Information and Computation 268 (2019) 104431

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

# Deciding probabilistic simulation between probabilistic pushdown automata and finite-state systems ☆

Mingzhang Huang [a], Hongfei Fu [b,*], Joost-Pieter Katoen [c]

[a] Basics Lab, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, PR China
[b] Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, PR China
[c] Lehrstuhl für Informatik 2, RWTH Aachen University, Ahornstraße 55, 52074 Aachen, Germany

ARTICLE INFO

*Article history:*
Received 31 March 2015
Received in revised form 22 April 2019
Available online 25 May 2019

*Keywords:*
Probabilistic pushdown automata
Simulation preorder
Infinite-state systems

ABSTRACT

Checking whether a pushdown automaton is simulated – in the sense of a simulation pre-order – by a finite-state automaton is EXPTIME-complete. This paper shows that the same computational complexity is obtained in a probabilistic setting. That is, the problem of deciding whether a probabilistic pushdown automaton (pPDA) is simulated by a finite Markov decision process (MDP) is EXPTIME-complete. The considered pPDA contain both probabilistic and non-deterministic branching. The EXPTIME-membership is shown by combining a partition-refinement algorithm with a tableaux system that is inspired by Stirling's technique for bisimilarity checking of ordinary pushdown automata. The hardness is obtained by exploiting the EXPTIME-hardness for the non-probabilistic case. Moreover, our decision problem is in PTIME when both the number of states of the pPDA and the number of states in the MDP are fixed.



## 1. Introduction

Probabilistic pushdown automata (pPDA) extend classical pushdown automata (PDA) with discrete probabilistic branching: the target of a transition is a finite probability distribution over pairs consisting of a control state and a stack. As a control state may have several transitions, a pPDA exhibits both probabilistic and non-deterministic branching. pPDA has been studied in language theory as automata model for probabilistic context-free grammars [2–4]. In concurrency theory, PDAs are a strict generalization of processes from basic process algebra [5], a subset of process algebra that has been intensively studied. The pPDA model plays a similar role for probabilistic process algebra. Finally, pPDA are a natural model for recursive Bayesian networks [6] and recursive probabilistic programs [7,8], programs that extend the usual programming language constructs with random assignments.

A natural question is whether all behaviours of a PDA, considered as a realization of recursive program, can be captured by a finite-state model, considered as an abstract specification. Simulation pre-orders are a prominent notion to study such relations. The key issue in a simulation pre-order is whether any possible computational step of the implementation can be mimicked by the specification. Simulation pre-orders preserve a safety fragment of temporal logic. They are therefore used as a basis for abstraction techniques where the model to be verified is replaced by a smaller abstract model and to verify

☆ A preliminary version of this article [1] was published in FSTTCS 2011.

* Corresponding author.
*E-mail address:* fuhf@cs.sjtu.edu.cn (H. Fu).

https://doi.org/10.1016/j.ic.2019.05.004
0890-5401/

**Table 1**
Complexity results for relating PDA and finite processes.

| | Non-probabilistic | Probabilistic |
|---|---|---|
| Bisimilarity | PSPACE-c. [16,9] | in EXPTIME [15] |
| Simulation pre-order | EXPTIME-c. [9] | EXPTIME-c. (this paper) |

the latter instead of the original one. A seminal result by Kučera and Mayr [9] is that deciding whether a PDA is being simulated – in the sense of a simulation pre-order – is EXPTIME-complete.

This paper shows that this computational complexity carries over to the probabilistic setting. More specifically, this paper shows that checking whether a pPDA is being simulated by a finite probabilistic automaton (in the sense of Segala and Lynch [10]), a slight variation of Markov decision processes, is EXPTIME-complete. Our decision procedure is in PTIME when both the number of states of the pPDA and the number of states in the probabilistic automaton (PA) are fixed. The basis is probabilistic simulation [11,12,10] whose kernel preserves a safety fragment of PCTL$^*$ [13]. Stated in terms of the probabilistic extension of Dijkstra's guarded command language pGCL [14], our results apply to checking whether a finite-state pGCL program correctly abstracts a recursive pGCL program. If the abstract program satisfies a safe PCTL$^*$-formula, then the infinite-state recursive program does so too.

Our result extends the result by Brázdil et al. [15] that probabilistic bisimilarity between pPDA and finite-state PA is in EXPTIME. An overview of related computational complexity results is given in Table 1.

The EXPTIME-completeness result is obtained as follows. The EXPTIME-membership is shown by combining a partition-refinement algorithm with a tableaux system for a probabilistic simulation pre-order between pPDA and finite PA. Our tableaux system is inspired by Stirling's technique [17,18] for bisimilarity checking of ordinary pushdown automata. The hardness is obtained by exploiting the EXPTIME-hardness for the non-probabilistic case [9]. Whereas the proof technique in [9] for EXPTIME-membership of checking whether a PDA is simulated by a finite-state process is based on a reduction to the model-checking problem of $\mu$-calculus on pushdown processes, our technique uses a different approach.

Recent works on equivalence checking on pPDA's include checking language equivalence [19], and characterizing probabilistic bisimilarity for various sub-classes of pPDA in terms of two-player games [20]. In contrast, the verification of pPDA has received quite some attention in recent years; for a survey see [21]. The primary focus has been on model-checking problems, whose task is to check whether a pPDA satisfies a given logical property or not. These include computing *termination probabilities* [22,23], checking *linear-time properties* [24,25], and verifying *branching-time properties* [26]. Other related works include relevant results on PDA [27,9,17,18,16,28].

The article is organized as follows. In Section 2, we introduce the notion of probabilistic automata which serves as the semantical backbone for probabilistic pushdown automata, and we introduce the notion of probabilistic pushdown automata, following the definitions from [24,15]. In Section 3, we introduce the notion of extended stack symbols which is crucial to the tableaux proof system for probabilistic simulation preorder. In Section 4, we present the tableaux proof system and a partition-refinement algorithm to the main complexity result. In Section 5, we briefly describe how one can apply the hardness result by Kučera and Mayr [9] to our case. Finally, Section 6 concludes the article. In our paper, the propositions are building blocks for theorems and the propositions are proved by lemmas.

## 2. Probabilistic models and simulation preorders

In this section, we introduce the notion of probabilistic automata [10] and pushdown automata (pPDA's). Probabilistic automata (PA) are an analogue of Markov decision processes [29] which, like Markov decision processes, involves both stochastic and non-deterministic features. The difference between PAs and Markov decision processes is that an action of a PA (under a current state) can lead to different probability distributions, while an action (under a current state) uniquely determines a probability distribution in a Markov decision process; in this sense, PAs focus more on reactive features than Markov decision processes do. PAs can also be viewed as an orthogonal extension of labelled transition systems [13] with probabilities.

Informally, pPDA extend (non-probabilistic) pushdown processes [30,27] with discrete probabilistic branching.

In this article, we focus on (probabilistic) simulation preorder and simulation equivalence between PAs and pPDA's. We regard every pPDA as a PA with many infinite states induced by it. We consider the simulation preorder between PA. Simulation preorder is a preorder on the set of states of a PA where a pair of states $(s, s')$ is in this preorder whenever $s'$ can mimic the behaviour of $s$; simulation equivalence is the equivalence relation induced by simulation preorder. Intuitively, simulation preorder is the one-sided version of (probabilistic) bisimulation equivalence [12,10], which in turn characterizes whether two states can mutually mimic the other. In [10], Segala and Lynch proved that (probabilistic) bisimulation equivalence preserves logical properties encoded in PCTL$^*$ [31,10], and the equivalence relation induced by (probabilistic) simulation preorder preserves a safety fragment of PCTL$^*$ [31,10] (cf. also [13]).

In the remainder, let $\mathbb{N}$ be the set of natural numbers including zero.

### 2.1. Probabilistic automata

To introduce the notion of probabilistic automata, we first introduce the notion of (discrete) probability distributions.

**Definition 1.** Let $X$ be a finite or countable set. A *(discrete) probability distribution* on $X$ is a function $\mu : X \to [0,1]$ such that $\sum_{x\in X} \mu(x) = 1$.

A probability distribution $\mu$ on $X$ is *finite* if the *support* of $\mu$, denoted by $\lfloor\mu\rfloor$ and defined by $\lfloor\mu\rfloor := \{x \in X \mid \mu(x) > 0\}$, is finite. The set of probability distributions (resp. finite probability distributions) on $X$ is denoted by $Dist(X)$ (resp. by $Dist_{\mathrm{f}}(X)$).

The notion of probabilistic automata [10] is given as follows.

**Definition 2.** [10] A *probabilistic automaton* (PA) is a tuple $(S, Act, \to)$ where

- $S$ is a non-empty, finite or countable set of *states*;
- $Act$ is a non-empty finite set of *actions*;
- $\to \subseteq S \times Act \times Dist(S)$ is a finite set of *transitions*.

A PA $(S, Act, \to)$ is *locally finite* if for all $s \in S$ and $a \in A$, it holds that $\{\mu \mid (s, a, \mu) \in \to\} \subseteq Dist_{\mathrm{f}}(S)$; it is *finite* if it is locally finite, and $S$, $Act$ and $\to$ are finite.

Technically speaking, the class of probabilistic automata defined here refers to the class of *simple* probabilistic automata in [10].

Let $(S, Act, \to)$ be a PA. For each $s \in S$, we define

- $Act(s) := \{a \in Act \mid \exists \mu.(s, a, \mu) \in \to\}$ be the set of actions *enabled* at $s$, and
- $Succ(s) := \{s' \in S \mid \exists (s, a, \mu) \in \to .\mu(s') > 0\}$ be the set of states which can be reached from $s$ within one step with non-zero probability.

Intuitively, the set $\to$ specifies all possible probabilistic transitions between states. A transition $(s, a, \mu)$ specifies that when the current state is $s$ and the action to be taken is $a$, the state at the next step is chosen w.r.t. the probability distribution $\mu$.

The following definition extends the notions of transitions to *combined transitions*. In [10], *combined transitions* are introduced to model randomized strategies.

**Definition 3.** Let $(S, Act, \to)$ be a PA. The set of *combined transitions* $\underset{\mathsf{c}}{\to} \subseteq S \times Act \times Dist(S)$ is defined as follows: $(s, a, \mu) \in \underset{\mathsf{c}}{\to}$ iff there exists a finite or infinite sequence $\{(\mu_n, d_n)\}_{n\in I}$ $(I \subseteq \mathbb{N})$ such that

- $\mu_n \in Dist(S)$ and $d_n \in [0, 1]$ for all $n \in I$, and
- $(s, a, \mu_n) \in \to$ for all $n \in I$, and
- $\sum_{n\in I} d_n = 1$, and
- $\mu(s) = \sum_{n\in I} d_n \cdot \mu_n(s)$ for all $s \in S$.

By definition, $\to \subseteq \underset{\mathsf{c}}{\to}$. We write "$s \xrightarrow[\mathsf{nc}]{a} \mu$" for "$(s, a, \mu) \in \to$" where nc stands for "standard" or "non-combined". And we write "$s \xrightarrow[\mathsf{c}]{a} \mu$" for "$(s, a, \mu) \in \underset{\mathsf{c}}{\to}$". We will use "$s \xrightarrow[op]{a} \mu$" to refer to either "$s \xrightarrow[\mathsf{nc}]{a} \mu$" or "$s \xrightarrow[\mathsf{c}]{a} \mu$", depending on whether $op = \mathsf{nc}$ or $op = \mathsf{c}$.

### 2.2. Simulation preorder on probabilistic automata

In this part, we fix a PA $(S, Act, \to)$. To define (probabilistic) simulation preorder, we first introduce a lifting operation which lifts a binary relation on states to a binary relation on probability distributions.

