Probabilistic Verification Beyond Context-Freeness [*]

Guanyan Li
University of Oxford
United Kingdom

Andrzej S. Murawski
University of Oxford
United Kingdom

C.-H. Luke Ong
University of Oxford
United Kingdom

ABSTRACT

Probabilistic pushdown automata (recursive state machines) are a widely known model of probabilistic computation associated with many decidable problems concerning termination (time) and linear-time model checking. Higher-order recursion schemes (HORS) are a prominent formalism for the analysis of higher-order computation.

Recent studies showed that, for the probabilistic variant of HORS, even the basic problem of determining whether a scheme terminates almost surely is undecidable. Moreover, the undecidability already holds for order-2 schemes (order-1 schemes are known to correspond to pushdown automata).

Motivated by these results, we study *restricted probabilistic tree-stack automata* (rPTSA), which in the nondeterministic setting are known to characterise a proper extension of context-free languages, namely, the multiple context-free languages. We show that several verification problems, such as almost-sure termination, positive almost-sure termination and ω-regular model checking are decidable for this class.

At the level of higher-order recursion schemes, this corresponds to being able to verify a probabilistic version of MAHORS (which are a multiplicative-additive version of higher-order recursion schemes). MAHORS extend order-1 recursion schemes and are incomparable with order-2 schemes.

CCS CONCEPTS

• **Theory of computation → Grammars and context-free languages**; **Probabilistic computation**; **Program verification**.

KEYWORDS

probabilistic (affine additive) higher-order recursion schemes; restricted (probabilistic) tree stack automata; computing termination probability; deciding almost sure termination; first-order theory of the reals

ACM Reference Format:
Guanyan Li, Andrzej S. Murawski, and C.-H. Luke Ong. 2022. Probabilistic Verification Beyond Context-Freeness . In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22), August 2–5, 2022, Haifa, Israel.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3531130.3533351

[*]This research was funded in whole, or in part, by Engineering and Physical Sciences Research Council (EP/T006579/1). For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

1 INTRODUCTION

Probabilistic (or randomised) programs are widely recognised as essential to solving numerous algorithmic problems efficiently [22]. In probabilistic programming [13, 23, 30], an approach to Bayesian statistical machine learning that utilises ideas and methods from programming languages, probabilistic programs are used to express generative models whose posterior probability can be computed by general purpose inference engines. The universal probabilistic programming languages (for example, Church [12], Anglican [28], Gen [7], Pyro [3], Edward [29] and Turing [10]) are typically higher-order, recursive functional programming language equipped with random primitives (and conditioning constructs).

This paper is concerned with a central property of probabilistic programs: *termination.* When a probabilistic program implements a solution to an algorithmic problem, we naturally require the computation to terminate with probability 1, in which case the program is called *almost surely terminating* (AST). Indeed, it is standard for designers and implementors of probabilistic programming systems to regard non-AST programs as defining invalid models, and hence inadmissible [12, 23]. (Yet none of these systems provides any support for the development or verification of AST programs.) Moreover, various theorems about probabilistic programs rely on the assumption that the program terminates almost surely (see e.g. [12, 19]). The AST problem is not just important, but also difficult: deciding AST of first-order imperative programs with discrete probabilities is Π_2^0-complete [15, 18] . In recent work [2], the AST problem for higher-order programs (with continuous distributions) was shown to have the same complexity.

To investigate scenarios involving probability and higher-order computation, Kobayashi, Dal Lago and Grellois introduced the notion of probabilistic higher-order recursion schemes (PHORS) [17]. In the deterministic case, higher-order recursive schemes (HORS) have been a fruitful approach to algorithmic analysis of higher-order programs [16], which applies to all orders [21]. In stark contrast, the authors of [17] report that the AST problem for PHORS is undecidable already at order 2, and provide decidability results for order-1 PHORS only.

Contributions. In this paper, we consider a subclass of PHORS, called *Probabilistic Additive HORS* (PAHORS). They are inspired by Multiplicative Additive HORS (MAHORS) [6], in which the order of computation is unbounded, but function definitions must obey the discipline of multiplicative additive linear logic [11], i.e. variable sharing is controlled by a linear additive type system. In the deterministic case, MAHORS are known to be more expressible than order-1 HORS and incomparable with order-2 HORS. We are interested in finding decision procedures for the almost-sure termination (AST) and *positive almost-sure termination* (PAST) problems for PAHORS. The latter concerns the question whether the expected time to termination is finite.

Our starting point is the model of *restricted tree-stack automata* (rTSA) [8], known to be expressively equivalent to MAHORS [6] (with respect to tree generation). A tree-stack is a tree-shaped stack (growing upwards) satisfying *restriction*: each node can be visited from below no more than an *a priori* fixed number (k, say) of times. We first adapt the automata to the probabilistic case—called ***rPTSA***—and show how to characterise the associated termination probability and expected time to termination equationally, over the real numbers with addition and multiplication.

The central idea that enables the equational characterisation is a decomposition of rPTSA runs into finitely many "patterns" (which we call $\mathit{Trips}_D^{\gamma}$), as succinctly captured in *Decomposition Formula 1* (Equation (1)). This (de)construction may be viewed as a generalisation of decomposition techniques for runs of pushdown automata [4]. To each pattern, one can associate a probability (weight), and the mutual dependency among the (subject node's and its children's) patterns defines a system of polynomial equations. Once the equations have been derived, the existence and properties of their least fixpoints can be decided using the existential fragment of the first-order theory of the reals (EToR), similarly to probabilistic pushdown automata [4]. Thanks to the decidability of EToR [5, 26], the AST and PAST problem of the automata model, rPTSA, can be solved effectively; moreover one can query the logic to approximate the exact values. The total number of these patterns depends on the restriction k. This makes it possible to determine a complexity bound (EXPSPACE) under the assumption that k is given in unary.

