Deciding Probabilistic Simulation between Probabilistic Pushdown Automata and Finite-State Systems

Hongfei Fu*[1] and Joost-Pieter Katoen†[2]

1,2 RWTH Aachen University, Ahornstraße 55, D-52074 Aachen, Germany

Abstract

This paper studies the decidability and computational complexity of checking probabilistic simulation pre-order between probabilistic pushdown automata (pPDA) and (probabilistic) finite-state systems. We show that checking classical and combined probabilistic similarity are EXPTIME-complete in both directions and become polynomial if both the number of control states of the pPDA and the size of the finite-state system are fixed. These results show that checking probabilistic similarity is as hard as checking similarity in the standard, i.e., non-probabilistic setting.

1998 ACM Subject Classification F.2.2, F.3.1

Keywords and phrases infinite-state systems, probabilistic simulation, probabilistic pushdown automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2011.445

1 Introduction

Probabilities are a convenient means to model uncertainty [15, 11] in transition systems. They are essential for modelling, e.g., randomized algorithms, unreliable and unpredictable system behaviour, or environmental uncertainties. Discrete-Time Markov Chains (DTMC) and Markov Decision Processes (MDPs) are popular probabilistic extensions of transition systems. An important technique in the formal verification of probabilistic systems is semantical equivalence or pre-order checking. Here, one focuses on comparing the behaviour of a probabilistic system (the "implementation") with its intended behaviour (the "specification"). Probabilistic bisimulation [16, 18, 3] and simulation [16, 12, 18, 3] are typical instances that act as a basis for comparison. For finite-state systems, efficient algorithms have been established for both notions [6, 1, 2]. An additional interest in checking probabilistic (bi)simulation between two systems is the preservation of temporal logic formulas: e.g., probabilistic bisimulation equivalence preserves PCTL while probabilistic simulation equivalence preserves a safety fragment of PCTL [3, 18].

This paper considers probabilistic pushdown automata (pPDA) [7] as implementation models and Segala's probabilistic automata [18] as specifications. pPDA are a natural abstract model for probabilistic procedural programs and are equally expressive as recursive Markov chains [9]. In fact, there are linear-time transformations between these two models [9]. Whereas the verification of pPDA and recursive Markov chains has been addressed quite extensively in the literature [8, 13], their use in semantical equivalence checking has so far been restricted to a study of (strong) probabilistic bisimulation [5]. Probabilistic

* Supported by a CSC scholarship.
† Supported by the EU FP7 project MoVeS.

31st Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011).
Editors: Supratik Chakraborty, Amit Kumar; pp. 445–456

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

automata [18] are an orthogonal extension of labelled transition systems with probabilities and subsume both DTMCs and MDPs. They are used as semantical model of probabilistic process algebras and constitute the core of PIOA, an input/output version that is frequently used for describing randomized distributed algorithms. In this paper we study the decidability and complexity of checking (combined) strong simulation pre-order between pPDA and finite-state probabilistic automata. Our motivation is that system specifications can often be captured by a finite system, thus for many practical problems it is useful and sufficient to check if a (possibly infinite) probabilistic system is semantically equivalent to a finite one.

Our approach to tackle the above problem is to attempt to lift techniques for related problems in the non-probabilistic setting to the probabilistic one. The obvious candidate is the result by Kučera and Mayr [14] stating that strong simulation pre-order between PDA and finite-state systems (in both directions) is decidable in EXPTIME. Their proof technique relies on the EXPTIME-completeness of model checking modal μ-Calculus against PDA. As model checking PCTL over fully probabilistic PDA is undecidable [4], extending this approach is however hopeless. Instead, we take an alternative route and extend the method by Stirling [19, 20] for showing the decidability of bisimulation equivalence over PDA's. This extension will be non-trivial as (non-probabilistic) strong simulation pre-order is undecidable over PDA's [10]. Our technique enables us to show that probabilistic (combined) simulation pre-order (in both directions) is in EXPTIME. The problem is decidable in PTIME if both the number of control states of the pPDA and the size of the finite-state system are fixed. By exploiting a hardness result in [14], we achieve that the considered problem is EXPTIME-complete.

2 Preliminaries

2.1 Probabilistic Transition Systems

In this subsection we introduce the notion of probabilistic transition systems (pTS) which corresponds to "simple probabilistic automata" defined in [18].

▶ **Definition 1** (Probability Distribution)**.** Let S be a countable set. A function $\mu : S \to [0,1]$ is a *probability distribution over* S if $\sum_{s\in S} \mu(s) = 1$. We say that μ is *finite*, if the set $\lfloor\mu\rfloor := \{s \in S \mid \mu(s) > 0\}$ is finite, and $\mu(s) \in \mathbb{Q}$ for all $s \in S$. Let $\mathcal{D}(S)$ (resp. $\mathcal{D}_{\mathrm{f}}(S)$) denote the set of probability distributions (resp. finite probability distributions) over S.

The definition of probabilistic transition system is given as follows:

▶ **Definition 2** (Probabilistic Transition System)**.** A *probabilistic transition system* (pTS) $\mathcal{T}$ is a triple (S, A, Ω) where S is a countable set of *states*, A is a countable set of *actions* and $\Omega \subseteq S \times A \times \mathcal{D}(S)$ is a set of *transitions*. We define the following notations related to $\mathcal{T}$:

- $\mathrm{der}^{\mathcal{T}}(s,a) := \{\mu \in \mathcal{D}(S) \mid (s,a,\mu) \in \Omega\}$ for $s \in S$ and $a \in A$.
- $\mathrm{Der}^{\mathcal{T}}(s) := \bigcup_{a\in A} \bigcup_{\mu\in \mathrm{der}^{\mathcal{T}}(s,a)} \lfloor\mu\rfloor$ for $s \in S$.
- $\mathrm{Act}^{\mathcal{T}}(s) := \{a \in A \mid \mathrm{der}^{\mathcal{T}}(s,a) \neq \emptyset\}$ for $s \in S$.

We write $s \xrightarrow{a}_{\mathrm{n}} \mu \in \Omega$ instead of $(s,a,\mu) \in \Omega$ (where 'n' stands for "non-combined"). We omit '$\mathcal{T}$', 'Ω' in the notations above if the context is clear.

The following definition illustrates the notion of "combined transitions", which is originally introduced by Segala [18] to model stochastic adversaries.

▶ **Definition 3** (Combined Transitions)**.** Let (S, A, Ω) be a pTS. We write $s \xrightarrow{a}_{\mathrm{c}} \mu$ (where 'c' stands for "combined") if there exists a finite or infinite sequence $\{(\mu_i, d_i)\}_i$ (where

$\mu_i \in \mathcal{D}(S)$ and $d_i \in \mathbb{R}_{\geq 0}$ for every i) such that: (i) $s \xrightarrow{a}_{\mathrm{n}} \mu_i$ for all i; (ii) $\sum_i d_i = 1$; and (iii) $\mu(s) = \sum_i d_i \cdot \mu_i(s)$ for all $s \in S$.

In this paper, we will also concern the notion of "finiteness" in a pTS.

▶ **Definition 4.** A pTS $\mathcal{T}$ is *locally finite* if for all $s \in S$ and $a \in A$, $\mathrm{der}(s,a)$ is a finite subset of $\mathcal{D}_{\mathrm{f}}(S)$. Further $\mathcal{T}$ is *finite* if $\mathcal{T}$ is locally finite and both S and A are finite.

2.2 Probabilistic Simulation

In this subsection we introduce the notion of probabilistic simulations which corresponds to *strong simulation* and *strong probabilistic simulation* defined by Segala [18]. Below we fix a pTS $\mathcal{T} = (S, A, \Omega)$. The following definition originates from [12].

▶ **Definition 5.** [12] Let $\mathcal{R} \subseteq S \times S$. The binary relation $\overline{\mathcal{R}} \subseteq \mathcal{D}(S) \times \mathcal{D}(S)$ is defined as follows: $\mu \overline{\mathcal{R}} \nu$ iff there is a weight function $w : S \times S \to [0,1]$ such that:
- for all $s \in S$, $\sum_{t \in S} w(s,t) = \mu(s)$;
- for all $t \in S$, $\sum_{s \in S} w(s,t) = \nu(t)$;
- for all $(s,t) \in S \times S$, if $w(s,t) > 0$ then $(s,t) \in \mathcal{R}$.