**Definition 4.** [11,12] Let $\mathcal{R} \subseteq S \times S$ be a binary relation on $S$. The *lifting relation* $\overline{\mathcal{R}} \subseteq Dist(S) \times Dist(S)$ of $\mathcal{R}$ is defined as follows: $(\mu, \nu) \in \overline{\mathcal{R}}$ iff there exists a *weight function* $w : S \times S \to [0, 1]$ such that

- $\sum_{v\in S} w(u, v) = \mu(u)$ for all $u \in S$, and
- $\sum_{u\in S} w(u, v) = \nu(v)$ for all $v \in S$, and
- for all $u, v \in S$, $(u, v) \in \mathcal{R}$ whenever $w(u, v) > 0$.

The weight function $w$ is a coupling distribution on $S \times S$. The third condition indicates that the distribution is based on $\mathcal{R}$. If $w(u, v)$ is positive, then we have $u\mathcal{R}v$. For the sake of convenience, we will simply write "$\mu\mathcal{R}\nu$" instead of "$\mu\overline{\mathcal{R}}\nu$".

Below we define the notion of (probabilistic) simulation preorder.

**Definition 5.** A binary relation $\mathcal{R}$ on $S$ is an *op-simulation* if for all $(s, s') \in \mathcal{R}$, the following conditions hold:

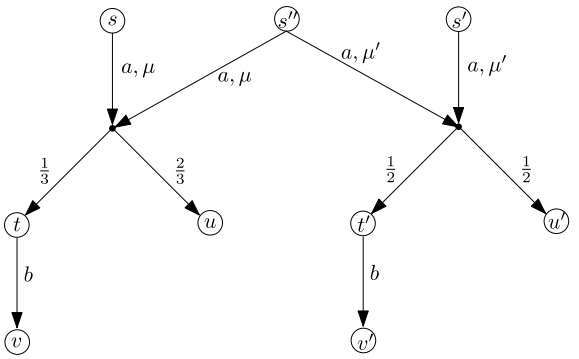


**Fig. 1.** $s \sqsubseteq^*_{\mathsf{nc}} s'$.

- $Act(s) = Act(s')$;
- for all $s \xrightarrow[\mathsf{nc}]{a} \mu$, there exists $s' \xrightarrow[op]{a} \mu'$ such that $\mu \mathcal{R} \mu'$.

The union of two *op*-simulations is also an *op*-simulation. The *op-similarity preorder*, denoted by $\sqsubseteq_{op}$, is defined by:

$$\sqsubseteq_{op} := \bigcup \{\mathcal{R} \subseteq S \times S \mid \mathcal{R} \text{ is an } op\text{-simulation}\} \ .$$

The *op-similarity equivalence* is defined as $\sqsubseteq_{op} \cap \sqsubseteq_{op}^{-1}$, where $\sqsubseteq_{op}^{-1}$ is the inverse relation of $\sqsubseteq_{op}$.

The relation $\sqsubseteq_{\mathsf{nc}}$ corresponds to strong simulation and $\sqsubseteq_{\mathsf{c}}$ corresponds to strong probabilistic simulation in [10]. It can be easily verified that $\sqsubseteq_{\mathsf{nc}}$ (resp. $\sqsubseteq_{\mathsf{c}}$) itself is an nc-simulation (resp. c-simulation).

**Remark 1.** The first condition $Act(s) = Act(s')$ in the Definition 5 corresponds to the third condition of Definition 5 in [10]. We can relax this condition $Act(s) \subseteq Act(s')$ thanks to the second simulation condition. We explain the difference between these two definitions by the example illustrated in Fig. 1. Let $\sqsubseteq^*_{\mathsf{nc}}$ be the simulation relation defined by replacing the first constraint in Definition 5 by the $Act(s) \subseteq Act(s')$. We have $\sqsubseteq_{\mathsf{nc}} \subseteq \sqsubseteq^*_{\mathsf{nc}}$.

It is clear that $s \sqsubseteq_{\mathsf{nc}} s''$, $s \sqsubseteq^*_{\mathsf{nc}} s''$, $s' \sqsubseteq^*_{\mathsf{nc}} s''$and $s' \sqsubseteq_{\mathsf{nc}} s''$.

We now consider the relation between $s$ and $s'$.

First, we have $s \sqsubseteq^*_{\mathsf{nc}} s'$, since there exists a weight function $w$ as given by the table:

| $w$ | $t'$ | $u'$ |
|---|---|---|
| $t$ | $\frac{1}{3}$ | 0 |
| $u$ | $\frac{1}{6}$ | $\frac{1}{2}$ |

and $u \sqsubseteq^*_{\mathsf{nc}} t'$. It is noticed that the trivial weight function

| $w'$ | $t'$ | $u'$ |
|---|---|---|
| $t$ | $\frac{1}{6}$ | $\frac{1}{6}$ |
| $u$ | $\frac{1}{3}$ | $\frac{1}{3}$ |

is not suitable for simulation. Since $w'(t, u') > 0$, we require that $t$ can be simulated by $u'$ by the third condition in Definition 4, which is not true for the action $b$.

Second, we have $s \not\sqsubseteq_{\mathsf{nc}} s'$. Since the action set of $u$ and $t'$ are not the same, we have $u \not\sqsubseteq_{\mathsf{nc}} t'$. Therefore, $s \not\sqsubseteq_{\mathsf{nc}} s'$.

For the rest of this paper, we use the simulation relation defined by Definition 5.

It can be verified that we can obtain the algorithms for $\sqsubseteq^*$ by modifying the part (check action sets) of the algorithms in Section 4 that check equality of the action sets. The proof of correctness of the modified algorithm is similar.

Up till now, the notion of simulation preorder are defined on a single probabilistic automaton. They are extended to the counterparts between two probabilistic automata by taking the disjoint union of the two PA, i.e., by making a PA which precisely comprises all the states, actions and transitions of the original two PA.

### 2.3. Probabilistic pushdown automata

In this subsection, we introduce probabilistic pushdown automata.

**Definition 6.** [24,15] A *probabilistic pushdown automaton (pPDA)* is a quadruple $(Q, \Gamma, L, \rightarrowtail)$, where:

- $Q$ is a finite non-empty set of *control states*;
- $\Gamma$ is a finite non-empty set of *stack symbols*;
- $L$ is a finite non-empty set of *labels*;

- $\rightarrowtail \subseteq (Q \times \Gamma) \times L \times Dist_{\mathrm{f}}(Q \times \Gamma^*)$ is a finite set of *transition rules*.

From now on, we use $p, q$ to range over control states $Q$, $A, B, C$ to range over stack symbols $\Gamma$, $\alpha, \beta, \gamma$ to range over $\Gamma^*$, $\mu, \nu$ to range over finite distributions on $Q \times \Gamma^*$ and $a, b, c$ to range over labels $L$. Instead of writing "$(pA, a, \mu) \in \rightarrowtail$", we write "$pA \overset{a}{\rightarrowtail} \mu$".

We define the probability distribution $\mu_\gamma \in Dist(Q \times \Gamma^*)$ with $\mu \in Dist(Q \times \Gamma^*)$ and $\gamma \in \Gamma^*$ by:

$$\mu_\gamma(p\alpha) = \begin{cases} \mu(p\beta) & \text{if } \alpha = \beta\gamma \text{ for some (unique) } \beta \in \Gamma^* \\ 0 & \text{otherwise} \end{cases} ,$$

for all $p\alpha \in Q \times \Gamma^*$. Let $(\mu_{\gamma_1})_{\gamma_2} = \mu_{\gamma_1\gamma_2}$. The following notion defines how a pPDA generates a probabilistic automaton.

**Definition 7.** The pPDA $M = (Q, \Gamma, L, \rightarrowtail)$ induces the PA $P_M = (S, Act, \rightarrow)$ with:

$$S := Q \times \Gamma^*;\ Act := L;\ \rightarrow := \{(pA\gamma, a, \mu_\gamma) \mid pA \overset{a}{\rightarrowtail} \mu, \gamma \in \Gamma^*\} .$$

Elements of $Q \times \Gamma^*$ are also called *configurations*.

Note that every pPDA induces a locally-finite PA. Intuitively, a pPDA generates a PA by expanding suffixes to transition rules. Below we state a simple property for such expansion of transition rules.

**Lemma 8.** *Let $(Q, \Gamma, L, \rightarrowtail)$ be a pPDA. Let $p\beta\gamma \in Q \times \Gamma^*$, where $\beta \in \Gamma^+$, $a \in L$ and $\mu \in Dist(Q \times \Gamma^*)$. Then $p\beta\gamma \xrightarrow[op]{a} \mu$ iff there exists $\mu' \in Dist(Q \times \Gamma^*)$ such that $p\beta \xrightarrow[op]{a} \mu'$ and $\mu = \mu'_\gamma$.*

**Proof.** We first consider the case when $op = \mathsf{nc}$. Let $\beta = A\beta'$. By Definition 7,

$$pA\beta'\gamma \xrightarrow[\mathsf{nc}]{a} \mu \text{ iff } \exists \mu'. \left(pA \overset{a}{\rightarrowtail} \mu' \wedge \mu = \mu'_{\beta'\gamma}\right) .$$

Note that $(\nu_\beta)_\gamma = \nu_{\beta\gamma}$ for all $\nu \in Dist(Q \times \Gamma^*)$. Thus, one obtains

$$\exists \mu'. \left(pA \overset{a}{\rightarrowtail} \mu' \wedge \mu = \mu'_{\beta'\gamma}\right) \text{ iff } \exists \mu''. \left(pA\beta' \xrightarrow[\mathsf{nc}]{a} \mu'' \wedge \mu = \mu''_\gamma\right)$$

by taking $\mu'' = \mu'_{\beta'}$.

Now we consider the case when $op = \mathsf{c}$. By Definition 3, $p\beta\gamma \xrightarrow[\mathsf{c}]{a} \mu$ iff

(a) there exists a finite or infinite sequence $\{(\mu_n, d_n)\}_n$ such that $p\beta\gamma \xrightarrow[\mathsf{nc}]{a} \mu_n$ for all $n$, $\sum_n d_n = 1$, and $\mu(s) = \sum_n d_n \cdot \mu_n(s)$ for all $s \in Q \times \Gamma^*$.

By the proof for the case $op = \mathsf{nc}$, we have (a) holds iff

(b) there exists a finite or infinite sequence $\{(\mu'_n, d_n)\}_n$ such that $p\beta \xrightarrow[\mathsf{nc}]{a} \mu'_n$ for all $n$, $\sum_n d_n = 1$, and $\mu(s) = \sum_n d_n \cdot (\mu'_n)_\gamma(s)$ for all $s \in Q \times \Gamma^*$.

We can further obtain that (b) holds iff

(c) there exists a finite or infinite sequence $\{(\mu'_n, d_n)\}_n$ such that $p\beta \xrightarrow[\mathsf{nc}]{a} \mu'_n$ for all $n$, $\sum_n d_n = 1$, and $\mu = \mu'_\gamma$, where $\mu'(s) := \sum_n d_n \cdot \mu'_n(s)$ for all $s \in Q \times \Gamma^*$. Then (c) holds iff $\exists \mu'. (p\beta \xrightarrow[\mathsf{c}]{a} \mu' \wedge \mu = \mu'_\gamma)$, from which the result follows. □

For any two sets $X, Y$, we define $X \odot Y := (X \times Y) \cup (Y \times X)$. Given a pPDA $M = (Q, \Gamma, L, \rightarrowtail)$, a finite PA $P = (S, Act, \rightarrow)$, we have that the simulation relation $\sqsubseteq_{op} \subseteq (Q \times \Gamma^*) \odot S$. This article studies the complexity of the following decision problems:

- **INPUT**: a pPDA $M$, a configuration $p\alpha$ of $M$, a finite PA $P$ and state $s$ of $P$;
- **OUTPUT**: whether $p\alpha \sqsubseteq_{op} s$ and whether $s \sqsubseteq_{op} p\alpha$ between the finite PA and the PA $P$ induced by the pPDA $M$, for $op \in \{\mathsf{nc}, \mathsf{c}\}$.

We prove that both problems are EXPTIME-complete, and are in PTIME only if both the number of control states of the pPDA $M$ and the number of states of the finite PA $P$ are fixed.

