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

Antonín Kučera[a], Richard Mayr[b]

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

[b] *University of Edinburgh, School of Informatics, LFCS, 10 Crichton Street, Edinburgh EH8 9AB, UK.*

Abstract

For a given process equivalence, we say that a process g is *fully equivalent* to a process f of a transition system $\mathcal{T}$ if g is equivalent to f and every reachable state of g is equivalent to some state of $\mathcal{T}$. We propose a generic method for deciding full equivalence between pushdown processes and finite-state processes applicable to every process equivalence satisfying certain abstract conditions. Then, we show that these conditions are satisfied by bisimulation-like equivalences (including weak and branching bisimilarity), weak simulation equivalence, and weak trace equivalence, which are the main conceptual representatives of the linear/branching time spectrum. The list of particular results obtained by applying our method includes items which are first of their kind, and the associated upper complexity bounds are essentially optimal.

Key words: Pushdown automata, semantic equivalences, bisimulation

1. Introduction

One of the main paradigms in formal verification is *equivalence-checking*, where the correctness of a given *implementation* is demonstrated by proving semantic equivalence with its intended behavior called the *specification*. Formally, the implementation and the specification are understood as *processes*, i.e., states in labeled transition systems, and the semantic equivalence is some equivalence over the class of all processes. Equivalence proofs are often long and tedious, especially when the implementation uses unbounded data structures such as counters, stacks, or queues making the state space infinite. A natural question is

whether such proofs can be produced automatically, i.e., whether a given process equivalence is decidable in a given class of processes, and what is the associated complexity. The *equivalence-checking* problem has been considered for various process equivalences and various classes of infinite-state processes in the last decades; we refer to, e.g., [49, 18, 37, 9, 43, 11, 53] for surveys of some subfields.

A special variant of the equivalence-checking problem is *regular equivalence-checking*, where the specification is a finite-state process. Hence, an instance of the regular equivalence-checking problem is a process g of a (possibly infinite-state) transition system $\mathcal{U}$, and a process f of a finite-state transition system $\mathcal{T}$. The question is whether g and f are equivalent for some fixed process equivalence. In general, the process g may reach states that are not equivalent to any state of $\mathcal{T}$, i.e., the system $\mathcal{T}$ does not necessarily characterize the *state space* of g up to the chosen equivalence. This motivates the problem of *full regular equivalence-checking*, where we require that g and f are *fully equivalent*, i.e., they are equivalent and each state reachable from g is equivalent to some state of $\mathcal{T}$. The concept of full equivalence was introduced in [40] and studied in [47], where it was shown that, for a large class of process equivalences, the problem of full regular equivalence-checking is reducible to the model-checking problem with a slightly extended version of the branching-time logic EF.

In this paper, we restrict our attention to implementations definable by *pushdown automata (PDA)*, a widely accepted model[1] for sequential programs with recursive procedure calls (see, e.g., [2, 3, 20, 22, 21]). The operational behavior of a given PDA Δ is formally defined by the associated transition system $\mathcal{T}_\Delta$, where the states are the configurations of Δ and the transitions are determined by the rules of Δ in the natural way (see Section 2). Hence, $\mathcal{T}_\Delta$ has infinitely many states. We use PDA^k to denote the subclass of PDA processes where the underlying pushdown automaton has at most k control states. Due to historical reasons, we also refer to PDA^1 processes as BPA processes.[2]

[1] From the language-theoretic point of view, the definition of PDA adopted in this paper corresponds to the subclass of real-time PDA. The concept of ε-transitions is replaced by "silent" transitions with a distinguished label τ which may (but do not have to) be treated in a special way by a given semantic equivalence.

[2] The "BPA" acronym stands for Basic Process Algebra, a natural fragment of ACP [5]. BPA algebra is expressively equivalent (up to strong bisimilarity) to PDA processes with one control state.

1.1. Our contribution

We give a generic algorithm for the full regular equivalence-checking problem where the implementation (i.e., the process g) is a PDA process. The algorithm is applicable to every process equivalence satisfying certain abstract criteria, and we show that these criteria are met by bisimulation-like equivalences (incl. weak, early, delay, and branching bisimilarity), weak simulation equivalence, and weak trace equivalence. These equivalences are the main conceptual representatives of the linear/branching time spectrum [57, 58], and the applicability of the presented algorithm extends to many (if not all) equivalences in this spectrum by modifying the techniques used for the aforementioned representatives. For PDA^k processes, where $k \geq 1$ is a fixed constant, the obtained algorithms are essentially *optimal*.

More specifically, we show that, given a PDA Δ and a finite-state system $\mathcal{T}$, the full equivalence between the processes of Δ and $\mathcal{T}$ is representable by a finite relation $\mathcal{B}$ called *base*. All pairs of fully equivalent processes can be generated from $\mathcal{B}$ by applying simple substitution rules assuming that the chosen process equivalence is a *right PDA congruence* (see Definition 5). Then, we show how to compute the base $\mathcal{B}$ as the greatest fixed-point of a certain monotonic function. This monotonic function depends on another function called *expansion* which must be tailored specifically for each process equivalence so that the criteria of Definition 12 are satisfied. Finally, we show how to design an appropriate expansion for the concrete process equivalences mentioned above. The list of particular results obtained in this way includes the following:

(a) Branching bisimilarity [59] between PDA^k and finite-state processes is decidable in polynomial time. To the best of our knowledge, this is the first result about computational tractability of branching bisimilarity for systems with infinitely many states (the same actually applies to early and delay bisimilarity). Branching bisimilarity plays a distinguished role in the semantics of systems with silent moves [56], similarly as strong bisimilarity [50] for processes without silent moves.

(b) For weak simulation equivalence, we prove that full equivalence between PDA^k and finite-state processes is decidable in polynomial time. Since checking (non-full) weak simulation equivalence between PDA^k and finite-state processes is **EXPTIME**-complete even for BPA [46], this result shows that full regular equivalence-checking can be more tractable than "ordinary" regular equivalence-checking.

(c) For weak trace equivalence, we show that full equivalence between PDA^k and finite-state processes is decidable in polynomial space, and the problem is **PSPACE**-hard even for BPA. Since checking (weak) trace equivalence between

BPA and finite-state processes is undecidable, we see that full regular equivalence-checking can be even "more decidable" than regular equivalence-checking.

Another generic outcome of our method is an algorithm deciding whether a given finite-state process is the $\sim$-quotient of a given PDA process for a given semantic equivalence $\sim$. Here we need to assume that $\sim$ is *preserved under quotients* (see Definition 18) which is not really restrictive because most of the existing process equivalences satisfy this property [40, 42].

1.2. Related work

Language equivalence is undecidable for general nondeterministic PDA and BPA [30]. However, for the deterministic subclass (dPDA), language equivalence is decidable [51] (see also [55, 34]). The computational complexity of this problem is open and no nontrivial lower bound is known. For the subclass of deterministic one-counter automata, language equivalence-checking is **NL**-complete [8].

Checking bisimulation equivalence is decidable for PDA processes [52]. A nonelementary lower bound has been shown in [6] (see also [35]), improving the previous **EXPTIME** lower bound of [46]; the exact complexity is still open. However, bisimilarity is known to be **PSPACE**-complete for the subclass of one-counter automata [7]. In the context of bisimilarity-checking, a special attention has been devoted BPA which are strictly less expressive than PDA w.r.t. bisimulation-like equivalences. The first positive result is due to Baeten, Bergstra, and Klop [4] who proved the decidability of strong bisimilarity for *normed* BPA (a PDA is normed if the stack can be emptied from every reachable configuration). Simpler proofs were given later in [14, 26, 32], and there is even a polynomial-time algorithm [28]. The decidability result was extended to all (not necessarily normed) BPA in [16], and a 2-**EXPTIME** upper complexity bound is due to [12, 33]. An **EXPTIME** lower complexity bound for BPA bisimilarity was shown in [39].

In the presence of silent τ-moves, the equivalence-checking problems become harder. Weak bisimilarity is undecidable for PDA [54], and in fact even for a very modest subclass of PDA known as one-counter nets[3] [48] . The decidability of weak bisimilarity for BPA is open. However, it is known that branching bisimilarity is decidable for normed BPA [24], which extends the previous results

[3]One-counter nets are a subclass of one-counter automata where the counter cannot be tested for zero explicitly; one-counter nets are expressively equivalent to Petri nets with at most one unbounded place.

about totally normed BPA [31, 15]. The best known upper complexity bound for checking branching bisimilarity for normed BPA processes is **EXPTIME** [27] (a **NEXPTIME** upper bound is due to [17]). It is open whether the decidability result can be extended to all BPA processes, but it is already clear that such extensions cannot go much beyond the BPA class [60].

For simulation-like and trace-like equivalences, the equivalence-checking problem is undecidable even for (normed) BPA; this follows directly from Friedman's result about the undecidability of language inclusion for simple grammars [23]. Simulation equivalence is decidable for one-counter nets [1] (see also [38, 29]), and the relationship between strong bisimilarity and simulation equivalence over one-counter nets was studied in [36].

Regular equivalence-checking for PDA processes is computationally easier. Strong and weak bisimilarity between PDA and finite-state processes is **PSPACE**-complete [46]. The complexity is lower for the subclass of one-counter automata; strong bisimilarity between processes of one-counter automata and finite-state processes is decidable in polynomial time [41], while checking weak bisimilarity is $\mathbf{P}^{\mathbf{NP}}$-complete [25]. Strong and weak bisimilarity between BPA and finite-state processes is decidable in polynomial time [45]. Checking strong and weak simulation equivalence between BPA and finite-state processes is **EXPTIME**-complete, and the same holds for general PDA [46, 44]. Trace-like equivalences between BPA and finite-state processes are undecidable (this is a direct consequence of the undecidability of the universality problem for context-free languages [30]).

Our results subsume and generalize the method used in [45] to decide weak bisimilarity between BPA and finite-state processes in polynomial time. The presence of control states in PDA makes the (de)composition technique more intricate, which is overcome by utilizing partial functions associating processes to control states (see Definitions 7 and 8). Further, the technique is no longer limited to weak bisimilarity, but applicable to an arbitrary right PDA congruence satisfying the abstract conditions formulated in Definition 12. This is achieved by abstracting the equivalence-specific part into the notion of expansion, and inventing new techniques applicable to simulation-like and trace-like equivalences.

Both [45] and the work presented in this paper build on the idea of finite bisimulation bases pioneered by Caucal [14]. Finite bisimulation bases exist for processes of BPA and BPP systems. For the *normed* subclasses of BPA and BPP, bisimulation bases are even computable in polynomial time. We refer to [11] for a detailed exposition of these results. Bisimulation bases use simple congruence rules to generate new pairs of bisimilar processes, and they are usually computed iteratively by "cleaning" a sufficiently large relation subsuming the base. The

same principle is used in [45] and the presented work, although the base and the way of generating new pairs of processes are technically different. Since the scope of our work includes equivalences with silent τ-moves, we have to deal with infinite sets of configurations reachable by sequences of silent moves of unbounded length, which complicates the "cleaning" phase. This difficulty is overcome by representing the relevant infinite sets of configurations by efficiently constructible finite-state automata. New insights are also required to handle the restrictions on intermediate states visited by silent moves imposed by early, delay, and branching bisimilarity. Simulation-like and trace-like equivalences need specific treatment which has no direct counterpart in previous works.

The paper is organized as follows. In Section 2, we recall transition systems, process equivalences, PDA systems, and the concept of full regular equivalence [40, 47]. In Section 3, we introduce a finite semantic base representing full equivalence between processes of a given PDA system and a given finite-state system. Finally, in Section 4 we show how to compute the base $\mathcal{B}$.

2. Basic Definitions

In this paper, processes are understood as states in transition systems.

Definition 1. *A* transition system *is a triple* $\mathcal{T} = (S, \mathcal{A}, \rightarrow)$ *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*. A* process *is a pair* $(\mathcal{T}, s)$*, where* $\mathcal{T}$ *is a transition system and* s *is a state of* $\mathcal{T}$*.*

When $\mathcal{T}$ is clearly determined by the context, we write just s instead of $(\mathcal{T}, s)$. Further, 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 $s \xrightarrow{w} t$. A *state space* of a process $(\mathcal{T}, s)$ is the set of all states reachable from s in $\mathcal{T}$. Two processes s, t are *isomorphic* if there is a one-to-one correspondence h between their state spaces such that $h(s) = t$ and for all states u, u' reachable from s we have that $u \xrightarrow{a} u'$ iff $h(u) \xrightarrow{a} h(u')$.

We assume that $\mathcal{A}$ always contains a special *silent action* denoted by τ. Intuitively, τ-labeled transitions model "internal" computational steps that are not directly visible to an external observer. For all $s, t \in S$, a "$\stackrel{\tau}{\Rightarrow}$" *move* from s to t is a sequence of transitions $s{=}u_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} u_i{=}t$ where $i \geq 0$, and a "$\stackrel{a}{\Rightarrow}$" move from s to t, where $a \neq \tau$, is a sequence of transitions $s{=}u_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} u_i \xrightarrow{a} v_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} v_j{=}t$

where $i, j \geq 0$. A "$\stackrel{a}{\Rightarrow}$" move from s to t, where $a \in \mathcal{A}$, is usually denoted by $s \stackrel{a}{\Rightarrow} t$. Note that there may exist infinitely many "$\stackrel{a}{\Rightarrow}$" moves from s to t, and if we needed to consider two different "$\stackrel{a}{\Rightarrow}$" moves from s to t at the same time, our notation would be insufficient (we could not denote both of these moves by $s \stackrel{a}{\Rightarrow} t$). Fortunately, there is no such need in this paper. When no confusion arises, we slightly abuse our notation and write $s \stackrel{a}{\Rightarrow} t$ to indicate the *existence* of a "$\stackrel{a}{\Rightarrow}$" move from s to t.

A *process equivalence* is an equivalence over the class[4] of all processes. In general, a process equivalence $\sim$ does not fully characterize the state space of a given process up to $\sim$. That is, even if $s \sim t$, a state reachable from s is not necessarily $\sim$-equivalent to any state reachable from t. This motivates the following definition.

Definition 2. *Let $\sim$ be a process equivalence and $(\mathcal{T}_1, s_1)$, $(\mathcal{T}_2, s_2)$ processes. We say that $(\mathcal{T}_1, s_1)$ is* fully equivalent *to $(\mathcal{T}_2, s_2)$, denoted by $(\mathcal{T}_1, s_1) \precsim (\mathcal{T}_2, s_2)$, if $s_1 \sim s_2$ and for every s_1' reachable from s_1 there is a state s_2' of $\mathcal{T}_2$ (not necessarily reachable from s_2) such that $s_1' \sim s_2'$.*

Remark 3. *Observe that if $(\mathcal{T}_1, s_1) \precsim (\mathcal{T}_2, s_2)$ and $s_1 \rightarrow^* s_1'$, then for every state s_2' of $\mathcal{T}_2$ such that $s_1' \sim s_2'$ we also have that $s_1' \precsim s_2'$.*

In this paper, we are interested in the problem of checking full equivalence with a finite-state process, formalized as follows:

Problem: Full regular equivalence-checking.
Instance: A process $(\mathcal{U}, g)$ and a finite-state process $(\mathcal{T}, f)$.
Question: Do we have $g \precsim f$?

Here, we assume some finite encoding of $(\mathcal{U}, g)$. In our setting, $(\mathcal{U}, g)$ will always be a pushdown process.

The concept of full regular equivalence was introduced and studied in [47]. For bisimulation-like equivalences, $g \sim f$ implies $g \precsim f$, because bisimilar processes have the same state space up to bisimilarity. However, for simulation-like and trace-like equivalences, this implication does not hold. As a simple example

[4] In our setting, we can safely assume that the class of all processes is actually a set; formally, this is achieved by fixing two sets *States* and *Actions* such that every transition system $\mathcal{T} = (S, \mathcal{A}, \rightarrow)$ satisfies $S \subseteq$ *States* and $\mathcal{A} \subseteq$ *Actions*.

illustrating the difference, consider a finite-state system $\mathcal{T}$ with just one state f where $f \xrightarrow{a} f$. Further, let $\sim$ be the trace equivalence[5]. Then,

- $g \sim f$ iff g can perform an arbitrarily long finite sequence of a's and no other action;
- $g \precsim f$ iff every state reachable from g can perform an arbitrarily long finite sequence of a's and no other action.

Obviously, the second condition is stronger and, in this particular case, encodes the same property as strong bisimilarity to f. If we extend $\mathcal{T}$ by another state f' which does not have any ingoing/outgoing transitions, then full trace equivalence to f means that each reachable state can either perform an arbitrarily long finite sequence of a's (and no other action) or it is terminated. This property cannot be encoded as bisimulation/simulation/trace equivalence with *any* finite-state process.

An important advantage of full regular equivalence is its *computational tractability*. As we shall see in Section 4, full regular equivalence-checking tends to be more tractable and more decidable than "ordinary" regular equivalence-checking (i.e., the question whether $g \sim f$).

