)$.*

From now on we assume that the system $\mathcal{T}$ is complete in the sense of Definition 4.25. This means that for all $f, g \in F$ and $w \in \mathcal{A}^*$ we have that $f \stackrel{w}{\Rightarrow} g$ iff $f \stackrel{w}{\rightarrow} g$.

Definition 4.29. *Let $R \subseteq \mathcal{P}(\Delta, F) \times F$ be a relation. For every $pw \in \mathcal{P}(\Delta, F)$ we define the set $\mathcal{M}_R(pw) = \{g \in F \mid (pw, g) \in R\}$.*

We say that a pair $(pw, f) \in R$ t-expands *in R iff the following two conditions are satisfied:*

- *for all $a \in \mathcal{A}$ and $pw \stackrel{a}{\rightarrow} qv$ there is a $\bar{g} \in F$ such that $(qv, \bar{g}) \in R$ and $Tr(\bar{g}) \subseteq \bigcup_{f \stackrel{a}{\rightarrow} g} Tr(g)$.*

• *for all* $a \in \mathcal{A}$ *and* $f \xrightarrow{a} g$ *we have that* $Tr(g) \subseteq \bigcup_{pw \stackrel{a}{\Rightarrow} qv} \bigcup_{\bar{g} \in \mathcal{M}_R(qv)} Tr(\bar{g})$.

We say that R *is a* $\prec$-trace similarity *iff every pair of* R *t-expands in* R.

Lemma 4.30. *The relation* $\precsim$ *over* $\mathcal{P}(\Delta, F) \times F$ *is a* $\prec$*-trace similarity.*

Proof. This follows immediatelly from Definition 4.29 and the assumption that $\mathcal{T}$ is complete. □

Lemma 4.31. *Let* $pw \in \mathcal{P}(\Delta, F)$ *and* $f \in F$. *If* $(pw, f) \in R$ *for some* $\prec$*-trace similarity* R, *then* $pw \precsim f$.

Proof. We show that for all $(pw, f) \in F$ and $u \in \mathcal{A}^*$ we have the following:

(1) If $pw \xrightarrow{u} qv$, then there is $g \in F$ such that $f \xrightarrow{u} g$.
(2) If $f \xrightarrow{u} g$, then there is $qv \in \mathcal{P}(\Delta, F)$ such that $pw \stackrel{u}{\Rightarrow} qv$.

We proceed by induction on the length of u. If $u = \varepsilon$, we are done immediatelly. Now let $u = ax$ where $a \in \mathcal{A}$.

"(1)": If $pw \stackrel{ax}{\Rightarrow} qv$, there is ry such that $pw \xrightarrow{a} ry \xrightarrow{x} qv$. Since R is a $\prec$-trace similarity, there is $\bar{g} \in F$ such that $(ry, \bar{g}) \in R$ and $Tr(\bar{g}) \subseteq \bigcup_{f \xrightarrow{a} g} Tr(g)$. Since $(ry, \bar{g}) \in R$ and $ry \xrightarrow{x} qv$, by induction hypothesis we get that $\bar{g} \xrightarrow{x} \bar{g}'$ for some $\bar{g}' \in F$. Hence, there is $g \in F$ such that $f \xrightarrow{a} g \xrightarrow{x} g'$ for some $g' \in F$ as required.

"(2)": If $f \xrightarrow{ax} g$, there is $g' \in F$ such that $f \xrightarrow{a} g' \xrightarrow{x} g$. Since R is a $\prec$-trace similarity, there is $pw \stackrel{a}{\Rightarrow} qv$ and $\bar{g} \in F$ such that $(qv, \bar{g}) \in R$ and $x \in Tr(\bar{g})$. Since $\mathcal{T}$ is complete, there is $\bar{g}' \in F$ such that $\bar{g} \xrightarrow{x} \bar{g}'$. As $(qv, \bar{g}) \in R$ and $\bar{g} \xrightarrow{x} \bar{g}'$, by induction hypothesis we get that $qv \stackrel{x}{\Rightarrow} ry$ for some $ry \in \mathcal{P}(\Delta, F)$. Hence, $pw \stackrel{ax}{\Rightarrow} ry$ as needed. □

Definition 4.32. *Let* K *be a well-formed set, and let* $R = \bigcup_{f \in F} Gen_f(K) \times \{f\}$. *The set* $TExp(K)$ *consists of all pairs* $(w, \mathcal{F}) \in K$ *such that for each* $p \in Q$ *we have that if* $\mathcal{F}(p) \neq \bot$, *then the pair* $(pw, \mathcal{F}(p))$ *t-expands in* R.