For the sake of simplicity, we write $\mu \mathcal{R} \nu$ instead of $\mu \overline{\mathcal{R}} \nu$.

In this paper, it is somewhat convenient to consider an equivalent definition of Definition 5.

▶ **Definition 6.** Let $\mathcal{R} \subseteq S \times S$. The binary relation $\overline{\mathcal{R}} \subseteq \mathcal{D}(S) \times \mathcal{D}(S)$ is defined as follows: $\mu \overline{\mathcal{R}} \nu$ iff there is a weight function $w : \lfloor \mu \rfloor \times \lfloor \nu \rfloor \to [0,1]$ such that:
- for all $s \in \lfloor \mu \rfloor$, $\sum_{t \in \lfloor \nu \rfloor} w(s,t) = \mu(s)$;
- for all $t \in \lfloor \nu \rfloor$, $\sum_{s \in \lfloor \mu \rfloor} w(s,t) = \nu(t)$;
- for all $(s,t) \in \lfloor \mu \rfloor \times \lfloor \nu \rfloor$, if $w(s,t) > 0$ then $(s,t) \in \mathcal{R}$.

Then the definition of probabilistic simulation is given as follows, where $\sqsubseteq_{\mathrm{n}}$ (resp. $\sqsubseteq_{\mathrm{c}}$) corresponds to *strong simulation* (resp. *strong probabilistic simulation*) in [18], respectively.

▶ **Definition 7** (κ-Simulation)**.** Let $\kappa \in \{\mathrm{n}, \mathrm{c}\}$. A binary relation $\mathcal{R} \subseteq S \times S$ is a *κ-simulation* iff for all $(s,t) \in \mathcal{R}$: (i) $\mathrm{Act}(s) = \mathrm{Act}(t)$ and (ii) whenever $s \xrightarrow{a}_{\mathrm{n}} \mu$ there is $t \xrightarrow{a}_{\kappa} \nu$ such that $\mu \mathcal{R} \nu$. The κ-similarity, denoted $\sqsubseteq_\kappa$, is the union of all κ-simulations.

By definition, one can verify that $\sqsubseteq_\kappa$ is a κ-simulation. Below we define approximants of κ-simulation. We use κ to indicate either 'n' or 'c'.

▶ **Definition 8.** The family $\{\sqsubseteq_\kappa^n\}_{n \in \mathbb{N}_0}$ of approximations of $\sqsubseteq_\kappa$ is inductively defined by:
- $\sqsubseteq_\kappa^0 = \{(s,t) \in S \times S \mid \mathrm{Act}(s) = \mathrm{Act}(t)\}$;
- $(s,t) \in \sqsubseteq_\kappa^{n+1}$ iff $(s,t) \in \sqsubseteq_\kappa^0$ and whenever $s \xrightarrow{a}_{\mathrm{n}} \mu$ there is $t \xrightarrow{a}_{\kappa} \nu$ such that $\mu \sqsubseteq_\kappa^n \nu$.

The following property can be easily proved by induction on n.

▶ **Lemma 9.** *For any $n \in \mathbb{N}_0$, $\sqsubseteq_\kappa^{n+1} \subseteq \sqsubseteq_\kappa^n$ and $\sqsubseteq_\kappa \subseteq \sqsubseteq_\kappa^n$.*

Then the relationship between $\sqsubseteq_\kappa$ and $\{\sqsubseteq_\kappa^n\}_{n \in \mathbb{N}_0}$ is clarified in the following lemma.

▶ **Lemma 10.** *If the underlying pTS $\mathcal{T}$ is locally finite, then $s \sqsubseteq_\kappa t$ iff $s \sqsubseteq_\kappa^n t$ for all $n \in \mathbb{N}_0$.*

Proof. Define $\sqsubseteq_\kappa^\omega := \bigcap_{n \in \mathbb{N}_0} \sqsubseteq_\kappa^n$. We prove that $\sqsubseteq_\kappa^\omega = \sqsubseteq_\kappa$. "$\sqsubseteq_\kappa \subseteq \sqsubseteq_\kappa^\omega$" follows directly from Lemma 9. For "$\sqsubseteq_\kappa^\omega \subseteq \sqsubseteq_\kappa$", we prove that $\sqsubseteq_\kappa^\omega$ is a κ-simulation. Fix any $(s,t) \in \sqsubseteq_\kappa^\omega$ and $a \in A$. Define $\mathcal{R} := \bigcup_{\mu \in \mathrm{der}^{\mathcal{T}}(s,a)} \lfloor \mu \rfloor \times \bigcup_{\nu \in \mathrm{der}^{\mathcal{T}}(t,a)} \lfloor \nu \rfloor$. Then $\mathcal{R}$ is finite as $\mathcal{T}$ is locally finite. Consider any $(s',t') \in \mathcal{R}$. If $(s',t') \notin \sqsubseteq_\kappa^\omega$, then there is a minimal $N(s',t') \in \mathbb{N}_0$ such that

$(s',t') \not\sqsubseteq_\kappa^{N(s',t')}$. Define $N = \max\{N(s',t') \mid (s',t') \in \mathcal{R} \setminus \sqsubseteq_\kappa^\omega\}$ (where $\max \emptyset = 0$). Together with Lemma 9, we have $\mathcal{R} \cap \sqsubseteq_\kappa^N = \mathcal{R} \cap \sqsubseteq_\kappa^\omega$. Since $s \sqsubseteq_\kappa^{N+1} t$, then for any $s \xrightarrow{a}_{\mathrm{n}} \mu$, there is $t \xrightarrow{a}_\kappa \nu$ such that $\mu \sqsubseteq_\kappa^N \nu$. Then $\mu(\sqsubseteq_\kappa^N \cap \mathcal{R})\nu$. Thus $\mu \sqsubseteq_\kappa^\omega \nu$ by $\mathcal{R} \cap \sqsubseteq_\kappa^N = \mathcal{R} \cap \sqsubseteq_\kappa^\omega$. ◀

In this paper we consider κ-similarity between two separate pTS's. This is interpreted in the standard way by taking the disjoint union of the two pTS's.

2.3 Probabilistic Pushdown Automata

In this subsection we extend the fully probabilistic pushdown automata defined by Kučera *et al* [13] to our setting.

▶ **Definition 11** (Probabilistic Pushdown Automata)**.** A *probabilistic pushdown automaton (pPDA)* is a quadruple (Q, Γ, L, Δ) where: Q is a finite set of *control states*; Γ is a finite set of *stack symbols*; L is a finite set of *labels*; $\Delta \subseteq (Q \times \Gamma) \times L \times \mathcal{D}_{\mathrm{f}}(Q \times \Gamma^*)$ is a finite set of *transition rules*. Instead of $(pX, a, \mu) \in \Delta$ (where $p \in Q, X \in \Gamma$) we write $pX \overset{a}{\rightarrowtail} \mu \in \Delta$, and omit '$\Delta$' if it is clear from the context.

Semantics of pPDA Let $P = (Q, \Gamma, L, \Delta)$ be a pPDA. For any $\mu \in \mathcal{D}(Q \times \Gamma^*)$ and $\gamma \in \Gamma^*$, the distribution $\mu_\gamma \in \mathcal{D}(Q \times \Gamma^*)$ is defined as follows: for any $p\alpha \in Q \times \Gamma^*$,

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

Then the pPDA P induces a locally finite pTS (S, A, Ω) with $S = Q \times \Gamma^*$, $A = L$ and $\Omega = \{pX\gamma \xrightarrow{a}_{\mathrm{n}} \mu_\gamma \mid pX \overset{a}{\rightarrowtail} \mu \in \Delta, \gamma \in \Gamma^*\}$. Here elements of the state set $Q \times \Gamma^*$ are called "configurations". Note that $p\beta\gamma \xrightarrow{a}_\kappa \mu$ with $\beta \in \Gamma^+$ iff $p\beta \xrightarrow{a}_\kappa \mu'$ and $\mu = \mu'_\gamma$ for some μ'.