One can argue that full regular equivalence is "asymmetric" in the sense that the state space of g must be included in the set of states of $\mathcal{T}$, but not vice versa. In fact, a "symmetric" variant of full regular equivalence, where we require that $g \sim f$ and the state spaces of g and f are the same up to $\sim$, is equally tractable as the asymmetric version, at least for processes generated by pushdown automata (see Remark 17). The main reasons for considering the asymmetric version are its simplicity and convenience—we usually aim at specifying some kind of safety property by describing the set of all admissible behaviors up to $\sim$ (by constructing the system $\mathcal{T}$), and then it is not so important whether g can exhibit all or just some of these behaviors.

Now we formally introduce pushdown automata and the corresponding class of infinite-state processes.

Definition 4. *A* pushdown automaton (PDA) *is a tuple* $\Delta = (Q, \Gamma, \mathcal{A}, \delta)$ *where* $Q \neq \emptyset$ *is a finite set of* control states*,* $\Gamma \neq \emptyset$ *is a finite* stack alphabet*,* $\mathcal{A} \neq \emptyset$ *is a finite* input alphabet*, and* $\delta \subseteq (Q \times \Gamma) \times \mathcal{A} \times (Q \times \Gamma^{\leq 2})$ *is a set of* rules *where* $\Gamma^{\leq 2} = \{\varepsilon\} \cup \Gamma \cup (\Gamma \times \Gamma)$*. A PDA*^{k}*, where* $k \geq 1$*, is a PDA with at most* k *control*

[5] A word $w \in Act^*$ is a *trace* of a process s if $s \xrightarrow{w} t$ for some t. Processes s, s' are *trace equivalent* if they have the same set of traces.

states.

A *configuration* of Δ is a pair $p\alpha \in Q \times \Gamma^*$. To Δ we associate the transition system $\mathcal{T}_\Delta$ where the states are the configurations of Δ, $\mathcal{A}$ is the set of actions, and the transition relation, denoted by "$\rightarrow$", is the least relation ϱ satisfying the following condition: If $(pX, a, q\beta) \in \delta$, then $(pX\alpha, a, q\beta\alpha) \in \varrho$ for all $\alpha \in \Gamma^*$. For every $p\alpha \in Q \times \Gamma^*$, we use $M_{p\alpha}$ to denote the set of all $r \in Q$ such that $p\alpha \rightarrow^* r\varepsilon$.

In the rest of this paper, the elements of Q and Γ are denoted by lower case and upper case Latin letters such as $p, q, \ldots$ and $X, Y, \ldots$, respectively. The elements of Γ^* are denoted by Greek letters $\alpha, \beta, \ldots$, where ε denotes the empty word with the standard conventions (in particular, $\varepsilon\alpha = \alpha\varepsilon = \alpha$ for every $\alpha \in \Gamma^*$).

3. A Finite Semantic Base for PDA

For the rest of this section, we 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 $\perp \notin F$ is formally understood as a process without any outgoing transitions.

Our aim is to show that the relation $\precsim$ between the processes of $\mathcal{T}_\Delta$ and $\mathcal{T}$ can be represented by a finite set $\mathcal{B}$ called *base* such that for all $p\alpha \in Q \times \Gamma^*$ and $f \in F$ we have that $p\alpha \precsim f$ iff the pair $(p\alpha, f)$ can be generated from $\mathcal{B}$ by applying simple substitution rules.

To get some intuition, let us first consider the case when Δ has only one control state p. For simplicity, we write just α instead of $p\alpha$. For every $\alpha \in \Gamma^*$ and a process s, we use αs to denote the process which behaves like α until α reaches ε; from this point on, αs behaves like s. Now, let us fix some $X_1 X_2 \cdots X_\ell \in \Gamma^*$ and $f \in F$ such that $X_1 X_2 \cdots X_\ell \precsim f$. There are two possibilities.

- $X_1 \rightarrow^* \varepsilon$. Then $X_1 X_2 \cdots X_\ell \rightarrow^* X_2 \ldots X_k$, and hence there exists $g \in F$ such that $X_2 \cdots X_\ell \sim g$. By substituting $X_2 \ldots X_\ell$ with g in $X_1 X_2 \cdots X_\ell$, we obtain the process $X_1 g$. We intuitively expect $X_1 \cdots X_\ell \sim X_1 g$, but this does not hold in general; we need to assume that $\sim$ is a *right congruence* w.r.t. sequential composition, i.e., if $s \sim t$ for some processes s, t, then $\alpha s \sim \alpha t$ for all $\alpha \in \Gamma^*$. Under this assumption, which is satisfied by most of the standard process equivalences, we easily obtain $X_1 g \precsim f$.
  Thus, $X_1 X_2 \cdots X_\ell \precsim f$ is "decomposed" into $X_1 g \precsim f$ and $X_2 \cdots X_\ell \sim g$. We "remember" the pair $(X_1 g, f)$ and continue by decomposing $X_2 \cdots X_\ell \precsim g$ (see Remark 3). If $\ell = 1$, i.e., $X_2 \cdots X_\ell = \varepsilon$, we stop and "remember" the pair (ε, g).
- $X_1 \not\rightarrow^* \varepsilon$. Then, we have that $X_1 s \precsim f$ for an *arbitrary* process s, assuming that $\sim$ does not distinguish between isomorphic processes. For technical reason

clarified below, we "remember" the pairs $(X\bot, f)$, $(\varepsilon, \bot)$, together with all pairs of the form $(Yg, \bot)$ where $Y \in \Gamma$ and $g \in F_\bot$.

Our base $\mathcal{B}$ consists of *all* pairs we need to remember when applying the decomposition procedure above. That is,

$$\begin{aligned}
\mathcal{B} \;=\; & \{(Xg, f) \mid Xg \precsim f, \text{where } X \in \Gamma, X \to {}^*\varepsilon, \text{ and } g, f \in F\} \\
\cup\; & \{(\varepsilon, f) \mid \varepsilon \precsim f, \text{where } f \in F\} \\
\cup\; & \{(X\bot, f) \mid X \precsim f, \text{where } X \twoheadrightarrow^* \varepsilon, \text{ and } f \in F\} \\
\cup\; & \{(Yg, \bot) \mid Y \in \Gamma, g \in F_\bot\} \\
\cup\; & \{(\varepsilon, \bot)\}
\end{aligned}$$

Note that $\mathcal{B}$ is finite and its size is $O(|\Gamma| \cdot |F|^2)$. Also observe that by "reversing" the above decomposition principle, the original pairs (α, f) such that $\alpha \precsim f$ can be generated from $\mathcal{B}$. Formally, let $Cl(\mathcal{B})$ be the least set L subsuming $\mathcal{B}$ such that whenever $(\alpha h, f) \in L$ and $\mathcal{B}$ contains a pair of the form (w, h), where $h \in F_\bot$, then $(\alpha w, f) \in L$. Now it is easy to see that for all $\alpha \in \Gamma^*$ and $f \in F$ we have that $\alpha \precsim f$ iff $(\alpha, f) \in Cl(\mathcal{B})$. The "$\Rightarrow$" direction follows by a simple induction on the length of α; in the inductive step, we argue by distinguishing the two possible ways of decomposing $\alpha \precsim f$ (see above). Note that if $X \precsim f$ where $X \twoheadrightarrow^* \varepsilon$, then $(X\beta, f) \in Cl(\mathcal{B})$ for every $\beta \in \Gamma^*$. This explains the role of the "auxiliary" pairs $(Yg, \bot)$ and $(\varepsilon, \bot)$; recall that g ranges over $F_\bot$, including $\bot$. For the "$\Leftarrow$" direction, we just check that the above closure rule is safe.

Now we extend the above construction to general PDA with arbitrarily many control states. We start by generalizing the notion of right congruence w.r.t. sequential composition, and then show how to adapt the base.

3.1. Right PDA Congruence

Recall that a process equivalence $\sim$ is a right congruence w.r.t. the standard sequential composition "$\cdot$" if for all processes s, t, u we have that $t \sim u$ implies $s \cdot t \sim s \cdot u$. A process $p\alpha$ of Δ can be seen as a sequential composition of recursive procedure calls stored in the stack. The process $p\alpha$ terminates by emptying its stack, i.e., by reaching a configuration $r\varepsilon$. Let $\mathcal{P}$ be the set of all processes[6], and let $\varphi : Q \to \mathcal{P}$ be a function assigning (some) process to every control state of Q.

[6]Since the set of states of every transition system $\mathcal{T} = (S, \mathcal{A}, \to)$ is finite or countably infinite, we may safely assume $S \subseteq \mathcal{S}$ for some fixed set $\mathcal{S}$. Similarly, we may assume that $\mathcal{A}$ is a subset of some fixed set. Then, the class of all processes $\mathcal{P}$ is also a set.

Now we can naturally define the sequential composition of $p\alpha$ and φ, denoted by $p\alpha\varphi$, behaving like $p\alpha$ until a terminated configuration $r\varepsilon$ is reached, where the process "switches" to $\varphi(r)$. Formally, the behaviour of $p\alpha\varphi$ is defined by the following rules:

$$\frac{p\alpha \xrightarrow{a} q\beta}{p\alpha\varphi \xrightarrow{a} q\beta\varphi} \qquad\qquad \frac{\varphi(p) \xrightarrow{a} s}{p\varphi \xrightarrow{a} p\varphi[s/p]}$$

Here, $\varphi[s/p] : Q \to \mathcal{P}$ is a function which returns the same result as φ for every argument except for p where $\varphi[s/p](p) = s$. In principle, we could simplify the second rule into

$$\frac{\varphi(p) \xrightarrow{a} s}{p\varphi \xrightarrow{a} s}$$

which makes a clear sense if the operational behaviour of s is known. For our purposes, the previous "more complicated" variant is more advantageous, because it simplifies the structure of the base $\mathcal{B}$ defined in Section 3.2.

Clearly, the processes $p\varphi$ and $\varphi(p)$ are isomorphic. Also observe that if $\varphi(r) = \psi(r)$ for all $r \in M_{p\alpha}$, then the processes $p\alpha\varphi$ and $p\alpha\psi$ are isomorphic. Now we formally define the notion of right PDA congruence.

Definition 5. *We say that a process equivalence $\sim$ is a* right PDA congruence *if the following conditions are satisfied:*

- *For every configuration $p\alpha$ of Δ and all $\varphi, \psi : Q \to \mathcal{P}$ such that $\varphi(q) \sim \psi(q)$ for all $q \in M_{p\alpha}$, it holds that $p\alpha\varphi \sim p\alpha\psi$.*
- *Isomorphic processes are $\sim$-equivalent.*

Intuitively, the first item of Definition 5 says that if the computation of $p\alpha$ is "prolonged" by $\sim$-equivalent processes, then all such extensions are $\sim$-equivalent. It is easy to check that bisimulation-like, simulation-like, and trace-like equivalences (even in their "weak" forms) are right PDA congruences.

3.2. The Base $\mathcal{B}$

Intuitively, the base $\mathcal{B}$ is obtained by generalizing the relation $\mathcal{B}$ introduced for BPA and strong bisimilarity at the beginning of Section 3. The only difference is that the elements of $F_\perp$ appearing in the pairs of $\mathcal{B}$ are replaced by the elements of $(F_\perp)^Q$, i.e., by *functions* from Q to $F_\perp$, which are consistently denoted by symbols such as $\mathcal{F}, \mathcal{G}, \mathcal{H}, \ldots$ in the rest of this paper.

The operational semantics of a process $p\alpha\mathcal{F}$ is defined by the rules given in Section 3.1 (recall that $\perp$ is formally understood as a process without any outgoing

transitions). Still, the role of $\bot$ is somewhat special, which is reflected in our next definition.

Definition 6. *A function $\mathcal{F} : Q \to F_\bot$ is* compatible *with $p\alpha$ iff for every $q \in M_{p\alpha}$ we have that $\mathcal{F}(q) \neq \bot$. The class of all functions compatible with $p\alpha$ is denoted by $C(p\alpha)$.*

Note that if $\mathcal{F} \in C(p\alpha)$ and $p\alpha \to^* q\beta$, then $\mathcal{F} \in C(q\beta)$.

For the rest of this section, we fix a right PDA congruence $\sim$. For all $\alpha \in \Gamma^*$ and $\mathcal{F}, \mathcal{G} : Q \to F_\bot$, we write

- $\alpha \precsim \mathcal{F}$ if $p\alpha \precsim \mathcal{F}(p)$ for all $p \in Q$ where $\mathcal{F}(p) \neq \bot$;
- $\alpha\mathcal{G} \precsim \mathcal{F}$ if $\mathcal{G} \in C(p\alpha)$ and $p\alpha\mathcal{G} \precsim \mathcal{F}(p)$ for all $p \in Q$ where $\mathcal{F}(p) \neq \bot$.

In particular, note that if $\mathcal{F}(p) = \bot$ for all $p \in Q$, then $\alpha \precsim \mathcal{F}$ and $\alpha\mathcal{G} \precsim \mathcal{F}$ for all $\alpha \in \Gamma^*$ and $\mathcal{G} : Q \to F_\bot$.

In the rest of this paper, the set of all processes of the form $p\alpha$ and $p\alpha\mathcal{F}$, where $p \in Q$, $\alpha \in \Gamma^*$, and $\mathcal{F} \in C(p\alpha)$, is denoted by $\mathcal{P}(\Delta, F)$, and we use $pw, qv, ru, \ldots$ to range over $\mathcal{P}(\Delta, F)$. When we write $pw = p\alpha$ or $pw = p\alpha\mathcal{F}$, we mean that pw takes the respective form.

Now we are ready to define the base $\mathcal{B}$. For technical reasons, we first introduce a more general notion of *well-formed* sets. As we shall see in Section 4, the base $\mathcal{B}$ is computable by "cleaning" the greatest well-formed set.

Definition 7. *A* well-formed set *is a set K consisting of*

- *all pairs of the form $(\varepsilon, \mathcal{F})$ such that $\varepsilon \precsim \mathcal{F}$;*
- *all pairs of the form $(\mathcal{G}, \mathcal{F})$ such that $\mathcal{G} \precsim \mathcal{F}$;*
- *some (possibly none) pairs of the form $(X\mathcal{G}, \mathcal{F})$ such that $\mathcal{F}(p) \neq \bot$ implies $\mathcal{G} \in C(pX)$.*

Further, we require[7] *that if $(X\mathcal{G}, \mathcal{F}) \in K$ and $\mathcal{F} \precsim \mathcal{H}$, then $(X\mathcal{G}, \mathcal{H}) \in K$. The* base *$\mathcal{B}$ is a well-formed set defined by*

$$\mathcal{B} \;=\; \{(\varepsilon, \mathcal{F}) \mid \varepsilon \precsim \mathcal{F}\} \;\cup\; \{(\mathcal{G}, \mathcal{F}) \mid \mathcal{G} \precsim \mathcal{F}\} \;\cup\; \{(X\mathcal{G}, \mathcal{F}) \mid X\mathcal{G} \precsim \mathcal{F}\}\,.$$

Note that if $\sim$ is decidable for finite-state processes, then the greatest well-formed set G is effectively constructible. The only possible difference between

[7] This condition is needed in Lemma 11(c).

G and $\mathcal{B}$ are some extra pairs of the form $(X\mathcal{G}, \mathcal{F})$ which are "cleaned" by the algorithm presented in Section 4.

Our next definition says how to generate new pairs of the form $(\alpha, \mathcal{F})$ and $(\alpha\mathcal{G}, \mathcal{F})$ from a given well-formed set K.

Definition 8. *Let K be a well-formed set. The* closure of *K, denoted by $Cl(K)$, is the least set L satisfying the following conditions:*

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

Clearly, $Cl(K) = \bigcup_{i=0}^{\infty} Cl^i(K)$ where

- $Cl^0(K) = K$;
- $Cl^{i+1}(K)$ 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 (2) and (3) of Definition 8.

We intuitively expect that $\mathcal{B}$ generates precisely the pairs $(\alpha, \mathcal{F})$ and $(\alpha\mathcal{G}, \mathcal{F})$ such that $\alpha \precsim \mathcal{F}$ and $\alpha\mathcal{G} \precsim \mathcal{F}$. The next theorem says that this is indeed the case.

Theorem 9. *For all $\alpha \in \Gamma^*$ and $\mathcal{F}, \mathcal{G} : Q \to F_\perp$ we have the following:*

(A) $\alpha\mathcal{G} \precsim \mathcal{F}$ iff $(\alpha\mathcal{G}, \mathcal{F}) \in Cl(\mathcal{B})$;
(B) $\alpha \precsim \mathcal{F}$ iff $(\alpha, \mathcal{F}) \in Cl(\mathcal{B})$.

Proof. For the "$\Leftarrow$" direction of (A) and (B), it suffices to show that the rules (2) and (3) of Definition 8 preserve the relation $\precsim$. Since the arguments are similar, we give an explicit proof just for the rule (3). Suppose $\alpha\mathcal{G} \precsim \mathcal{F}$ and $X\mathcal{H} \precsim \mathcal{G}$. We need to show that $\alpha X\mathcal{H} \precsim \mathcal{F}$, i.e., for every $p \in Q$ such that $\mathcal{F}(p) \neq \perp$ we have that

(a) $\mathcal{H} \in C(p\alpha X)$,
(b) $p\alpha X\mathcal{H} \precsim \mathcal{F}(p)$.