A similar decomposition principle underpins ω-regular model checking of rPTSA. Now, using an additional type of "patterns" (called Up_D^{γ}) for infinite runs, we obtain a corresponding equational characterisation in *Decomposition Formula 2* (Equation (6)). Moreover, these Up_D^{γ} patterns form themselves into a finite Markov chain. We can determine whether each bottom SCC (BSCC) of the digraph underlying the Markov chain is accepting, thanks to the decomposition formulas. Further the probability of reaching an accepting BSCC can be specified in EToR as a solution to a system of equations. Thus, as before, both the qualitative and quantitative verification problems are decidable (in EXPSPACE).

We can now transfer decidability results from rPTSA to PAHORS. Our first result about PAHORS states that the AST problem for PAHORS is decidable and that the associated termination probabilities can be approximated effectively. This subsumes analogous results for order-1 PHORS [17]. Moreover, we investigate the associated PAST problem. We can show that, for PAHORS, the problem is also decidable and the expected time to termination can be approximated effectively too.

When it comes to the AST problem, PAHORS are the first decidable extension of PHORS that goes beyond order 1 [17]. For PAST, we are not aware of any prior results, so we believe we are the first to identify a non-trivial class of higher-order programs with decidable PAST.

All EToR queries used in the paper are of exponential size and can be constructed in (at most) exponential space. This implies that the respective decision problems are in EXPSPACE.

Outline. In Section 2 we present the basic definitions and the technical preliminaries. Section 3 discusses the termination probability of rPTSA (and hence PAHORS). In Section 4, we investigate the expected time to termination. In Section 5 we present our ω-regular model checking results. In Section 6 we show how to relate PAHORS to rPTSA, thus enabling us to derive decidability results for PAHORS. We conclude in Section 7.

2 TECHNICAL PRELIMINARIES

In this section, we introduce two formalisms, namely rPTSA and PAHORS, which will be related in Section 6.

2.1 Restricted probabilistic tree stack automata

We introduce restricted probabilistic tree-stack automata (rPTSA), which will be the main model of probabilistic computation studied in the paper. The automata can be thought of as finite-control devices walking on trees built from the stack alphabet, starting at the root. Each node will have a local memory and the number of times a node can be visited *from below* will be restricted. rPTSA can be viewed as a probabilistic variant of tree-stack automata introduced by Denkinger [8]. Given a finite set X, we write $\mathit{Dist}(X)$ for the set of probability distributions over X.

Definition 2.1. A ***probabilistic tree stack automaton*** (PTSA) is a tuple $\langle Q, \mathcal{M}, \Gamma, \Delta, q_{init}, \gamma_{init}, m_{init}\rangle$ where

- Q is a *finite* set of states, $q_{init} \in Q$ is the initial state;
- $\mathcal{M}$ is a *finite* set of possible local memories, and $m_{init} \in \mathcal{M}$ is the initial memory;
- Γ is a *finite* set called the stack alphabet, $\gamma_{init} \in \Gamma$ is the initial stack symbol;
- $\Delta : Q \times \mathcal{M} \times \Gamma \to \mathit{Dist}(\mathit{Op})$ is the transition function, where $\mathit{Op} := Q + \big(Q \times \mathcal{M} \times (\{\mathit{up}_\gamma \mid \gamma \in \Gamma\} + \{\mathit{down}\})\big)$ and $+$ stands for the disjoint union of sets.

We will refer to specific transitions as tuples $\delta = (q, m, \gamma, op, p)$, where $op \in \mathit{Op}$, which stands for $\Delta(q, m, \gamma)(op) = p$.

Configurations of PTSA will rely on a $(\Gamma, \mathcal{M})$***-tree-stack***, which is a partial function $\Gamma^* \rightharpoonup \mathcal{M}$ with a prefix-closed domain. Thus, a tree-stack can be viewed as a tree whose edges are labelled with elements of Γ and nodes with elements of $\mathcal{M}$. The latter play the role of local memory. As we shall see, a PTSA will maintain a tree-stack during its lifetime and will walk inside it, possibly extending it and updating the local memories.

A ***configuration***, typically written c, is a triple (q, t, ρ) where $q \in Q$, t is a $(\Gamma, \mathcal{M})$-tree-stack, and $\rho \in \mathrm{dom}(t)$. We call ρ the *pointer* (in the tree-stack t) of the configuration. The last symbol of ρ, if any, is called the *current stack symbol*. If the pointer is not empty, we often write it as $\rho\gamma$ to make the current stack symbol explicit. Configurations with the empty pointer ϵ are called ***terminating***. The *initial configuration* is $c_{init} := (q_{init}, t_{init}, \gamma_{init})$ where $t_{init} := \{(\gamma_{init}, m_{init})\}$.

We distinguish three types of transitions according to the shape of op: ***horizontal***, up_γ (or simply ***up***) and ***down***. Horizontal transitions do not change the stack pointer, i.e. the automaton stays in the same node; non-horizontal transitions move the pointer up or down in the tree-stack, initialising local memory to m_{init} on a first visit, and preserving the memory otherwise. Formally, suppose

$\delta = (q, m, \gamma, op, p) \in \Delta$ and $c = (q, t, \rho\gamma)$ is a configuration such that $t(\rho\gamma) = m$. Figure 1 specifies when and how c may evolve into another configuration c'. We then write $c \xrightarrow{\delta} c'$. We say that the *probability* of $c \xrightarrow{\delta} c'$ is equal to the last component of δ, written p_δ.