Theorem 4.33. *Let* K *be a well-formed set. If* $K = TExp(K)$, *then* $K \subseteq \mathcal{B}$.

Proof. The proof has the same structure and is based on similar observations as proofs of Theorem 4.13 and Theorem 4.26. □

The complexity issues are different from the ones of Section 4.1 and Section 4.3. The problem of checking trace equivalence / inclusion over finite-state processes in **PSPACE**-complete. Hence, even computing the largest well-formed set takes time exponential in m, n. Further, deciding whether a given pair $(X, \mathcal{F})$ of K belongs to $TExp(K)$ also takes time exponential in m, n (the same holds for pairs of the form $(X\mathcal{G}, \mathcal{F})$). So, the whole algorithm for computing $\mathcal{B}$ takes $\mathcal{O}(2^{pol(m,n)})$ time where *pol* is a polynomial in two variables. Thus, we obtain the following:

Theorem 4.34. *The problem of full trace equivalence between PDA and finite-state processes is decidable in time exponential in* m, n. *Moreover, this problem is* ***PSPACE****-hard even for BPA.*

References

[AEM04] R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Proceedings of TACAS 2004*, volume 2988 of *Lecture Notes in Computer Science*, pages 467–481. Springer, 2004.

[AEY01] R. Alur, K. Etessami, and M. Yannakakis. Analysis of recursive state machines. In *Proceedings of CAV 2001*, volume 2102 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 2001.

[BBK93] 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.

[BCMS01] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 545–623. Elsevier, 2001.

[BCS95] 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.

[BEM97] 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.

[Bou01] A. Bouajjani. Languages, rewriting systems, and verification of infinite-state systems. In *Proceedings of ICALP 2001*, volume 2076 of *Lecture Notes in Computer Science*, pages 24–39. Springer, 2001.

[BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Number 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[Cau90] D. Caucal. Graphes canoniques des graphes algébriques. *Informatique Théorique et Applications (RAIRO)*, 24(4):339–352, 1990.

[CHS95] S. Christensen, H. Hüttel, and C. Stirling. Bisimulation equivalence is decidable for all context-free processes. *Information and Computation*, 121:143–148, 1995.

[EHRS00] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceed-*

ings of CAV 2000, volume 1855 of *Lecture Notes in Computer Science*, pages 232–247. Springer, 2000.

[EK99] 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.

[EKS03] J. Esparza, A. Kučera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. *Information and Computation*, 186(2):355–376, 2003.

[ES01] J. Esparza and S. Schwoon. A BDD-based model checker for recursive programs. In *Proceedings of CAV 2001*, volume 2102 of *Lecture Notes in Computer Science*, pages 324–336. Springer, 2001.

[Esp97] J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.

[Fri76] E.P. Friedman. The inclusion problem for simple languages. *Theoretical Computer Science*, 1(4):297–316, 1976.

[Gro92] J.F. Groote. A short proof of the decidability of bisimulation for normed BPA processes. *Information Processing Letters*, 42:167–171, 1992.

[HJM96] 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.

[HS98] H. Hüttel and C. Stirling. Actions speak louder than words: Proving bisimilarity for context-free processes. *Journal of Logic and Computation*, 8(4):485–509, 1998.

[JM99] P. Jančar and F. Moller. Techniques for decidability and undecidability of bisimilarity. In *Proceedings of CONCUR'99*, volume 1664 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 1999.

[KE03] A. Kučera and J. Esparza. A logical viewpoint on process-algebraic quotients. *Journal of Logic and Computation*, 13(6):863–880, 2003.