Let us fix some $p \in Q$ such that $\mathcal{F}(p) \neq \perp$. We start by proving (a). Clearly, $C(p\alpha X) = \bigcap_{r \in M_{p\alpha}} C(rX)$, and hence it suffices to prove $\mathcal{H} \in C(rX)$ for an arbitrary fixed $r \in M_{p\alpha}$. Since $\alpha\mathcal{G} \precsim \mathcal{F}$ and $\mathcal{F}(p) \neq \perp$, we obtain $\mathcal{G} \in C(p\alpha)$, which implies $\mathcal{G}(r) \neq \perp$ because $r \in M_{p\alpha}$. Since $X\mathcal{H} \precsim \mathcal{G}$ and $\mathcal{G}(r) \neq \perp$, we obtain $\mathcal{H} \in C(rX)$ as needed. Now we prove (b). Recall that for every $r \in M_{p\alpha}$ we have that $\mathcal{G}(r) \neq \perp$, which implies $rX\mathcal{H} \precsim \mathcal{G}(r)$ because $X\mathcal{H} \precsim \mathcal{G}$. For every $r \in M_{p\alpha}$, we put $\varphi(r) = rX\mathcal{H}$ and $\psi(r) = \mathcal{G}(r)$. Since $\sim$ is a right PDA congruence, for every $q\beta$ such that $p\alpha \to^* q\beta$ we obtain $q\beta\varphi \sim q\beta\psi$. We prove that

(1) $p\alpha X\mathcal{H} \sim \mathcal{F}(p)$,
(2) for every tw such that $p\alpha X\mathcal{H} \to^* tw$ there is $g \in F$ such that $tw \sim g$.

(1) follows immediately, because $p\alpha\varphi \sim p\alpha\psi$, and the processes $p\alpha\varphi$ and $p\alpha\psi$ are isomorphic to $p\alpha X\mathcal{H}$ and $p\alpha\mathcal{G}$, respectively. Hence, $p\alpha X\mathcal{H} \sim p\alpha\varphi \sim p\alpha\psi \sim p\alpha\mathcal{G}$. Further, $p\alpha\mathcal{G} \sim \mathcal{F}(p)$ because $\alpha\mathcal{G} \precsim \mathcal{F}$ and $\mathcal{F}(p) \neq \bot$. Thus, $p\alpha X\mathcal{H} \sim \mathcal{F}(p)$. It remains to prove (2). Let $p\alpha X\mathcal{H} \to^* tw$. We distinguish two cases:

- $tw = q\beta X\mathcal{H}$ where $p\alpha \to^* q\beta$. Then $p\alpha\mathcal{G} \to^* q\beta\mathcal{G}$. Since $\alpha\mathcal{G} \precsim \mathcal{F}$ and $\mathcal{F}(p) \neq \bot$, there is $g \in F$ such that $q\beta\mathcal{G} \sim g$. Further, $q\beta\varphi \sim q\beta\psi$, where $q\beta\varphi$ and $q\beta\psi$ are isomorphic to $q\beta X\mathcal{H}$ and $q\beta\mathcal{G}$, respectively. Thus, we obtain $q\beta X\mathcal{H} \sim g$.
- $p\alpha X\mathcal{H} \to^* rX\mathcal{H} \to^* tw$ where $p\alpha \to^* r\varepsilon$. Then $r \in M_{p\alpha}$, which implies $\mathcal{G}(r) \neq \bot$ (see above). Since $X\mathcal{H} \precsim \mathcal{G}$ and $\mathcal{G}(r) \neq \bot$, we obtain $rX\mathcal{H} \precsim \mathcal{G}(r)$. As $rX\mathcal{H} \to^* tw$, there exists $g \in F$ such that $tw \sim g$.

The "$\Rightarrow$" direction of (A) is proven by induction on the length of α. If $\alpha = \varepsilon$, we are done immediately because if $\mathcal{G} \precsim \mathcal{F}$, then $(\mathcal{G}, \mathcal{F}) \in \mathcal{B}$ by Definition 7. Now assume $\alpha = \beta Y$ where $\beta Y\mathcal{G} \precsim \mathcal{F}$. We prove that $(\beta Y\mathcal{G}, \mathcal{F}) \in Cl(\mathcal{B})$. Let $M_{\mathcal{F},\beta}$ be the union of all $M_{p\beta}$ such that $\mathcal{F}(p) \neq \bot$. Since $\beta Y\mathcal{G} \precsim \mathcal{F}$, for every $r \in M_{\mathcal{F},\beta}$ we can fix $f_r \in F$ such that $rY\mathcal{G} \sim f_r$. Let $\mathcal{H} : Q \to F_\bot$ be a function defined by $\mathcal{H}(r) = f_r$ for all $r \in M_{\mathcal{F},\beta}$, and $\mathcal{H}(r) = \bot$ for all $r \in Q \setminus M_{\mathcal{F},\beta}$. We show that

(i) $Y\mathcal{G} \precsim \mathcal{H}$,
(ii) $\beta\mathcal{H} \precsim \mathcal{F}$.

Using (i) and (ii), the proof can be completed as follows. If $\beta = \varepsilon$, we have that $Y\mathcal{G} \precsim \mathcal{H}$ and $\mathcal{H} \precsim \mathcal{F}$. This implies $Y\mathcal{G} \precsim \mathcal{F}$, hence $(Y\mathcal{G}, \mathcal{F}) \in \mathcal{B}$ by Definition 7. If $\beta \neq \varepsilon$, we have that $(Y\mathcal{G}, \mathcal{H}) \in \mathcal{B}$ by Definition 7, $(\beta\mathcal{H}, \mathcal{F}) \in Cl(\mathcal{B})$ by induction hypothesis, and $(\beta Y\mathcal{G}, \mathcal{F}) \in Cl(\mathcal{B})$ by applying the rule (3) of Definition 8.

So, it remains to prove (i) and (ii). Observe that (i) follows directly from the definition of $\mathcal{H}$ (see also Remark 3). Let $\varphi(r) = rY\mathcal{G}$ and $\psi(r) = f_r$ for all $r \in M_{\mathcal{F},\beta}$. To prove (ii), let us fix some $p \in Q$ such that $\mathcal{F}(p) \neq \bot$. Clearly, $\mathcal{H} \in C(p\beta)$. We show that $p\beta\mathcal{H} \precsim \mathcal{F}(p)$. Since $\sim$ is a right PDA congruence, we obtain $p\beta\varphi \sim p\beta\psi$, where $p\beta\varphi$ and $p\beta\psi$ are isomorphic to $p\beta Y\mathcal{G}$ and $p\beta\mathcal{H}$, respectively. Hence, $p\beta Y\mathcal{G} \sim p\beta\mathcal{H}$, and since $p\beta Y\mathcal{G} \sim \mathcal{F}(p)$, we obtain $p\beta\mathcal{H} \sim \mathcal{F}(p)$. It remains to show that every process reachable from $p\beta\mathcal{H}$ is $\sim$-equivalent to some process of F. Let $p\beta\mathcal{H} \to^* tw$. We distinguish two cases:

- $tw = q\gamma\mathcal{H}$ where $p\beta \to^* q\gamma$. Then $p\beta Y\mathcal{G} \to^* q\gamma Y\mathcal{G}$. Since $\beta Y\mathcal{G} \precsim \mathcal{F}$, there is $g \in F$ such that $q\gamma Y\mathcal{G} \sim g$. Further, $q\gamma\varphi \sim q\gamma\psi$, where $q\gamma\varphi$ and $q\gamma\psi$ are isomorphic to $q\gamma Y\mathcal{G}$ and $q\gamma\mathcal{H}$, respectively. Hence, $q\gamma\mathcal{H} \sim g$.

- $p\beta\mathcal{H} \to^* r\mathcal{H} \to^* tw$ where $p\beta \to^* r\varepsilon$. Note that then $t = r$ and $w = \mathcal{J}$ for some $\mathcal{J} : Q \to F_\perp$, and the process tw is isomorphic to $\mathcal{J}(r)$. Hence, $tw \sim \mathcal{J}(r)$.

The "$\Rightarrow$" direction of (B) follows easily now. Let $\alpha \precsim \mathcal{F}$. If $\alpha = \varepsilon$, we have that $\varepsilon \precsim \mathcal{F}$, hence $(\varepsilon, \mathcal{F}) \in \mathcal{B}$ by Definition 7. Now let $\alpha \neq \varepsilon$, and let $M_{\mathcal{F},\alpha}$ be the union of all $M_{p\alpha}$ such that $\mathcal{F}(p) \neq \perp$. Since $\alpha \precsim \mathcal{F}$, for every $r \in M_{\mathcal{F},\alpha}$ we can fix $f_r \in F$ such that $r\varepsilon \sim f_r$. Let $\mathcal{H} : Q \to F_\perp$ be a function defined by $\mathcal{H}(r) = f_r$ for all $r \in M_{\mathcal{F},\alpha}$, and $\mathcal{H}(r) = \perp$ for all $r \in Q \setminus M_{\mathcal{F},\alpha}$. Similarly as above, we obtain $\varepsilon \precsim \mathcal{H}$ and $\alpha\mathcal{H} \precsim \mathcal{F}$. Hence, $(\varepsilon, \mathcal{H}) \in \mathcal{B}$ by Definition 7, and $(\alpha\mathcal{H}, \mathcal{F}) \in Cl(\mathcal{B})$ due to (A) which has already been proven. By applying the rule (2) of Definition 8, we obtain $(\alpha, \mathcal{F}) \in Cl(\mathcal{B})$. □

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. The applicability of the method to concrete process equivalences is demonstrated in the subsequent subsections.

For the rest of this section, we fix

- a pushdown automaton $\Delta = (Q, \Gamma, \mathcal{A}, \delta)$ of size n;
- a finite-state system $\mathcal{T} = (F, \mathcal{A}, \to)$ of size m;
- a process equivalence $\sim$ such that
  - $\sim$ is a right PDA congruence;
  - $\sim$ is decidable for finite-state processes.

Note that we can safely assume that the input alphabet of Δ and the set of actions of $\mathcal{T}$ are the same. In our complexity estimations we also use the parameter $z = |F|^{|Q|}$.

Definition 10. *Let K be a well-formed set. For every $i \in \mathbb{N}_0$, we define the set $Gen^i(K) \subseteq \mathcal{P}(\Delta, F) \times F$ as follows:*

$$\begin{aligned} Gen^i(K) \;=\; & \{(p\alpha, f) \mid \text{there is } \mathcal{F} \text{ such that } \mathcal{F}(p) = f \text{ and } (\alpha, \mathcal{F}) \in Cl^i(K)\} \\ \cup\; & \{(p\alpha\mathcal{G}, f) \mid \text{there is } \mathcal{F} \text{ such that } \mathcal{F}(p) = f \text{ and } (\alpha\mathcal{G}, \mathcal{F}) \in Cl^i(K)\} \end{aligned}$$

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

Some useful properties of *Gen* are given in the next lemma (all items follow immediately from Definitions 7 and 8).

Lemma 11. *Let K be a well-formed set. Then we have the following:*

(a) $(p\mathcal{G}, f) \in Gen(K)$ *iff* $\mathcal{G}(p) \neq \bot$ *and* $p\mathcal{G} \precsim f$.
(b) $(p\varepsilon, f) \in Gen(K)$ *iff* $p\varepsilon \precsim f$.
(c) If $(pw, f) \in Gen(K)$ *and* $f \sim g$, *then* $(pw, g) \in Gen(K)$.

The base $\mathcal{B}$ is computed by taking the greatest well-formed set G as the initial over-approximation of $\mathcal{B}$, and then applying one or more "cleaning steps" until reaching a fixed-point. In each cleaning step, all pairs of the form $(X\mathcal{G}, \mathcal{F})$ contained in a current approximation K of the base are examined, and it is checked whether every $(pX\mathcal{G}, \mathcal{F}(p))$, where $\mathcal{F}(p) \neq \bot$, *expands* in $Gen(K)$. Intuitively, the expansion condition is designed so that it implies $pX\mathcal{G} \precsim \mathcal{F}(p)$ if all pairs of $Gen(K)$ are "correct", i.e., $Gen(K) \subseteq \precsim$. For example, in the case of strong bisimilarity, $(pX\mathcal{G}, \mathcal{F}(p))$ expands in $Gen(K)$ if for every $\xrightarrow{a}$ move of one process there is a matching $\xrightarrow{a}$ move of the other process such that the resulting pair of processes belongs to $Gen(K)$. If $(pX\mathcal{G}, \mathcal{F}(p))$ does *not* expand in $Gen(K)$ for some $p \in Q$ such that $\mathcal{F}(p) \neq \bot$, the pair $(X\mathcal{G}, \mathcal{F})$ is deleted from K, together with all $(X\mathcal{G}, \mathcal{H})$ such that $\mathcal{H} \precsim \mathcal{F}$ (hence, the resulting relation is again well-formed). Formally, an *expansion* is a function which to a given relation $R \subseteq \mathcal{P}(\Delta, F) \times F$ assigns a subset $R' \subseteq R$ of all pairs $(pw, f) \in R$ that expand in R. The desired properties of expansion are formulated in the next definition.

Definition 12. *We say that a pair of the form $(X\mathcal{G}, \mathcal{F})$ is* contained *in a relation $R \subseteq \mathcal{P}(\Delta, F) \times F$ if $(pX\mathcal{G}, \mathcal{F}(p)) \in R$ for all $p \in Q$ such that $\mathcal{F}(p) \neq \bot$.*

An expansion *for $\sim$ is a function Exp which to every relation $R \subseteq \mathcal{P}(\Delta, F) \times F$ assigns a set $Exp(R) \subseteq R$ so that the following conditions are satisfied (where $\precsim$ is interpreted as a subset of $\mathcal{P}(\Delta, F) \times F$ consisting of all (pw, f) such that $pw \precsim f$):*

(1) Exp is monotonic.
(2) $Exp(\precsim) = \precsim$.
(3) For every $R \subseteq \mathcal{P}(\Delta, F) \times F$, *if* $Exp(R) = R$ *then* $R \subseteq \precsim$.
(4) For every well-formed set K, we have that if every $(X\mathcal{G}, \mathcal{F}) \in K$ is contained in $Exp(Gen(K))$, then $Exp(Gen(K)) = Gen(K)$.
(5) Given a well-formed set K and a pair $(X\mathcal{G}, \mathcal{F}) \in K$, it is decidable whether $(X\mathcal{G}, \mathcal{F})$ is contained in $Exp(Gen(K))$.

The next theorem says that if *Exp* is an expansion for $\sim$, then the base $\mathcal{B}$ is computable in the way indicated above, i.e., by applying the algorithm of Fig. 1.

Theorem 13. *If Exp is an expansion for $\sim$, then the algorithm of Fig. 1 computes the base $\mathcal{B}$.*

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

```
1:  K := the greatest well-formed set G
2:  repeat
3:      U := K
4:      for each (XG, F) ∈ U do
5:          if (XG, F) is not contained in Exp(Gen(U))
6:              then K := K ∖ {(XG, H) | H ⋦ F}
7:  until K = U
8:  return K
```

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

PROOF. First, realize that the greatest well-formed set G used at line 1 is computable because the set $\{(\varepsilon, \mathcal{F}) \mid \varepsilon \precsim \mathcal{F}\} \cup \{(\mathcal{G}, \mathcal{F}) \mid \mathcal{G} \precsim \mathcal{F}\}$ is computable (recall that $\sim$ is decidable for finite-state processes), and the set of all pairs of the form $(X\mathcal{G}, \mathcal{F})$, where $\mathcal{F}(p) \neq \bot$ implies $\mathcal{G} \in \mathcal{C}(pX)$, is computable in time polynomial in m, n, z, because the set M_{pX} is computable in time polynomial in n (see, e.g., [30]). Further, the condition in the **if** statement at line 5 is effective due to Definition 12(5). The correctness of the algorithm is implied by the following observations:

- If K is a well-formed set such that $\mathcal{B} \subseteq K$ and $(X\mathcal{G}, \mathcal{F}) \in K$ is not contained in $Exp(Gen(K))$, then $(X\mathcal{G}, \mathcal{F}) \notin \mathcal{B}$ (and hence $(X\mathcal{G}, \mathcal{H}) \notin \mathcal{B}$ for all $\mathcal{H} \precsim \mathcal{F}$). To see this, realize that $\precsim = Exp(\precsim) = Exp(Gen(\mathcal{B}))$ (here we use Definition 12(2) and Theorem 9), hence every $(X\mathcal{G}, \mathcal{F}) \in \mathcal{B}$ is contained in $Exp(Gen(\mathcal{B}))$. Further, $Exp(Gen(\mathcal{B})) \subseteq Exp(Gen(K))$ because Exp is monotonic (Definition 12(1)). Thus, every $(X\mathcal{G}, \mathcal{F}) \in \mathcal{B}$ is contained also in $Exp(Gen(K))$.
- Consider the well-formed set K returned by the algorithm at line 8. The previous observation implies $\mathcal{B} \subseteq K$. Further, every $(X\mathcal{G}, \mathcal{F}) \in K$ is contained in $Exp(Gen(K))$. This means $Exp(Gen(K)) = Gen(K)$ by Definition 12(4), hence $Gen(K) \subseteq \precsim$ by applying Definition 12(3). Hence, for each $(X\mathcal{F}, \mathcal{G}) \in K$ we have that $X\mathcal{F} \precsim G$, which implies $K \subseteq \mathcal{B}$. □

As we shall see, an appropriate expansion can be designed for almost every process equivalence of the linear/branching time spectrum [57, 58].