A sequence $\pi = c_0 \cdots c_n$ of configurations such that $c_j \xrightarrow{\delta_j} c_{j+1}$ $(0 \le j < n)$, will be called a ***run*** (from c_0 to c_n). Note that π uniquely determines the corresponding sequence of transitions. We define the ***weight of a run*** to be $wt(\pi) := \prod_{j=0}^{n-1} p_{\delta_j}$. A run is *terminating* if c_n is terminating, i.e. the stack pointer in c_n is empty. Let us write R_{term} for the set of all terminating runs from c_{init}. Then the ***termination probability*** of a PTSA $\mathcal{P}$, written $\mathbb{P}(\mathcal{P})$, is defined to be $\sum_{\pi \in R_{term}} wt(\pi)$.

A run π is said to satisfy ***k-restriction*** if, for all ρ and γ, it contains at most k pairs of contiguous configurations that match $(-, -, \rho)\,(-, -, \rho\gamma)$. Intuitively, π cannot enter any node of the tree-stack *from below* more than k times.

Definition 2.2. Given $k \ge 1$, a PTSA is called ***k-restricted*** if all of its runs satisfy k-restriction. It is called ***restricted*** (rPTSA) if there exists k such that it is k-restricted.

From now on, we shall assume that we work with a k-restricted PTSA $\mathcal{P}$, where the parameter $k \ge 1$ is given in *unary* alongside $\mathcal{P}$. When we talk of exponential complexity for PTSA, we mean exponentiality in $(|\mathcal{P}| + k)$, i.e. an upper bound of the shape $\exp(poly(|\mathcal{P}| + k))$.

Example 2.3. We consider an rPTSA that imitates the work of a probabilistic printer and generates all words of the form $w\,w$, where $w \in \{A, B\}^*$. Initially, it will choose w at random by deciding at each step whether to print A (with probability p_A), print B (with probability p_B) or stop generating w (with probability p_E) in order to proceed to re-printing the chosen w. Reprinting is possible, because w will be recorded in the tree-stack using m_A and m_B. We assume $p_A + p_B + p_E = 1$. The automaton is presented in Figure 2, where each transition is accompanied by a short explanation.The rPTSA is 2-restricted, because it makes two up-and-down passes over the stack-tree.

Remark 2.4. It is well known that the language $\{w\,w \mid w \in \{A, B\}^*\}$ is not context-free. Indeed, the nondeterministic variant of rPTSA has been introduced to capture *multiple context-free languages* (MCFL) [25], which subsume context-free languages (CFL). Inside the higher-order (collapsible) pushdown hierarchy [14], MCFLs are situated between order 1 (CFL) and order 3 [24], and are incomparable with order 2 (indexed languages) [25]. The fact that rPTSA can simulate probabilistic PDA is not completely obvious. Due to the k-restriction, one cannot simply replace push and pop with up_γ and *down*. Nevertheless, it is possible to simulate probabilistic PDA with 1-restriction only.

Definition 2.5. A ***threshold problem*** for the termination probability of an rPTSA $\mathcal{P}$ asks whether $\mathbb{P}(\mathcal{P}) \sim r$ for a given rational r and $\sim \in \{<, \le, >, \ge\}$. The threshold problem corresponding to $\mathbb{P}(\mathcal{P}) \ge 1$ (equivalently, $\mathbb{P}(\mathcal{P}) = 1$) is known as the ***almost sure termination (AST)*** problem.

Note that, using queries to the threshold problem, one can approximate $\mathbb{P}(\mathcal{P})$ to an arbitrary level of precision via bisection.

We will also be interested in the *expected time to termination* for rPTSA. rPTSA has a natural notion of time: each transition can be viewed as taking a single unit of time. However, in order to apply our techniques to higher-order recursive computation, we will consider a more selective way of timing, where only designated horizontal transitions are counted. Specifically, we assume that each horizontal transition comes with a flag that indicates whether or not it should be counted, and *up*, *down* transitions do *not* contribute to time. The expected time to termination for an rPTSA is then calculated with respect to the *flagged* transitions only.

Definition 2.6. Given an rPTSA $\mathcal{P}$ and $\pi \in R_{term}$, let $\ell(\pi)$ be the number of flagged transitions in π. Then the ***expected time to termination*** is $ett(\mathcal{P}) := \sum_{\pi \in R_{term}} (\ell(\pi) \cdot wt(\pi))$.

Remark 2.7. Our definition of $ett(\mathcal{P})$ is equivalent to *expected total accumulated reward*, when $\ell(\pi)$ is taken to be the sum of rewards accumulated along π, where the reward of each transition is given by a customised reward function. In this paper we focus on termination time, but all our developments can easily be adapted to rewards.

Definition 2.8. The ***PAST problem*** for rPTSA $\mathcal{P}$ asks whether $ett(\mathcal{P})$ is finite. A ***threshold problem*** for the expected time to termination of rPTSA $\mathcal{P}$ asks whether $ett(\mathcal{P}) \sim r$, given a rational r and $\sim \in \{<, \le, >, \ge\}$.

2.2 Probabilistic additive HORS (PAHORS)

Higher-order recursion schemes (HORS) are a computational formalism based on rewriting typed terms. Each reduction step corresponds to unfolding a non-terminal according to a corresponding rule, which corresponds to substituting actual arguments (which may well be functions) into the function body. In standard HORS, the type discipline follows that of the simply-typed λ-calculus. In recent work [17], a probabilistic extension of HORS was introduced, and undecidability (of almost sure termination) was shown to hold already at order 2, i.e. when arguments can be of base type or first-order functions. In this paper, we will not introduce any restrictions on the type-theoretic order. However, we are going to use a stricter type discipline to restrict the shape of allowable terms. That discipline will come from linear logic [11].