**Example 1.** Let the pPDA $(Q, \Gamma, L, \rightarrowtail)$ and the finite PA $(S, Act, \rightarrow)$ be such that:

- $Q = \{p\}$, $L = \{a, b\}$, $\Gamma = \{A, Z\}$;
- $\rightarrowtail = \{pA \overset{a}{\rightarrowtail} \{pAA \mapsto 0.5, p \mapsto 0.5\}, pZ \overset{b}{\rightarrowtail} \{pAZ \mapsto 1\}\}$;
- $S = \{s_1, s_2\}$, $Act = \{a, b\}$;
- $\rightarrow = \{s_1 \xrightarrow[\mathsf{nc}]{a} \{s_1 \mapsto 1\}, s_1 \xrightarrow[\mathsf{nc}]{a} \{s_1 \mapsto 0.5, s_2 \mapsto 0.5\}, s_2 \xrightarrow[\mathsf{nc}]{b} \{s_1 \mapsto 1\}\}$.

Intuitively, the pPDA models a counter with random increase and decrease operation. It can be verified that $pZ \sqsubseteq_{\mathsf{nc}} s_2$ since the relation

$$\{(pZ, s_2)\} \cup \{(p\alpha Z, s_1) \mid \alpha \in A^+\}$$

is an nc-simulation.

## 3. Extended stack symbols

In this section, we extend the notion of "extended stack symbols" by Stirling [17,18], which is originally used to establish a tableaux proof system that is sound and complete for the bisimulation equivalence on (non-probabilistic) pushdown processes; we follow Stirling's method to establish extended stack symbols for a simulation preorder between a (PA induced by a) pPDA and a finite PA.

Later in Section 4, we present a tableaux proof system for simulation preorder between a pPDA and a finite PA and demonstrate our complexity results, and our partition-refinement algorithm also uses this notion.

The following definition illustrates the notion of extended stack symbols. Now we fix the given pPDA $(Q, \Gamma, L, \rightarrowtail)$ and finite PA $(S, Act, \rightarrow)$.

**Definition 9.** An *extended stack symbol* $U$ is a function $U : Q \to 2^S$. Let $E$ denote the set of all extended stack symbols.

W.l.o.g., we assume that $E \cap \Gamma = \emptyset$. Intuitively, an extended stack symbol represents some finite sequence $\gamma \in \Gamma^*$ of (original) stack symbols in the following sense: $U(q)$ tries to capture the set of all states $u \in S$ that either $q\gamma \sqsubseteq_{op} u$ or $u \sqsubseteq_{op} q\gamma$ (depending on the context). Thus, $U$ encodes all information for the "fragment" $\gamma$ under the context of a simulation preorder. Note that the set $E$ of extended stack symbols is finite because both $Q$ and $S$ are finite.

**Remark 2.** The extended stack symbol is a variant of the *simple constant* [17,18] which describes the capabilities of a suffix. We suppose that the pPDA process makes an action $p\gamma \xrightarrow[op]{a} \mu$, and the PA process simulates it by the transition $t \xrightarrow[op]{a} \mu'$ with $\lfloor \mu' \rfloor = S'$. We further suppose that for any $s, s' \in S'$ we have $s \not\sqsubseteq_{op} s'$. If $s \sqsubseteq_{op} s'$, we have $p'\gamma' \sqsubseteq_{op} s'$ from $p'\gamma' \sqsubseteq_{op} s$. However, there is no such transitivity in $S'$, and we possibly need to check almost every pair in $\lfloor \mu \rfloor \times \lfloor \mu' \rfloor$ for a particular weight function by definition of simulation preorder. We now see that in our context, it is enough to represent this by a set of states, while a single state is not enough.

We now extend the pPDA $(Q, \Gamma, L, \rightarrowtail)$ with $E$.

**Definition 10.** The extended pPDA is defined as $(Q, \Gamma \cup E, L, \rightarrowtail)$. The set of *extended configurations* (resp. basically-extended configurations), denoted by $\mathcal{E}$ (resp. by $\mathcal{E}_{\mathrm{b}}$), is defined by $\mathcal{E} := Q \times (\Gamma^* \cdot (E + \epsilon))$ (resp. $\mathcal{E}_{\mathrm{b}} := Q \times E$).

Instead of considering $Q \times (\Gamma + E)^*$ as the set of extended configurations, we consider elements from $\mathcal{E}$. This is because we only need elements in $\mathcal{E}$ to complete the tableaux proof system to be established in Section 4. Moreover, the elements from $\mathcal{E}_{\mathrm{b}}$ will serve as certain terminal leaves in the tableaux proof system. Note that $\mathcal{E} \backslash \mathcal{E}_{\mathrm{b}}$ $(= Q \times (\epsilon + \Gamma^+ \cdot (E + \epsilon)))$ is the set of all configurations where the extended stack symbol (if it occurs) is preceded by a non-empty sequence of (original) stack symbols. Also note that we do not modify $\rightarrowtail$ in the extension of the pPDA; thus an extended configuration $pU$ with $U \in E$ has no outgoing transitions in the PA induced by the extended pPDA.

In the remaining part of this article, if not stated otherwise, we will refer to the extended pPDA whenever the notion of pPDA is encountered (e.g., when the PA induced by the pPDA is of concern). We use $U, V$ to range over extended stack symbols, while using $A, B, C$ to range over the set of original stack symbols $\Gamma$ and $\alpha, \beta, \gamma$ to range over $\Gamma^* \cdot (E + \epsilon)$. We also override $\rightarrow$ to be the disjoint union of the transitions of both the finite PA and the PA induced by the extended pPDA.

The following definition extends the notion of simulation preorder to extended configurations. Since we focus only on the simulation preorder between an extended pPDA and a finite PA, we only consider binary relations between them.

**Definition 11.** Let

$$\mathcal{R}_{\mathrm{b}} := \{(qU, s) \in \mathcal{E}_{\mathrm{b}} \times S \mid s \in U(q)\} \cup \{(s, qU) \in S \times \mathcal{E}_{\mathrm{b}} \mid s \in U(q)\} \ .$$

A binary relation $\mathcal{R} \subseteq \mathcal{E} \odot S$ is an *extended op-simulation* iff for all $(s, s') \in \mathcal{R}$:

- if $(s, s') \in \mathcal{E}_b \odot S$ then $(s, s') \in \mathcal{R}_b$;
- if $(s, s') \in (\mathcal{E} \backslash \mathcal{E}_b) \odot S$ then (i) $Act(s) = Act(s')$ and (ii) whenever $s \xrightarrow[\mathsf{nc}]{a} \mu$ there exists $s' \xrightarrow[op]{a} \mu'$ such that $\mu \mathcal{R} \mu'$.

The *extended op-simulation preorder*, denoted by $\sqsubseteq_{e,op}$, is the union of all extended *op*-simulations.

By definition, $\sqsubseteq_{e,op}$ extends $\sqsubseteq_{op}$ by adding pairs in $\mathcal{R}_b$. It can be easily verified that $\sqsubseteq_{e,op}$ itself is an extended *op*-simulation. Intuitively, $\mathcal{R}_b$ contains all pairs of the form $(qU, s)$ such that "$qU \sqsubseteq s$" or of the form $(s, qU)$ such that "$s \sqsubseteq qU$". The following fact shows that $\sqsubseteq_{e,op}$ is a legitimate extension of $\sqsubseteq_{op}$.

**Proposition 12.** *$\sqsubseteq_{e,op} \cap ((Q \times \Gamma^*) \odot S) = \sqsubseteq_{op} \cap ((Q \times \Gamma^*) \odot S)$, where $\sqsubseteq_{op}$ refers to the simulation preorder on the disjoint union of the finite PA and the PA induced by the pPDA.*

**Proof.** The result follows from the facts that $\sqsubseteq_{op} \cap (Q \times \Gamma^*) \odot S$ is an extended *op*-simulation, and $\sqsubseteq_{e,op} \cap ((Q \times \Gamma^*) \odot S)$ is an *op*-simulation on the disjoint union of the finite PA and the original pPDA. □

Below we define stepwise approximations of $\sqsubseteq_{e,op}$. Their purpose to introduce such notion is to prove the soundness of the tableaux proof system for $\sqsubseteq_{e,op}$.

**Definition 13.** The family $\{\sqsubseteq^n_{e,op}\}_{n \in \mathbb{N}}$ is inductively defined as follows:

- $\sqsubseteq^0_{e,op} := \{(s, s') \in (\mathcal{E} \backslash \mathcal{E}_b) \odot S \mid Act(s) = Act(s')\} \cup \mathcal{R}_b$;
- $\sqsubseteq^{n+1}_{e,op}$ is defined as the following set:

$$\sqsubseteq^{n+1}_{e,op} := \mathcal{R}_b \cup \Big\{(s, s') \in (\mathcal{E} \backslash \mathcal{E}_b) \odot S \mid Act(s) = Act(s'), \text{ and for all } s \xrightarrow[\mathsf{nc}]{a} \mu, \text{ there exists } s' \xrightarrow[op]{a} \mu' \text{ such that } \mu \sqsubseteq^n_{e,op} \mu'\Big\} .$$

By definition and an induction on $n \in \mathbb{N}$, we can easily obtain the following fact.

**Lemma 14.** *For all $n \in \mathbb{N}$, $\sqsubseteq^{n+1}_{e,op} \subseteq \sqsubseteq^n_{e,op}$ and $\sqsubseteq_{e,op} \subseteq \sqsubseteq^n_{e,op}$.*

**Proof.** We proceed by induction on $n \in \mathbb{N}$. The base step $n = 0$ is straightforward. Consider the inductive step. Assuming $\sqsubseteq^{n+1}_{e,op} \subseteq \sqsubseteq^n_{e,op}$ and $\sqsubseteq_{e,op} \subseteq \sqsubseteq^n_{e,op}$, we prove that the arguments hold for $n + 1$.

Let $(s, s') \in \sqsubseteq^{n+2}_{e,op}$. By definition, either $(s, s') \in \mathcal{R}_b$, or $Act(s) = Act(s')$ and for all $s \xrightarrow[\mathsf{nc}]{a} \mu$, there exists $s' \xrightarrow[op]{a} \mu'$ such that $\mu \sqsubseteq^{n+1}_{e,op} \mu'$. If $(s, s') \in \mathcal{R}_b$, then $(s, s') \in \sqsubseteq^{n+1}_{e,op}$; otherwise, by $\sqsubseteq^{n+1}_{e,op} \subseteq \sqsubseteq^n_{e,op}$ we have $(s, s') \in \sqsubseteq^{n+1}_{e,op}$ by definition. Since $(s, s')$ is arbitrarily chosen, we obtain $\sqsubseteq^{n+2}_{e,op} \subseteq \sqsubseteq^{n+1}_{e,op}$.

Now let $(s, s') \in \sqsubseteq_{e,op}$. By definition, either $(s, s') \in \mathcal{R}_b$, or $Act(s) = Act(s')$ and for all $s \xrightarrow[\mathsf{nc}]{a} \mu$, there exists $s' \xrightarrow[op]{a} \mu'$ such that $\mu \sqsubseteq_{e,op} \mu'$. If $(s, s') \in \mathcal{R}_b$, then $(s, s') \in \sqsubseteq^{n+1}_{e,op}$; otherwise, by $\sqsubseteq_{e,op} \subseteq \sqsubseteq^n_{e,op}$ we also have $(s, s') \in \sqsubseteq^{n+1}_{e,op}$ by definition. Thus $\sqsubseteq_{e,op} \subseteq \sqsubseteq^{n+1}_{e,op}$ by the arbitrary choice of $(s, s')$. □

Intuitively, $s \sqsubseteq^n_{e,op} s'$ holds iff $s'$ can mimic the behaviour of $s$ up to $n$ steps, and then $\bigcap_{n \in \mathbb{N}} \sqsubseteq^n_{e,op}$ equals $\sqsubseteq_{e,op}$ due to image-finiteness. Below we prove that they are equal.