4.1. Finite multi-automata

A general problem related to equivalences with silent τ-moves is that the set of states reachable in one $\stackrel{a}{\Rightarrow}$ move can be infinite. For example, in the case of weak bisimilarity, it is natural to stipulate that $(pX\mathcal{G}, \mathcal{F}(p))$ expands in $Gen(K)$ if for every $\stackrel{a}{\rightarrow}$ move of one process there is a matching $\stackrel{a}{\Rightarrow}$ move of the other process such that the resulting pair of processes belongs to $Gen(K)$ (cf. the paragraph preceding Definition 12). It is not hard to verify that the associated *Exp* indeed satisfies the conditions (1)–(4) of Definition 12. Now consider the condition (5). Given a well-formed set K, a pair $(X\mathcal{G}, \mathcal{F}) \in K$, and $p \in Q$ such that $\mathcal{F}(p) \neq \bot$, we need to decide whether $(pX\mathcal{G}, \mathcal{F}(p))$ expands in $Gen(K)$. In particular, this means to check whether for every $\mathcal{F}(p) \stackrel{a}{\rightarrow} f$ there exists a matching move $pX\mathcal{G} \stackrel{a}{\Rightarrow} qw$ such that $(qw, f) \in Gen(K)$. Since there may exist infinitely many candidates for qw, we cannot try all of them one by one. Instead, we construct

- a finite-state automaton recognizing all words qw such that $pX\mathcal{G} \stackrel{a}{\Rightarrow} qw$;
- a finite-state automaton recognizing all words qw such that $(qw, f) \in Gen(K)$.

Then, we just check whether the languages recognized by the two finite-state automata have a non-empty intersection. Interestingly, this is achievable in time *polynomial* in m, n, z.

Now we develop these tools formally. The next definition is borrowed from [10].

Definition 14. *A* multi-automaton *is a tuple* $\mathcal{M} = (S, \Sigma, \gamma, Acc)$ *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* $\mathcal{M}$*);*
- $\Sigma = \Gamma \cup \{\mathcal{F} \mid \mathcal{F} : Q \to F_\bot\}$ *is the* input alphabet *(i.e., the alphabet has a special symbol for each* $\mathcal{F} : Q \to F_\bot$*);*
- $\gamma \subseteq S \times \Sigma \times S$ *is a* transition relation*, which is extended to the elements of* $S \times \Sigma^* \times S$ *in the natural way;*
- $Acc \subseteq S$ *is a set of* accepting states.

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

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

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

Now we show that the set of all configurations reachable from a set of configurations represented by a given multi-automaton is also representable by an

efficiently constructible multi-automaton. For every set of processes $\mathcal{P}$ and every action a, we define the sets

- $Post_a(\mathcal{P}) = \{t \mid \text{there is } s \in \mathcal{P} \text{ such that } s \xrightarrow{a} t\}$,
- $Post^*(\mathcal{P}) = \{t \mid \text{there is } s \in \mathcal{P} \text{ such that } s \rightarrow^* t\}$,
- $Post^*_\tau(\mathcal{P}) = \{t \mid \text{there is } s \in \mathcal{P} \text{ such that } s \stackrel{\tau}{\Rightarrow} t\}$.

Note that if $\mathcal{P} \subseteq \mathcal{P}(\Delta, F)$, then $Post_a(\mathcal{P})$, $Post^*(\mathcal{P})$, and $Post^*_\tau(\mathcal{P})$ are also subsets of $\mathcal{P}(\Delta, F)$. A proof of the following lemma is obtained by applying the standard saturation technique for pushdown automata (see, e.g., [13] for an overview of recent results). An explicit proof can be found in [19].

Lemma 15. *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 $\mathcal{M}$.*

In the next lemma we prove that for every well-formed set K and every $f \in F$, one can efficiently construct a multi-automaton recognizing all pw such that $(pw, f) \in Gen(K)$. Here we utilize the simplicity of $Cl(K)$, see Definition 8.

Lemma 16. *Let K be a well-formed set and $f \in F$. There is a multi-automaton $\mathcal{M}_{K,f}$ constructible in time polynomial in the size of K recognizing the set $Gen_f(K) = \{pw \mid (pw, f) \in Gen(K)\}$.*

Proof. The multi-automaton $\mathcal{M}_{K,f}$ is constructed as follows:

- the set of states is $Q \cup \{fin\} \cup \{\hat{\mathcal{F}} \mid \mathcal{F} : Q \to F_\perp\}$;
- the transition relation is the least γ satisfying the following conditions:
  - for all $(X\mathcal{G}, \mathcal{F}) \in K$ and $p \in Q$ such that $\mathcal{F}(p) = f$ we have that $(p, X, \hat{\mathcal{G}}) \in \gamma$;
  - for all $(\mathcal{G}, \mathcal{F}) \in K$ and $p \in Q$ such that $\mathcal{F}(p) = f$ we have that $(p, \mathcal{G}, fin) \in \gamma$;
  - if $(X\mathcal{G}, \mathcal{F}) \in K$, then $(\hat{\mathcal{F}}, X, \hat{\mathcal{G}}) \in \gamma$;
  - if $(y, X, \hat{\mathcal{G}}) \in \delta$ and $(\varepsilon, \mathcal{G}) \in K$, then $(y, X, fin) \in \gamma$;
  - $(\hat{\mathcal{F}}, \mathcal{F}, fin) \in \gamma$ for every $\mathcal{F} : Q \to F_\perp$;
- the set of accepting states is

$$\{fin\} \cup \{p \in Q \mid \text{ there is } (\varepsilon, \mathcal{F}) \in K \text{ such that } \mathcal{F}(p) = f\}.$$

One can easily verify that $\mathcal{L}(\mathcal{M}_{K,f}) = Gen_f(K)$. □

Remark 17. *Lemma 15 and Lemma 16 can also be used to decide the "symmetric" variant of full regular equivalence discussed in Section 2 where we require that $g \sim f$ and the sets of states reachable from g and f are the same up to $\sim$. Given a PDA process $p\alpha$ and a process f of a finite-state transition system $\mathcal{T}$, we first restrict the set of states of $\mathcal{T}$ to the subset of states reachable from f, then compute the base $\mathcal{B}$, and finally check whether $p\alpha \in Gen_f(\mathcal{B})$ and $Post^*(\{p\alpha\}) \cap Gen_{f'}(\mathcal{B}) \neq \emptyset$ for every f' such that $f \rightarrow^* f'$.*

It is worth noting that the base $\mathcal{B}$ together with Lemma 15 and Lemma 16 can also be used to decide whether a given finite-state process f is the $\sim$-*quotient* of a given PDA process $p\alpha$, under the condition that $\sim$ is *preserved under quotients*.

Definition 18. *Let s be a process with state space S and $\sim$ a process equivalence. The* $\sim$-quotient *of s is the process $[s]$ of the transition system $(S/\sim, \mathcal{A}, \rightarrowtail)$, where $\mathcal{A}$ consists of all actions a such that $t \xrightarrow{a} u$ for some $t, u \in S$, and $[t] \overset{a}{\rightarrowtail} [u]$ iff $t' \xrightarrow{a} u'$ for some $t', u' \in S$ such that $t \sim t'$ and $u \sim u'$. We say that $\sim$ is* preserved under quotients *iff $s \sim [s]$ for every process s.*

It has been shown in [40] (see also [42]) that all reasonable process equivalences are preserved under quotients.

An algorithm deciding whether f is isomorphic to the $\sim$-quotient of $p\alpha$ (where $\sim$ is preserved under quotients) is given in Fig. 2. First, it is verified that the set of actions $\mathcal{A}_{\mathcal{T}}$ of $\mathcal{T}$ is equal to the set of actions executable in the configurations reachable from $p\alpha$ (line 1). Note that the set

$$\{a \in \mathcal{A} \mid \text{ there exist } (qX, a, r\gamma) \in \delta \text{ and } \beta \in \Gamma^* \text{ such that } p\alpha \rightarrow^* qX\beta\}$$

is computable in time polynomial in the size of $p\alpha$ and Δ (see Lemma 15). At line 2, it is verified that the states of F are pairwise non-equivalent. Then, it is checked whether the state space of $p\alpha$ is included in F up to $\sim$ (lines 3 and 4). Note that $\mathcal{T}$ may still contain some extra states and transitions which are not present in the $\sim$-quotient of $p\alpha$. At line 6, it is checked that for each transition $f' \xrightarrow{a} f''$ of $\mathcal{T}$ there exists a transition $q\beta \xrightarrow{a} r\gamma$ such that $q\beta \precsim f'$, $r\gamma \precsim f''$, and $p\alpha \rightarrow^* q\beta$. Similarly, at line 7 it is checked that no transition of the $\sim$-quotient of $p\alpha$ is missing in $\mathcal{T}$. Observe that the **if** statements at lines 6 and 7 can be implemented by computing a multi-automaton $\mathcal{M}$ recognizing the set

$$Post_a(Post^*(\{p\alpha\}) \cap Gen_{f'}(\mathcal{B})) \cap Gen_{f''}(\mathcal{B})$$

Input: A process $p\alpha$ of a PDA $\Delta = (Q, \Gamma, \mathcal{A}, \delta)$,
a process f of a finite-state transition system $\mathcal{T} = (F, \mathcal{A}_\mathcal{T}, \rightarrow)$.
Output: YES if f is the $\sim$-quotient of $p\alpha$, NO otherwise.

1: **if** $\mathcal{A}_\mathcal{T} \neq \{a \in \mathcal{A} \mid$ there exist $(qX, a, r\gamma) \in \delta$ and $\beta \in \Gamma^*$ such that $p\alpha \rightarrow^* qX\beta\}$
then return NO
2: **if** there are $f', f'' \in F$ such that $f' \neq f''$ and $f' \sim f''$ **then return** NO
3: compute the base $\mathcal{B}$
4: **if** $p\alpha \notin Gen_f(\mathcal{B})$ **then return** NO
5: **for all** $f', f'' \in F$, $a \in \mathcal{A}$ **do**
6: **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
7: **if** (**not** $f' \xrightarrow{a} f''$) **and** $Post_a(Post^*(\{p\alpha\}) \cap Gen_{f'}(\mathcal{B})) \cap Gen_{f''}(\mathcal{B}) \neq \emptyset$
then return NO
8: **od**
9: **return** YES

Figure 2: An algorithm deciding whether f is the $\sim$-quotient of $p\alpha$.

and checking whether $\mathcal{L}(\mathcal{M}) = \emptyset$ (note that $\mathcal{M}$ is effectively constructible due to Lemma 15 and Lemma 16). Moreover, if the base $\mathcal{B}$ is computable in time polynomial in m, n, z, then the algorithm of Fig. 2 also terminates in time polynomial in m, n, z.

4.2. Bisimulation Equivalences with Silent Moves

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

Definition 19. *Let $\mathcal{T}_1 = (S_1, \rightarrow, \mathcal{A})$ and $\mathcal{T}_2 = (S_2, \rightarrow, \mathcal{A})$ be transition systems*[8] *such that $S_1 \cap S_2 = \emptyset$. Further, let $R \subseteq S_1 \times S_2$ and $(s, a, s') \in S_1 \times \mathcal{A} \times S_1$ (note that (s, a, s') is not necessarily a transition of $\mathcal{T}_1$). We say that a move*

[8] Although we use the same symbol "$\rightarrow$" to denote the transition relations of $\mathcal{T}_1$ and $\mathcal{T}_2$, no confusion arises because the sets S_1 and S_2 are disjoint.

$t \stackrel{a}{\Rightarrow} t'$ is R-consistent *with (s, a, s') in a weak, early, delay, or branching style, if $(s,t), (s',t') \in R$ and one of the following conditions holds:*

- *$a = \tau$ and $t = t'$;*
- *the move $t \stackrel{a}{\Rightarrow} t'$ takes the form $t{=}u_0 \stackrel{\tau}{\to} \cdots \stackrel{\tau}{\to} u_i \stackrel{a}{\to} v_0 \stackrel{\tau}{\to} \cdots \stackrel{\tau}{\to} v_j{=}t'$ where $i, j \geq 0$, and*

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

Further, we say that a move $t \stackrel{a}{\Rightarrow} t'$ is tightly R-consistent *with (s, a, s') (in a given style) if stronger variants of the above conditions are satisfied, where (i) requires $(s, u_k) \in R$ for all $k \leq i$, and (ii) requires $(s', v_k) \in R$ for all $k \leq j$.*

A pair $(s,t) \in R$ b-expands in R *if*

- *for every $s \stackrel{a}{\to} s'$ there is a move $t \stackrel{a}{\Rightarrow} t'$ which is R-consistent with $s \stackrel{a}{\to} s'$;*
- *for every $t \stackrel{a}{\to} t'$ there is a move $s \stackrel{a}{\Rightarrow} s'$ which is R^{-1}-consistent with $t \stackrel{a}{\to} t'$.*

Let BExp be a function which to every $R \subseteq S_1 \times S_2$ assigns the set of all $(s,t) \in R$ that b-expand in R. A tight *b-expansion and a function TBExp are defined analogously using tight R-consistency instead of R-consistency.*

We say that R is a weak, early, delay, or branching bisimulation *if $R \subseteq BExp(R)$, where the function BExp is parameterized by the respective style. Processes s, t are weakly, early, delay, or branching* bisimilar *if they are related by some weak, early, delay, or branching bisimulation, respectively.*

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. As bisimilar processes have the same state space up to $\sim$, we do not distinguish between the relations $\sim$ and $\precsim$.

The next lemma recalls the standard property of bisimilarity used in the proof of Lemma 23.

Lemma 20. *Let $\mathcal{T}_1 = (S_1, \to, \mathcal{A})$ and $\mathcal{T}_2 = (S_2, \to, \mathcal{A})$ be transition systems where $S_1 \cap S_2 = \emptyset$, and let $\sim$ be the relation of bisimilarity over $S_1 \times S_2$. Then $TBExp(\sim) = \sim$.*

Proof. Let $R \subseteq S_1 \times S_2$ be a relation defined by $(s,t) \in R$ if one of the following conditions holds:

- There are $u, u' \in S_1$ and moves $u \stackrel{\tau}{\Rightarrow} s$, $s \stackrel{\tau}{\Rightarrow} u'$ such that $u \sim t$, and $u' \sim t$.
- There are $v, v' \in S_2$ and moves $v \stackrel{\tau}{\Rightarrow} t$, $t \stackrel{\tau}{\Rightarrow} v'$ such that $s \sim v$, and $s \sim v'$.

It is easy to verify that R is a bisimulation, and hence $R \subseteq \sim$. From this we immediately obtain $\sim \subseteq TBExp(\sim)$, hence $TBExp(\sim) = \sim$. □

For technical reasons that become clear in the proof of Lemma 23, 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 over $F \times F$. We say that $\mathcal{T}$ is* complete *if, for all $f, f' \in F$ and $a \in \mathcal{A}$, the existence of a move $f \stackrel{a}{\Rightarrow} f'$ $\sim_F$-consistent with (f, a, f') implies $f \stackrel{a}{\rightarrow} f'$.*

The completeness assumption is not restrictive because if we add the missing transitions to $\mathcal{T}$ (which can be done in time polynomial in m), each state f of $\mathcal{T}$ stays bisimilar to itself. Note that this would not be true if we added a $f \stackrel{a}{\rightarrow} f'$ transition for *every* move $f \stackrel{a}{\Rightarrow} f'$ (consider, e.g., branching bisimilarity).

Our aim is to design a suitable function Exp satisfying the conditions (1)–(5) of Definition 12. A natural idea is to employ the function $BExp$ introduced in Definition 19. Obviously, $BExp$ is monotonic, $BExp(\sim) = \sim$, and if $BExp(R) = R$, then $R \subseteq \sim$. Now let K be a well-formed set such that every $(X\mathcal{G}, \mathcal{F}) \in K$ is contained in $BExp(Gen(K))$. We need to show $Gen(K) \subseteq BExp(Gen(K))$, which can be achieved by proving $Gen^i(K) \subseteq BExp(Gen(K))$ for every $i \geq 0$ (by induction on i). However, there are some difficulties in the induction step. To see this, consider a pair $(p\alpha X\mathcal{G}, f) \in Gen^{i+1}(K)$ such that $(p\alpha\mathcal{F}, f) \in Gen^i(K)$ and $(X\mathcal{G}, \mathcal{F}) \in K$. Then

(a) $(p\alpha\mathcal{F}, f)$ b-expands in $Gen(K)$ by induction hypothesis,
(b) for every $q \in Q$ such that $\mathcal{F}(q) \neq \bot$ we have that $(qX\mathcal{G}, \mathcal{F}(q))$ b-expands in $Gen(K)$ (because $(X\mathcal{G}, \mathcal{F})$ is contained in $BExp(Gen(K))$).