In this paper, we study the decidability and complexity of the following decision problems: given a configuration $p\alpha$ of a pPDA P and a state f of a finite pTS $\mathcal{F}$, decide whether $p\alpha \sqsubseteq_\kappa f$ and whether $f \sqsubseteq_\kappa p\alpha$, where $\kappa \in \{\mathrm{n}, \mathrm{c}\}$. We prove that both of these problems are EXPTIME-complete.

3 Extended Stack Symbols

In this section we adopt and extend the method by Colin Stirling for PDA's, which is called "extended stack symbols" [19, 20], to our setting. Based on extended stack symbols, Stirling presented a tableaux proof system that decides the bisimilarity between two PDA's. Here we establish extended stack symbols for probabilistic simulation. Then in Section 4 we present a tableaux proof system demonstrating the EXPTIME-decidability of probabilistic simulation between pPDA and a finite pTS. Our extended stack symbols will take a different form from Stirling's.

Below we fix a pPDA $P = (Q, \Gamma, L, \Delta)$ and a finite pTS $\mathcal{F} = (F, A, \Omega)$. For any sets $\mathcal{X}, \mathcal{Y}$, we denote $\Pi[\mathcal{X}, \mathcal{Y}] := \mathcal{X} \times \mathcal{Y} \cup \mathcal{Y} \times \mathcal{X}$.

▶ **Definition 12** (Extended Stack Symbol)**.** An *extended stack symbol* U is a function from Q to 2^F. The set of all extended stack symbols is denoted by $\mathcal{E}$.

Intuitively, an extended stack symbol represents the behaviour of configurations with more symbols on the stack and maps this to corresponding states in F. We now extend pPDA P by setting its stack symbol set to $\Gamma \cup \mathcal{E}$. In the extension of P we do not modify Δ, thus a configuration $pU\alpha$ with $U \in \mathcal{E}$ has no outgoing transitions.

Let us compare Stirling's extended symbols with ours. In [19, 20], an extended stack symbol U is a function that maps a control state p to a configuration $q\alpha \in Q \times \Gamma^*$. Then the semantics is rather straightforward: the extended configuration pU behaves exactly as $q\alpha$. However in our setting, U is much more syntactical: qU can be deemed as the set of all $f \in F$ such that $f \sqsubseteq q\alpha$ (or $q\alpha \sqsubseteq f$, which depends on the context) for some $\alpha \in \Gamma^+$. For example, consider $pX\alpha$ (where $p \in Q$, $X \in \Gamma$) and $f \in F$. If $pX\alpha \sqsubseteq f$ then we expect that $pXU \sqsubseteq f$ where $U(q) := \{g \in F \mid q\alpha \sqsubseteq g\}$, for arbitrary $q \in Q$. Analoguously if $f \sqsubseteq pX\alpha$ then we expect that $f \sqsubseteq pXU$, where $U(q) := \{g \in F \mid g \sqsubseteq q\alpha\}$ for arbitrary $q \in Q$. In this way we reduce $pX\alpha$ to pXU with shorter length, which is the key to make the contruction of our tableaux trees finite. Note that to talk about "$pXU \sqsubseteq f$" or "$f \sqsubseteq pXU$" we need to extend simulation preorders to extended stack symbols, which will be captured by "extended κ-simulations". First we define extended configurations concerned in this paper.

▶ **Definition 13** (Extended Configuration)**.** Define $\mathcal{C} := Q \times (\Gamma^* \cdot (\mathcal{E} + \epsilon))$ and $\mathcal{C}_e := Q \times \mathcal{E}$.

$\mathcal{C}$ is the set of extended configurations of concern: we only consider extended configurations that contain at most one extended stack symbol at the end of the configuration. Note that $\mathcal{C} \backslash \mathcal{C}_e = Q \times (\epsilon + \Gamma^+ \cdot (\mathcal{E} + \epsilon))$ is the set of all configurations where the extended stack symbol (if it occurs) is guarded by a non-empty sequence of (normal) stack symbols.

▶ **Definition 14** (Extended κ-Simulation)**.** Define

$$\mathcal{R}_e := \{(qU, f) \in \mathcal{C}_e \times F \mid f \in U(q)\} \cup \{(f, qU) \in F \times \mathcal{C}_e \mid f \in U(q)\}$$

A relation $\mathcal{R} \subseteq \Pi[\mathcal{C}, F]$ is an *extended κ-simulation* iff for all $(s,t) \in \mathcal{R}$

- if $(s,t) \in \Pi[\mathcal{C}_e, F]$ then $(s,t) \in \mathcal{R}_e$; and
- if $(s,t) \in \Pi[\mathcal{C} \backslash \mathcal{C}_e, F]$ then $\mathrm{Act}(s) = \mathrm{Act}(t)$ and whenever $s \xrightarrow{a}_n \mu$ there is $t \xrightarrow{a}_\kappa \nu$ such that $\mu \mathcal{R} \nu$.

The *extended κ-similarity*, denoted $\preceq_\kappa$, is the union of all extended κ-simulations.

Next we extend simulation approximants to extended stack symbols.

▶ **Definition 15.** The family $\{\preceq_\kappa^n\}_{n \in \mathbb{N}_0}$ is inductively defined as follows:

- $\preceq_\kappa^0 := \{(s,t) \in \Pi[\mathcal{C} \backslash \mathcal{C}_e, F] \mid \mathrm{Act}(s) = \mathrm{Act}(t)\} \cup \mathcal{R}_e$;
- $\preceq_\kappa^{n+1} := \{(s,t) \in \Pi[\mathcal{C} \backslash \mathcal{C}_e, F] \mid \mathrm{Act}(s) = \mathrm{Act}(t)$ and $(\forall s \xrightarrow{a}_n \mu)(\exists t \xrightarrow{a}_\kappa \nu).(\mu \preceq_\kappa^n \nu)\} \cup \mathcal{R}_e$.

By similar proofs for $\sqsubseteq_\kappa$, we have the following two lemmas:

▶ **Lemma 16.** *For any $n \in \mathbb{N}_0$, $\preceq_\kappa^{n+1} \subseteq \preceq_\kappa^n$ and $\preceq_\kappa \subseteq \preceq_\kappa^n$.*

▶ **Lemma 17.** *For any $(s,t) \in \Pi[\mathcal{C}, F]$, $s \preceq_\kappa t$ iff $s \preceq_\kappa^n t$ for all $n \in \mathbb{N}_0$.*

The following theorem clarifies the relation between $\preceq_\kappa$ and $\sqsubseteq_\kappa$.

▶ **Theorem 18.** *For any $(s,t) \in \Pi[Q \times \Gamma^*, F]$, $s \preceq_\kappa t$ iff $s \sqsubseteq_\kappa t$.*

Proof. It can be shown by induction on n that $s \preceq_\kappa^n t$ iff $s \sqsubseteq_\kappa^n t$ for all $n \in \mathbb{N}_0$. Then the result follows from Lemma 10 and Lemma 17. ◀

Theorem 18 allows us to decide $\preceq_\kappa$ instead of $\sqsubseteq_\kappa$.

4 Tableaux Proof System

In this section we demonstrate the EXPTIME-decidability of κ-similarity through a tableaux proof system. Below we fix a pPDA $P = (Q, \Gamma, L, \Delta)$ and a finite pTS $\mathcal{F} = (F, A, \Omega)$. We use p, q to range over Q, X, Y to range over Γ, a to range over $A \cup L$, f, g to range over F,

U to range over $\mathcal{E}$, and α, β, γ to range over $\Gamma^* \cdot (\mathcal{E} + \epsilon)$. We extend P with extended stack symbols as described in Section 3.

Due to Theorem 18, the decision problem for κ-similarity can be reformulated and simplified as follows: Given a pair $(s,t) \in \Pi[Q \times \Gamma, F]$, decide if $s \preceq_\kappa t$. We only consider elements in $Q \times \Gamma$ since for any $pX\alpha \in Q \times \Gamma^*$, we can always add a new control state p_{init} and a new stack symbol X_{init} and augment Δ with the set $\{p_{\text{init}}X_{\text{init}} \xrightarrow{a} \mu_\alpha \mid pX \xrightarrow{a} \mu\}$, so that $p_{\text{init}}X_{\text{init}}$ mimics $pX\alpha$.