**Proposition 15.** *For all $(s, s') \in \mathcal{E} \odot S$, $s \sqsubseteq_{e,op} s'$ iff $s \sqsubseteq^n_{e,op} s'$ for all $n \in \mathbb{N}$.*

**Proof.** Define $\sqsubseteq^\omega_{e,op} := \bigcap_{n \in \mathbb{N}} \sqsubseteq^n_{e,op}$. We prove that $\sqsubseteq^\omega_{e,op} = \sqsubseteq_{e,op}$. One direction $\sqsubseteq_{e,op} \subseteq \sqsubseteq^\omega_{e,op}$ follows directly from Lemma 14. As to $\sqsubseteq^\omega_{e,op} \subseteq \sqsubseteq_{e,op}$, we prove that $\sqsubseteq^\omega_{e,op}$ is an extended *op*-simulation.

Fix arbitrarily $(s, s') \in \sqsubseteq^\omega_{e,op}$ and $a \in L \cup Act$. If $(s, s') \in \mathcal{E}_b \odot S$, then by definition $(s, s') \in \mathcal{R}_b$. Assume $(s, s') \in (\mathcal{E} \backslash \mathcal{E}_b) \odot S$. Clearly $Act(s) = Act(s')$. Define

$$\mathcal{R} := \Big\{(u, v) \in \mathcal{E} \odot S \mid \exists \mu, \nu. \Big(s \xrightarrow[\mathsf{nc}]{a} \mu \wedge s' \xrightarrow[\mathsf{nc}]{a} \nu \wedge (u, v) \in \lfloor \mu \rfloor \times \lfloor \nu \rfloor\Big)\Big\} .$$

Then $\mathcal{R}$ is finite. Consider any $(u, v) \in \mathcal{R}$. If $(u, v) \notin \sqsubseteq^\omega_{e,op}$, there is a minimal $N(u, v) \in \mathbb{N}$ such that $(u, v) \notin \sqsubseteq^{N(u,v)}_{e,op}$. Define

$$N := \max\{N(u, v) \mid (u, v) \in \mathcal{R} \backslash \sqsubseteq^\omega_{e,op}\}$$

where $\max\emptyset := 0$. By Lemma 14, we have $\mathcal{R}\cap\sqsubseteq^N_{e,op} = \mathcal{R}\cap\sqsubseteq^\omega_{e,op}$. Since $s \sqsubseteq^{N+1}_{e,op} s'$, for all $s \xrightarrow[nc]{a} \mu$, there exists $s' \xrightarrow[op]{a} \mu'$ such that $\mu \sqsubseteq^N_{e,op} \mu'$. Then $\mu(\sqsubseteq^N_{e,op}\cap\mathcal{R})\mu'$. Thus $\mu \sqsubseteq^\omega_{e,op} \mu'$ by $\mathcal{R}\cap\sqsubseteq^N_{e,op} = \mathcal{R}\cap\sqsubseteq^\omega_{e,op}$. □

## 4. Algorithm for deciding preorder

In this section, we present an algorithm for the simulation preorder between a pPDA and a finite PA. It is based on the notion of extended stack symbols. First, we present a tableaux proof system for the preorder. The leaves of the tableaux proof system can be regarded as a simulation base and we prove the soundness and completeness of the tableaux. Secondly, we develop a partition-refinement algorithm to obtain the simulation base directly, based on which we prove the EXPTIME-membership of the *op*-simulation preorder.

Below we fix a pPDA $(Q, \Gamma, L, \rightarrowtail)$ and a finite PA $(S, Act, \rightarrow)$. We extend $(Q, \Gamma, L, \rightarrowtail)$ with extended stack symbols as described in Section 3. As before, we let $\rightarrow$ be the set of transitions of both the finite PA and the PA induced by the extended pPDA.

### 4.1. Tableaux proof system

By Proposition 12, the decision problem for *op*-simulation preorder can be reformulated as follows: given a pair $(s, s') \in (Q \times \Gamma) \odot S$, decide if $s \sqsubseteq_{e,op} s'$. Note that we only consider elements in $Q \times \Gamma$ (instead of $Q \times \Gamma^*$). This is because for any $pX\alpha \in Q \times \Gamma^+$, we can always (i) add a fresh control state $p_{\text{init}}$ and a new stack symbol $X_{\text{init}}$, and (ii) augment $\rightarrowtail$ with the set

$$\{p_{\text{init}}X_{\text{init}} \overset{a}{\rightarrowtail} \mu_\alpha \mid pX \overset{a}{\rightarrowtail} \mu\}$$

of extra transition rules so that $p_{\text{init}}X_{\text{init}}$ mimics the behaviour of $pX\alpha$. For the sake of simplicity, we abbreviate $\sqsubseteq_{e,op}$ (resp. $\sqsubseteq^n_{e,op}$) as $\sqsubseteq_{op}$ (resp. $\sqsubseteq^n_{op}$).

We follow Stirling's tableaux proof system [18,17] to develop the tableaux system in our setting. A tableaux is a goal-directed proof system that consists of a set of goals Goals and a set RULE of rules which describes how a goal can be expanded into sub-goals. Graphically, a rule can be viewed as a proof step:

$$\frac{\text{goal}}{\text{goal}_1, \ldots, \text{goal}_n},$$

where goal is what is currently to be "proved" and $\text{goal}_1, \ldots, \text{goal}_n$ are the subgoals to which goal is reduced. Each rule is backward sound: in the rule depicted above, if all $\text{goal}_i$ $(1 \le i \le n)$ are true then so is goal. An application of a rule is to make all the sub-goals children of goal (in a tree). Then a tableaux is a tree built from a specified goal (the root of the tree) and repeated application of rules. The leaves of a tableaux are divided into *terminal* and *nonterminal* leaves. Terminal leaves are further divided into *successful* and *unsuccessful* leaves. A tableaux is *successful* iff it is finite and all its leaves are successful.

Below we formulate our tableaux proof system. Firstly, we introduce goals in our tableaux proof system.

Define $\text{Goals} := \mathcal{E} \odot S$ to be the set of *goals*. A goal $(s, s') \in \text{Goals}$ is *successful* if either one of the following three conditions hold:

- $(s, s') = (pU, s')$ such that $s' \in U(p)$;
- $(s, s') = (s, pU)$ such that $s \in U(p)$;
- $(s, s') \in (\mathcal{E}\backslash\mathcal{E}_b) \odot S$ and $Act(s) = Act(s') = \emptyset$.

A goal $(s, s') \in \text{Goals}$ is *unsuccessful* if either one of the following three conditions hold:

- $(s, s') = (pU, s')$ such that $s' \notin U(p)$;
- $(s, s') = (s, pU)$ such that $s \notin U(p)$;
- $(s, s') \in (\mathcal{E}\backslash\mathcal{E}_b) \odot S$ and $Act(s) \neq Act(s')$.

We abbreviate $(s, s') \in \text{Goals}$ as "$s \sqsubseteq s'$". Intuitively, the goal $s \sqsubseteq s'$ corresponds to a guess that the claim $s \sqsubseteq_{op} s'$ is correct.

Then, we introduce rules of our tableaux proof system. Formally, a rule

$$\frac{\text{goal}}{\text{goal}_1, \ldots, \text{goal}_n}$$

can be viewed as a pair $(\text{goal}, \{\text{goal}_1, \ldots, \text{goal}_n\})$, which is an element of $\text{Goals} \times 2^{\text{Goals}}$. There are two kinds of rules: UNF (unfolding) and RED (reduction). Intuitively, a rule of type UNF expands a goal $s \sqsubseteq s'$ one-step further and a rule of RED

reduces a goal $pA\alpha \sqsubseteq u$ (resp. $u \sqsubseteq pA\alpha$) to $pAU \sqsubseteq u$ (resp. $u \sqsubseteq pAU$) together with all information at $\alpha$ encoded in $U$. For the sake of clarity, we use subscript 'a' to indicate the case $s \sqsubseteq s' \in \mathcal{E} \times S$ and subscript 'b' to indicate the case for $s \sqsubseteq s' \in S \times \mathcal{E}$.

**Non-probabilistic case.** Before introducing the rules for the probabilistic case, we illustrate how the tableaux system with extended stack symbols prove the simulation relation for the non-probabilistic case. This corresponds to the case all probability distributions are Dirac (i.e., with probability one the distribution is concentrated at a single state). Suppose that $pA \sqsubseteq u$ is the goal we want to prove. Since $pA$ is of the form $Q \times \Gamma$, we first unfold the goal by one-step expanding. For a transition $pA\alpha \xrightarrow{a} p'B\beta\alpha$ (non-probabilistic transition), we guess the simulation transition of $u \xrightarrow{a} u'$, and then one of the subgoals is $p'B\beta\alpha \sqsubseteq u'$. The goal is the preorder if and only if the subgoals by unfolding are in the preorder. If the stack content of this subgoal is more than one symbol, we need to reduce it by extended stack symbol. We guess an extended stack symbol $U$ which maps a state of the PDA onto a state of the automata. Then, we replace the subgoal $p'B\beta\alpha \sqsubseteq u'$ by $p'BU \sqsubseteq u'$ and $q\alpha \sqsubseteq v$, for every $\{v\} = U(q)$. Intuitively, the extended symbol $U$ describe the whole behaviour of the suffix $\alpha$. If $p'B$ goes to $\epsilon$, then the current pair is just one of the subgoals. The soundness of the reducing rule is inherited from the right congruence of PDA. Since such extended stack symbols are finite, the tableaux tree is finite. Thus, we can prove the goal by a tableaux system.

For non-probabilistic models and relations, Kučera and Mayr [32] present an algorithm to compute the finite semantic base for PDA. The pairs representing suffix in the base are of the form $(\alpha\mathcal{G}, \mathcal{F})$, where $\alpha \in \Gamma^*$ and $\mathcal{G}, \mathcal{F} : Q \to S$. Functions $Q \to S$ are enough for non-probabilistic models to describe the suffix behaviour.

**Probabilistic case.** When it comes to probabilistic models, we need to consider all possible simulating states (cf. Remark 2) as defined in extended stack symbols rather than one state. Now we introduce the rules formally. Compared to the non-probabilistic case, the unfolding must consider the distribution simulation rather than state simulation. The set $\mathrm{UNF}_{op} \subseteq \mathsf{Goals} \times 2^{\mathsf{Goals}}$ of *unfolding rules* is defined as follows: $(s \sqsubseteq s', \mathcal{R}) \in \mathrm{UNF}_{op}$ iff

- $s \sqsubseteq s' \in (Q \times (\Gamma \cdot (E + \epsilon))) \odot S$ and $Act(s) = Act(s') \neq \emptyset$, and
- $\mathcal{R} \subseteq Succ(s) \times Succ(s')$ and for all $s \xrightarrow[\mathsf{nc}]{a} \mu$, there exists $s' \xrightarrow[op]{a} \nu$ such that $\mu \mathcal{R} \nu$.

The set $\mathrm{RED}_{op} \subseteq \mathsf{Goals} \times 2^{\mathsf{Goals}}$ of *reduction rules* is defined as $\mathrm{RED}^{\mathrm{a}}_{op} \cup \mathrm{RED}^{\mathrm{b}}_{op}$, for which:

- $(s \sqsubseteq s', \mathcal{R}) \in \mathrm{RED}^{\mathrm{a}}_{op}$ iff $s \sqsubseteq s' = pA\alpha \sqsubseteq u$ with $u \in S$ such that
  - $Act(pA) = Act(u) \neq \emptyset$ and $\alpha \in \Gamma^+ \cdot (E + \epsilon)$, and
  - $\mathcal{R} = \{pAU \sqsubseteq u\} \cup \{q\alpha \sqsubseteq v \mid q \in Q, v \in U(q)\}$ for some $U \in E$.
- $(s \sqsubseteq s', \mathcal{R}) \in \mathrm{RED}^{\mathrm{b}}_{op}$ iff $s \sqsubseteq s' = u \sqsubseteq pA\alpha$ with $u \in S$ such that
  - $Act(u) = Act(pA) \neq \emptyset$ and $\alpha \in \Gamma^+ \cdot (E + \epsilon)$, and
  - $\mathcal{R} = \{u \sqsubseteq pAU\} \cup \{v \sqsubseteq q\alpha \mid q \in Q, v \in U(q)\}$ for some $U \in E$.