We use types defined by the grammar $\theta ::= o^n \mid \theta \multimap \theta$, where o is a base type and o^n stands for the n-fold product of o, i.e. $\underbrace{o\& \cdots \& o}_{n}$ $(n \ge 1)$, and $\multimap$ is the linear function space. Let $\mathcal{N}$ be a finite set of typed variables, called *non-terminals*. We use upper-case letters $F, G, \cdots$ to range over $\mathcal{N}$ and write $\mathcal{N}(F)$ for the type of F. We shall consider probabilistic (affine additive) λ-terms with non-terminals from $\mathcal{N}$ and constant (terminal) $\top : o$, which represents termination. The typing judgments $\mathcal{N} \mid \Delta \vdash t : \theta$, where Δ is a partial function from variable names to types and $\mathrm{dom}(\Delta) \cap \mathcal{N} = \emptyset$, are defined by induction over the rules given in Figure 3.

The rules restrict the way in which function parameters (free variables) occur inside a term. However, non-terminals from $\mathcal{N}$ are

Transition type	op	c'	pre-condition
Horizontal	q'	$(q', t, \rho\gamma)$	none
Up	$(q', m', up_{\gamma'})$	$(q', t[\rho\gamma \mapsto m'], \rho\gamma\gamma')$	$\rho\gamma\gamma' \in \mathrm{dom}(t)$
		$(q', t[\rho\gamma \mapsto m'][\rho\gamma\gamma' \mapsto m_{init}], \rho\gamma\gamma')$	$\rho\gamma\gamma' \notin \mathrm{dom}(t)$
Down	$(q', m', down)$	$(q', t[\rho\gamma \mapsto m'], \rho)$	none

Figure 1: Specification of $c \xrightarrow{\delta} c'$, where $c = (q, t, \rho\gamma)$ is such that $t(\rho\gamma) = m$ and $\delta = (q, m, \gamma, op, p)$

The rPTSA is given by $\langle \{q_{init}, q_E\}, \{m_{init}, m_A, m_B, m_E\}, \{\gamma_{init}, \gamma\}, \Delta, q_{init}, m_{init}, \gamma_{init}\rangle$, where Δ contains the following transitions.

$(q_{init}, m_{init}, \gamma_{init}, (q_{init}, m_A, up_\gamma), p_A)$	choose A	$(q_E, m_A, \gamma_{init}, (q_{init}, m_E, up_\gamma), 1)$	re-print first letter of w
$(q_{init}, m_{init}, \gamma_{init}, (q_{init}, m_B, up_\gamma), p_B)$	choose B	$(q_E, m_B, \gamma_{init}, (q_{init}, m_E, up_\gamma), 1)$	re-print first letter of w
$(q_{init}, m_{init}, \gamma_{init}, (q_E, m_E, down), p_E)$	$w = \epsilon$, terminate	$(q_{init}, m_A, \gamma, (q_{init}, m_E, up_\gamma), 1)$	re-print A
$(q_{init}, m_{init}, \gamma, (q_{init}, m_A, up_\gamma), p_A)$	choose A	$(q_{init}, m_B, \gamma, (q_{init}, m_E, up_\gamma), 1)$	re-print B
$(q_{init}, m_{init}, \gamma, (q_{init}, m_B, up_\gamma), p_B)$	choose B	$(q_{init}, m_E, \gamma, (q_E, m_E, down), 1)$	end second w
$(q_{init}, m_{init}, \gamma, (q_E, m_E, down), p_E)$	end first w	$(q_E, m_E, \gamma, (q_E, m_E, down), 1)$	descend after second w
$(q_E, m_A, \gamma, (q_E, m_A, down), 1)$	descend after first w	$(q_E, m_E, \gamma_{init}, (q_E, m_E, down), 1)$	terminate
$(q_E, m_B, \gamma, (q_E, m_B, down), 1)$	descend after first w		

Figure 2: Automaton for Example 2.3

$$\frac{\mathcal{N}(F) = \theta}{\mathcal{N} \mid \Delta \vdash F : \theta} \qquad \frac{\Delta(x) = \theta}{\mathcal{N} \mid \Delta \vdash x : \theta} \qquad \frac{}{\mathcal{N} \mid \Delta \vdash \top : o} \qquad \frac{\mathcal{N} \mid \Delta_1 \vdash t_1 : \theta \multimap \theta' \qquad \mathcal{N} \mid \Delta_2 \vdash t_2 : \theta}{\mathcal{N} \mid \Delta_1, \Delta_2 \vdash t_1\, t_2 : \theta'}$$

$$\frac{\mathcal{N} \mid \Delta \vdash t_i : o \quad i = 1, \cdots, n}{\mathcal{N} \mid \Delta \vdash \langle t_1, \cdots, t_n\rangle : o^n} \qquad \frac{\mathcal{N} \mid \Delta \vdash t : o^n \quad i \in \{1, \cdots, n\}}{\mathcal{N} \mid \Delta \vdash \pi_i\, t : o} \qquad \frac{\mathcal{N} \mid \Delta, x : \theta \vdash t : \theta'}{\mathcal{N} \mid \Delta \vdash \lambda x.t : \theta \multimap \theta'}$$