The Tableaux System We use tableaux to solve this problem along the lines of [19, 20]. A tableaux is a goal-directed proof system that consists of a set of goals Goals and a set RULE of rules which is essentially a decidable subset of $\mathsf{Goals} \times \mathcal{P}_{\text{f}}(\mathsf{Goals})$, where $\mathcal{P}_{\text{f}}(\mathsf{Goals})$ is the set of finite subsets of Goals. Graphically, a rule $(\mathsf{goal}, \{\mathsf{goal}_1, \ldots, \mathsf{goal}_n\}) \in \mathsf{RULE}$ can be viewed as a proof step:

$$\frac{\mathsf{goal}}{\mathsf{goal}_1 \ldots \mathsf{goal}_n}$$

where goal is what currently is to be proved and $\mathsf{goal}_1 \ldots \mathsf{goal}_n$ are the subgoals what it reduces to. Each rule is backward sound: in this instance if all goal_i are true then so is goal. An application of a rule $(\mathsf{goal}, \{\mathsf{goal}_1, \ldots, \mathsf{goal}_n\})$ to a goal goal is to make all the subgoals $\mathsf{goal}_1 \ldots \mathsf{goal}_n$ children of goal. Then a tableaux tree is a proof tree built from a specified goal (the root of the tree) and repeated application of rules. The leaves of a tableaux tree are divided into *terminal* and *nonterminal* leaves. Terminal leaves are divided into *successful* and *unsuccessful* leaves. A tableaux tree is *successful* iff it is finite and all its leaves are successful.

Below we formulate our tableaux proof system. Given a (ordinary) rooted tree T, we define $V(T)$ to be the set of vertices of T, and $lf(T) \subseteq V(T)$ to be the set of leaves of T.

▶ **Definition 19** (Goals). Define $\mathsf{Goals} := \Pi[\mathcal{C}, F]$. A goal $(p\alpha, f) \in \mathsf{Goals}$ (resp. $(f, p\alpha) \in \mathsf{Goals}$) is also written as $p\alpha \preceq f$ (resp. $f \preceq p\alpha$), which corresponds to the proof analogue of $p\alpha \preceq_\kappa f$ (resp. $f \preceq_\kappa p\alpha$). Below we define the notion of basic successfulness of goals:

A goal $s \preceq t$ is *basically successful* iff:	A goal $s \preceq t$ is *basically unsuccessful* iff:
1. $s \preceq t = pU \preceq f$ such that $f \in U(p)$; or	1. $s \preceq t = pU \preceq f$ such that $f \notin U(p)$; or
2. $s \preceq t = f \preceq pU$ such that $f \in U(p)$; or	2. $s \preceq t = f \preceq pU$ such that $f \notin U(p)$; or
3. $(s,t) \in \Pi[\mathcal{C}\backslash\mathcal{C}_{\text{e}}, F]$ and $\text{Act}(s) = \text{Act}(t) = \emptyset$.	3. $(s,t) \in \Pi[\mathcal{C}\backslash\mathcal{C}_{\text{e}}, F]$ and $\text{Act}(s) \neq \text{Act}(t)$.

▶ **Definition 20** (Proof Tree). A *proof tree* is a pair $(T, \mathcal{L})$ for which T is a (possibly infinite) rooted tree and $\mathcal{L} : V(T) \to \mathsf{Goals}$ is a labeling function. Denote $\mathcal{L}(T)$ be the range of $\mathcal{L}$. A leaf $v \in lf(T)$ is *successful* iff either $\mathcal{L}(v)$ is basically successful, or there is $v' \in V(T)$ such that $v' \neq v$, v' lies on the path from the root of T to v and $\mathcal{L}(v') = \mathcal{L}(v)$. A leaf $v \in lf(T)$ is *unsuccessful* iff $\mathcal{L}(v)$ is basically unsuccessful. A leaf $v \in lf(T)$ is *terminal* if either v is successful or unsuccessful; otherwise it is *non-terminal*.

Note that if v with $\mathcal{L}(v) = (s,t)$ is non-terminal, then $(s,t) \in \Pi[\mathcal{C}\backslash\mathcal{C}_{\text{e}}, F]$ and $\text{Act}(s) = \text{Act}(t) \neq \emptyset$. Below we define rules in our tableaux system. There are two kinds of rules in our tableaux system: UNF (Unfolding) and RED (Reduction).

▶ **Definition 21** (Rules). $\text{UNF}_\kappa \subseteq \mathsf{Goals} \times \mathcal{P}_{\text{f}}(\mathsf{Goals})$ is defined by: $(s \preceq t, \mathcal{G}) \in \text{UNF}_\kappa$ iff

- $s \preceq t \in \Pi[Q \times (\Gamma \cdot (\mathcal{E} + \epsilon)), F]$ and $\text{Act}(s) = \text{Act}(t) \neq \emptyset$;
- $\mathcal{G} \subseteq \text{Der}(s) \times \text{Der}(t)$ and for any $s \xrightarrow{a}_{\text{n}} \mu$, there is $t \xrightarrow{a}_\kappa \nu$ such that $\mu \mathcal{G} \nu$.

$\text{RED}_\kappa \subseteq \mathsf{Goals} \times \mathcal{P}_{\text{f}}(\mathsf{Goals})$ is defined as $\text{RED}^{\text{a}}_\kappa \cup \text{RED}^{\text{b}}_\kappa$, where

- $(s \preceq t, \mathcal{G}) \in \mathrm{RED}^{\mathrm{a}}_{\kappa}$ iff $s \preceq t = pX\alpha \preceq f$ such that
  - $\mathrm{Act}(pX) = \mathrm{Act}(f) \neq \emptyset$ and $\alpha \in \Gamma^{+} \cdot (\mathcal{E} + \epsilon)$; and
  - $\mathcal{G} = \{pXU \preceq f\} \cup \{q\alpha \preceq g \mid q \in Q, g \in U(q)\}$ for some $U \in \mathcal{E}$.
- $(s \preceq t, \mathcal{G}) \in \mathrm{RED}^{\mathrm{b}}_{\kappa}$ iff $s \preceq t = f \preceq pX\alpha$ such that
  - $\mathrm{Act}(f) = \mathrm{Act}(pX) \neq \emptyset$ and $\alpha \in \Gamma^{+} \cdot (\mathcal{E} + \epsilon)$; and
  - $\mathcal{G} = \{f \preceq pXU\} \cup \{g \preceq q\alpha \mid q \in Q, g \in U(q)\}$ for some $U \in \mathcal{E}$.

We denote $\mathrm{RULE}_{\kappa} := \mathrm{UNF}_{\kappa} \cup \mathrm{RED}_{\kappa}$

Intuitively, a rule of UNF_{κ} expands a goal $s \preceq t$ one-step further (cf. Lemma 25) and a rule of RED_{κ} reduce a goal $pX\alpha \preceq f$ (resp. $f \preceq pX\alpha$) to $pXU \preceq f$ (resp. $f \preceq pXU$) together with all information at α encoded in U. Here we use 'a' to indicate the case $s \preceq t \in \mathcal{C} \times F$ and 'b' to indicate the case $s \preceq t \in F \times \mathcal{C}$; we will continue using this in the sequel.

The following definition illustrates the application of rules.

▶ **Definition 22** (Rule Application)**.** Suppose $B = (T, \mathcal{L})$ be a proof tree, $v \in lf(T)$ and $(\mathsf{gl}, \mathcal{G}) \in \mathrm{RULE}_{\kappa}$ such that v is non-terminal and $\mathcal{L}(v) = \mathsf{gl}$. The proof tree $(T', \mathcal{L}')$ *after the application* of $(\mathsf{gl}, \mathcal{G})$ at v, denoted $B_v[\mathsf{gl}, \mathcal{G}]$, is defined as follows: $T' = T \cup \{(v, v') | v' \in V^{\mathrm{new}}\}$ and $\mathcal{L}' = \mathcal{L} \cup \mathcal{L}^{\mathrm{new}}$, where $V^{\mathrm{new}} \cap V(T) = \emptyset$ and $\mathcal{L}^{\mathrm{new}}$ a bijection from V^{new} to $\mathcal{G}$.

Then the set of tableaux trees which is closed under rule application is defined as follows.