The set $\mathrm{RULE}_{op}$ of rules is defined by: $\mathrm{RULE}_{op} := \mathrm{UNF}_{op} \cup \mathrm{RED}_{op}$.

**Remark 3.** $\mathrm{UNF}_{op}$ guesses the distribution simulation and $\mathrm{RED}_{op}$ reduces a suffix to an extended stack symbol. By the syntactic nature of definition of $\mathrm{UNF}_{op}$ and $\mathrm{RED}_{op}$, we have that the unfolding rules can only be applied when there is only one normal stack symbol, and reduction rules are applied if there are multiple normal stack symbols. This is different from the standard use of non-probabilistic unfolding rules [18,17].

With goals and rules defined, we present our tableaux proof system. Firstly, we introduce the notion of vertex-labelled rooted tree.

A *vertex-labelled rooted tree* is a pair $(\mathcal{T}, \mathcal{L})$ for which $\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}))$ is a rooted directed tree (with vertex set $V(\mathcal{T})$ and edge set $E(\mathcal{T})$) and $\mathcal{L}$ is a function which assigns to each vertex of $\mathcal{T}$ a goal.

Let $(\mathcal{T}, \mathcal{L})$ be a vertex-labelled rooted tree. A leaf $z$ of $\mathcal{T}$ is *successful* if either $\mathcal{L}(z)$ is successful, or there is $z' \in V(\mathcal{T})$ such that (i) $z' \neq z$, (ii) $z'$ lies on the path from the root of $\mathcal{T}$ to $z$ and (iii) $\mathcal{L}(z') = \mathcal{L}(z)$. A leaf $z$ of $\mathcal{T}$ is *unsuccessful* if $\mathcal{L}(z)$ is unsuccessful. A leaf $z$ of $\mathcal{T}$ is *terminal* if either $z$ is successful or unsuccessful; otherwise it is *non-terminal*. The notion of tableaux tree (which is the core notion of our tableaux proof system) is defined as a subclass of vertex-labelled rooted trees.

An *op-tableaux tree* is a vertex-labelled rooted tree inductively defined as follows:

- a single vertex labelled with a goal is an *op*-tableaux tree (in this base case the sole vertex is the root of the tree);
- if
  1. $(\mathcal{T}, \mathcal{L})$ is an *op*-tableaux tree, and
  2. $z \in V(\mathcal{T})$ is non-terminal in $\mathcal{T}$ and is either a leaf of $\mathcal{T}$ or the sole vertex of $\mathcal{T}$ (which in this case is also the root of $\mathcal{T}$), and

$$\dfrac{\dfrac{\dfrac{pZ \sqsubseteq s_2}{pAZ \sqsubseteq s_1}\text{UNF}}{pZ \sqsubseteq s_2,\ \dfrac{pAU \sqsubseteq s_1}{\text{to subtree } \mathbf{l},\ pU \sqsubseteq s_2}\text{UNF}}\text{RED}}{}$$

$$\dfrac{\dfrac{\dfrac{\text{subtree } \mathbf{l} : pAAU \sqsubseteq s_1}{pAV \sqsubseteq s_1}\text{RED}}{pV \sqsubseteq s_1,\ \dfrac{pAAV \sqsubseteq s_1}{pAV \sqsubseteq s_1}\text{RED}}\text{UNF},\ pAU \sqsubseteq s_1}{}$$

**Fig. 2.** A tableaux tree.

3. $(\mathcal{L}(z), \mathcal{R})$ is a rule in $\text{RULE}_{op}$,
then $\left((V(\mathcal{T}) \cup V', E(\mathcal{T}) \cup \{(z, z') \mid z' \in V'\}), \mathcal{L} \cup \mathcal{L}'\right)$ is an *op*-tableaux tree, where $V'$ is a fresh set of vertices with $|V'| = |\mathcal{R}|$ and $\mathcal{L}'$ is a bijection from $V'$ to $\mathcal{R}$.

An *op*-tableaux tree is *successful* if either it consists of a sole successful vertex (which is the root) or all its leaves are successful. Intuitively, a tableaux tree is constructed inductively from an initial goal and (finite) repeated applications of rules.

Below we use our tableaux proof system to "prove" that $pZ \sqsubseteq_{\text{nc}} s_2$ in Example 1.

**Example 2.** A successful nc-tableaux tree for the pPDA and PA of Example 1 is depicted in Fig. 2, where $U := \{p \mapsto \{s_2\}\}$ and $V := \{p \mapsto \{s_1\}\}$ are extended symbols. UNF (resp. RED) is marked beside the inference when a UNF (resp. RED) rule is used. Let subtree $\mathbf{l}$ be the subtree rooted by $pAAU \sqsubseteq s_1$, and we write the whole tree into two parts for simplicity. The nc-tableaux tree is successful, since every leaf is a successful goal or has an ancestor with the same goal in the tree.

### 4.2. Soundness and completeness of tableaux system

In the following, we show the soundness and completeness of our tableaux proof system, i.e., $s \sqsubseteq_{op} s'$ iff there exists an *op*-tableaux tree rooted at $s \sqsubseteq s'$, for all goals $s \sqsubseteq s'$.

We first show that the tableaux trees possess a finiteness property. In the following, we define a *suffix* predicate $\text{suff}(\cdot, \cdot)$ by: $\text{suff}(\beta, \alpha)$ holds iff $\alpha = \gamma\beta$ for some $\gamma$.

**Lemma 16.** *Let* Suff *be the following set:*

$$\left\{\beta \in \Gamma^* \mid \exists \alpha \in \Gamma^*.\ \left(\left(\exists q \in Q\, \exists pA \xrightarrow{a} \mu.(\mu(q\alpha) > 0)\right) \wedge \text{suff}(\beta, \alpha)\right)\right\}\ .$$

*Define*

- $\text{GI}^{\text{a}}_{\text{suff}} := \{p\beta\alpha \sqsubseteq u \mid \beta \in \Gamma \cup \text{Suff}, \alpha \in E \cup \{\epsilon\}, p \in Q, u \in S\}$ *and*
- $\text{GI}^{\text{b}}_{\text{suff}} := \{u \sqsubseteq p\beta\alpha \mid \beta \in \Gamma \cup \text{Suff}, \alpha \in E \cup \{\epsilon\}, p \in Q, u \in S\}$.

*If* $s \sqsubseteq s' \in \text{GI}^{\text{a}}_{\text{suff}}$, *then* $\mathcal{L}(V(\mathcal{T})) \subseteq \text{GI}^{\text{a}}_{\text{suff}}$ *for all op-tableaux trees* $(\mathcal{T}, \mathcal{L})$ *with root label* $s \sqsubseteq s'$. *Analogously, if* $s \sqsubseteq s' \in \text{GI}^{\text{b}}_{\text{suff}}$, *then* $\mathcal{L}(V(\mathcal{T})) \subseteq \text{GI}^{\text{b}}_{\text{suff}}$ *for all op-tableaux trees* $(\mathcal{T}, \mathcal{L})$ *with root label* $s \sqsubseteq s'$.

**Proof.** The proof is a simple induction on the construction of tableaux trees. The base step where $(\mathcal{T}, \mathcal{L})$ is a single root is straightforward. It is also clear that $\text{GI}^{\text{a}}_{\text{suff}}$ and $\text{GI}^{\text{b}}_{\text{suff}}$ are closed under rule application of $\text{UNF}_{op}$ and $\text{RED}_{op}$. □

Now we show the soundness of our tableaux proof system, i.e., if there exists a successful tableaux tree rooted at $s \sqsubseteq s'$ then $s \sqsubseteq_{op} s'$. We first show that rules $\text{UNF}_{op}$ and $\text{RED}_{op}$ are backward sound, i.e., if all the subgoals are correct then the goal "to be proved" is correct.

**Lemma 17** *(Backward Soundness). Rule* $\text{UNF}_{op}$, $\text{RED}^{\text{a}}_{op}$ *and* $\text{RED}^{\text{b}}_{op}$ *satisfy the following properties:*

1. *Let* $(s \sqsubseteq s', \mathcal{R}) \in \text{UNF}_{op}$. *If* $r \sqsubseteq^n_{op} r'$ *for all* $r \sqsubseteq r' \in \mathcal{R}$, *then* $s \sqsubseteq^{n+1}_{op} s'$.
2. *Let* $(pA\alpha \sqsubseteq u, \{pAU \sqsubseteq u\} \cup \text{GI}^{\text{a}}_{\alpha,U}) \in \text{RED}^{\text{a}}_{op}$ *where*

$$\text{GI}^{\text{a}}_{\alpha,U} := \{q\alpha \sqsubseteq v \mid q \in Q, v \in U(q)\}\ .$$

*For all* $n \in \mathbb{N}$, *if* $pAU \sqsubseteq^{n+1}_{op} u$ *and* $q\alpha \sqsubseteq^n_{op} v$ *for all* $q\alpha \sqsubseteq v \in \text{GI}^{\text{a}}_{\alpha,U}$, *then* $pA\alpha \sqsubseteq^{n+1}_{op} u$.

3. *Let* $(u \sqsubseteq pA\alpha, \{u \sqsubseteq pAU\} \cup \mathrm{GI}^{\mathrm{b}}_{\alpha,U}) \in \mathrm{RED}^{\mathrm{b}}_{op}$ *where*

$$\mathrm{GI}^{\mathrm{b}}_{\alpha,U} := \{v \sqsubseteq q\alpha \mid q \in Q\,,\, v \in U(q)\}\ .$$

*For all* $n \in \mathbb{N}$, *if* $u \sqsubseteq^{n+1}_{op} pAU$ *and* $v \sqsubseteq^{n}_{op} q\alpha$ *for all* $v \sqsubseteq q\alpha \in \mathrm{GI}^{\mathrm{b}}_{\alpha,U}$, *then* $u \sqsubseteq^{n+1}_{op} pA\alpha$.

**Proof.** The backward soundness property for rule $\mathrm{UNF}_{op}$ follows directly from the rules of the tableaux and Definition 13.

For the backward soundness property for reduction, we only prove the case of $\mathrm{RED}^{\mathrm{a}}_{op}$; $\mathrm{RED}^{\mathrm{b}}_{op}$ is a symmetric case. We prove by induction on $n$ a more general lemma: for all $p\gamma\alpha \sqsubseteq u \in \mathrm{Goals}$ with $\gamma \in \Gamma^{+}$, it holds that for all $U \in E$, if $p\gamma U \sqsubseteq^{n+1}_{op} u$ and $q\alpha \sqsubseteq^{n}_{op} v$ for all $q\alpha \sqsubseteq v \in \mathrm{GI}^{\mathrm{a}}_{\alpha,U}$, then $p\gamma\alpha \sqsubseteq^{n+1}_{op} u$.