We need to show that $(p\alpha X\mathcal{G}, f)$ b-expands in $Gen(K)$. In particular, this means to prove that for every $f \stackrel{a}{\rightarrow} g$, where $a \neq \tau$, there is a "$\stackrel{a}{\Rightarrow}$" move of $p\alpha X\mathcal{G}$ which is $Gen(K)^{-1}$-consistent with $f \stackrel{a}{\rightarrow} g$. Due to (a), we know that such a "$\stackrel{a}{\Rightarrow}$" move exists for $p\alpha\mathcal{F}$. One of the problematic cases is when the style is branching and this move takes the form

$$p\alpha\mathcal{F} \stackrel{\tau}{\Rightarrow} r\mathcal{F} \stackrel{\tau}{\Rightarrow} r\mathcal{J} \stackrel{a}{\Rightarrow} r\mathcal{H}$$

where $p\alpha \stackrel{\tau}{\Rightarrow} r\varepsilon$ and $(r\mathcal{H}, g) \in Gen(K)$. Here, we would like to conclude that there exists a move $\mathcal{F}(r) \stackrel{a}{\Rightarrow} \mathcal{H}(r)$ $\sim_F$-consistent with $(\mathcal{F}(r), a, \mathcal{H}(r))$ (see Definition 21), hence $\mathcal{F}(r) \stackrel{a}{\rightarrow} \mathcal{H}(r)$, and due to (b) there is a move $rX\mathcal{G} \stackrel{a}{\Rightarrow} tw$ which is $Gen(K)^{-1}$-consistent with $\mathcal{F}(r) \stackrel{a}{\rightarrow} \mathcal{H}(r)$. Further, the sequence

$$p\alpha X\mathcal{G} \stackrel{\tau}{\Rightarrow} rX\mathcal{G} \stackrel{a}{\Rightarrow} tw$$

should form a "$\stackrel{a}{\Rightarrow}$" move which is $Gen(K)^{-1}$-consistent with $f \stackrel{a}{\rightarrow} g$. Unfortunately, there is no clear justification for the existence of a move $\mathcal{F}(r) \stackrel{a}{\Rightarrow} \mathcal{H}(r)$ $\sim_F$-consistent with $(\mathcal{F}(r), a, \mathcal{H}(r))$. If we used *TBExp* instead of *BExp*, the above argument would work, because then $(r\mathcal{F}, f) \in Gen(K)$, hence $r\mathcal{F} \sim f$ by Lemma 11(a), and $\mathcal{F}(r) \sim f$ because $r\mathcal{F} \sim \mathcal{F}(r)$. However, deciding whether a given $(X\mathcal{G}, \mathcal{F}) \in K$ is contained in $TBExp(Gen(K))$ (cf. condition (5) of Definition 12) is complicated because of the constraints that must be satisfied by all of the intermediate configurations visited along the move $pX\mathcal{G} \stackrel{a}{\Rightarrow} tw$. So, *TBExp* is not an ideal choice either. After considering possible ways of resolving these problems, it turned out that a simple solution is to modify the function *BExp* as follows:

Definition 22. *Let $R \subseteq \mathcal{P}(\Delta, F) \times F$. We say that a pair $(pw, f) \in R$* quasi-expands *in R if the following conditions hold:*

- *for every $pw \stackrel{a}{\rightarrow} qv$, there is $f \stackrel{a}{\rightarrow} g$ such that $(qv, g) \in R$;*
- *for every $f \stackrel{a}{\rightarrow} g$, one of the following conditions hold:*
  - *$a = \tau$ and $(pw, g) \in R$;*
  - *there is a "$\stackrel{a}{\Rightarrow}$" move of pw which is R^{-1}-consistent with $f \stackrel{a}{\rightarrow} g$. Further, this move contains at most one transition of the form $r\mathcal{G} \stackrel{x}{\rightarrow} r\mathcal{H}$ (which can appear only at the end of the whole move).*

A function which to every $R \subseteq \mathcal{P}(\Delta, F) \times F$ assigns the set of all $(pw, f) \in R$ which quasi-expand in R is denoted by QExp.

We immediately obtain that *QExp* is monotonic, and if $QExp(R) = R$, then $R \subseteq \sim$. It remains to check the conditions (2), (4), and (5) of Definition 12.

Lemma 23. *Let $\sim$ be the relation of bisimilarity over $\mathcal{P}(\Delta, F) \times F$. Then $QExp(\sim) = \sim$.*

Proof. Let $(pw, f) \in \mathcal{P}(\Delta, F) \times F$ be a bisimilar pair of processes. We need to show that (pw, f) quasi-expands in $\sim$. Let $pw \stackrel{a}{\rightarrow} qv$. Since (pw, f) tightly b-expands in $\sim$ (see Lemma 20), there are two possibilities.

(a) $a = \tau$ and $qv \sim f$. Since the sequence consisting only of f is a $f \stackrel{\tau}{\Rightarrow} f$ move $\sim_F$-consistent with (f, τ, f), we obtain $f \stackrel{\tau}{\rightarrow} f$ due to the completeness of $\mathcal{T}$.

(b) There is a move $f \stackrel{a}{\Rightarrow} g$ tightly $\sim$-consistent with $pw \stackrel{a}{\rightarrow} qv$ (see Lemma 20). Then, the move $f \stackrel{a}{\Rightarrow} g$ is $\sim_F$-consistent with (f, a, g), and due to the completeness of $\mathcal{T}$ we obtain $f \stackrel{a}{\rightarrow} g$.

Now let $f \stackrel{a}{\to} g$. Since (pw, f) tightly b-expands in $\sim$ (see Lemma 20), we either have that $a = \tau$ and $pw \sim g$ (and we are done), or there is a move $pw \stackrel{a}{\Rightarrow} qv$ which is tightly $\sim^{-1}$-consistent with $f \stackrel{a}{\to} g$. The only problematic case not admitted by Definition 22 is when the move $pw \stackrel{a}{\Rightarrow} qv$ takes the form

$$p\alpha\mathcal{G} \stackrel{x_1}{\longrightarrow} \cdots \stackrel{x_n}{\longrightarrow} q\mathcal{G} = q\mathcal{G}_0 \stackrel{y_1}{\longrightarrow} \cdots \stackrel{y_m}{\longrightarrow} q\mathcal{G}_m = qv$$

where $m \geq 2$. However, the sequence $\mathcal{G}_0(q) \stackrel{y_1}{\to} \cdots \stackrel{y_m}{\to} \mathcal{G}_m(q)$ is then a "$\stackrel{x}{\Rightarrow}$" move $\sim_F$-consistent with $(\mathcal{G}_0(q), x, \mathcal{G}_m(q))$, where $x = a$ or $x = \tau$. Observe that here we need the *tight* $\sim^{-1}$-consistency of $pw \stackrel{a}{\Rightarrow} qv$ with $f \stackrel{a}{\to} g$. Thus, we obtain $\mathcal{G}_0(q) \stackrel{x}{\to} \mathcal{G}_m(q)$ because $\mathcal{T}$ is complete. Now it is easy to check that

$$p\alpha\mathcal{G} \stackrel{x_1}{\longrightarrow} \cdots \stackrel{x_n}{\longrightarrow} q\mathcal{G} = q\mathcal{G}_0 \stackrel{x}{\longrightarrow} q\mathcal{G}_m = qv$$

is a "$\stackrel{a}{\Rightarrow}$" move $\sim^{-1}$-consistent with $f \stackrel{a}{\to} g$. □

Lemma 24. *Let K be a well-formed set such that every $(X\mathcal{G}, \mathcal{F}) \in K$ is contained in $QExp(Gen(K))$. Then $QExp(Gen(K)) = Gen(K)$.*

Proof. By induction on i, we show that $Gen^i(K) \subseteq QExp(Gen(K))$ for all $i \geq 0$. In the base case (when $i = 0$), we need to consider pairs of the form

(A) $(p\varepsilon, \mathcal{F}(p))$ where $(\varepsilon, \mathcal{F}) \in K$ and $\mathcal{F}(p) \neq \bot$;
(B) $(p\mathcal{G}, \mathcal{F}(p))$ where $(\mathcal{G}, \mathcal{F}) \in K$ and $\mathcal{F}(p) \neq \bot$;
(C) $(pX\mathcal{G}, \mathcal{F}(p))$ where $(X\mathcal{G}, \mathcal{F}) \in K$ and $\mathcal{F}(p) \neq \bot$.

We start with (A). Observe that there is no transition of the form $p\varepsilon \stackrel{a}{\to} q\beta$. Now let $\mathcal{F}(p) \stackrel{a}{\to} g$. As $p\varepsilon \sim \mathcal{F}(p)$, there is a "$\stackrel{a}{\Rightarrow}$" move of $p\varepsilon$ which leads to a configuration bisimilar to g. Since the only "$\stackrel{a}{\Rightarrow}$" move of $p\varepsilon$ is $p\varepsilon \stackrel{\tau}{\Rightarrow} p\varepsilon$, we obtain $a = \tau$ and $p\varepsilon \sim g$. Then $(p\varepsilon, g) \in Gen(K)$ by Lemma 11(b), and we are done. Case (B) is also simple (we use Lemma 11(a)), and case (C) follows by applying the assumption of our lemma.

Now assume $(pw, f) \in Gen^{i+1}(K)$. If $(pw, f) \in Gen^i(K)$, we apply induction hypothesis. Otherwise, there are two possibilities (cf. the rules (2) and (3) of Definition 8):

(a) $pw = p\alpha$ where $\alpha \neq \varepsilon$, and there is $\mathcal{F}$ such that $(p\alpha\mathcal{F}, f) \in Gen^i(K)$ and $(\varepsilon, \mathcal{F}) \in K$.
(b) $pw = p\alpha X\mathcal{G}$ where $\alpha \neq \varepsilon$, and there is $\mathcal{F}$ such that $(p\alpha\mathcal{F}, f) \in Gen^i(K)$ and $(X\mathcal{G}, \mathcal{F}) \in K$.

We show that (pw, f) quasi-expands in $Gen(K)$. Consider Case (a). Here we need to show that if $(p\alpha\mathcal{F}, f)$ quasi-expands in $Gen(K)$ and $(\varepsilon, \mathcal{F}) \in K$, then $(p\alpha, f)$ quasi-expands in $Gen(K)$.

- Let $p\alpha \xrightarrow{a} r\beta$. Since $(p\alpha\mathcal{F}, f)$ quasi-expands in $Gen(K)$ and $p\alpha\mathcal{F} \xrightarrow{a} r\beta\mathcal{F}$, there is $f \xrightarrow{a} g$ such that $(r\beta\mathcal{F}, g) \in Gen(K)$. Since $(\varepsilon, \mathcal{F}) \in K$, we have that $(r\beta, g) \in Gen(K)$ as needed.
- Let $f \xrightarrow{a} g$. Since $(p\alpha\mathcal{F}, f)$ quasi-expands in $Gen(K)$, there is a move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ which is $Gen(K)^{-1}$-consistent with $f \xrightarrow{a} g$. We distinguish two cases.
  - The move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ takes the form

    $$p\alpha\mathcal{F} = p_0\gamma_0\mathcal{F} \xrightarrow{x_1} p_1\gamma_1\mathcal{F} \xrightarrow{x_2} \cdots \xrightarrow{x_n} p_n\gamma_n\mathcal{F} = rw$$

    where $n \geq 0$ and $\gamma_j \neq \varepsilon$ for all $j < n$. Then $p_0\gamma_0 \stackrel{x_1}{\Rightarrow} p_1\gamma_1 \stackrel{x_2}{\Rightarrow} \cdots \stackrel{x_n}{\Rightarrow} p_n\gamma_n$ is a "$\stackrel{a}{\Rightarrow}$" move of $p\alpha$ which is $Gen(K)^{-1}$-consistent with $f \xrightarrow{a} g$ because $(\varepsilon, \mathcal{F}) \in K$.
  - Otherwise, the move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ takes the form

    $$p\alpha\mathcal{F} = p_0\gamma_0\mathcal{F} \xrightarrow{x_1} \cdots \xrightarrow{x_{n-1}} p_{n-1}\gamma_{n-1}\mathcal{F} \xrightarrow{x_n} q\mathcal{F} \xrightarrow{x} q\mathcal{H} = rw$$

    where $n \geq 1$, $\gamma_j \neq \varepsilon$ for all $j < n$, and $(q\mathcal{H}, g) \in Gen(K)$. First, we show that $x = \tau$ and $q\mathcal{F} \sim q\mathcal{H} \sim q\varepsilon \sim g$. Since $(\varepsilon, \mathcal{F}) \in K$, $\mathcal{F} \in C(p\alpha)$, and $q \in M_{p\alpha}$, we have that $(q\varepsilon, \mathcal{F}(q))$ quasi-expands in $Gen(K)$. As $\mathcal{F}(q) \xrightarrow{x} \mathcal{H}(q)$, there is a "$\stackrel{x}{\Rightarrow}$" move of $q\varepsilon$ which is $Gen(K)^{-1}$-consistent with $\mathcal{F}(q) \xrightarrow{x} \mathcal{H}(q)$. The only candidate for this move is $q\varepsilon \stackrel{\tau}{\Rightarrow} q\varepsilon$, which means that $x = \tau$ and $(q\varepsilon, \mathcal{H}(q)) \in Gen(K)$. Hence, $q\varepsilon \sim \mathcal{H}(q) \sim q\mathcal{H}$, and since $(q\mathcal{H}, g) \in Gen(K)$, we further obtain $q\mathcal{H} \sim \mathcal{H}(q) \sim g$ (see Lemma 11).
    Since $q\varepsilon \sim g$, we have that $(q\varepsilon, g) \in Gen(K)$ due to Lemma 11(b). Now consider the sequence

    $$p\alpha = p_0\gamma_0 \xrightarrow{x_1} p_1\gamma_1 \xrightarrow{x_2} \cdots \xrightarrow{x_n} q\varepsilon$$

    If this sequence forms a "$\stackrel{a}{\Rightarrow}$" move which is $Gen(K)^{-1}$-consistent with $f \xrightarrow{a} g$, we are done. Otherwise, the style is early or branching, $x_i = \tau$ for all $1 \leq i \leq n$ (hence $a = \tau$), and $(q\mathcal{F}, f) \in Gen(K)$. But then $f \sim g$, and as $(p\alpha, f) \in Gen(K)$ (this is because $(p\alpha\mathcal{F}, f) \in Gen(K)$ and $(\varepsilon, \mathcal{F}) \in K$), we also obtain $(p\alpha, g) \in Gen(K)$ by Lemma 11(c). So, the condition of quasi-expansion is satisfied.

Now consider Case (b). We need to show that if $(p\alpha\mathcal{F}, f)$ quasi-expands in $Gen(K)$ and $(X\mathcal{G}, \mathcal{F}) \in K$, then $(p\alpha X\mathcal{G}, f)$ quasi-expands in $Gen(K)$. If $p\alpha X\mathcal{G} \xrightarrow{a} q\beta X\mathcal{G}$, then $p\alpha\mathcal{F} \xrightarrow{a} q\beta\mathcal{F}$ (recall $\alpha \neq \varepsilon$) and hence there is $f \xrightarrow{a} g$ such that $(q\beta\mathcal{F}, g) \in Gen(K)$. Then also $(q\beta X\mathcal{G}, g) \in Gen(K)$ because $(X\mathcal{G}, \mathcal{F}) \in K$. Now let $f \xrightarrow{a} g$. Then there is a move $p\alpha\mathcal{F} \overset{a}{\Rightarrow} rw$ which is $Gen(K)^{-1}$-consistent with $f \xrightarrow{a} g$. Similarly as in Case (b), we distinguish two cases.