▶ **Definition 23** (Tableaux Trees)**.** Let $s \preceq t \in \Pi[\mathcal{C}, F]$. The set of *tableaux trees* rooted at $s \preceq t$, denoted $\mathrm{Tab}[s \preceq t]$, is the smallest set satisfying the following conditions:

- The proof tree with only a single root labeled with $s \preceq t$ belongs to $\mathrm{Tab}[s \preceq t]$.
- If $B = (T, \mathcal{L}) \in \mathrm{Tab}[s \preceq t]$, then $B_v[\mathsf{gl}, \mathcal{G}] \in \mathrm{Tab}[s \preceq t]$ for all $v \in lf(T)$, $(\mathsf{gl}, \mathcal{G}) \in \mathrm{RULE}_{\kappa}$ such that v is non-terminal and $\mathcal{L}(v) = \mathsf{gl}$.
- If there is an infinite sequence $\{(T_n, \mathcal{L}_n)\}_{n \in \mathbb{N}_0}$ with each $(T_n, \mathcal{L}_n) \in \mathrm{Tab}[s \preceq t]$ such that $T_n \subseteq T_{n+1}$ and $\mathcal{L}_n \subseteq \mathcal{L}_{n+1}$ for all n, then $(\bigcup_n T_n, \bigcup_n \mathcal{L}_n) \in \mathrm{Tab}[s \preceq t]$.

The tableaux trees have the following finiteness property which can be proved inductively from Definition 23.

▶ **Lemma 24.** *For each $\alpha \in \Gamma^{*}$, we define* $\mathsf{Suffix}(\alpha) := \{\alpha' \mid \alpha'$ *is a suffix of* $\alpha\}$ *(here ϵ is a suffix of any $\alpha \in \Gamma^{*}$). Let* $\mathsf{Suffix} := \bigcup\{\mathsf{Suffix}(\alpha) \mid \exists q \in Q \exists pX \overset{a}{\rightarrowtail} \mu \in \Delta.(q\alpha \in \lfloor \mu \rfloor)\}$*. Define*

- $\mathcal{G}^{\mathrm{a}}_{*} = \{p\beta\alpha \preceq f \mid \beta \in \Gamma \cup \mathsf{Suffix}, \alpha \in \mathcal{E} \cup \{\epsilon\}, p \in Q, f \in F\}$
- $\mathcal{G}^{\mathrm{b}}_{*} = \{f \preceq p\beta\alpha \mid \beta \in \Gamma \cup \mathsf{Suffix}, \alpha \in \mathcal{E} \cup \{\epsilon\}, p \in Q, f \in F\}$

Then if $s \preceq t \in \mathcal{G}^{\mathrm{a}}_{*}$, then for any $(T, \mathcal{L}) \in \mathrm{Tab}[s \preceq t]$, $\mathcal{L}(T) \subseteq \mathcal{G}^{\mathrm{a}}_{*}$. Analoguously if $s \preceq t \in \mathcal{G}^{\mathrm{b}}_{*}$, then for any $(T, \mathcal{L}) \in \mathrm{Tab}[s \preceq t]$, $\mathcal{L}(T) \subseteq \mathcal{G}^{\mathrm{b}}_{*}$.

Below we prove that rules of UNF_{κ} and RED_{κ} are backward sound.

▶ **Lemma 25.** *Let $(s \preceq t, \mathcal{G}) \in \mathrm{UNF}_{\kappa}$. If $s' \preceq^{n}_{\kappa} t'$ for all $s' \preceq t' \in \mathcal{G}$, then $s \preceq^{n+1}_{\kappa} t$.*

Proof. Directly from Definition 15 and Definition 21. ◀

▶ **Lemma 26.** *Let $(pX\alpha \preceq f, \{pXU \preceq f\} \cup \mathcal{G}^{\mathrm{a}}_{\alpha,U}) \in \mathrm{RED}^{\mathrm{a}}_{\kappa}$ where*

$$\mathcal{G}^{\mathrm{a}}_{\alpha,U} := \{q\alpha \preceq g \mid q \in Q, g \in U(q)\}.$$

If $pXU \preceq^{n+1}_{\kappa} f$ and $q\alpha \preceq^{n}_{\kappa} g$ for all $q\alpha \preceq g \in \mathcal{G}^{\mathrm{a}}_{\alpha,U}$, then $pX\alpha \preceq^{n+1}_{\kappa} f$.

Proof. We prove by induction on n that for all $p\gamma\alpha \preceq f \in \mathsf{Goals}$ with $\gamma \in \Gamma^+$ it holds that for any $U \in \mathcal{E}$, if $p\gamma U \preceq_\kappa^{n+1} f$ and $q\alpha \preceq_\kappa^n g$ for all $q\alpha \preceq g \in \mathcal{G}^{\mathrm{a}}_{\alpha,U}$, then $p\gamma\alpha \preceq_\kappa^{n+1} f$.

Base Step: $n = 0$. Since $p\gamma U \preceq_\kappa^1 f$, for any $p\gamma \xrightarrow{a}_{\mathrm{n}} \mu$, there is $f \xrightarrow{a}_\kappa \nu$ such that $\mu_U \preceq_\kappa^0 \nu$. Let $w : \lfloor \mu_U \rfloor \times \lfloor \nu \rfloor \to [0,1]$ be a weight function for $\mu_U \preceq_\kappa^0 \nu$ (cf. Definition 6). We define a weight function $w' : \lfloor \mu_\alpha \rfloor \times \lfloor \nu \rfloor \to [0,1]$ by: $w'(q\beta\alpha, g) = w(q\beta U, g)$ for all $q\beta \in \lfloor \mu \rfloor$ (note that $\beta \in \Gamma^*$) and $g \in \lfloor \nu \rfloor$. We prove that w' is a weight function for $\mu_\alpha \preceq_\kappa^0 \nu$. The first two conditions in Definition 6 are straightforward to verify. For the third condition, suppose $w'(q\beta\alpha, g) > 0$ with $q\beta \in \lfloor \mu \rfloor$. Then $w(q\beta U, g) > 0$ and hence $q\beta U \preceq_\kappa^0 g$. If $\beta \neq \epsilon$ then by Definition 15 $\mathrm{Act}(q\beta) = \mathrm{Act}(g)$ and we have $q\beta\alpha \preceq_\kappa^0 g$. If $\beta = \epsilon$ then $g \in U(q)$ and we have $q\alpha \preceq g \in \mathcal{G}^{\mathrm{a}}_{\alpha,U}$; thus $q\alpha \preceq_\kappa^0 g$. In either case $q\beta\alpha \preceq_\kappa^0 g$. So w' is a weight function for $\mu_\alpha \preceq_\kappa^0 \nu$. Also from $p\gamma U \preceq_\kappa^1 f$ we have $\mathrm{Act}(p\gamma\alpha) = \mathrm{Act}(f)$. So $p\gamma\alpha \preceq_\kappa^1 f$.

Inductive Step: Suppose $p\gamma U \preceq_\kappa^{n+2} f$ and $q\alpha \preceq_\kappa^{n+1} g$ for all $q\alpha \preceq g \in \mathcal{G}^{\mathrm{a}}_{\alpha,U}$. We prove that $p\gamma\alpha \preceq_\kappa^{n+2} f$. Since $p\gamma U \preceq_\kappa^{n+2} f$, for any $p\gamma \xrightarrow{a}_{\mathrm{n}} \mu$, there is $f \xrightarrow{a}_\kappa \nu$ such that $\mu_U \preceq_\kappa^{n+1} \nu$. Consider any $(q\beta U, g) \in \preceq_\kappa^{n+1}$ with $\beta \in \Gamma^*$: if $\beta = \epsilon$ then $q\beta\alpha \preceq_\kappa^{n+1} g$ since $q\alpha \preceq g \in \mathcal{G}^{\mathrm{a}}_{\alpha,U}$; if $\beta \in \Gamma^+$ then we have $q\beta\alpha \preceq_\kappa^{n+1} g$ by induction hypothesis. So by the same construction of weight function in the base step we have $\mu_\alpha \preceq_\kappa^{n+1} \nu$. Also we have $\mathrm{Act}(p\gamma\alpha) = \mathrm{Act}(f)$. Thus $p\gamma\alpha \preceq_\kappa^{n+2} f$. ◀

Similarly we can prove the following analogue to Lemma 26.