Figure 3: Affine additive typing rules

$$\frac{\mathcal{R}(F) = \lambda x_1 \cdots x_k.t}{F t_1 \cdots t_k \xrightarrow{1} t[t_1/x_1, \cdots, t_k/x_k]} \qquad \pi_i\langle t_1, \cdots, t_n\rangle \xrightarrow{1} t_i \qquad t_1 \oplus_p t_2 \xrightarrow{p} t_1 \qquad t_1 \oplus_p t_2 \xrightarrow{1-p} t_2$$

Figure 4: PAHORS reduction rules

excluded from these restrictions. In addition, we include probabilistic branching

$$\frac{\mathcal{N} \mid \Delta \vdash t_1 : o \quad \mathcal{N} \mid \Delta \vdash t_2 : o \quad p \in \mathbb{Q}}{\mathcal{N} \mid \Delta \vdash t_1 \oplus_p t_2 : o}$$

with the intention that p and $1 - p$ will be the probabilities of choosing the left and right branches respectively. Note that, like in the rule for pairing, the variables in Δ can be shared. In contrast, in the rule for function application, they cannot be shared.

Definition 2.9. A ***Probabilistic Additive HORS (PAHORS)*** is a tuple $\mathcal{G} = \langle \mathcal{N}, \mathcal{R}, S\rangle$, where $\mathcal{N}$ is a finite set of typed non-terminals, $S \in \mathcal{N}$ is the *start symbol* such that $\mathcal{N}(S) = o$, and $\mathcal{R}$ is a function that associates to each $F \in \mathcal{N}$ a term $\mathcal{N} \mid \emptyset \vdash \lambda x_1 \cdots x_k.t : \mathcal{N}(F)$ such that t does not contain λ-abstractions and $\mathcal{N} \mid x_1, \cdots, x_k \vdash t : o$.

Example 2.10. Let $p_1, p_2, p_3 \in \mathbb{Q}$. Consider the PAHORS $\mathcal{G} = \langle \mathcal{N}, \mathcal{R}, S\rangle$, where $\mathcal{N} = \{(S, o), (I, o \multimap o), (F, (o \multimap o) \multimap o), (G, (o \multimap o) \multimap o \multimap o)\}$, and $\mathcal{R}$ is specified below.

$$S = F\,(G\,I) \qquad\qquad I\,x = x$$
$$F\,f = F(Gf) \oplus_{p_1} f\,\top \qquad G\,f\,x = f\,(x \oplus_{p_3} \top) \oplus_{p_2} \top$$

Because F, G use functional arguments, $\mathcal{G}$ is classified as order 2.

Remark 2.11. PAHORS subsume order-1 probabilistic HORS [17]. To simulate an order-1 non-terminal $F : \underbrace{o \to \cdots \to o}_{n} \to o$, one can take $F : \underbrace{o\& \cdots \& o}_{n} \multimap o$. A typical rule, such as $F\,x\,y = F(F\,x\,x)\,(F\,y\,y) \oplus_p x$, can then be simulated by

$$F\,z = F\,\langle F\,\langle \pi_1 z, \pi_1 z\rangle, F\,\langle \pi_2\,z, \pi_2\,z\rangle\rangle \oplus_p (\pi_1\,z).$$

In order to define the termination probability $\mathbb{P}(\mathcal{G})$, we rely on reduction rules of the shape $t \xrightarrow{p} t'$ given in Figure 4, where $p \in \mathbb{Q}$.

We write $Red^\top(\mathcal{G})$ for the set of *terminating* reduction sequences, i.e. those from S to $\top$. The subset of $Red^\top(\mathcal{G})$ consisting of reductions of length not exceeding n will be written $Red^\top_{\leq n}(\mathcal{G})$. Similarly, we write $Red_n^{\top}(\mathcal{G})$ for the subset of $Red_{\leq n}^{\top}(\mathcal{G})$ comprising reductions of length n. Given $\zeta = (S \xrightarrow{p_1} \dots \xrightarrow{p_n} \top) \in Red_n^{\top}(\mathcal{G})$, we define $\mathbb{P}(\zeta) := \prod_{i=1}^{n} p_i$.

Definition 2.12. The ***termination probability*** $\mathbb{P}(\mathcal{G})$ of a PAHORS $\mathcal{G}$ is $\sum_{\zeta \in Red^{\top}(\mathcal{G})} \mathbb{P}(\zeta)$. Given a PAHORS $\mathcal{G}$, the associated ***almost sure termination*** (AST) problem asks whether $\mathbb{P}(\mathcal{G}) = 1$.

Definition 2.13. The ***expected time to termination*** $ett(\mathcal{G})$ of a PAHORS $\mathcal{G}$ is $\sum_{n=0}^{\infty}(1 - \sum_{\zeta \in Red_{\leq n}^{\top}(\mathcal{G})} \mathbb{P}(\zeta))$. Given a PAHORS $\mathcal{G}$, the associated ***positive almost sure-termination*** problem (PAST) asks whether $ett(\mathcal{G})$ is finite.

Remark 2.14. [15] It is known that PAST implies AST, i.e. $\mathbb{P}(\mathcal{G}) < 1$ implies $ett(\mathcal{G}) = \infty$. Moreover, if $\mathbb{P}(\mathcal{G}) = 1$, then $ett(\mathcal{G}) = \sum_{n=0}^{\infty} n \cdot (\sum_{\zeta \in Red_n^{\top}(\mathcal{G})} \mathbb{P}(\zeta))$.

By analogy to rPTSA, one can also study threshold problems for $\mathbb{P}(\mathcal{G})$ and $ett(\mathcal{G})$.