**Base step**: $n = 0$. Assume that $\gamma \in \Gamma^{+}$, $p\gamma U \sqsubseteq^{1}_{op} u$ and $q\alpha \sqsubseteq^{0}_{op} v$ for all $q\alpha \sqsubseteq v \in \mathrm{GI}^{\mathrm{a}}_{\alpha,U}$. Since $p\gamma U \sqsubseteq^{1}_{op} u$, for all $p\gamma \xrightarrow[\mathsf{nc}]{a} \mu$, there is $u \xrightarrow[op]{a} \nu$ such that $\mu_U \sqsubseteq^{0}_{op} \nu$ (cf. Lemma 8). Let $w : \lfloor \mu_U \rfloor \times \lfloor \nu \rfloor \to [0,1]$ be a weight function for $\mu_U \sqsubseteq^{0}_{op} \nu$. (We can restrict the domain to $\lfloor \mu_U \rfloor \times \lfloor \nu \rfloor$ since all other values are zero.) We define a weight function $w' : \lfloor \mu_\alpha \rfloor \times \lfloor \nu \rfloor \to [0,1]$ by: $w'(q\beta\alpha, \nu) = w(q\beta U, \nu)$ for all $q\beta \in \lfloor \mu \rfloor$ (note that $\beta \in \Gamma^{*}$) and $\nu \in \lfloor \nu \rfloor$. We prove that $w'$ is a weight function for $\mu_\alpha \sqsubseteq^{0}_{op} \nu$. The first two conditions in Definition 4 are straightforward to verify. For the third condition, suppose $w'(q\beta\alpha, \nu) > 0$ with $q\beta \in \lfloor \mu \rfloor$. Then $w(q\beta U, \nu) > 0$ and hence $q\beta U \sqsubseteq^{0}_{op} \nu$. If $\beta \neq \epsilon$, then by Definition 13, $Act(q\beta) = Act(\nu)$ and we have $q\beta\alpha \sqsubseteq^{0}_{op} \nu$; if $\beta = \epsilon$, then $\nu \in U(q)$ and $q\alpha \sqsubseteq \nu \in \mathrm{GI}^{\mathrm{a}}_{\alpha,U}$, which further implies $q\alpha \sqsubseteq^{0}_{op} \nu$. In either case $q\beta\alpha \sqsubseteq^{0}_{op} \nu$. So $w'$ is a weight function for the statement $\mu_\alpha \sqsubseteq^{0}_{op} \nu$. Also from $p\gamma U \sqsubseteq^{1}_{op} u$, we have $Act(p\gamma\alpha) = Act(u)$. Thus $p\gamma\alpha \sqsubseteq^{1}_{op} u$.

**Inductive step**: Assume that $\gamma \in \Gamma^{+}$, $p\gamma U \sqsubseteq^{n+2}_{op} u$ and $q\alpha \sqsubseteq^{n+1}_{op} v$ for all $q\alpha \sqsubseteq v \in \mathrm{GI}^{\mathrm{a}}_{\alpha,U}$. We prove that $p\gamma\alpha \sqsubseteq^{n+2}_{op} u$. Since $p\gamma U \sqsubseteq^{n+2}_{op} u$, for any $p\gamma \xrightarrow[\mathsf{nc}]{a} \mu$, there exists $u \xrightarrow[op]{a} \nu$ such that $\mu_U \sqsubseteq^{n+1}_{op} \nu$. Consider any $(q\beta U, \nu) \in \sqsubseteq^{n+1}_{op}$ with $\beta \in \Gamma^{*}$ and $\nu \in S$: if $\beta = \epsilon$ then $q\beta\alpha \sqsubseteq^{n+1}_{op} \nu$ since $q\alpha \sqsubseteq \nu \in \mathrm{GI}^{\mathrm{a}}_{\alpha,U}$; if $\beta \in \Gamma^{+}$ then we have $q\beta\alpha \sqsubseteq^{n+1}_{op} \nu$ by induction hypothesis; in any case, we have $q\beta\alpha \sqsubseteq^{n+1}_{op} \nu$. Thus by the same construction of weight function in the base step, we have $\mu_\alpha \sqsubseteq^{n+1}_{op} \nu$. Also we have $Act(p\gamma\alpha) = Act(u)$. Thus $p\gamma\alpha \sqsubseteq^{n+2}_{op} u$. □

Based on Lemma 17, we prove the soundness of our tableaux system as follows.

**Proposition 18.** *For all goals* $s \sqsubseteq s'$, *if there exists a successful op-tableaux tree rooted at a vertex labelled with* $s \sqsubseteq s'$, *then* $s \sqsubseteq_{op} s'$.

**Proof.** We only prove the case when $s \sqsubseteq s' \in \mathcal{E} \times S$, the other case is completely symmetric. Let $p_0\alpha_0 \sqsubseteq u_0 = s \sqsubseteq s'$. Suppose $p_0\alpha_0 \not\sqsubseteq_{op} u_0$ and $(\mathcal{T}, \mathcal{L})$ is a successful *op*-tableaux tree with root label $p_0\alpha_0 \sqsubseteq u_0$. Let the root of $(\mathcal{T}, \mathcal{L})$ be $z_0$. By Proposition 15, there exists $n_0 \in \mathbb{N}$ such that $p_0\alpha_0 \sqsubseteq^{n_0}_{op} u_0$ but $p_0\alpha_0 \not\sqsubseteq^{n_0+1}_{op} u_0$. Note that $(p_0\alpha_0, u_0) \in \sqsubseteq^{0}_{op}$ or otherwise the goal $p_0\alpha_0 \sqsubseteq u_0$ would be unsuccessful. By the backward soundness of $\mathrm{UNF}_{op}$ and $\mathrm{RED}_{op}$ (cf. Lemma 17), we can obtain the following statements (†):

- if the rule applied to $p_0\alpha_0 \sqsubseteq u_0$ (in the inductive construction of $(\mathcal{T}, \mathcal{L})$) belongs to $\mathrm{UNF}_{op}$, then there exists a child $z_1$ of $z_0$ labelled with $p'\alpha' \sqsubseteq u'$ such that $p'\alpha' \not\sqsubseteq^{n_0}_{op} u'$;
- if the rule applied to $p_0\alpha_0 \sqsubseteq u_0$ lies in $\mathrm{RED}^{\mathrm{a}}_{op}$, then there exists a child $z_1$ of $z_0$ whose label is either $p_0AU \sqsubseteq u_0$ with $p_0AU \not\sqsubseteq^{n_0+1}_{op} u_0$, or $q\alpha \sqsubseteq \nu$ with $q\alpha \not\sqsubseteq^{n_0}_{op} \nu$ for some $q\alpha \sqsubseteq \nu \in \mathrm{GI}^{\mathrm{a}}_{\alpha,U}$, where $A \in \Gamma$, $U \in E$ and $\alpha \in \Gamma^{+} \cdot (E + \epsilon)$ are specified by the rule.

In either case, there is a child $z_1$ of $z_0$ labelled with $p_1\alpha_1 \sqsubseteq u_1$ such that $p_1\alpha_1 \not\sqsubseteq^{n_0+1}_{op} u_1$. Choose such $z_1$ according to (†) arbitrarily. Let $n_1 \in \mathbb{N}$ be such that $p_1\alpha_1 \not\sqsubseteq^{n_1+1}_{op} u_1$ and $p_1\alpha_1 \sqsubseteq^{n_1}_{op} u_1$. Then $n_1 \leq n_0$. By performing (†) on $z_1$ and $p_1\alpha_1 \sqsubseteq u_1$ and so forth, we can recursively construct a finite sequence $\{(z_i, p_i\alpha_i \sqsubseteq u_i, n_i)\}_{0 \leq i \leq k}$ of length $k+1$ $(k \geq 1)$ such that $p_i\alpha_i \not\sqsubseteq^{n_i+1}_{op} u_i$ and $p_i\alpha_i \sqsubseteq^{n_i}_{op} u_i$, $\{n_i\}$ is decreasing, and the last vertex $z_k$ is a successful leaf. Since $p_k\alpha_k \not\sqsubseteq^{n_k+1}_{op} u_k$, the goal $p_k\alpha_k \sqsubseteq u_k$ cannot be successful. So the only possibility is that there is $j < k$ such that $p_k\alpha_k \sqsubseteq u_k = p_j\alpha_j \sqsubseteq u_j$. Consider the rule application from $(z_{k-1}, p_{k-1}\alpha_{k-1} \sqsubseteq u_{k-1})$ to $(z_k, p_k\alpha_k \sqsubseteq u_k)$. By (†):

- if the rule lies in $\mathrm{UNF}_{op}$, then $n_k < n_{k-1} \leq n_j$;
- if the rule lies in $\mathrm{RED}^{\mathrm{a}}_{op}$ and $p_k\alpha_k \sqsubseteq u_k \in \mathrm{GI}^{\mathrm{a}}_{\alpha,U}$ where $\alpha \in \Gamma^{+} \cdot (E + \epsilon)$ and $U \in E$ are determined by the rule, then $n_k < n_{k-1} \leq n_j$;
- if the rule lies in $\mathrm{RED}^{\mathrm{a}}_{op}$ and $p_k\alpha_k \sqsubseteq u_k = p_kAU \sqsubseteq u_k$ where $A \in \Gamma$ and $U \in E$ are determined by the rule, then $p_{k-1}\alpha_{k-1} \sqsubseteq u_{k-1} \neq p_kAU \sqsubseteq u_k$ and so $j < k-1$. By $p_k\alpha_k \sqsubseteq u_k = p_j\alpha_j \sqsubseteq u_j$, the rule application from $z_j$ to $z_{j+1}$ belongs to $\mathrm{UNF}_{op}$, which implies $n_{j+1} < n_j$. Hence $n_k < n_j$.

In either case, we have $n_k < n_j$. But then we have $p_k\alpha_k \not\sqsubseteq^{n_k+1}_{op} u_k$ and $p_k\alpha_k \sqsubseteq^{n_j}_{op} u_k$. Contradiction. □

Finally, we prove the completeness of our tableaux proof system, i.e., if $s \sqsubseteq_{op} s'$ then there exists a successful *op*-tableaux tree with root label $s \sqsubseteq s'$. We first prove a useful lemma below.

**Lemma 19.** *Let $pA\alpha \sqsubseteq_{op} u$ and $U \in E$ be an extended stack symbol such that $U(q) = \{v \in S \mid q\alpha \sqsubseteq_{op} v\}$ for all $q \in Q$. Then $pAU \sqsubseteq_{op} u$.*

**Proof.** We prove that the binary relation

$$\mathcal{R} := \{(q\beta U, v) \mid q \in Q, \beta \in \Gamma^*, v \in S, q\beta\alpha \sqsubseteq_{op} v\}$$

is an extended *op*-simulation. Consider any $(q\beta U, v) \in \mathcal{R}$. If $\beta = \epsilon$, then $q\alpha \sqsubseteq_{op} v$ and $v \in U(q)$; it follows that $(qU, v) \in \mathcal{R}_{\mathrm{b}}$. On the other hand, assume that $\beta \in \Gamma^+$. Then $Act(q\beta U) = Act(q\beta\alpha) = Act(v)$. Furthermore, for all $q\beta \xrightarrow[\mathrm{nc}]{a} \mu$, by $q\beta\alpha \sqsubseteq_{op} v$ there is $v \xrightarrow[op]{a} \nu$ such that $\mu_\alpha \sqsubseteq_{op} \nu$. We prove that $\mu_U \mathcal{R} \nu$. By $\mu_\alpha \sqsubseteq_{op} \nu$, there exists a weight function $w : \lfloor \mu_\alpha \rfloor \times \lfloor \nu \rfloor \to [0, 1]$ for the statement $\mu_\alpha \sqsubseteq_{op} \nu$. (We can restrict the domain of the weight function to $\lfloor \mu_\alpha \rfloor \times \lfloor \nu \rfloor$ since values at other places are zero.) We construct a weight function $w' : \lfloor \mu_U \rfloor \times \lfloor \nu \rfloor \to [0, 1]$ by: $w'(q'\gamma U, v') = w(q'\gamma\alpha, v')$ for all $q'\gamma \in \lfloor \mu \rfloor$ and $v' \in \lfloor \nu \rfloor$ (note that $\gamma \in \Gamma^*$). Then we show that $w'$ is a weight function for $\mu_U$ and $\nu$. The first two conditions in Definition 4 are straightforward to verify. For the third condition, consider any $q'\gamma \in \lfloor \mu \rfloor$ and $v' \in \lfloor \nu \rfloor$: assume that $w'(q'\gamma U, v') > 0$; then $w(q'\gamma\alpha, v') > 0$ and $q'\gamma\alpha \sqsubseteq_{op} v'$, which implies that $(q'\gamma U, v') \in \mathcal{R}$ by definition. Thus $\mathcal{R}$ is an extended *op*-simulation. □

By a symmetric proof, we obtain the following lemma.