▶ **Lemma 27.** *Let* $(f \preceq pX\alpha, \{f \preceq pXU\} \cup \mathcal{G}^{\mathrm{b}}_{\alpha,U}) \in \mathrm{RED}^{\mathrm{b}}_\kappa$ *where*

$$\mathcal{G}^{\mathrm{b}}_{\alpha,U} := \{g \preceq q\alpha \mid q \in Q, g \in U(q)\}.$$

If $f \preceq_\kappa^{n+1} pXU$ *and* $g \preceq_\kappa^n q\alpha$ *for all* $g \preceq q\alpha \in \mathcal{G}^{\mathrm{b}}_{\alpha,U}$, *then* $f \preceq_\kappa^{n+1} pX\alpha$.

Based on Lemma 25 through Lemma 27, we obtain the soundness of our tableaux system:

▶ **Proposition 28.** *If there is a successful tableaux tree rooted at* $s \preceq t$, *then* $s \preceq_\kappa t$.

Proof. We only prove the case when $s \preceq t \in \mathcal{C} \times F$, the other case is similar. The proof is by contraposition. Let $p_0\alpha_0 \preceq f_0 = s \preceq t$. Suppose $(T, \mathcal{L})$ is a successful tableaux tree rooted at $p_0\alpha_0 \preceq f_0$ however $p_0\alpha_0 \not\preceq_\kappa f_0$. Then by Lemma 17, there is $n_0 \in \mathbb{N}_0$ such that $p_0\alpha_0 \preceq_\kappa^{n_0} f_0$ however $p_0\alpha_0 \not\preceq_\kappa^{n_0+1} f_0$. Note that we have $(p_0\alpha_0, f_0) \in \preceq_\kappa^0$ or otherwise the goal $p_0\alpha_0 \preceq f_0$ would be unsuccessful. By the backward soundness of UNF_κ and RED_κ, if the rule applied to $p_0\alpha_0 \preceq f_0$ belongs to UNF_κ, then there is a child $p_1\alpha_1 \preceq f_1$ of $p_0\alpha_0 \preceq f_0$ such that $p_1\alpha_1 \not\preceq_\kappa^{n_0} f_1$; and if the rule applied to $p_0\alpha_0 \preceq f_0$ belongs to $\mathrm{RED}^{\mathrm{a}}_\kappa$, then either $pXU \not\preceq_\kappa^{n_0+1} f$ or $p'\alpha \not\preceq_\kappa^{n_0} f'$ for some $p'\alpha \preceq f' \in \mathcal{G}^{\mathrm{a}}_{\alpha,U}$ with corresponding X, U and α. In either case there is a child $p_1\alpha_1 \preceq f_1$ of $p_0\alpha_0 \preceq f_0$ such that $p_1\alpha_1 \not\preceq_\kappa^{n_0+1} f_1$. Let $n_1 \in \mathbb{N}_0$ such that $p_1\alpha_1 \not\preceq_\kappa^{n_1+1} f_1$ and $p_1\alpha_1 \preceq_\kappa^{n_1} f_1$. Then $n_1 \leq n_0$. In this way we can recursively construct a finite sequence $\{(p_i\alpha_i \preceq f_i, n_i)\}_{1 \leq i \leq k}$ such that $p_i\alpha_i \not\preceq_\kappa^{n_i+1} f_i$ and $p_i\alpha_i \preceq_\kappa^{n_i} f_i$, and the last goal $p_k\alpha_k \preceq f_k$ is a successful leaf. Since $p_k\alpha_k \not\preceq_\kappa^{n_k+1} f_k$, $p_k\alpha_k \preceq_\kappa f_k$ cannot be basically successful. So the only possibility is that there is $j < k$ such that $p_k\alpha_k \preceq f_k = p_j\alpha_j \preceq f_j$. Consider the step from $p_{k-1}\alpha_{k-1} \preceq f_{k-1}$ to $p_k\alpha_k \preceq f_k$:

- if the step is due to UNF_κ, then $n_k < n_{k-1} \leq n_j$;
- if the step is due to $\mathrm{RED}^{\mathrm{a}}_\kappa$ and $p_k\alpha_k \preceq f_k \in \mathcal{G}^{\mathrm{a}}_{\alpha,U}$ with corresponding α and U, then $n_k < n_{k-1} \leq n_j$;
- if the step is due to $\mathrm{RED}^{\mathrm{a}}_\kappa$ and $p_k\alpha_k \preceq f_k = pXU \preceq f$ with corresponding X and U, then $p_{k-1}\alpha_{k-1} \preceq f_{k-1} \neq pXU \preceq f$ and so $j < k-1$. Then from j to $j+1$ the rule is a UNF_κ which implies $n_{j+1} < n_j$, hence $n_k < n_j$.

Thus in either case $n_k < n_j$. But then we have $p_k\alpha_k \not\preceq_\kappa^{n_k+1} f_k$ and $p_k\alpha_k \preceq_\kappa^{n_j} f_k$. Contradiction. ◀

To show the completeness of the tableaux, we first prove a useful lemma below.

▶ **Lemma 29.** *Suppose $pX\alpha \preceq_\kappa f$ and U be an extended symbol such that $U(q) := \{g \in F \mid q\alpha \preceq_\kappa g\}$ for all $q \in Q$. Then $pXU \preceq_\kappa f$.*

Proof. We prove that $\mathcal{R} := \{(q\beta U, g) \mid q \in Q, \beta \in \Gamma^*, q\beta\alpha \preceq_\kappa g\}$ is an extended κ-simulation. Consider any $(q\beta U, g) \in \mathcal{R}$. If $\beta = \epsilon$, then $q\alpha \preceq_\kappa g$ and thus $g \in U(q)$; then $(qU, g) \in \mathcal{R}_{\mathrm{e}}$. On the other hand, suppose that $\beta \in \Gamma^+$. Then $\mathrm{Act}(q\beta U) = \mathrm{Act}(q\beta\alpha) = \mathrm{Act}(g)$. Further for any $q\beta \xrightarrow{a}_{\mathrm{n}} \mu$, by $q\beta\alpha \preceq_\kappa g$ there is $g \xrightarrow{a}_\kappa \nu$ such that $\mu_\alpha \preceq_\kappa \nu$. We prove that $\mu_U \mathcal{R} \nu$. By $\mu_\alpha \preceq_\kappa \nu$, there is a weight function $w : \lfloor \mu_\alpha \rfloor \times \lfloor \nu \rfloor \to [0,1]$ for μ_α and ν. We construct a function $w' : \lfloor \mu_U \rfloor \times \lfloor \nu \rfloor \to [0,1]$ by: $w'(q'\gamma U, g') = w(q'\gamma\alpha, g')$ for all $q'\gamma \in \lfloor \mu \rfloor$ and $g' \in \lfloor \nu \rfloor$ (note that $\gamma \in \Gamma^*$). Then we show that w' is a weight function for μ_U and ν. The first two conditions in Definition 6 are straightforward to verify. For the third condition, consider any $q'\gamma \in \lfloor \mu \rfloor$ and $g' \in \lfloor \nu \rfloor$: if $w'(q'\gamma U, g') > 0$, then $w(q'\gamma\alpha, g') > 0$ and hence $q'\gamma\alpha \preceq_\kappa g'$; then by definition we obtain that $(q'\gamma U, g') \in \mathcal{R}$. Thus $\mathcal{R}$ is an extended κ-simulation. ◀

Similarly, we can obtain the following lemma:

▶ **Lemma 30.** *Suppose $f \preceq_\kappa pX\alpha$ and U be an extended symbol such that $U(q) := \{g \in F \mid g \preceq_\kappa q\alpha\}$ for all $q \in Q$. Then $f \preceq_\kappa pXU$.*

Then the completeness of the tableaux system is as follows:

▶ **Proposition 31.** *Let $s \preceq t \in \mathcal{G}_*^{\mathrm{a}} \cup \mathcal{G}_*^{\mathrm{b}}$. If $s \preceq_\kappa t$ then there is a successful tableaux tree rooted at $s \preceq t$.*

Proof. We only prove the case when $s \preceq t \in \mathcal{G}_*^{\mathrm{a}}$, the other case is similar. Define

$$\mathrm{Tab}'[s \preceq t] := \{(T, \mathcal{L}) \in \mathrm{Tab}[s \preceq t] \mid T \text{ is finite and } s' \preceq_\kappa t' \text{ for all } s' \preceq t' \in \mathcal{L}(T)\}$$

Below we recursively construct a sequence $\{B^n\}_n$ with each $B^n \in \mathrm{Tab}'[s \preceq t]$.

