A GENERIC FRAMEWORK FOR CHECKING SEMANTIC EQUIVALENCES BETWEEN PUSHDOWN AUTOMATA AND FINITE-STATE AUTOMATA

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

Richard Mayr[†]
Department of Computer Science,
Albert-Ludwigs-University Freiburg
Georges-Koehler-Allee 51,
D-79110 Freiburg, Germany.
mayrri@informatik.uni-freiburg.de

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.

Keywords: Formal verification; Pushdown automata; Semantic equivalences;

[*]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.
[†]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., [1, 2, 13, 15, 14]). 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 achieved during the last decade (see [30, 11, 20, 5, 23, 7, 33] 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 [40, 42].

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 [3] 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 [9, 17, 19], and there is even a polynomial-time algorithm [18]. The decidability result has been extended to all (not necessarily normed) BPA in [10], and an elementary upper complexity bound is due to [8]. Recently, **PSPACE**-hardness of this problem has been established in [34]. Strong bisimilarity was shown to be decidable also for

[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.

[2]This is because stateless PDA correspond to a natural fragment of ACP known as "BPA" (Basic Process Algebra; see [4]). BPA cannot model global data, but they are sufficiently powerful to model, e.g., the interprocedural data-flow [13]. 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.

normed PDA [36]. Later, Sénizergues proved that bisimilarity is decidable for all PDA processes [32]. For simulation-like and trace-like equivalences, the equivalence-checking problem is undecidable even for (normed) BPA; this follows directly from Friedman's result [16]. In the presence of silent moves, the situation gets even worse. Weak bisimilarity is undecidable for PDA [35], and in fact for a very modest subclass of PDA known as one-counter nets [28].

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 [25]. For general pushdown automata, both problems are **PSPACE**-complete [24]. Checking strong and weak simulation equivalence between a BPA and a finite-state system is **EXPTIME**-complete [24], 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 4). 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 explicitly 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 [43] 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 [39], similarly as strong bisimilarity [31] 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 [29] is substantially more developed in this sense. One reason is that weak bisimilarity admits a simple game-theoretic characterization [37, 38] 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 [27, 41] 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 [24], 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 existing process equivalences are preserved under their respective quotients [21, 22], 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 [9], and can also be seen as a generalization of the method used in [25] 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.

Due to the lack of space, we had to omit all proofs and also the parts devoted to probabilistic bisimilarity, simulation-like equivalences, and trace-like equivalences. These can be found in a full version of this paper [26].

2 Basic Definitions

DEFINITION 1 *A* transition system *is a triple $\mathcal{T} = (S, \to, \mathcal{A})$ where S is a finite or countably infinite set of* states*, $\mathcal{A}$ is a finite set of* actions*, and $\to \subseteq S \times \mathcal{A} \times S$ is a* transition relation.

We write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \to$, 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 $s \to^* t$, 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 $\overset{a}{\Rightarrow} \subseteq S \times S$ as follows:

- $s \overset{\tau}{\Rightarrow} t$ iff there is a sequence of the form $s = p_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_k = t$ where $k \geq 0$;
- $s \overset{a}{\Rightarrow} t$ where $a \neq \tau$ iff there are p, q such that $s \overset{\tau}{\Rightarrow} p \xrightarrow{a} q \overset{\tau}{\Rightarrow} t$.

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 *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 $\delta : (Q \times \Gamma) \to 2^{\mathcal{A} \times Q \times \Gamma^{\leq 2}}$ 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 $pX \xrightarrow{a} q\beta \in \delta$ 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 $pX\alpha \xrightarrow{a} q\beta\alpha$ iff $pX \xrightarrow{a} q\beta \in \delta$.

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 $\mathcal{T} = (F, \mathcal{A}, \to)$. The symbol $F_\perp$ denotes the set $F \cup \{\perp\}$, where $\perp \notin F$ stands for "undefined".

DEFINITION 3 *For every process $p\alpha$ of Δ we define the set $M_{p\alpha} = \{q \in Q \mid p\alpha \to^* q\varepsilon\}$. A function $\mathcal{F} : Q \to F_\perp$ 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:

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

Here $\mathcal{F}[f/p] : Q \to F_\perp$ 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 $p\alpha \to^* q\beta$, then $\mathcal{F} \in \mathcal{C}(q\beta)$. We put $Stack(\Delta, F) = \Gamma^* \cup \{p\alpha\mathcal{F} \mid p \in Q, \alpha \in \Gamma^*, \mathcal{F} \in (F_\perp)^Q\}$, 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 4 *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 $w, v \in Stack(\Delta, F)$ 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 5 *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 $p\alpha \to^* q\beta$ 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 6 *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 $\mathcal{H} : Q \to F_\perp$. 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 $p\alpha\gamma\mathcal{H} \precsim f$.

DEFINITION 7 *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 8 *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 9 *Let* K *be a well-formed set. The* closure of K*, denoted* $\mathit{Cl}(K)$*, 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 $\mathit{Cl}(K) = \bigcup_{i=0}^{\infty} \mathit{Cl}^i(K)$ where $\mathit{Cl}^0(K) = K$ and $\mathit{Cl}^{i+1}(K)$ consists of exactly those pairs which are either in $\mathit{Cl}^i(K)$ or can be derived from K and $\mathit{Cl}^i(K)$ by applying one of the rules (1)–(5) of Definition 9. Another simple observation (which will be useful later) is the following:

LEMMA 10 *Let* K *be a well-formed set, and let* $(\mathcal{F},\mathcal{H}) \in K$*. If* $(\alpha,\mathcal{F}) \in \mathit{Cl}^i(K)$*, then also* $(\alpha,\mathcal{H}) \in \mathit{Cl}^i(K)$*. Similarly, if* $(\alpha\mathcal{G},\mathcal{F}) \in \mathit{Cl}^i(K)$*, then also* $(\alpha\mathcal{G},\mathcal{H}) \in \mathit{Cl}^i(K)$*.*

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

DEFINITION 11 *The* base $\mathcal{B}$ *is defined as follows:*

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

THEOREM 12 *Let* $\alpha \in \Gamma^*$ *and* $\mathcal{F},\mathcal{G} : Q \to F_\perp$*. We have*

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

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 [26]). 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:   $K := \mathcal{B}$; $\mathcal{B} := \emptyset$
4:   **for all** $(w, \mathcal{F}) \in K$ **do**
5:       **if** $(w, \mathcal{F}) \in Exp(K)$ **then** $\mathcal{B} := \mathcal{B} \cup \{(w, \mathcal{F})\}$ **fi**
6:   **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 [40, 42]. 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

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

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 [6]:

DEFINITION 13 *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 [12].

LEMMA 14 *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 15 *Let* K *be a well-formed set. For all* $f \in F$ *and* $i \in \mathbf{N}_0$ *we define the set* $Gen^i_f(K) =$

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

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

LEMMA 16 *The relation* $\precsim$ *over* $\mathcal{P}(\Delta, F) \times F$ *is exactly* $\bigcup_{f \in F} Gen_f(\mathcal{B}) \times \{f\}$.

LEMMA 17 *Let* K *be a well-formed set and* $f \in F$. *The set* $Gen_f(K)$ *is recognized by a multi-automaton* $\mathcal{M}_{K,f}$ *which is constructible in time polynomial in* m, n, z.

Proof: We refer to [25] where a similar result is proven explicitly; the construction required for Lemma 17 differs from the one presented in [25] only in minor details. □

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

LEMMA 18 *Let* K *be a well-formed set. The following conditions hold:*

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

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 19 *Let R be a binary relation over processes, and let $(s,t) \in R$. We say that a move $t \overset{a}{\Rightarrow} t'$ is R*-consistent with a move $s \xrightarrow{a} s'$ 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 $t \overset{a}{\Rightarrow} t'$ is of the form $t{=}u_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} u_i \xrightarrow{a} v_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} v_j{=}t'$, 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 $s \xrightarrow{a} s'$ there is a move $t \overset{a}{\Rightarrow} t'$ which is R-consistent with $s \xrightarrow{a} s'$. 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 20 *An important fact (which will be used in the proof of Lemma 23) is that the* same *notion of weak, early, delay, and branching bisimilarity is obtained when the conditions* (i) *and* (ii) *of Definition 19 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 19 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 19, we need to assume that the transition relation of $\mathcal{T}$ is "complete" in the following sense:

DEFINITION 21 *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 $f \overset{a}{\Rightarrow} f'$ move which is $\sim_F$-consistent with a* hypothetical *transition $f \xrightarrow{a} f'$ (note that the condition of $\sim_F$-consistency with $f \xrightarrow{a} f'$ makes a clear sense even if $f \xrightarrow{a} f'$ is not a transition of $\mathcal{T}$), then $f \xrightarrow{a} f'$ 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 "$\overset{a}{\Rightarrow}$" moves of f; it suffices to consider the "$\xrightarrow{a}$" ones.

DEFINITION 22 *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 $pw \overset{a}{\Rightarrow} qv$ such that $(qv, g) \in R$. Moreover, we require that if pw is of the form $p\alpha\mathcal{G}$, then the move $p\alpha\mathcal{G} \overset{a}{\Rightarrow} qv$ 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 20 and the assumption that $\mathcal{T}$ is complete):

LEMMA 23 *The relation $\sim$ restricted to $\mathcal{P}(\Delta, F) \times F$ is a quasi-bisimulation.*

DEFINITION 24 *Let K be a well-formed set, and let $R = \bigcup_{f \in F} Gen_f(K) \times \{f\}$. The set $BExp(K)$ 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 16 and Lemma 23 we obtain $BExp(\mathcal{B}) = \mathcal{B}$. Now we prove that if $K = BExp(K)$ 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 25 *Let K be a well-formed set. If $K = BExp(K)$, then $K \subseteq \mathcal{B}$.*

Now we show how to decide the membership to $BExp(K)$. At the same time, we perform a (rough) complexity analysis. Pairs of the form $(\mathcal{G}, \mathcal{F})$ and $(\varepsilon, \mathcal{F})$ belong to $BExp(K)$ 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 24 we have that $(X, \mathcal{F}) \in BExp(K)$ 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 17), 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 17.
  - there is a sequence $pX \overset{\tau}{\Rightarrow} q\alpha \xrightarrow{a} r\beta \overset{\tau}{\Rightarrow} s\gamma$ 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 17 and Lemma 14, 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 26 *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).*

References

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

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

[3] 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.

[4] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. No. 18 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

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

[6] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: application to model checking. In *Proceedings of CONCUR'97*, vol. 1243 of *Lecture Notes in Computer Science*, pp. 135–150. Springer, 1997.

[7] 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*, pp. 545–623. Elsevier, 2001.

[8] O. Burkart, D. Caucal, and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proceedings of MFCS'95*, vol. 969 of *Lecture Notes in Computer Science*, pp. 423–433. Springer, 1995.

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

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

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

[12] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of CAV 2000*, vol. 1855 of *Lecture Notes in Computer Science*, pp. 232–247. Springer, 2000.

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

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

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

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

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

[18] 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.

[19] 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.

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

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

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

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

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

[25] 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.

[26] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[43] 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.