**Lemma 20.** *Let $u \sqsubseteq_{op} pA\alpha$ and $U$ be an extended stack symbol such that $U(q) = \{v \in S \mid v \sqsubseteq_{op} q\alpha\}$ for all $q \in Q$. Then $u \sqsubseteq_{op} pAU$.*

The completeness of our tableaux proof system is shown as follows.

**Proposition 21.** *Let $s \sqsubseteq s' \in \mathrm{GI}^{\mathrm{a}}_{\mathrm{suff}} \cup \mathrm{GI}^{\mathrm{b}}_{\mathrm{suff}}$. If $s \sqsubseteq_{op} s'$ then there is a successful op-tableaux tree with root label $s \sqsubseteq s'$.*

**Proof.** We only prove the case for $s \sqsubseteq s' \in \mathrm{GI}^{\mathrm{a}}_{\mathrm{suff}}$, the other case is symmetric. Below we inductively construct a sequence $\{(\mathcal{T}_n, \mathcal{L}_n)\}_{1 \le n \le k}$ of *op*-tableaux trees, each with root label $s \sqsubseteq s'$, such that $(\mathcal{T}_k, \mathcal{L}_k)$ is a successful *op*-tableaux tree.

Initially, $(\mathcal{T}_1, \mathcal{L}_1)$ is the tableaux tree which contains only a root labelled with $s \sqsubseteq s'$. Then assume that $(\mathcal{T}_n, \mathcal{L}_n)$ is constructed. If all leaves of $(\mathcal{T}_n, \mathcal{L}_n)$ are terminal, then the construction is ended. Otherwise, we choose an arbitrary non-terminal leaf $z$ of $(\mathcal{T}_n, \mathcal{L}_n)$ and construct $(\mathcal{T}_{n+1}, \mathcal{L}_{n+1})$ as follows:

1. if $\mathcal{L}_n(z) = pA\alpha \sqsubseteq u$ with $\alpha \in E \cup \{\epsilon\}$, then we apply the $\mathrm{UNF}_{op}$ rule

   $$(pA\alpha \sqsubseteq u, \{(q\beta, v) \in Succ(pA\alpha) \times Succ(u) \mid q\beta \sqsubseteq_{op} v\})$$

   to the vertex $z$ to form $(\mathcal{T}_{n+1}, \mathcal{L}_{n+1})$;
2. if $\mathcal{L}_n(z) = pA\alpha \sqsubseteq u$ with $\alpha \in \Gamma^+ \cdot (E + \epsilon)$, then we apply the $\mathrm{RED}_{op}$ rule

   $$(pA\alpha \sqsubseteq u, \{pAU \sqsubseteq u\} \cup \mathrm{GI}^{\mathrm{a}}_{\alpha,U})$$

   where $U$ is defined by: $U(q) := \{v \in S \mid q\alpha \sqsubseteq_{op} v\}$ for all $q \in Q$. Note that by Lemma 19, we have $pAU \sqsubseteq_{op} u$.

In the inductive construction above, one easily sees that we only "append" goals $s \sqsubseteq s'$ such that $s \sqsubseteq_{op} s'$. Thus if the inductive construction ends in a finite number of steps, then the last tableaux tree $(\mathcal{T}_k, \mathcal{L}_k)$ should be successful, i.e., it won't contain any unsuccessful leaves. Below we show that the inductive construction above ends in a finite number of steps. This is done by contraposition. Suppose that we obtain an infinite sequence $\{(\mathcal{T}_n, \mathcal{L}_n)\}_{n \in \mathbb{N}}$ of tableaux trees with $\mathcal{T}_n = (V(\mathcal{T}_n), E(\mathcal{T}_n))$ from the inductive construction above. Let $\mathcal{T} := \left(\bigcup_{n \in \mathbb{N}} V(\mathcal{T}_n), \bigcup_{n \in \mathbb{N}} E(\mathcal{T}_n)\right)$ and $\mathcal{L} := \bigcup_{n \in \mathbb{N}} \mathcal{L}_n$. Then $(\mathcal{T}, \mathcal{L})$ is an infinite vertex-labelled tree. By the way of rule application, $\mathcal{T}$ is also finitely branching. Thus by König's Lemma, there is an infinite path in $\mathcal{T}$. Then by Lemma 16, on such infinite path there must be vertices $z, z'$ such that $z \neq z'$ and $\mathcal{L}(z) = \mathcal{L}(z')$. It follows that either $z$ or $z'$ is a successful leaf of the tableaux tree $\mathcal{T}_{n^*}$, where $n^*$ is the smallest number such that $z, z'$ are leaves of $\mathcal{T}_{n^*}$. Then the inductive construction should stop expanding at $z$ (or $z'$, depending on which one is a leaf), which leads to a contradiction. □

We have ended the demonstration of the soundness and completeness of our tableaux proof system. The finite tableaux proof system yields a non-deterministic algorithm for checking $s \sqsubseteq_{op} s'$.

**Corollary 22.** *The problem whether $s \sqsubseteq_{op} s'$ for a given $(s, s') \in (Q \times \Gamma) \odot S$ is decidable.*

### 4.3. Partition-refinement algorithm

Based on the tableaux proof system, we develop an algorithm that decides $\sqsubseteq_{op}$. The algorithm will use a partition-refinement technique to achieve the EXPTIME upper bound, which is the main result of this article. Recall that we have fixed a pPDA $(Q, \Gamma, L, \rightarrowtail)$ and a finite PA $(S, Act, \rightarrow)$. Below we denote by $K$ the integrated size of the pPDA and the finite PA, where the numerical values (probability values) are assumed to be rational and are represented in binary.

**Theorem 1.** *The problem whether $s \sqsubseteq_{op} s'$ for a given $(s, s') \in (Q \times \Gamma) \odot S$ can be decided in $\mathcal{O}(poly(K) \cdot 8^{|S|\cdot|Q|})$ time. Thus, if $|Q|$ and $|S|$ are fixed, then the problem is in PTIME.*

**Proof.** We assume that $s \sqsubseteq s' \in (Q \times \Gamma) \times S$ and $op =$ 'c'. The other cases are similar. We present a partition-refinement algorithm to decide whether $s \sqsubseteq_{\mathsf{c}} s'$. Formally, we construct a finite decreasing sequence of sets of goals $\{X_n\}_{1 \le n \le k}$ where the last element $X_k$ contains all the correct goals in $\mathrm{Gl}^{\mathrm{a}}_{\mathrm{suff}}$.

The construction is as follows. Initially, $X_1 = \mathrm{Gl}^{\mathrm{a}}_{\mathrm{suff}}$. Then $X_{n+1} \subseteq \mathrm{Gl}^{\mathrm{a}}_{\mathrm{suff}}$ is constructed from $X_n$ as follows: $s \sqsubseteq s' \in X_{n+1}$ iff (i) $s \sqsubseteq s' \in X_n$ and (ii) either $s \sqsubseteq s'$ is successful or there exists $Y \subseteq X_n$ such that $(s \sqsubseteq s', Y) \in \mathrm{RULE}_{\mathsf{c}}$. Note that $|\mathrm{Gl}^{\mathrm{a}}_{\mathrm{suff}}| = \mathcal{O}(K^4 \cdot 2^{|S|\cdot|Q|})$.

The computation from $X_n$ to $X_{n+1}$ can be done in $\mathcal{O}(poly(K) \cdot 4^{|S|\cdot|Q|})$ time by the following procedure. Let $s \sqsubseteq s' \in X_n$. We check whether $s \sqsubseteq s' \in X_{n+1}$ as follows:

- if $s \sqsubseteq s' = pA\alpha \sqsubseteq u$ with $\alpha \in \Gamma^+ \cdot (E + \epsilon)$, we check whether $\{pAU \sqsubseteq u\} \cup \mathrm{Gl}^{\mathrm{a}}_{\alpha,U} \subseteq X_n$ for some $U \in E$;
- if $s \sqsubseteq s' = pA\alpha \sqsubseteq u$ with $\alpha \in E \cup \{\epsilon\}$, we check whether for all $pA\alpha \xrightarrow[\mathsf{nc}]{a} \mu$, there exists $u \xrightarrow[\mathsf{c}]{a} \nu$ such that $\mu X_n \nu$ (treat $X_n$ as a binary relation such that $s_1 \sqsubseteq s_2 \in X_n$ iff $(s_1, s_2) \in X_n$); this can be checked by examining whether the following linear inequation system (with variables $\{x_\nu\}_{u \xrightarrow[\mathsf{nc}]{a} \nu}$ and $\{y_{(s'',\nu)}\}_{(s'',\nu) \in \lfloor\mu\rfloor \times S}$) has a solution:
  - $\sum_{u \xrightarrow[\mathsf{nc}]{a} \nu} x_\nu = 1$;
  - $x_\nu \ge 0$ for all $u \xrightarrow[\mathsf{nc}]{a} \nu$;
  - $\sum_{\nu \in S} y_{(s'',\nu)} = \mu(s'')$ for all $s'' \in \lfloor\mu\rfloor$;
  - $\sum_{s'' \in \lfloor\mu\rfloor} y_{(s'',\nu)} = \sum_{u \xrightarrow[\mathsf{nc}]{a} \nu} x_\nu \cdot \nu(\nu)$ for all $\nu \in S$;
  - $y_{(s'',\nu)} \ge 0$ for all $(s'', \nu) \in \lfloor\mu\rfloor \times S$;
  - $y_{(s'',\nu)} = 0$ whenever $(s'', \nu) \notin X_n$.

  This can be solved in polynomial time in $K$ (cf. [33]).

Since $X_{n+1} \subseteq X_n$, there exists $k \le |\mathrm{Gl}^{\mathrm{a}}_{\mathrm{suff}}|$ such that $X_{k+1} = X_k$. We show that for all $s \sqsubseteq s' \in \mathrm{Gl}^{\mathrm{a}}_{\mathrm{suff}}$, $s \sqsubseteq_{\mathsf{c}} s'$ iff $s \sqsubseteq s' \in X_k$. On one hand, assume that $s \sqsubseteq_{\mathsf{c}} s'$. Let $(\mathcal{T}, \mathcal{L})$ be the tableaux tree constructed in the proof of Proposition 21 for the goal $s \sqsubseteq s'$. An easy induction on $n$ shows that $\mathcal{L}(V(\mathcal{T})) \subseteq X_n$ for all $n \in \mathbb{N}$. Thus $s \sqsubseteq s' \in X_k$. On the other hand, assume that $s \sqsubseteq s' \in X_k$. Since for all goals $s_1 \sqsubseteq s_2 \in X_k$ which are not successful, there is $(s_1 \sqsubseteq s_2, Y) \in \mathrm{RULE}_{\mathsf{c}}$ such that $Y \subseteq X_k$. Thus we can iteratively apply rules to the root labelled with $s \sqsubseteq s'$ (and non-terminal leaves), which results in a successful c-tableaux tree similar to the construction in the proof of Proposition 21. Thus, $s \sqsubseteq_{\mathsf{c}} s'$ by Proposition 18. Then the result follows from the fact that $k \le |\mathrm{Gl}^{\mathrm{a}}_{\mathrm{suff}}|$. □

Our partition-refinement algorithm gives a complete base for the preorder relation. If we obtain the base, then for any pair we can decide whether they are in the preorder relation rather than compute a new base. The bases of tableaux system for different pairs may be different and not complete for all pairs. Kučera and Mayr [32]'s algorithm to compute the finite semantic base for non-probabilistic BPA is a simplified tableaux. For every subgoal, they check whether it is in the closure of the current base, and regard it as a leaf if so.

## 5. EXPTIME-hardness

In this section, we show that deciding $\sqsubseteq_{op}$ is EXPTIME-hard, whenever $op =$ nc or $op =$ c. We prove this by providing a straightforward reduction from the non-probabilistic EXPTIME-hardness result obtained in [9]. Our main efforts lie in the treatment of the additional "$Act(s) = Act(s')$" condition in Definition 5 which is not part of the definition of non-probabilistic simulation preorder. Firstly, we define a variation of $\sqsubseteq_{op}$.