- The move $p\alpha\mathcal{F} \overset{a}{\Rightarrow} rw$ takes the form

$$p\alpha\mathcal{F} = p_0\gamma_0\mathcal{F} \xrightarrow{x_1} p_1\gamma_1\mathcal{F} \xrightarrow{x_2} \cdots \xrightarrow{x_n} p_n\gamma_n\mathcal{F} = rw$$

  where $n \geq 0$ and $\gamma_j \neq \varepsilon$ for all $j < n$. Then $p_0\gamma_0 X\mathcal{G} \xrightarrow{x_1} \cdots \xrightarrow{x_n} p_n\gamma_n X\mathcal{G}$ is a "$\overset{a}{\Rightarrow}$" move of $p\alpha X\mathcal{G}$ which is $Gen(K)^{-1}$-consistent with $f \xrightarrow{a} g$ because $(X\mathcal{G}, \mathcal{F}) \in K$.
- The move $p\alpha\mathcal{F} \overset{a}{\Rightarrow} rw$ takes the form

$$p\alpha\mathcal{F} = p_0\gamma_0\mathcal{F} \xrightarrow{x_1} \cdots \xrightarrow{x_n} q\mathcal{F} \xrightarrow{x} q\mathcal{H} = rw$$

  where $n \geq 1$, $\gamma_j \neq \varepsilon$ for all $j < n$, and $(q\mathcal{H}, g) \in Gen(K)$. Since $(q X\mathcal{G}, \mathcal{F}(q))$ quasi-expands in $Gen(K)$, there is a move $qX\mathcal{G} \overset{x}{\Rightarrow} sv$ which is $Gen(K)^{-1}$-consistent with $\mathcal{F}(q) \xrightarrow{x} \mathcal{H}(q)$. Now it is easy to check that the sequence of transition obtained by concatenating $p\alpha X\mathcal{G} = p_0\gamma_0 X\mathcal{G} \xrightarrow{x_1} \cdots \xrightarrow{x_n} qX\mathcal{G}$ with the move $qX\mathcal{G} \overset{x}{\Rightarrow} sv$ forms a "$\overset{a}{\Rightarrow}$" move which is $Gen(K)^{-1}$-consistent with $f \xrightarrow{a} g$. □

Lemma 25. *The problem whether $(X\mathcal{G}, \mathcal{F})$ is contained in $QExp(Gen(K))$ for a given well-formed set K and a given pair $(X\mathcal{G}, \mathcal{F}) \in K$ is decidable in time polynomial in m, n, z.*

Proof. Let us fix some $p \in Q$ such that $\mathcal{F}(p) = f \neq \bot$. To decide whether $(pX\mathcal{G}, f)$ quasi-expands in $Gen(K)$, we need to verify the following conditions:

- For each $pX\mathcal{G} \xrightarrow{a} q\beta\mathcal{G}$ there is some $f \xrightarrow{a} g$ such that $(q\beta\mathcal{G}, g) \in Gen(K)$. However, it suffices check whether $f \xrightarrow{a} g$ for some $g \in F$ such that $q\beta\mathcal{G} \in \mathcal{L}(\mathcal{M}_{K,g})$, where $\mathcal{M}_{K,g}$ is the multi-automaton of Lemma 16. Obviously, this is achievable in time polynomial in m, n, z.
- For each $f \xrightarrow{a} g$, one of the following conditions is satisfied:

  (A) $a = \tau$ and $(pX\mathcal{G}, g) \in Gen(K)$;
  (B) there is a sequence $pX \overset{\tau}{\Rightarrow} q\alpha \xrightarrow{a} r\beta \overset{\tau}{\Rightarrow} s\gamma$ such that $(s\gamma\mathcal{G}, g) \in Gen(K)$ and

  $*$ if the style is early or branching, then $(q\alpha\mathcal{G}, f) \in Gen(K)$;

$*$ if the style is delay or branching, then $(r\beta\mathcal{G}, g) \in Gen(K)$;

(C) there exist a sequence $pX \stackrel{\tau}{\Rightarrow} q\alpha \stackrel{a}{\rightarrow} r\beta \stackrel{\tau}{\Rightarrow} s\varepsilon$ and $\mathcal{H}$ such that $s\mathcal{G} \stackrel{\tau}{\rightarrow} s\mathcal{H}$, $(s\mathcal{H}, g) \in Gen(K)$, and

$*$ if the style is early or branching, then $(q\alpha\mathcal{G}, f) \in Gen(K)$;
$*$ if the style is delay or branching, then $(r\beta\mathcal{G}, g) \in Gen(K)$;

(D) there exist a move $pX \stackrel{\tau}{\Rightarrow} s\varepsilon$ and $\mathcal{H}$ such that $s\mathcal{G} \stackrel{a}{\rightarrow} s\mathcal{H}$, $(s\mathcal{H}, g) \in Gen(K)$, and if the style is early or branching, then also $(s\mathcal{G}, f) \in Gen(K)$.

Condition (A) can be decided by checking whether $pX\mathcal{G} \in \mathcal{L}(\mathcal{M}_{K,g})$. Now consider Condition (B). Clearly, for every $h \in F$ there is a multi-automaton $\mathcal{M}^{\mathcal{G}}_{K,h}$ constructible in time polynomial in m, n, z recognizing the set

$$Gen^{\mathcal{G}}_h(K) = \{p\alpha \mid (p\alpha\mathcal{G}, h) \in Gen(K)\}.$$

Note that $\mathcal{M}^{\mathcal{G}}_{K,h}$ can be obtained by a trivial modification of $\mathcal{M}_{K,h}$. Depending on whether the style is weak, early, delay, or branching, Condition (B) can be reformulated as follows:

– $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}))) \cap Gen^{\mathcal{G}}_g(K) \neq \emptyset$;
– $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}) \cap Gen^{\mathcal{G}}_f(K))) \cap Gen^{\mathcal{G}}_g(K) \neq \emptyset$;
– $Post^*_\tau(Post_a(Post^*_\tau(\{pX\})) \cap Gen^{\mathcal{G}}_g(K)) \cap Gen^{\mathcal{G}}_g(K) \neq \emptyset$;
– $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}) \cap Gen^{\mathcal{G}}_f(K)) \cap Gen^{\mathcal{G}}_g(K)) \cap Gen^{\mathcal{G}}_g(K) \neq \emptyset$.

Due to Lemma 16 and Lemma 15, 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 can be implemented in time polynomial in m, n, z. Also observe that the last two lines can actually be simplified into

– $Post_a(Post^*_\tau(\{pX\})) \cap Gen^{\mathcal{G}}_g(K) \neq \emptyset$;
– $Post_a(Post^*_\tau(\{pX\}) \cap Gen^{\mathcal{G}}_f(K)) \cap Gen^{\mathcal{G}}_g(K) \neq \emptyset$.

Condition (C) is handled similarly. Let $T^{\mathcal{G}}_g(K)$ be the set of all $s\varepsilon$ such that $s\mathcal{G} \stackrel{\tau}{\rightarrow} s\mathcal{H}$ for some $\mathcal{H}$ satisfying $(s\mathcal{H}, g) \in Gen(K)$. Clearly, the set $T^{\mathcal{G}}_g(K)$ is constructible in time polynomial in m, n, z. Depending on whether the style is weak, early, delay, or branching, Condition (C) can now be stated as follows:

– $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}))) \cap T^{\mathcal{G}}_g(K) \neq \emptyset$;
– $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}) \cap Gen^{\mathcal{G}}_f(K))) \cap T^{\mathcal{G}}_g(K) \neq \emptyset$;
– $Post^*_\tau(Post_a(Post^*_\tau(\{pX\})) \cap Gen^{\mathcal{G}}_g(K)) \cap T^{\mathcal{G}}_g(K) \neq \emptyset$;

– $Post^*_\tau(Post_a(Post^*_\tau(\{pX\}) \cap Gen^{\mathcal{G}}_f(K)) \cap Gen^{\mathcal{G}}_g(K)) \cap T^{\mathcal{G}}_g(K) \neq \emptyset$.

Again, these conditions can be checked symbolically it time polynomial in m, n, z. Condition (D) can be reformulated and verified similarly. □

As a direct corollary to Lemmata 23, 24, and 25, we obtain the following theorem:

Theorem 26. *The problems of weak, early, delay, and branching bisimilarity between PDA processes and finite-state processes are decidable in time polynomial in m, n, z. For PDA^k processes, where $k \geq 1$ is a fixed constant, the problems are decidable in time polynomial in m, n.*

According to Theorem 26, bisimulation-like equivalences between PDA^k processes and finite-state processes are decidable in polynomial time. In particular, this holds for BPA (i.e., PDA^1) processes. Thus, we obtain a substantial generalization of the polynomial-time algorithm for deciding weak bisimilarity between BPA and finite-state processes presented in [45].

4.3. Simulation-Like Equivalences

In this section we show how to design an appropriate expansion for simulation-like equivalences. We have chosen *weak simulation equivalence* as a representative example.

Definition 27. *Let $\mathcal{T}_1 = (S_1, \rightarrow, \mathcal{A})$ and $\mathcal{T}_2 = (S_2, \rightarrow, \mathcal{A})$ be transition systems such that $S_1 \cap S_2 = \emptyset$, and let $R \subseteq S_1 \times S_2$. We say that R is a* weak simulation *if for all $(s,t) \in R$ and $s \xrightarrow{a} s'$ there is a move $t \stackrel{a}{\Rightarrow} t'$ such that $(s', t') \in R$.*

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 weakly simulate each other.*

Similarly as in Section 4.2, we need to assume that $\mathcal{T}$ is *complete* in the following sense: For all $f, g \in F$ and $a \in \mathcal{A}$ we have that if $f \stackrel{a}{\Rightarrow} g$, then also $f \xrightarrow{a} g$. Again, this assumption is not restrictive because the missing transitions can be added in polynomial time and each state of F stays weakly simulation equivalent to itself.

Note that if $pw \precsim f$, then $pw \sqsubseteq f$, $f \sqsubseteq pw$, and for every qv reachable from pw there is $\bar{g} \in F$ such that $qv \precsim \bar{g}$. Observe the following:

- If $pw \xrightarrow{a} qv$, there is a matching $f \stackrel{a}{\Rightarrow} g$ (and hence also $f \xrightarrow{a} g$) such that $qv \sqsubseteq g$. Further, there is $\bar{g} \in F$ such that $qv \precsim \bar{g}$, hence $\bar{g} \sqsubseteq g$. This is illustrated in Fig. 3 (left).

Figure 3: An expansion for weak simulation equivalence.

- If $f \xrightarrow{a} g$, there is a matching $pw \stackrel{a}{\Rightarrow} qv$ such that $g \sqsubseteq qv$. Further, there is $\bar{g} \in F$ such that $qv \precsim \bar{g}$, hence $g \sqsubseteq \bar{g}$. This is illustrated in Fig. 3 (right).

The above properties are directly reflected in the expansion condition for simulation equivalence.

Definition 28. *Let $R \subseteq \mathcal{P}(\Delta, F) \times F$ be a relation. We say that a pair $(pw, f) \in R$* sim-expands *in R if the following conditions are satisfied:*

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

A function which to every $R \subseteq \mathcal{P}(\Delta, F) \times F$ assigns the set of all $(pw, f) \in R$ which sim-expand in R is denoted by $SExp(R)$.

The function *SExp* is clearly monotonic. Further, it is easy to check that $SExp(\precsim) = \precsim$. It remains to verify the conditions (3), (4), and (5) of Definition 12.

Lemma 29. *Let $R \subseteq \mathcal{P}(\Delta, F) \times F$ such that $SExp(R) = R$. Then $R \subseteq \precsim$.*

Proof. Let us fix some $R \subseteq \mathcal{P}(\Delta, F) \times F$ such that $SExp(R) = R$. It suffices to show $R \subseteq \sim$, because for every $(pw, f) \in R$ and every qv reachable from pw there clearly exists $g \in F$ such that $(qv, g) \in R$ (cf. the first item of Definition 28). Let

- $R_{\sqsubseteq} = \{(pw, f) \mid$ there is $\bar{f} \in F$ such that $(pw, \bar{f}) \in R$ and $\bar{f} \sqsubseteq f\}$;
- $R_{\sqsupseteq} = \{(pw, f) \mid$ there is $\bar{f} \in F$ such that $(pw, \bar{f}) \in R$ and $f \sqsubseteq \bar{f}\}$.

Clearly, the relations $R_{\sqsubseteq}$ and $R_{\sqsupseteq}$ subsume R, and it is straightforward to check that both $R_{\sqsubseteq}$ and $R_{\sqsupseteq}^{-1}$ are weak simulations, which implies $R \subseteq \sim$. □

Lemma 30. *Let K be a well-formed set such that every $(X\mathcal{G}, \mathcal{F}) \in K$ is contained in $SExp(Gen(K))$. Then $SExp(Gen(K)) = Gen(K)$.*

PROOF. By induction on i, we show that $Gen^i(K) \subseteq SExp(Gen(K))$ for all $i \geq 0$. The base case (when $i = 0$) is similar as in the proof of Lemma 24. Now assume $(pw, f) \in Gen^{i+1}(K)$. If $(pw, f) \in Gen^i(K)$, we apply induction hypothesis. Otherwise, there are two possibilities (cf. the rules (2) and (3) of Definition 8):

(a) $pw = p\alpha$ where $\alpha \neq \varepsilon$, and there is $\mathcal{F}$ such that $(p\alpha\mathcal{F}, f) \in Gen^i(K)$ and $(\varepsilon, \mathcal{F}) \in K$.
(b) $pw = p\alpha X\mathcal{G}$ where $\alpha \neq \varepsilon$, and there is $\mathcal{F}$ such that $(p\alpha\mathcal{F}, f) \in Gen^i(K)$ and $(X\mathcal{G}, \mathcal{F}) \in K$.

We show that (pw, f) sim-expands in $Gen(K)$. In Case (a), we need to show that if $(p\alpha\mathcal{F}, f)$ sim-expands in $Gen(K)$ and $(\varepsilon, \mathcal{F}) \in K$, then $(p\alpha, f)$ sim-expands in $Gen(K)$.

- Let $p\alpha \xrightarrow{a} r\beta$. Since $(p\alpha\mathcal{F}, f)$ sim-expands in $Gen(K)$ and $p\alpha\mathcal{F} \xrightarrow{a} r\beta\mathcal{F}$, there are $\bar{g} \in F$ and $f \xrightarrow{a} g$ such that $(r\beta\mathcal{F}, \bar{g}) \in Gen(K)$ and $\bar{g} \sqsubseteq g$. Since $(\varepsilon, \mathcal{F}) \in K$, we have that $(r\beta, \bar{g}) \in Gen(K)$ as needed.
- Let $f \xrightarrow{a} g$. Since $(p\alpha\mathcal{F}, f)$ sim-expands in $Gen(K)$, there are $\bar{g} \in F$ and a move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ such that $(rw, \bar{g}) \in Gen(K)$ and $g \sqsubseteq \bar{g}$. We distinguish two cases.
  - The move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ is of the form $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} r\beta\mathcal{F}$ where $p\alpha \stackrel{a}{\Rightarrow} r\beta$. Since $(r\beta\mathcal{F}, \bar{g}) \in Gen(K)$ and $(\varepsilon, \mathcal{F}) \in K$, we obtain $(r\beta, \bar{g}) \in Gen(K)$ as needed.
  - Otherwise, the move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ takes the form $p\alpha\mathcal{F} \stackrel{x}{\Rightarrow} r\mathcal{F} \stackrel{y}{\Rightarrow} r\mathcal{H}$ where $p\alpha \stackrel{x}{\Rightarrow} r\varepsilon$. Now it suffices to show that $x = a$ and $r\varepsilon \sim \bar{g}$, because then also $r\varepsilon \precsim \bar{g}$ and hence $(r\varepsilon, \bar{g}) \in Gen(K)$ by Lemma 11(b). Since $(\varepsilon, \mathcal{F}) \in K$ and $\mathcal{F}(r) \neq \bot$ (this is because $\mathcal{F} \in C(p\alpha)$ and $r \in M_{p\alpha}$), we obtain $r\varepsilon \sim \mathcal{F}(r) \sim r\mathcal{F}$. This implies that every process reachable from $r\mathcal{F}$ can execute only τ-labeled transitions, and hence it is weakly simulation equivalent to $r\varepsilon$. In particular, $y = \tau$ (hence $x = a$) and $r\varepsilon \sim r\mathcal{H} \sim \bar{g}$ as needed.

Now consider Case (b). We need to show that if $(p\alpha\mathcal{F}, f)$ sim-expands in $Gen(K)$ and $(X\mathcal{G}, \mathcal{F}) \in K$, then $(p\alpha X\mathcal{G}, f)$ sim-expands in $Gen(K)$. If $p\alpha X\mathcal{G} \xrightarrow{a} q\beta X\mathcal{G}$, then $p\alpha\mathcal{F} \xrightarrow{a} q\beta\mathcal{F}$ (recall $\alpha \neq \varepsilon$) and hence there are $\bar{g} \in F$ and $f \xrightarrow{a} g$ such that $(q\beta\mathcal{F}, \bar{g}) \in Gen(K)$ and $\bar{g} \sqsubseteq g$. Since $(X\mathcal{G}, \mathcal{F}) \in K$, we have that $(q\beta X\mathcal{G}, \bar{g}) \in Gen(K)$ as needed. Now let $f \xrightarrow{a} g$. Since $(p\alpha\mathcal{F}, f)$ sim-expands in $Gen(K)$, there are $\bar{g} \in F$ and a move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ such that $(rw, \bar{g}) \in Gen(K)$ and $g \sqsubseteq \bar{g}$. There are two possibilities.