Remark 2.15. For $\mathcal{G}$ from Example 2.10, one can show that $\mathbb{P}(\mathcal{G}) = 1$ and $ett(\mathcal{G})$ is finite as long as $p_1 \neq 1$ (otherwise $\mathbb{P}(\mathcal{G}) = 0$ and $ett(\mathcal{G}) = \infty$). For example, for $p_1 = p_2 = p_3 = 0.5$, we have $ett(\mathcal{G}) = \frac{176}{21}$. In the setting of probabilistic HORS, the problem of establishing AST is already undecidable for order-2 probabilistic HORS [17]. Nonetheless, we will show that the AST and PAST problems for PAHORS are decidable, even though the order of schemes in PAHORS is unrestricted.

Remark 2.16. If we replace probabilistic branching $\oplus_{p_1}, \oplus_{p_2}, \oplus_{p_3}$ in $\mathcal{G}$ from Example 2.10 with terminal symbols $a, b, c : o \to o \to o$, we obtain an order-2 recursion scheme featuring the following rules.

$$\begin{aligned} S &= F\,(G\,I) & I\,x &= x \\ F\,f &= a\,(F\,(G\,f))(f\,\top) & G\,f\,x &= b\,(f\,(c\,x\,\top))\,\top \end{aligned}$$

Unfolding the start symbol S will now generate an infinite tree where terminating branches (those that end with $\top$) will have one of the forms: $a^n\,\top$, $a^n\,b^m\,\top$ and $a^n\,b^n\,c^m\,\top$, where $1 \leq m \leq n$. Note that the corresponding word language is not context-free (because of the third case). Consequently, no order-1 probabilistic scheme can mimic the branching structure of $\mathcal{G}$.

First-order theory of the reals. In this work, we design algorithms that rely on (decision) procedures for the first-order theory of the reals, $(\mathbb{R}, +, \times, \leq)$. Given a closed first-order formula ϕ over the signature $\{+, \times, \leq\}$, the problem whether ϕ holds in the universe of real numbers, with the standard interpretation of $+$ and $\times$, is decidable [26]. The existential fragment of $(\mathbb{R}, +, \times, \leq)$, which will be referred to as ***EToR***, is decidable in polynomial space [5].

3 CHARACTERISING TERMINATION PROBABILITIES

Let us fix an rPTSA $\langle Q, \mathcal{M}, \Gamma, \Delta, q_{init}, m_{init}, \gamma_{init} \rangle$. The goal of this section is to construct a system of equations that captures the dependencies between parent and child nodes. In order to construct it, we introduce several auxiliary notions that will help us to decompose runs.

Definition 3.1. A ***prerun*** is a sequence of runs. Its weight is defined to be the product of weights of the constituent runs.

We will use preruns to represent incomplete runs: we take the points of contact between adjacent runs in a prerun to be places (or "gaps") that are missing transition sequences, and we will show that transition sequences of certain shapes can be slotted into these places to complete a prerun to a run.

Definition 3.2. A run from a configuration of the form (q, t, γ) to a configuration of the form (q', t', ϵ) is called a ***trip***.

Note that the last transition in a trip is necessarily a *down* transition and that terminating runs are instances of trips.

Definition 3.3. Let $D = [q_1, \cdots, q_{2i}]$ $(i \geq 0)$. Let $Trips_D^{\gamma}$ be the set of preruns obtained by concatenating i trips $\tau_1 \cdots \tau_i$ such that, for all $1 \leq j \leq i$, τ_j is a trip from (q_{2j-1}, t_j, γ) to $(q_{2j}, t_{j+1}, \epsilon)$, and $t_1 = \{(\gamma, m_{init})\}$. For $i = 0$, this degenerates to $Trips_{[]}^{\gamma} = \{\epsilon\}$.

For $i > 0$, it will be useful to see preruns from $Trips_D^{\gamma}$ as sequences in which $i-1$ gaps have been left between trips. Intuitively, the $i-1$ gaps (between $(q_{2j}, t_{j+1}, \epsilon)$ to $(q_{2j+1}, t_{j+1}, \gamma)$ for $1 \leq j < i$; notice that the tree-stack t_{j+1} does not change) can be viewed as placeholding transitions with weight 1. Our decomposition principle will consist in filling the gaps with appropriate transitions.

Overall, $Trips_D^{\gamma}$ captures the history of a node, which we will call ***the subject node***, that has been visited i times and ultimately left by executing *down*, along with the history of all nodes above it. Thanks to the requirement that $t_1 = \{(\gamma, m_{init})\}$, the first entry into the node is consistent with creating the node.

Figure 5 shows a typical element of $Trips_D^{\gamma}$: the solid arrow starting from q_{2j-1} (jth entry) and ending at q_{2j} (jth exit) depicts the jth trip.

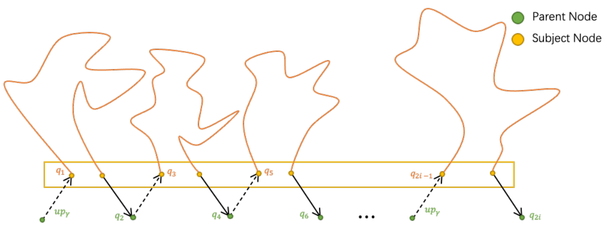

Figure 5: A prerun from $Trips_D^{\gamma}$ with $D = [q_1, \cdots, q_{2i}]$

Remark 3.4. Define $Ter_{q_{init}}^{\gamma_{init}} := \bigcup_{q \in Q} Trips_{[q_{init}, q]}^{\gamma_{init}}$. Note that $Ter_{q_{init}}^{\gamma_{init}}$ contains exactly the terminating runs. Consequently, $\mathbb{P}(\mathcal{P}) = \sum_{\pi \in Ter_{q_{init}}^{\gamma_{init}}} wt(\pi)$.