**Definition 23.** $\mathcal{M} = (S, Act, \rightarrow)$ be a PA. Define $\preccurlyeq_{op}$ to be the union of all binary relations $\mathcal{R} \subseteq S \times S$ such that for all $(s, s') \in \mathcal{R}$, whenever $s \xrightarrow[\mathsf{nc}]{a} \mu$ there is $s' \xrightarrow[op]{a} \mu'$ with $\mu \mathcal{R} \mu'$.

In other words, $\preccurlyeq_{op}$ is defined in a similar way as $\sqsubseteq_{op}$, however without the "$Act(s) = Act(s')$" requirement. Then we embed non-probabilistic transition systems into PA.

**Definition 24.** A PA $(S, Act, \rightarrow)$ is *Dirac* if $\mu$ is Dirac for all $(s, a, \mu) \in \rightarrow$. A pPDA $(Q, \Gamma, L, \rightarrowtail)$ is *Dirac* if $\mu$ is Dirac for all $(pA, a, \mu) \in \rightarrowtail$.

Note that a Dirac pPDA induces a Dirac PA. Dirac PA correspond to transition systems. It is not hard to verify that $\preccurlyeq_{\mathsf{nc}}$ (over Dirac PA) coincides with the (non-probabilistic) simulation preorder (cf. [13]) over non-probabilistic transition systems. From [9], deciding $\preccurlyeq_{\mathsf{nc}}$ is EXPTIME-complete between Dirac pPDA and Dirac finite PA in both direction. Below we reduce $\preccurlyeq_{op}$ to $\sqsubseteq_{op}$ under Dirac PA. The following proposition allows us to focus solely on the case $op = \mathsf{nc}$. Below let $\mathcal{D}[x]$ denote the Dirac distribution at the element $x$.

**Proposition 25.** *For any Dirac PA $\mathcal{M} = (S, Act, \rightarrow)$, $\preccurlyeq_{nc} = \preccurlyeq_{c}$ and $\sqsubseteq_{nc} = \sqsubseteq_{c}$.*

**Proof.** It is clear that $\preccurlyeq_{\mathsf{nc}} \subseteq \preccurlyeq_{\mathsf{c}}$ and $\sqsubseteq_{\mathsf{nc}} \subseteq \sqsubseteq_{\mathsf{c}}$. Below we prove the reverse direction. We only prove the case $\sqsubseteq_{\mathsf{c}} \subseteq \sqsubseteq_{\mathsf{nc}}$, since the proof for the other is similar. Let $s \sqsubseteq_{\mathsf{c}} s'$ and $s \xrightarrow[\mathsf{nc}]{a} \mu$. By definition, there exists $s' \xrightarrow[\mathsf{c}]{a} \mu'$ such that $\mu \sqsubseteq_{\mathsf{c}} \mu'$. Since $\mu$ is Dirac, there exists $s'' \in \lfloor \mu' \rfloor$ such that $s' \xrightarrow[\mathsf{nc}]{a} \mathcal{D}[s'']$ and $\mu \sqsubseteq_{\mathsf{c}} \mathcal{D}[s'']$. It follows from the arbitrary choice of $s, s'$ and $s \xrightarrow[\mathsf{nc}]{a} \mu$ that $\sqsubseteq_{\mathsf{c}}$ is an nc-simulation, which implies $\sqsubseteq_{\mathsf{c}} \subseteq \sqsubseteq_{\mathsf{nc}}$. □

Now we reduce $\preccurlyeq_{\mathsf{nc}}$ between a Dirac pPDA $M = (Q, \Gamma, L, \rightarrowtail)$ and a Dirac finite PA $P = (S, Act, \rightarrow)$, to $\sqsubseteq_{\mathsf{nc}}$ between a Dirac pPDA $(Q', \Gamma', L, \rightarrowtail')$ and a Dirac finite PA $(S', Act', \rightarrow')$. The reduction is as follows:

1. $Q' = Q \cup \{p_\perp\}$ and $S' = S \cup \{s_\perp\}$ where $p_\perp \notin Q$ and $s_\perp \notin S$ are fresh elements;
2. $\Gamma' = \Gamma \cup \{A_\perp\}$ where $A_\perp \notin \Gamma$ is a fresh (bottom) stack symbol;
3. $Act' = Act$;
4. $\rightarrowtail' = \rightarrowtail \cup \{(pA, a, \mathcal{D}[p_\perp]) \mid p \in Q, A \in \Gamma', a \in L \cup Act\}$;
5. $\rightarrow' = \rightarrow \cup \{(s, a, \mathcal{D}[s_\perp]) \mid s \in S, a \in L \cup Act\}$.

The basic idea is that we try to "amend" $P$ and $M$ so that pairs in $\preccurlyeq_{\mathsf{nc}}$ will have the same enabled-actions. It is not hard to prove that for all $p\alpha \in Q \times \Gamma^*$ and $s \in S$, $p\alpha \preccurlyeq_{\mathsf{nc}} s$ (resp. $s \preccurlyeq_{\mathsf{nc}} p\alpha$) iff $p\alpha A_\perp \sqsubseteq_{\mathsf{nc}} s$ (resp. $s \sqsubseteq_{\mathsf{nc}} p\alpha A_\perp$). Thus deciding $\sqsubseteq_{\mathsf{nc}}$ between Dirac pPDA and Dirac finite PA is EXPTIME-hard. Then we have the following theorem.

**Theorem 2.** *Deciding $\sqsubseteq_{nc}$ and $\sqsubseteq_{c}$ between probabilistic pushdown automata and finite probabilistic automata in both directions (of the simulation preorder) is EXPTIME-complete.*

## 6. Conclusion

In this article, we showed that (probabilistic) simulation preorder between a probabilistic pushdown automaton $(Q, \Gamma, L, \rightarrowtail)$ and a finite probabilistic automaton $(S, Act, \rightarrow)$ is EXPTIME-complete; this result holds for both directions. Furthermore, when both $|Q|$ and $|S|$ are fixed, the problem is decidable in polynomial time. These results extend their non-probabilistic counterparts in [9]. The tableaux are obtained by extending Stirling's method of extended stack symbols [17, 18] to simulation preorder, which is originally used to demonstrate the decidability of bisimilarity on non-probabilistic pushdown processes. And the algorithm is based on the principle of partition refinement.

## Declaration of Competing Interest

There is no competing interest.

## References

[1] H. Fu, J.-P. Katoen, Deciding probabilistic simulation between probabilistic pushdown automata and finite-state systems, in: S. Chakraborty, A. Kumar (Eds.), FSTTCS, in: LIPIcs, vol. 13, Schloss Dagstuhl–Leibniz–Zentrum Fuer Informatik, 2011, pp. 445–456.
[2] J. Hromkovic, G. Schnitger, On probabilistic pushdown automata, Inf. Comput. 208 (8) (2010) 982–995.
[3] S.P. Abney, D.A. McAllester, F. Pereira, Relating probabilistic grammars and automata, in: 37th Annual Meeting of the Association for Computational Linguistics, 1999, pp. 542–549.
[4] I.I. Macarie, M. Ogihara, Properties of probabilistic pushdown automata, Theor. Comput. Sci. 207 (1) (1998) 117–130.
[5] J.A. Bergstra, J.W. Klop, Process algebra for synchronous communication, Inf. Control 60 (1–3) (1984) 109–137.
[6] M. Jaeger, Complex probabilistic modeling with recursive relational Bayesian networks, Ann. Math. Artif. Intell. 32 (1–4) (2001) 179–220.
[7] F. Olmedo, B.L. Kaminski, J-P. Katoen, C. Matheja, Reasoning about recursive probabilistic programs, in: LICS, ACM, 2016, pp. 672–681.
[8] O. Evans, A. Stuhlmüller, J. Salvatier, D. Filan, Modeling agents with probabilistic programs, http://agentmodels.org, 2017. (Accessed 30 November 2018).
[9] A. Kučera, R. Mayr, On the complexity of checking semantic equivalences between pushdown processes and finite-state processes, Inf. Comput. 208 (7) (2010) 772–796.
[10] R. Segala, N.A. Lynch, Probabilistic simulations for probabilistic processes, Nord. J. Comput. 2 (2) (1995) 250–273.
[11] B. Jonsson, K.G. Larsen, Specification and refinement of probabilistic processes, in: LICS, IEEE Computer Society, 1991, pp. 266–277.
[12] K.G. Larsen, A. Skou, Bisimulation through probabilistic testing, Inf. Comput. 94 (1) (1991) 1–28.

[13] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT Press, 2008.
[14] A. McIver, C. Morgan, Abstraction, Refinement and Proof for Probabilistic Systems, Monographs in Computer Science, Springer, 2005.
[15] T. Brázdil, A. Kučera, O. Strazovský, Deciding probabilistic bisimilarity over infinite-state probabilistic systems, Acta Inform. 45 (2) (2008) 131–154.
[16] P. Jancar, A. Kucera, R. Mayr, Deciding bisimulation-like equivalences with finite-state processes, Theor. Comput. Sci. 258 (1–2) (2001) 409–433.
[17] C. Stirling, Decidability of bisimulation equivalence for normed pushdown processes, Theor. Comput. Sci. 195 (2) (1998) 113–131.
[18] C. Stirling, Decidability of DPDA equivalence, Theor. Comput. Sci. 255 (1–2) (2001) 1–31.
[19] V. Forejt, P. Jancar, S. Kiefer, J. Worrell, Language equivalence of probabilistic pushdown automata, Inf. Comput. 237 (2014) 1–11.
[20] V. Forejt, P. Jancar, S. Kiefer, J. Worrell, Game characterization of probabilistic bisimilarity, and applications to pushdown automata, Log. Methods Comput. Sci. 14 (4) (2018).
[21] T. Brázdil, J. Esparza, S. Kiefer, A. Kucera, Analyzing probabilistic pushdown automata, Form. Methods Syst. Des. 43 (2) (2013) 124–163.
[22] K. Etessami, M. Yannakakis, Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations, J. ACM 56 (1) (2009) 1.
[23] K. Etessami, M. Yannakakis, Recursive Markov decision processes and recursive stochastic games, in: L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (Eds.), ICALP, in: Lecture Notes in Computer Science, vol. 3580, Springer, 2005, pp. 891–903.
[24] A. Kucera, J. Esparza, R. Mayr, Model checking probabilistic pushdown automata, Log. Methods Comput. Sci. 2 (1) (2006).
[25] T. Brázdil, A. Kučera, O. Strazovský, On the decidability of temporal properties of probabilistic pushdown automata, in: V. Diekert, B. Durand (Eds.), STACS, in: Lecture Notes in Computer Science, vol. 3404, Springer, 2005, pp. 145–157.
[26] T. Brázdil, V. Brozek, V. Forejt, A. Kucera, Branching-time model-checking of probabilistic pushdown automata, J. Comput. Syst. Sci. 80 (1) (2014) 139–156.
[27] I. Walukiewicz, Pushdown processes: games and model-checking, Inf. Comput. 164 (2) (2001) 234–263.
[28] J.F. Groote, H. Hüttel, Undecidable equivalences for basic process algebra, Inf. Comput. 115 (2) (1994) 354–371.
[29] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edition, John Wiley & Sons, Inc., New York, NY, USA, 1994.
[30] O. Burkart, D. Caucal, F. Moller, B. Steffen, Verification on infinite structures, in: J. Bergstra, A. Ponse, S. Smolka (Eds.), Handbook of Process Algebra, Elsevier Science, Amsterdam, The Netherlands, 2001, pp. 545–623, Ch. 9.
[31] A. Bianco, L. de Alfaro, Model checking of probabilistic and nondeterministic systems, in: P.S. Thiagarajan (Ed.), FSTTCS, in: Lecture Notes in Computer Science, vol. 1026, Springer, 1995, pp. 499–513.
[32] A. Kucera, R. Mayr, A generic framework for checking semantic equivalences between pushdown automata and finite-state automata, J. Comput. Syst. Sci. 91 (2018) 82–103.
[33] A. Schrijver, Theory of Linear and Integer Programming, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, 1999.