- The move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ is of the form $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} r\beta\mathcal{F}$ where $p\alpha \stackrel{a}{\Rightarrow} r\beta$. Then $p\alpha X\mathcal{G} \stackrel{a}{\Rightarrow} r\beta X\mathcal{G}$, and since $(r\beta\mathcal{F}, \bar{g}) \in Gen(K)$ and $(X\mathcal{G}, \mathcal{F}) \in K$, we obtain $(r\beta X\mathcal{G}, \bar{g}) \in Gen(K)$ as needed.

- The move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ takes the form $p\alpha\mathcal{F} \stackrel{x}{\Rightarrow} r\mathcal{F} \stackrel{y}{\Rightarrow} r\mathcal{H}$ where $p\alpha X\mathcal{G} \stackrel{x}{\Rightarrow} rX\mathcal{G}$ and $(r\mathcal{H}, \bar{g}) \in Gen(K)$. Due to Lemma 11(a), we have that $r\mathcal{H} \sim \bar{g}$, hence $g \sqsubseteq \bar{g} \sqsubseteq r\mathcal{H} \sqsubseteq \mathcal{H}(r)$. Since $r\mathcal{F} \stackrel{y}{\Rightarrow} r\mathcal{H}$, we also have $\mathcal{F}(r) \stackrel{y}{\Rightarrow} \mathcal{H}(r)$, hence $\mathcal{F}(r) \stackrel{y}{\rightarrow} \mathcal{H}(r)$ because $\mathcal{T}$ is complete. Since $(X\mathcal{F}, \mathcal{F})$ is contained in $SExp(Gen(K))$, the pair $(rX\mathcal{G}, \mathcal{F}(r))$ sim-expands in $Gen(K)$, and therefore there exist $\bar{h} \in F$ and a move $rX\mathcal{G} \stackrel{y}{\Rightarrow} qv$ such that $(qv, \bar{h}) \in Gen(K)$ and $\mathcal{H}(r) \sqsubseteq \bar{h}$. Hence, $p\alpha X\mathcal{G} \stackrel{x}{\Rightarrow} rX\mathcal{G} \stackrel{y}{\Rightarrow} qv$, where $(qv, \bar{h}) \in Gen(K)$ and $g \sqsubseteq \mathcal{H}(r) \sqsubseteq \bar{h}$. □

Given a well-formed set K and a pair $(X\mathcal{G}, \mathcal{F}) \in K$, the problem whether $(X\mathcal{G}, \mathcal{F})$ is contained in $SExp(Gen(K))$ can be decided in time polynomial in m, n, z by using the same technique as in Lemma 25. That is, we use Lemma 16 and Lemma 15 to check the required conditions symbolically. Thus, we obtain the following:

Theorem 31. *The problem of full weak simulation equivalence between PDA and finite-state processes is decidable in time polynomial in m, n, z. For PDA^k processes, where $k \geq 1$ is a fixed constant, the problem is decidable in time polynomial in m, n.*

Let us note that the problem of checking weak (and also strong) simulation equivalence between PDA and finite-state processes is **EXPTIME**-complete, and the **EXPTIME**-hardness holds even for BPA (i.e., PDA^1) processes [46].

4.4. Trace-Like Equivalences

In this section we consider trace-like equivalences. We show how to design an appropriate expansion for *weak trace equivalence*.

Definition 32. *Let $\mathcal{T} = (S, \mathcal{A}, \rightarrow)$ be a transition system. For all $s, t \in S$ and all finite words $x = a_1 \ldots a_k \in \mathcal{A}^*$ (where $k \geq 0$), we write $s \stackrel{x}{\Rightarrow} t$ if there are $s_0, \ldots, s_k \in S$ such that $s = s_0$, $t = s_k$, and $s_{i-1} \stackrel{a_i}{\Longrightarrow} s_i$ for all $1 \leq i \leq k$. A* trace *of $s \in S$ is a word $x \in \mathcal{A}^*$ such that $s \stackrel{x}{\Rightarrow} t$ for some $t \in S$. The set of all traces of s is denoted by $Tr(s)$. Processes s, t are* weakly trace equivalent*, written $s \sim t$, if $Tr(s) = Tr(t)$.*

To get some intuition behind the next definition, realize that if $pw \precsim f$, the following conditions are satisfied:

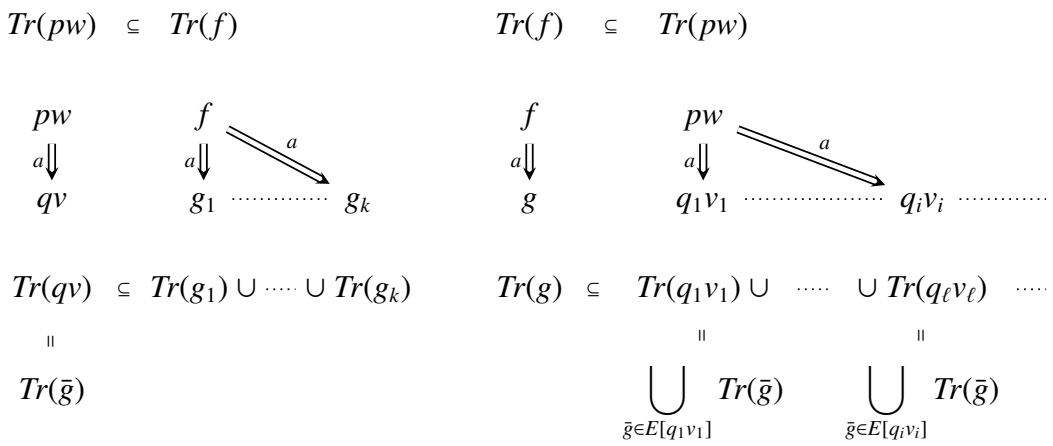

Figure 4: An expansion for weak trace equivalence.

- If $pw \stackrel{a}{\Rightarrow} qv$, there is $\bar{g} \in F$ such that $qv \precsim \bar{g}$. Further, each trace of qv is a trace of some $g \in F$ such that $f \stackrel{a}{\Rightarrow} g$. Hence,

$$Tr(qv) \quad = \quad Tr(\bar{g}) \quad \subseteq \quad \bigcup_{f \stackrel{a}{\Rightarrow} g} Tr(g)\,.$$

  See Fig. 4 (left).
- If $f \stackrel{a}{\Rightarrow} g$, then $Tr(g) \subseteq \bigcup_{pw \stackrel{a}{\Rightarrow} qv} Tr(qv)$. Further, for every qv such that $pw \stackrel{a}{\Rightarrow} qv$, the set $E[qv] = \{\bar{g} \in F \mid qv \precsim \bar{g}\}$ is non-empty. We have that

$$Tr(g) \quad \subseteq \quad \bigcup_{pw \stackrel{a}{\Rightarrow} qv} Tr(qv) \quad = \quad \bigcup_{pw \stackrel{a}{\Rightarrow} qv} \bigcup_{\bar{g} \in E[qv]} Tr(\bar{g})\,.$$

  See Fig. 4 (right). Note that all $\bar{g} \in E[qv]$ have the same set of traces, so the above inclusion holds even if we used just one representative of each $E[qv]$. For our purposes (see Definition 33), it is more convenient to consider the union $\bigcup_{\bar{g} \in E[qv]} Tr(\bar{g})$.

Definition 33. *Let $R \subseteq \mathcal{P}(\Delta, F) \times F$ be a relation. For every $pw \in \mathcal{P}(\Delta, F)$, we define the set $R[pw] = \{g \in F \mid (pw, g) \in R\}$.*

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

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

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

A function which to every $R \subseteq \mathcal{P}(\Delta, F) \times F$ *assigns the set of all* $(pw, f) \in R$ *which trace-expand in* R *is denoted by* $TExp(R)$.

The monotonicity of $TExp$ and the equality $TExp(\precsim) = \precsim$ are easy to verify. We prove the remaining properties of Definition 12.

Lemma 34. *Let* $R \subseteq \mathcal{P}(\Delta, F) \times F$ *such that* $TExp(R) = R$. *Then* $R \subseteq \precsim$.

Proof. Let $R \subseteq \mathcal{P}(\Delta, F) \times F$ such that $TExp(R) = R$. It suffices to show $R \subseteq \sim$ (then also $R \subseteq \precsim$ due to the first item of Definition 33). By induction on the length of $x \in \mathcal{A}^*$, we show that for all $(pw, f) \in R$ we have that $x \in Tr(pw)$ iff $x \in Tr(f)$. The base case when $x = \varepsilon$ is immediate, because ε is a trace of every process. Now let $x = ay$ where $a \in \mathcal{A}$, and let $(pw, f) \in R$.

- If $ay \in Tr(pw)$, there are qv and ru such that $pw \stackrel{a}{\Rightarrow} qv \stackrel{y}{\Rightarrow} ru$. Since the pair (pw, f) trace-expands in R, there is $\bar{g} \in F$ such that $(qv, \bar{g}) \in R$ (hence, $y \in Tr(\bar{g})$ by induction hypothesis) and $Tr(\bar{g}) \subseteq \bigcup_{f \stackrel{a}{\Rightarrow} g} Tr(g)$. This implies $y \in Tr(g)$ for some $g \in F$ such that $f \stackrel{a}{\Rightarrow} g$, which means $ay \in Tr(f)$.
- If $ay \in Tr(f)$, there are $g, h \in F$ such that $f \stackrel{a}{\Rightarrow} g \stackrel{y}{\Rightarrow} h$. Since the pair (pw, f) trace-expands in R, we have that $y \in \bigcup_{pw \stackrel{a}{\Rightarrow} qv} \bigcup_{\bar{g} \in R[qv]} Tr(\bar{g})$. Hence, there exist qv and $\bar{g}$ such that $pw \stackrel{a}{\Rightarrow} qv$, $(qv, \bar{g}) \in R$, and $y \in Tr(\bar{g})$. By induction hypothesis, we obtain $y \in Tr(qv)$, hence $ay \in Tr(pw)$. □

Lemma 35. *Let* K *be a well-formed set such that every* $(X\mathcal{G}, \mathcal{F}) \in K$ *is contained in* $TExp(Gen(K))$. *Then* $TExp(Gen(K)) = Gen(K)$.

Proof. By induction on i, we show that $Gen^i(K) \subseteq TExp(Gen(K))$ for all $i \geq 0$. The base case (when $i = 0$) is similar as in the proof of Lemma 24. Now assume $(pw, f) \in Gen^{i+1}(K)$. If $(pw, f) \in Gen^i(K)$, we apply induction hypothesis. Otherwise, there are two possibilities (cf. the rules (2) and (3) of Definition 8):

(a) $pw = p\alpha$ where $\alpha \neq \varepsilon$, and there is $\mathcal{F}$ such that $(p\alpha\mathcal{F}, f) \in Gen^i(K)$ and $(\varepsilon, \mathcal{F}) \in K$.
(b) $pw = p\alpha X\mathcal{G}$ where $\alpha \neq \varepsilon$, and there is $\mathcal{F}$ such that $(p\alpha\mathcal{F}, f) \in Gen^i(K)$ and $(X\mathcal{G}, \mathcal{F}) \in K$.

We show that (pw, f) trace-expands in $Gen(K)$. In Case (a), we need to show that if $(p\alpha\mathcal{F}, f)$ trace-expands in $Gen(K)$ and $(\varepsilon, \mathcal{F}) \in K$, then $(p\alpha, f)$ trace-expands in $Gen(K)$.

- Let $p\alpha \stackrel{a}{\Rightarrow} q\beta$. Since $(p\alpha\mathcal{F}, f)$ trace-expands in $Gen(K)$ and $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} q\beta\mathcal{F}$, there is $\bar{g} \in Gen(K)[q\beta\mathcal{F}]$ such that $Tr(\bar{g}) \subseteq \bigcup_{f \stackrel{a}{\Rightarrow} g} Tr(g)$. Since $(\varepsilon, \mathcal{F}) \in K$, we have that $\bar{g} \in Gen(K)[q\beta]$ as needed.
- Let $f \stackrel{a}{\Rightarrow} g$. Since $(p\alpha\mathcal{F}, f)$ trace-expands in $Gen(K)$, we have that $Tr(g) \subseteq \bigcup_{p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv} \bigcup_{\bar{g} \in Gen(K)[qv]} Tr(\bar{g})$. Now it suffices to show that for every move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv$ there is a move $p\alpha \stackrel{a}{\Rightarrow} qu$ such that, for every $\bar{g} \in F$, $(qv, \bar{g}) \in Gen(K)$ implies $(qu, \bar{g}) \in Gen(K)$. So, let us fix a move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv$. We distinguish two cases.
  - The move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv$ takes the form $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} q\beta\mathcal{F}$, where $p\alpha \stackrel{a}{\Rightarrow} q\beta$. Since $(q\beta\mathcal{F}, \bar{g}) \in Gen(K)$ implies $(q\beta, \bar{g}) \in Gen(K)$ (because $(\varepsilon, \mathcal{F}) \in K$), we are done.
  - The move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} rw$ takes the form $p\alpha\mathcal{F} \stackrel{x}{\Rightarrow} q\mathcal{F} \stackrel{y}{\Rightarrow} q\mathcal{H}$ where $p\alpha \stackrel{x}{\Rightarrow} q\varepsilon$ and $x, y \in \{\tau, a\}$. It suffices to show that $x = a$, and if $(q\mathcal{H}, \bar{g}) \in Gen(K)$, then also $(q\varepsilon, \bar{g}) \in Gen(K)$. Since $(\varepsilon, \mathcal{F}) \in K$ and $\mathcal{F}(q) \neq \varepsilon$, we obtain $q\varepsilon \sim \mathcal{F}(q) \sim q\mathcal{F}$. This implies that every process reachable from $q\mathcal{F}$ can execute only τ-labeled transitions, and hence it is weakly trace equivalent to $q\varepsilon$. In particular, $y = \tau$ (hence $x = a$), and $q\varepsilon \sim q\mathcal{H}$. Now suppose $(q\mathcal{H}, \bar{g}) \in Gen(K)$. Then $q\mathcal{H} \precsim \bar{g}$ (see Lemma 11(a)), hence $q\varepsilon \precsim \bar{g}$, and $(q\varepsilon, \bar{g}) \in Gen(K)$ by Lemma 11(b).

In Case (b), we need to show that if $(p\alpha\mathcal{F}, f)$ trace-expands in $Gen(K)$ and $(X\mathcal{G}, \mathcal{F}) \in K$, then $(p\alpha X\mathcal{G}, f)$ trace-expands in $Gen(K)$. Let $p\alpha X\mathcal{G} \stackrel{a}{\Rightarrow} qv$. There are two possibilities.

- The move $p\alpha X\mathcal{G} \stackrel{a}{\Rightarrow} qv$ takes the form $p\alpha X\mathcal{G} \stackrel{a}{\Rightarrow} q\beta X\mathcal{G}$ where $p\alpha \stackrel{a}{\Rightarrow} q\beta$. Then $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} q\beta\mathcal{F}$ and hence there is $\bar{g} \in Gen(K)[q\beta\mathcal{F}]$ such that $Tr(\bar{g}) \subseteq \bigcup_{f \stackrel{a}{\Rightarrow} g} Tr(g)$. Since $(X\mathcal{G}, \mathcal{F}) \in K$, we have that $\bar{g} \in Gen(K)[q\beta X\mathcal{G}]$ as needed.
- The move $p\alpha X\mathcal{G} \stackrel{a}{\Rightarrow} qv$ takes the form $p\alpha X\mathcal{G} \stackrel{x}{\Rightarrow} rX\mathcal{G} \stackrel{y}{\Rightarrow} qv$, where $p\alpha \stackrel{x}{\Rightarrow} r\varepsilon$ and $x, y \in \{\tau, a\}$. Then $p\alpha\mathcal{F} \stackrel{x}{\Rightarrow} r\mathcal{F}$ and since $(p\alpha\mathcal{F}, f)$ trace-expands in $Gen(K)$, there is $\bar{\ell} \in Gen(K)[r\mathcal{F}]$ such that $Tr(\bar{\ell}) \subseteq \bigcup_{f \stackrel{x}{\Rightarrow} \ell} Tr(\ell)$. By Lemma 11(a), we have that $\mathcal{F}(r) \sim r\mathcal{F} \sim \bar{\ell}$. Further, since $(rX\mathcal{G}, \mathcal{F}(r))$ trace-expands in $Gen(K)$ and $rX\mathcal{G} \stackrel{y}{\Rightarrow} qv$, there is $\bar{h} \in Gen(K)[qv]$ such that $Tr(\bar{h}) \subseteq \bigcup_{\mathcal{F}(r) \stackrel{y}{\Rightarrow} h} Tr(h)$. Hence, it suffices to show $Tr(\bar{h}) \subseteq \bigcup_{f \stackrel{a}{\Rightarrow} g} Tr(g)$. However, by applying the above inclusions we immediately obtain

$$Tr(\bar{h}) \subseteq \bigcup_{\mathcal{F}(r) \stackrel{y}{\Rightarrow} h} Tr(h) = \bigcup_{\bar{\ell} \stackrel{y}{\Rightarrow} h} Tr(h) \subseteq \bigcup_{f \stackrel{x}{\Rightarrow} \ell} \bigcup_{\ell \stackrel{y}{\Rightarrow} h} Tr(h) \subseteq \bigcup_{f \stackrel{a}{\Rightarrow} g} Tr(g).$$

Now let $f \stackrel{a}{\Rightarrow} g$. As $(p\alpha\mathcal{F}, f)$ trace-expands in $Gen(K)$, we obtain

$$Tr(g) \quad \subseteq \bigcup_{p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv} \; \bigcup_{\bar{g} \in Gen(K)[qv]} Tr(\bar{g}) \,.$$

Hence, it suffices to show that for every $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv$ and every $\bar{g} \in Gen(K)[qv]$,

$$Tr(\bar{g}) \quad \subseteq \bigcup_{p\alpha X\mathcal{G} \stackrel{a}{\Rightarrow} ru} \; \bigcup_{\bar{h} \in Gen(K)[ru]} Tr(\bar{h}) \,.$$

So, let us fix some $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv$ and $\bar{g} \in Gen(K)[qv]$. There are two possibilities.