3.1 Inductive characterisation of $Trips_D^{\gamma}$

Let $n \geq 1$, we write $[n] := \{1, \cdots, n\}$. Let $\Gamma = \{\gamma_1, \cdots, \gamma_{|\Gamma|}\}$. Then the set $Trips_D^{\gamma}$ *of preruns* admits the following decomposition with respect to the child nodes of the subject node. (We view tree-stacks the way trees occur in nature: the root is at the bottom.) We use $\mathcal{D}$ to denote an assignment of an even-length list of states $\mathcal{D}_i$ to each direction γ_i, such that $|\mathcal{D}_i| \leq 2k$. Formally $\mathcal{D}$ is a function on $[|\Gamma|] := \{1, \cdots, |\Gamma|\}$, but we sometimes present it as a tuple $\mathcal{D} = (\mathcal{D}_1, \cdots, \mathcal{D}_{|\Gamma|})$. The decomposition expresses the fact that each prerun from $Trips_D^{\gamma}$ is an interleaving of

- a prerun of $\mathit{Trips}^{\gamma_i}_{\mathcal{D}_i}$ (whose subject node is the γ_i-child of the subject node of $\mathit{Trips}^{\gamma}_{D}$), for each $1 \le i \le |\Gamma|$, with
- a "synchronising" prerun consisting of horizontal transitions inside the subject node, and up_{γ_i} transitions and *down* transitions from the subject node,

for some $\mathcal{D}$. Note that $\mathcal{D}$ captures assumptions on the behaviour of each child node of the subject node.

We can describe the decomposition equationally. Defining $wt(\mathit{Trips}^{\gamma}_{D}) := \sum_{\pi \in \mathit{Trips}^{\gamma}_{D}} wt(\pi)$, we have:

Decomposition formula 1: $\mathit{Trips}^{\gamma}_{D}$

$$wt(\mathit{Trips}^{\gamma}_{D}) = \sum_{\mathcal{D}} \Big(\big(\prod_{i \in [|\Gamma|]} wt(\mathit{Trips}^{\gamma_i}_{\mathcal{D}_i}) \big) \cdot \sum_{\pi \in \mathit{SynP}^{\mathcal{D}}_{\gamma,D}} wt(\pi) \Big) \tag{1}$$

where $\mathit{SynP}^{\mathcal{D}}_{\gamma,D}$ consists of (synchronising) preruns of $\mathit{Trips}^{\gamma}_{D}$ in which transitions from descendant nodes of the subject node are erased, creating gaps that are consistent with $\mathcal{D}_i$ for each γ_i, (i.e., for each contiguous pair $q'_{2j-1}\, q'_{2j}$ from $\mathcal{D}_i$ there must be a gap from q'_{2j-1} to q'_{2j} and the gaps must occur in the same order as the list $\mathcal{D}_i$). See Example 3.5 for an illustration of a synchronising prerun.

Now if we use $\mathcal{D}_i$ to fix the subject node's up-and-down behaviour in the direction γ_i, constrain its down-and-up behaviour by D, and interpret the gaps as placeholding transitions of weight 1 then the node becomes a finite Markov chain (see Definition 3.6), such that each $\pi \in \mathit{SynP}^{\mathcal{D}}_{\gamma,D}$ is a trajectory of the chain (that hits a certain set of states). Consequently, $\sum_{\pi \in \mathit{SynP}^{\mathcal{D}}_{D}} wt(\pi)$ coincides with the hitting probability, and can be computed effectively.

Example 3.5. Suppose $D = [q_1, q_2, q_3, q_4, q_5, q_6]$, $\mathcal{D}_1 = [q^1_1, q^1_2, q^1_3, q^1_4]$, $\mathcal{D}_2 = [q^2_1, q^2_2]$ with $\Gamma = \{\gamma_1, \gamma_2\}$. A possible decomposition described by Equation (1) is depicted in Figure 6. The blue and grey preruns are from $\mathit{Trips}^{\gamma_1}_{\mathcal{D}_1}$ and $\mathit{Trips}^{\gamma_2}_{\mathcal{D}_2}$ respectively. The prerun consisting of the black (solid) arrows is a (synchronising) prerun in $\mathit{SynP}^{\mathcal{D}}_{\gamma,D}$.

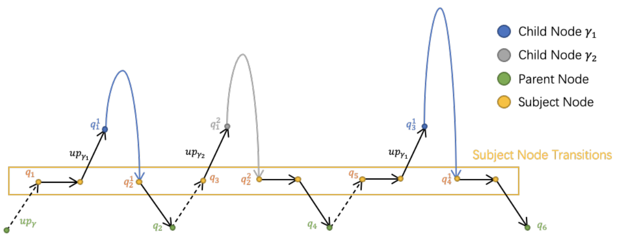

Figure 6: A decomposition of $\mathit{Trips}^{\gamma}_{D}$ as per Equation (1)

3.2 Synchronising Markov chain

Given γ, D and $\mathcal{D}$, we define $\mathit{SynP}^{\mathcal{D}}_{\gamma,D}$ as the set of trajectories (that hit a certain set of states) of a Markov chain. Intuitively, the Markov chain is a product construction on $(|\Gamma| + 1)$ finite-state automata (Markov chains): for each $i \in \{1, \cdots, |\Gamma|\}$, the i-automaton is the straight-line code given by the list $\mathcal{D}_i = [q^i_1, \cdots, q^i_{|\mathcal{D}_i|}]$. The 0-automaton, synchronising with the i-automata ($i > 0$), then generates elements of $\mathit{SynP}^{\mathcal{D}}_{\gamma,D}$.