[KJ02] A. Kučera and P. Jančar. Equivalence-checking with infinite-state systems: Techniques and results. In *Proceedings of SOFSEM'2002*, volume 2540 of *Lecture Notes in Computer Science*. Springer, 2002.

[KM02a] A. Kučera and R. Mayr. On the complexity of semantic equivalences for pushdown automata and BPA. In *Proceedings of MFCS 2002*, volume 2420 of *Lecture Notes in Computer Science*, pages 433–445. Springer, 2002.

[KM02b] A. Kučera and R. Mayr. Weak bisimilarity between finite-state systems and BPA or normed BPP is decidable in polynomial time. *Theoretical Computer Science*, 270(1–2):677–700, 2002.

[KM04] A. Kučera and R. Mayr. A generic framework for checking semantic equivalences between pushdown automata and finite-state automata. Technical report FIMU-RS-2004-01, Faculty of Informatics, Masaryk University, 2004.

[KS] A. Kučera and Ph. Schnoebelen. A general approach to comparing infinite-state systems with their finite-state specifications. Submitted.

[Kuč99] A. Kučera. On finite representations of infinite-state behaviours. *Information Processing Letters*, 70(1):23–30, 1999.

[LS91] K. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

[May03] R. Mayr. Undecidability of weak bisimulation equivalence for 1-counter processes. In *Proceedings of ICALP 2003*, volume 2719 of *Lecture Notes in Computer Science*, pages 570–583. Springer, 2003.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Mol96] F. Moller. Infinite results. In *Proceedings of CONCUR'96*, volume 1119 of *Lecture Notes in Computer Science*, pages 195–216. Springer, 1996.

[Par81] 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.

[Sén98] G. Sénizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree. In *Proceedings of FOCS'98*, pages 120–129. IEEE Computer Society Press, 1998.

[Srb02a] J. Srba. Roadmap of infinite results. *EATCS Bulletin*, (78):163–175, 2002.

[Srb02b] J. Srba. Strong bisimilarity and regularity of basic process algebra is PSPACE-hard. In *Proceedings of ICALP 2002*, volume 2380 of *Lecture Notes in Computer Science*, pages 716–727. Springer, 2002.

[Srb02c] J. Srba. Undecidability of weak bisimilarity for pushdown processes. In *Proceedings of CONCUR 2002*, volume 2421 of *Lecture Notes in Computer Science*, pages 579–593. Springer, 2002.

[Sti98a] C. Stirling. Decidability of bisimulation equivalence for normed pushdown processes. *Theoretical Computer Science*, 195:113–131, 1998.

[Sti98b] C. Stirling. The joys of bisimulation. In *Proceedings of MFCS'98*, volume 1450 of *Lecture Notes in Computer Science*, pages 142–151. Springer, 1998.

[Tho93] W. Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science. In *Proceedings of TAPSOFT'93*, volume 668 of *Lecture Notes in Computer Science*, pages 559–568. Springer, 1993.

[vG93] R.J. van Glabbeek. The linear time—branching time spectrum II: The semantics of sequential systems with silent moves. In *Proceedings of CONCUR'93*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1993.

[vG94] R. van Glabbeek. What is branching time semantics and why to use it? *EATCS Bulletin*, (53):191–198, 1994.

[vG99] R. van Glabbeek. The linear time—branching time spectrum. *Handbook of Process Algebra*, pages 3–99, 1999.

[vGSST90] R. van Glabbeek, A. Smolka, B. Steffen, and C. Tofts. Reactive, generative, and stratified models for probabilistic processes. In *Proceedings of LICS'90*, pages 130–141. IEEE Computer Society Press, 1990.

[vGW96] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the Association for Computing Machinery*, 43(3):555–600, 1996.

Publications in the FI MU Report Series are in general accessible via WWW and anonymous FTP:

`http://www.fi.muni.cz/informatics/reports/`
`ftp ftp.fi.muni.cz (cd pub/reports)`

Copies may be also obtained by contacting:

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**