- The move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv$ is of the form $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} q\beta\mathcal{F}$ where $p\alpha \stackrel{a}{\Rightarrow} q\beta$. Then $p\alpha X\mathcal{G} \stackrel{a}{\Rightarrow} q\beta X\mathcal{G}$, and since $(q\beta\mathcal{F}, \bar{g}) \in Gen(K)$ implies $(q\beta X\mathcal{G}, \bar{g}) \in Gen(K)$ (because $(X\mathcal{G}, \mathcal{F}) \in K$), we are done.
- The move $p\alpha\mathcal{F} \stackrel{a}{\Rightarrow} qv$ takes the form $p\alpha\mathcal{F} \stackrel{x}{\Rightarrow} q\mathcal{F} \stackrel{y}{\Rightarrow} q\mathcal{H}$ where $p\alpha X\mathcal{G} \stackrel{x}{\Rightarrow} qX\mathcal{G}$, $\mathcal{F}(q) \stackrel{y}{\Rightarrow} \mathcal{H}(q)$, and $x, y \in \{\tau, a\}$. Since $(q\mathcal{H}, \bar{g}) \in Gen(K)$, we have $\mathcal{H}(q) \sim q\mathcal{H} \sim \bar{g}$ by Lemma 11(a). As $\mathcal{F}(q) \stackrel{a}{\Rightarrow} \mathcal{H}(q)$ and $(qX\mathcal{G}, \mathcal{F}(q))$ trace-expands in $Gen(K)$, we obtain

  $$Tr(\mathcal{H}(q)) \quad \subseteq \bigcup_{qX\mathcal{G} \stackrel{y}{\Rightarrow} ru} \; \bigcup_{\bar{h} \in Gen(K)[ru]} Tr(\bar{h}) \,.$$

  Thus,

  $$Tr(\bar{g}) \;=\; Tr(\mathcal{H}(q)) \;\subseteq \bigcup_{qX\mathcal{G} \stackrel{y}{\Rightarrow} ru} \; \bigcup_{\bar{h} \in Gen(K)[ru]} Tr(\bar{h}) \;\subseteq \bigcup_{p\alpha X\mathcal{G} \stackrel{a}{\Rightarrow} ru} \; \bigcup_{\bar{h} \in Gen(K)[ru]} Tr(\bar{h}) \,.$$

□

Since the trace equivalence problem for $\mathcal{T}$ is **PSPACE**-complete[9], the problem of checking full weak trace equivalence between PDA and finite-state processes is **PSPACE**-hard even for BPA (i.e., PDA^1) processes. We prove the following:

[9]Trace equivalence is defined similarly as weak trace equivalence. The only difference is that τ is treated as an "ordinary" action; a *trace* of a given process s is a sequence $w \in \mathcal{A}^*$ such that $s \stackrel{w}{\rightarrow} t$ for some t. The **PSPACE**-hardness of trace equivalence for finite-state processes follows immediately from **PSPACE**-completeness of language inclusion/equivalence problem for non-deterministic finite automata; see, e.g., [30].

Lemma 36. *The problem whether $(X\mathcal{G},\mathcal{F})$ is contained in $TExp(Gen(K))$ for a given well-formed set K and a given pair $(X\mathcal{G},\mathcal{F}) \in K$ is decidable in space polynomial in m, n, z.*

Proof. Let K be a well-formed set, $(X\mathcal{G},\mathcal{F}) \in K$, and $p \in F$ such that $\mathcal{F}(p) \neq \bot$. To decide whether $(pX\mathcal{G},\mathcal{F}(p))$ trace-expands in $TExp(K)$, we need to check the following conditions:

- For all $a \in \mathcal{A}$ and $pX\mathcal{G} \stackrel{a}{\Rightarrow} qv$, there is $\bar{g} \in Gen(K)[qv]$ such that $Tr(\bar{g}) \subseteq \bigcup_{f \stackrel{a}{\Rightarrow} g} Tr(g)$. For a given $a \in \mathcal{A}$, this condition can be verified as follows. First, we compute the set $\bar{G} \subseteq F$ of all $\bar{g}$ such that $Tr(\bar{g}) \subseteq \bigcup_{f \stackrel{a}{\Rightarrow} g} Tr(g)$. This is achievable in space polynomial in m. Then, for each $\bar{g} \in \bar{G}$, we compute a multi-automaton $\mathcal{M}[\bar{g}]$ recognizing the set $\{qv \mid pX\mathcal{G} \stackrel{a}{\Rightarrow} qv \text{ and } (qv,\bar{g}) \in Gen(K)\}$. We also compute a multi-automaton $\mathcal{M}[a]$ recognizing the set $\{qv \mid pX\mathcal{G} \stackrel{a}{\Rightarrow} qv\}$. Clearly, these multi-automata are constructible in time polynomial in m, n, z, and their size is polynomial in m, n, z. Now we check whether $\mathcal{L}(\mathcal{M}[a]) \subseteq \bigcup_{\bar{g} \in \bar{G}} \mathcal{L}(\mathcal{M}[\bar{g}])$, which is achievable in space polynomial in m, n, z.
- For all $a \in \mathcal{A}$ and $f \stackrel{a}{\Rightarrow} g$ we have that $Tr(g) \subseteq \bigcup_{pX\mathcal{G} \stackrel{a}{\Rightarrow} qv} \bigcup_{\bar{g} \in Gen(K)[qv]} Tr(\bar{g})$. Here we use Lemma 16 and Lemma 15 to construct the set

  $$\bar{G} = \{\bar{g} \in F \mid (qv,\bar{g}) \in Gen(K) \text{ for some } qv \text{ such that } pX\mathcal{G} \stackrel{a}{\Rightarrow} qv\}$$

  and then check whether $Tr(g) \subseteq \bigcup_{\bar{g} \in \bar{G}} Tr(\bar{g})$. Obviously, this is achievable in space polynomial in m, n, z. □

Theorem 37. *The problem of full weak trace equivalence between PDA and finite-state processes is decidable in space polynomial in m, n, z. For PDA^k processes, where $k \geq 1$ is a fixed constant, the problem is decidable in space polynomial in m, n. Moreover, the problem is **PSPACE**-hard even for BPA processes.*

Note that checking trace-like equivalences between BPA and finite-state systems is undecidable (this follows easily from the undecidability of the universality problem for context-free grammars, i.e., the question whether $L(G) = \Sigma^*$ for a given CFG; see, e.g., [30]).

Remark 38. *It is worth noting that the problem of full* language *equivalence between PDA and finite-state processes is easily reducible to the problem of full weak trace equivalence. Let $\Delta = (Q,\Gamma,\mathcal{A},\delta)$ be a PDA (where the τ-labeled rules correspond to the ε-moves, cf. [30]) which accepts by empty stack, i.e., the*

language accepted by a given configuration $p\alpha$ consists of all $w \in \mathcal{A}^*$ such that $p\alpha \stackrel{w}{\Rightarrow} q\varepsilon$ for some $q \in Q$. Further, let $\mathcal{T} = (F, \mathcal{A}, \rightarrow)$ be a finite-state system which accepts by entering a final state $g \in F$. We construct another PDA Δ' and a finite-state system $\mathcal{T}'$ as follows:

- $\Delta' = (Q \cup \{p_u\}, \Gamma \cup \{Z\}, \mathcal{A} \cup \{\#\}, \delta')$, *where p_u, Z, $\#$ are fresh symbols, and δ' contains all rules of δ together with*
  - $pX \stackrel{a}{\rightarrow} p_uX$ *for all $p \in Q$, $X \in \Gamma \cup \{Z\}$, and $a \in \mathcal{A}$;*
  - $p_uX \stackrel{a}{\rightarrow} p_uX$ *for all $X \in \Gamma \cup \{Z\}$ and $a \in \mathcal{A}$;*
  - $pZ \stackrel{\#}{\rightarrow} p\varepsilon$ *for all $p \in Q$.*

  The stack symbol Z is used as the bottom-of-the-stack marker, and the action $\#$ marks the end of an accepted word. It is easy to check that for every configuration $p\alpha$ of Δ and every $w \in \mathcal{A}^*$ we have that $p\alpha$ accepts w iff the configuration $p\alpha Z$ of Δ' has a trace $w\#$. Further, observe that every $v \in \mathcal{A}^*$ is a trace of $p\alpha Z$.
- $\mathcal{T}' = (F \cup \{f_\#, f_u\}, \mathcal{A} \cup \{\#\}, \rightarrowtail)$ *where $f_\#, f_u$ are fresh states, and $\rightarrowtail$ contains all transitions of $\rightarrow$ together with*
  - $g \stackrel{\#}{\rightarrowtail} f_\#$;
  - $f \stackrel{a}{\rightarrowtail} f_u$ *for all $f \in F$ and $a \in \mathcal{A}$;*
  - $f_u \stackrel{a}{\rightarrowtail} f_u$ *for all $a \in \mathcal{A}$.*

Now it is easy to check that for every configuration $p\alpha$ of Δ and every state f of $\mathcal{T}$ we have that $p\alpha$ is fully language equivalent to f iff the configuration $p\alpha Z$ of Δ' is fully weak trace equivalent to the state f of $\mathcal{T}'$.

5. Conclusions

We have shown that the problem of checking full regular equivalence with PDA processes is decidable for selected conceptual representatives of the linear/branching time spectrum. For bisimulation and simulation-like equivalences, our algorithm is *polynomial* if the number of control states in PDA is bounded by some fixed constant. Since we aimed mainly at demonstrating the versatility and efficiency of the designed method, we have not paid much attention to the implementation details and performed only a rough complexity analysis. Nevertheless, this is sufficient for separating the problems solvable in polynomial time from the computationally hard ones.

A crucial parameter influencing the complexity of our algorithm is the number of control states of Δ (recall $z = |F|^{|Q|}$). A closer look reveals that we can actually refine z into $|F|^{Ret}$, where $Ret = \max\{|M_{pX}| \mid pX \in Q \times \Gamma\}$. Intuitively,

Ret represents the maximal amount of information returned by a procedure call in the recursive program represented by Δ. We can easily modify Δ so that for every $pX \in Q \times \Gamma$ we have that $M_{pX} \subseteq \{f_1, \ldots, f_{Ret}\}$, where $f_1, \ldots, f_{Ret}$ are some fixed control states (the modification may increase the size of Γ, but only polynomially). Then, we can safely restrict the range of the functions $\mathcal{F}, \mathcal{G}, \mathcal{H}, \ldots$ into $\{f_1, \ldots, f_{Ret}\}$. Hence, the presented complexity bounds remain valid even if we put $z = |F|^{Ret}$ and define PDA^k as the class of all PDA where $Ret \leq k$.

Acknowledgements

We thank the reviewers for their many useful comments and suggestions. Antonín Kučera is supported by the Czech Science Foundation, grant No. P202/12/G061.

References

[1] P.A. Abdulla and K. Čerāns. Simulation is decidable for one-counter nets. In *Proceedings of CONCUR'98*, volume 1466 of *Lecture Notes in Computer Science*, pages 253–268. Springer, 1998.

[2] 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.

[3] 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.

[4] 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(3):653–682, 1993.

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

[6] M. Benedikt, S. Göller, S. Kiefer, and A. Murawski. Bisimilarity of pushdown automata is nonelementary. In *Proceedings of LICS 2013*, pages 488–498. IEEE Computer Society Press, 2013.

[7] S. Böhm, S. Göller, and P. Jančar. Bisimilarity of one-counter processes is PSPACE-complete. In *Proceedings of CONCUR 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 177–191. Springer, 2010.

[8] S. Böhm, S. Göller, and P. Jančar. Equivalence of deterministic one-counter automata is NL-complete. In *Proceedings of STOC 2013*, pages 131–140. ACM Press, 2013.

[9] 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.

[10] 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.

[11] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. *Handbook of Process Algebra*, pages 545–623, 2001.

[12] 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.

[13] A. Carayol and M. Hague. Saturation algorithms for model-checking pushdown systems. *Electronic Proceedings in Theoretical Computer Science*, 151:1–24, 2014.

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

[15] D. Caucal, D.T. Huynh, and L. Tian. Deciding branching bisimilarity of normed context-free processes in in Σ_2^p. *Information and Computation*, 118(2):306–315, 1995.

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

[17] W. Czerwiński and P. Jančar. Branching bisimilarity of normed BPA processes is in NEXPTIME. In *Proceedings of LICS 2015*, pages 168–179, 2015.

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

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

[20] 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.

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

[22] 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.

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

[24] Y. Fu. Checking equality and regularity for normed BPA with silent moves. In *Proceedings of ICALP 2013, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 238–249. Springer, 2013.

[25] S. Göller, R. Mayr, and A.W. To. On the computational complexity of verifying one-counter processes. In *Proceedings of LICS 2009*, pages 235–244. IEEE Computer Society Press, 2009.

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

[27] C. He and M. Huang. Branching bisimilarity on normed BPA is EXPTIME-complete. In *Proceedings of LICS 2015*, pages 180–191, 2015.

[28] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science*, 158(1–2):143–159, 1996.

[29] P. Hofman, S. Lasota, R. Mayr, and P. Totzke. Simulation problems over one-counter nets. *Logical Methods in Computer Science*, 12(1:6):1–46, 2016.

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

[31] H. Hüttel. Silence is golden: Branching bisimilarity is decidable for context-free processes. In *Proceedings of CAV'91*, volume 575 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 1992.

[32] 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.

[33] P. Jančar. Bisimilarity on basic process algebra is in 2-ExpTime (an explicit proof). *Logical Methods in Computer Science*, 9(1), 2012.

[34] P. Jančar. Bisimulation equivalence of first-order grammars. In *Proceedings of ICALP 2014, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 232–243. Springer, 2014.

[35] P. Jančar. Equivalences of pushdown systems are hard. In *Proceedings of FoSSaCS 2014*, volume 8412 of *Lecture Notes in Computer Science*, pages 1–28. Springer, 2014.

[36] P. Jančar, A. Kučera, and F. Moller. Simulation and bisimulation over one-counter processes. In *Proceedings of STACS 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 334–345. Springer, 2000.

[37] 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.

[38] P. Jančar, F. Moller, and Z. Sawa. Simulation problems for one-counter machines. In *Proceedings of SOFSEM'99*, volume 1725 of *Lecture Notes in Computer Science*, pages 404–413. Springer, 1999.

[39] S. Kiefer. BPA bisimilarity is EXPTIME-hard. *Information Processing Letters*, 113(4):101–106, 2013.

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

[41] A. Kučera. The complexity of bisimilarity-checking for one-counter processes. *Theoretical Computer Science*, 304(1–3):157–183, 2003.

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

[43] A. Kučera and P. Jančar. Equivalence-checking on infinite-state systems: Techniques and results. *Theory and Practice of Logic Programming*, 6(3):226–264, 2006.

[44] A. Kučera and R. Mayr. Simulation preorder over simple process algebras. *Information and Computation*, 173(2):184–198, 2002.

[45] 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.

[46] A. Kučera and R. Mayr. On the complexity of checking semantic equivalences between pushdown processes and finite-state processes. *Information and Computation*, 208(7):772–796, 2010.

[47] A. Kučera and Ph. Schnoebelen. A general approach to comparing infinite-state systems with their finite-state specifications. *Theoretical Computer Science*, 358(2–3):315–333, 2006.

[48] 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.

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

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

[51] G. Sénizergues. L(A)=L(B)? Decidability results from complete formal systems. *Theoretical Computer Science*, 251(1–2):1–166, 2001.

[52] G. Sénizergues. The bisimulation problem for equational graphs of finite out-degree. *SIAM Journal of Computing*, 34(5):1025–1106, 2005.

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

[54] 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.

[55] C. Stirling. Decidability of DPDA equivalence. *Theoretical Computer Science*, 255:1–31, 2001.

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

[57] R. van Glabbeek. The linear time—branching time spectrum. *Handbook of Process Algebra*, pages 3–99, 2001.

[58] 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.

[59] 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.

[60] Q. Yin, Y. Fu, C. He, M. Huang, and X. Tao. Branching bisimilarity checking for PRS. In *Proceedings of ICALP 2014, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2014.