Initially B^0 is the tableaux tree which contains only a fixed root labeled with $s \preceq t$. Then suppose $B^n \in \mathrm{Tab}'[s \preceq t]$ is constructed. If all leaves $s' \preceq t'$ of B^n are terminal (and successful since $s' \preceq_\kappa t'$), then the construction is ended. Otherwise, we fix arbitrarily a non-terminal leaf v of B^n and the construction of B^{n+1} is divided into two cases below:

1. $\mathcal{L}(v) = pX\alpha \preceq f$ with $\alpha \in \mathcal{E} \cup \{\epsilon\}$. Then $B^{n+1} = \mathcal{B}_v^n[\mathcal{L}(v), \mathcal{G}]$ with
$$\mathcal{G} = \{(q\beta, g) \in \mathrm{Der}(pX\alpha) \times \mathrm{Der}(f) \mid q\beta \preceq_\kappa g\}.$$
2. $\mathcal{L}(v) = pX\alpha \preceq f$ with $\alpha \in \Gamma^+ \cdot (\mathcal{E} + \epsilon)$. Then $B^{n+1} = B_v^n[\mathcal{L}(v), \mathcal{G}]$ with
$$\mathcal{G} = \{pXU \preceq f\} \cup \mathcal{G}_{\alpha,U}^{\mathrm{a}}$$
where U is defined by: $U(q) = \{g \in F \mid q\alpha \preceq_\kappa g\}$ for all $q \in Q$. Following Lemma 29, we obtain $pXU \preceq_\kappa f$.

Then in either case $B^{n+1} \in \mathrm{Tab}'[s \preceq t]$. The construction of $\{B^n\}_n$ ends in finitely many steps. This is shown by contraposition. Suppose this is not the case. Denote $B^n = (T_n, \mathcal{L}_n)$ and $B = (\bigcup_n T_n, \bigcup_n \mathcal{L}_n)$. Then B is an infinite finitely-branching proof tree. Thus by König's Lemma there is an infinite path in B. However, by Lemma 24 on such infinite path there must be v, v' with $v \neq v'$ such that $\mathcal{L}(v) = \mathcal{L}(v')$. By Definition 20, either v or v' is successful, thus the construction should end at v or v'. Contradiction. Then the tableaux tree B^l which is the last element of $\{B^n\}_n$ is a successful tableaux tree. ◀

Below we illustrate our main result, where we use a refinement technique to achieve the EXPTIME-upperbound. We define $|P|$ (resp. $|\mathcal{F}|$) to be the integrated size of Q, Γ, L, Δ (resp. F, A, Ω).

▶ **Theorem 32.** *The problem whether $s \preceq_\kappa t$ for a given $(s,t) \in \Pi[Q \times \Gamma, F]$ is decidable in $\mathcal{O}(H(|P|, |\mathcal{F}|) \cdot 8^{|F||Q|})$ time where H is a fixed multivariate polynomial. Thus, if $|Q|$ and $|F|$ are fixed, then the problem can be decided in PTIME.*

Proof. We assume that $s \preceq t \in (Q \times \Gamma) \times F$ and $\kappa = \mathrm{c}$, the other cases are similar. We present a refinement algorithm to decide if $s \preceq_\mathrm{c} t$. Formally, we construct a finite decreasing sequence of sets of goals $\{\mathcal{G}_n\}_n$ where the last element $\mathcal{G}_m$ is expected to contain all the correct goals in $\mathcal{G}^\mathrm{a}_*$. The construction is as follows: Initially $\mathcal{G}_0 = \mathcal{G}^\mathrm{a}_*$. Then $\mathcal{G}_{n+1} \subseteq \mathcal{G}^\mathrm{a}_*$ is constructed from $\mathcal{G}_n$ as follows: $s \preceq t \in \mathcal{G}_{n+1}$ iff either $s \preceq t$ is basically successful, or $s \preceq t \in \mathcal{G}_n$ and there is $(s \preceq t, \mathcal{G}) \in \mathrm{RULE}_\mathrm{c}$ such that $\mathcal{G} \subseteq \mathcal{G}_n$. Here note that $|\mathcal{G}^\mathrm{a}_*| = \mathcal{O}(|P|^3 |F| 2^{|F||Q|})$.

The computation from $\mathcal{G}_n$ to $\mathcal{G}_{n+1}$ can be done in $\mathcal{O}(H'(|P|, |\mathcal{F}|) \cdot 4^{|F||Q|})$ time where H' is a fixed multivariate polynomial. Given $s \preceq t \in \mathcal{G}_n$, we can check whether $s \preceq t \in \mathcal{G}_{n+1}$ as follows: If $s \preceq t = pX\alpha \preceq f$ with $\alpha \in \Gamma^+ \cdot (\mathcal{E} + \epsilon)$, we check if $\{pXU \preceq f\} \cup \mathcal{G}^\mathrm{a}_{\alpha,U} \subseteq \mathcal{G}_n$ for some $U \in \mathcal{E}$. If $s \preceq t = pX\alpha \preceq f$ with $\alpha \in \mathcal{E} \cup \{\epsilon\}$, we check if for any $pX\alpha \xrightarrow{a}_\mathrm{n} \mu$, there is $f \xrightarrow{a}_\mathrm{c} \nu$ such that $\mu \mathcal{G}_n \nu$; this can be checked by checking if the following linear inequality system (with variables $\{x_\nu\}_{\nu \in \mathrm{der}(f,a)}$ and $\{y_{(s',t')}\}_{(s',t') \in \lfloor \mu \rfloor \times F}$) has a solution:

- $\sum_{\nu \in \mathrm{der}(f,a)} x_\nu = 1$, and $x_\nu \geq 0$ for all $\nu \in \mathrm{der}(f,a)$.
- $\sum_{t' \in F} y_{(s',t')} = \mu(s')$ for all $s' \in \lfloor \mu \rfloor$.
- $\sum_{s' \in \lfloor \mu \rfloor} y_{(s',t')} = \sum_{\nu \in \mathrm{der}(f,a)} x_\nu \cdot \nu(t')$ for all $t' \in F$.
- $y_{(s',t')} \geq 0$ for all $(s',t') \in \lfloor \mu \rfloor \times F$, and $y_{(s',t')} = 0$ for all $(s',t') \notin \mathcal{G}_n$.

This can be solved in polynomial time in $|P|$ and $|\mathcal{F}|$ [17].

Since $\mathcal{G}_{n+1} \subseteq \mathcal{G}_n$ there is $m \leq |\mathcal{G}^\mathrm{a}_*|$ such that $\mathcal{G}_{m+1} = \mathcal{G}_m$. We show that for any $s \preceq t \in \mathcal{G}^\mathrm{a}_*$, $s \preceq_\mathrm{c} t$ iff $s \preceq t \in \mathcal{G}_m$. Suppose $s \preceq_\mathrm{c} t$. Let $(T, \mathcal{L}) \in \mathrm{Tab}'[s \preceq t]$ be the tableaux tree constructed in the proof of Proposition 31. It follows by induction on n that $\mathcal{L}(T) \subseteq \mathcal{G}_n$ for all $n \in \mathbb{N}_0$. Thus $s \preceq t \in \mathcal{G}_m$. Suppose now that $s \preceq t \in \mathcal{G}_m$. Since for any $s' \preceq t' \in \mathcal{G}_m$ which is not basically successful, there is $(s' \preceq t', \mathcal{G}) \in \mathrm{RULE}_\mathrm{c}$ such that $\mathcal{G} \subseteq \mathcal{G}_m$. Thus we can iteratively apply rules to the root $s \preceq t$ and form a successful tableaux tree similar to the construction in the proof of Proposition 31. Thus, $s \preceq_\mathrm{c} t$ by Proposition 28. ◀

▶ Remark. The major difference between our tableaux proof system and Colin Stirling's [19, 20] is at the RED (Reduction) rules: here we need to tackle extended stack symbols in our setting, which is different from Stirling's version. Another difference is that we are able to derive a primitive upperbound by a refinement technique, which is not feasible in [19, 20].

5 EXPTIME-Hardness

In this section we show that deciding $\sqsubseteq_\kappa$ is EXPTIME-hard, whenever $\kappa = \mathrm{n}$ or $\kappa = \mathrm{c}$. We prove this by providing a rather straightforward reduction from the non-probabilistic EXTPIME-hardness result obtained in [14]. Our main efforts lie in the treatment of the additional "$\mathrm{Act}(s) = \mathrm{Act}(t)$" condition in Definition 7 which is not involved in the definition of non-probabilistic simulation preorder. First we define a variation of $\sqsubseteq_\kappa$.

▶ **Definition 33.** Let $\mathcal{T} = (S, A, \Omega)$ be a pTS. Define $\preccurlyeq_\kappa$ to be the union of all binary relations $\mathcal{R} \subseteq S \times S$ such that for any $(s,t) \in \mathcal{R}$, whenever $s \xrightarrow{a}_\mathrm{n} \mu$ there is $t \xrightarrow{a}_\kappa \nu$ with $\mu \mathcal{R} \nu$.

In other words, $\preccurlyeq_\kappa$ is defined in a similar way of $\preceq_\kappa$, however without the "$\mathrm{Act}(s) = \mathrm{Act}(t)$" requirement. Then we embed non-probabilistic transition systems into pTS's.

▶ **Definition 34.** A distribution μ is *Dirac* if $||\lfloor\mu\rfloor|| = 1$. The Dirac distribution μ with $\lfloor\mu\rfloor = \{s\}$ is also written as $\delta[s]$. A pTS (S, A, Ω) is *Dirac* if μ is dirac for any $(s, a, \mu) \in \Omega$. A pPDA (Q, Γ, L, Δ) is *Dirac* if μ is Dirac for any $pX \xrightarrow{a} \mu \in \Delta$.

Note that a Dirac pPDA induces a Dirac pTS. Dirac pTS's correspond to labeled transition systems without probability. It is easy to see that $\preccurlyeq_\mathrm{n}$ is the non-probabilistic simulation preorder over labeled transition systems. By [14], deciding $\preccurlyeq_\mathrm{n}$ is EXPTIME-hard between Dirac pPDA and finite Dirac pTS in both direction. Below we reduce $\preccurlyeq_\kappa$ to $\sqsubseteq_\kappa$ under Dirac pTS's. The following proposition allows us to focus solely on the case $\kappa = \mathrm{n}$.

▶ **Proposition 35.** *If the underlying pTS $\mathcal{T} = (S, A, \Omega)$ is Dirac, then $\preccurlyeq_\mathrm{n}$=$\preccurlyeq_\mathrm{c}$ and $\sqsubseteq_\mathrm{n}$=$\sqsubseteq_\mathrm{c}$.*

Now we reduce $\preccurlyeq_\mathrm{n}$ between a Dirac pPDA $P = (Q, \Gamma, L, \Delta)$ and a Dirac finite pTS $\mathcal{F} = (F, A, \Omega)$, to $\sqsubseteq_\mathrm{n}$ between a Dirac pPDA $(Q', \Gamma', L, \Delta')$ and a Dirac finite pTS (F', A, Ω'). The reduction is as follows:

1. $Q' = Q \cup \{p_\perp\}$ and $F' = F \cup \{f_\perp\}$ where $p_\perp \notin Q$ and $f_\perp \notin F$.
2. $\Gamma' = \Gamma \cup \{Z_\perp\}$ where $Z_\perp \notin \Gamma$ is a new bottom stack symbol.
3. $\Delta' = \Delta \cup \{(pX, a, \delta[p_\perp]) \mid p \in Q, X \in \Gamma', a \in L \cup A\}$
4. $\Omega' = \Omega \cup \{(f, a, \delta[f_\perp]) \mid f \in F, a \in L \cup A\}$.

It is not hard to prove that for all $p\alpha \in Q \times \Gamma^*$ and $f \in F$, $p\alpha \preccurlyeq_\mathrm{n} f$ (resp. $f \preccurlyeq_\mathrm{n} p\alpha$) iff $p\alpha Z_\perp \sqsubseteq_\mathrm{n} f$ (resp. $f \sqsubseteq_\mathrm{n} p\alpha Z_\perp$). Thus deciding $\sqsubseteq_\mathrm{n}$ between Dirac pPDA's and Dirac finite pTS's is EXPTIME-hard. Then:

▶ **Theorem 36.** *Deciding $\sqsubseteq_\mathrm{n}$ and $\sqsubseteq_\mathrm{c}$ between probabilistic pushdown automata and finite probabilistic transition systems in both directions is EXPTIME-complete.*

6 Conclusion

We have shown that deciding probabilistic simulation preorder between a probabilistic pushdown automata (Q, Γ, L, Δ) and a finite probabilistic transition system (F, A, Ω) is EXPTIME-complete. This result holds for both directions. Further if $|Q|$ and $|F|$ are fixed, then the problem is decidable in polynomial time. These results extend their non-probabilistic counterparts in [14]. We obtain these results by extending Colin Stirling's method [19, 20] which is originally used to demonstrate the decidability of bisimulation over pushdown automata. Our extension is nontrivial and has a different form from the original one. A future direction is to explore if this method can be extended to weak semantical equivalences such as weak probabilistic bisimulation or weak probabilistic simulation [3, 18].

Acknowledgement

Thanks to anonymous referees for valuable comments. The first author is supported by a CSC scholarship. The second author is supported by the EU FP7 project MoVeS.

References

1 Christel Baier, Bettina Engelen, and Mila E. Majster-Cederbaum. Deciding bisimilarity and similarity for probabilistic processes. *J. Comput. Syst. Sci.*, 60(1):187–231, 2000.

2 Christel Baier, Holger Hermanns, and Joost-Pieter Katoen. Probabilistic weak simulation is decidable in polynomial time. *Inf. Process. Lett.*, 89(3):123–130, 2004.

3 Christel Baier, Joost-Pieter Katoen, Holger Hermanns, and Verena Wolf. Comparative branching-time semantics for Markov chains. *Inf. Comput.*, 200(2):149–214, 2005.

4 Tomás Brázdil, Antonín Kučera, and Oldrich Strazovský. On the decidability of temporal properties of probabilistic pushdown automata. In *STACS*, pages 145–157, 2005.

5 Tomás Brázdil, Antonín Kučera, and Oldrich Strazovský. Deciding probabilistic bisimilarity over infinite-state probabilistic systems. *Acta Inf.*, 45(2):131–154, 2008.

6 Stefano Cattani and Roberto Segala. Decision algorithms for probabilistic bisimulation. In *CONCUR*, pages 371–385, 2002.

7 Javier Esparza, Antonín Kučera, and Richard Mayr. Model checking probabilistic pushdown automata. In *LICS*, pages 12–21, 2004.

8 Kousha Etessami and Mihalis Yannakakis. Algorithmic verification of recursive probabilistic state machines. In *TACAS*, pages 253–270, 2005.

9 Kousha Etessami and Mihalis Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1), 2009.

10 Jan Friso Groote and Hans Hüttel. Undecidable equivalences for basic process algebra. *Inf. Comput.*, 115(2):354–371, 1994.

11 Bengt Jonsson, Kim G. Larsen, and Wang Yi. Probabilistic extensions of process algebras. In *Handbook of Process Algebra*, pages 685–710. Elsevier, 2001.

12 Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *LICS*, pages 266–277, 1991.

13 Antonín Kučera, Javier Esparza, and Richard Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1), 2006.

14 Antonín Kučera and Richard Mayr. On the complexity of checking semantic equivalences between pushdown processes and finite-state processes. *Inf. Comput.*, 208(7):772–796, 2010.

15 Marta Z. Kwiatkowska. Model checking for probability and time: from theory to practice. In *LICS*, pages 351–360, 2003.

16 Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.

17 Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

18 Roberto Segala and Nancy A. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR*, pages 481–496, 1994.

19 Colin Stirling. Decidability of bisimulation equivalence for normed pushdown processes. *Theor. Comput. Sci.*, 195(2):113–131, 1998.

20 Colin Stirling. Decidability of bisimulation equivalence for pushdown processes. *Unpublished manuscript, available at http://homepages.inf.ed.ac.uk/cps/*, 2000.