Definition 3.6. Set $D = [q^0_1, \cdots, q^0_{|D|}]$. The state set of the ***synchronising Markov chain***, $\mathit{Markov}^{\mathcal{D}}_{\gamma,D}$, consists of triples of the form $(\alpha, \mu, \boldsymbol{d})$ where $\alpha \in Q$, $\mu \in \mathcal{M}$, and $\boldsymbol{d}$ is a $(|\Gamma| + 1)$-tuple with $\boldsymbol{d}_0 \in \{0, 1, \cdots, |D|\}$, and for each $i > 0$, $\boldsymbol{d}_i \in \{0, 1, \cdots, |\mathcal{D}_i|\}$. (We view $\boldsymbol{d}$ as a function on the set $\{0, 1, \cdots, |\Gamma|\}$, and use the standard function update notation $\boldsymbol{d}[i \mapsto d]$.)

The transition matrix

$$(\alpha, \mu, \boldsymbol{d}) \xrightarrow{\tau} (\alpha', \mu', \boldsymbol{d}') \tag{2}$$

is defined by cases of τ as follows:

(1) *horizontal transition (within the subject node of $\mathit{Trips}^{\gamma}_{D}$)*, where $(\alpha, \mu, \gamma, q', p) \in \Delta$ and $\alpha \neq q^0_{|D|}$.
Then $\alpha' := q'$, $\mu' := \mu$, and $\boldsymbol{d}' := \boldsymbol{d}$.

(2) *up_{γ_i} transition (from the subject node of $\mathit{Trips}^{\gamma}_{D}$)*, where $(\alpha, \mu, \gamma, (q', m', up_{\gamma_i}), p) \in \Delta$, and $\boldsymbol{d}_i$ is even, and $\boldsymbol{d}_i < |\mathcal{D}_i|$, and $q' = q^i_{\boldsymbol{d}_i+1}$.
Then $\alpha' := q'$, $\mu' := m'$, and $\boldsymbol{d}' := \boldsymbol{d}[i \mapsto \boldsymbol{d}_i + 1]$.

(3) *down transition (from the subject node of $\mathit{Trips}^{\gamma}_{D}$)*, where $(\alpha, \mu, \gamma, (q', m', down), p) \in \Delta$, and $\boldsymbol{d}_0$ is odd, and $q' = q^0_{\boldsymbol{d}_0+1}$.
Then $\alpha' := q'$, $\mu' := m'$, and $\boldsymbol{d}' := \boldsymbol{d}[0 \mapsto \boldsymbol{d}_0 + 1]$.

(4) *odd-to-even (in $\mathcal{D}_i$)*, where $\boldsymbol{d}_i$ is odd, and $\alpha = q^i_{\boldsymbol{d}_i}$.
Then $\alpha' := q^i_{\boldsymbol{d}_i+1}$, and $\mu' := \mu$, and $\boldsymbol{d}' := \boldsymbol{d}[i \mapsto \boldsymbol{d}_i + 1]$.

(5) *even-to-odd (in D)*, where $\boldsymbol{d}_0$ is even, and $\boldsymbol{d}_0 + 1 < |D|$, and $\alpha = q^0_{\boldsymbol{d}_0}$.
Then $\alpha' := q^0_{\boldsymbol{d}_0+1}$, and $\mu' := \mu$, and $\boldsymbol{d}' := \boldsymbol{d}[0 \mapsto \boldsymbol{d}_0 + 1]$.

In the *odd-to-even* and *even-to-odd* cases (i.e. the gaps), the probability of the transition (2) is defined to be 1; otherwise we take it to be p from the transition in question. Finally, set

$$\mathit{EndStates} := \{(q^0_{|D|}, m, (|D|, |D_{\gamma_1}|, \cdots, |D_{\gamma_{|\Gamma|}}|)) \mid m \in \mathcal{M}\}$$

and define states in *EndStates* to be absorbing. Thus defined, the transition matrix is a stochastic matrix, because Δ satisfies stochasticity.

Each trajectory of the Markov chain starts from a state of the form $(q^0_1, m_{init}, (1, 0, \cdots, 0))$. Preruns in $\mathit{SynP}^{\mathcal{D}}_{\gamma,D}$ then correspond to the trajectories of the chain $\mathit{Markov}^{\mathcal{D}}_{\gamma,D}$ that hit *EndStates*. It follows that $\sum_{\pi \in \mathit{SynP}^{\mathcal{D}}_{\gamma,D}} wt(\pi)$ is the hitting probability of *EndStates*, denoted $hp^{\mathcal{D}}_{\gamma,D}$, which is expressible as a system of linear equations (see e.g. [20, Theorem 1.3.2]). Since the Markov chain is finite, this system is finite, and hence algorithmically solvable [26].

We write $mht^{\mathcal{D}}_{\gamma,D}$ for the *mean hitting time* of *EndStates*, where time is calculated with respect to the flagged transitions only (this could be viewed as a particular case of expected total accumulated reward).

3.3 Solution via fixpoints and EToR

Recall that we work in the setting of rPTSA for some fixed restriction $k \geq 1$. Then Equation (1) induces an operator $F : \mathbb{R}^n \to \mathbb{R}^n$, where $n = \sum_{i=0}^{k}(|Q|^{2i}|\Gamma|)$ is the number of $Trips_D^\gamma$ variables. The following lemma is a direct consequence of the decomposition.

Lemma 3.7. *$(F^h(\vec{0}))_{Trips_D^\gamma}$, meaning the $Trips_D^\gamma$-component of the tuple $F^h(\vec{0})$, corresponds to the aggregate weight of preruns from $Trips_D^\gamma$ that can visit up to h levels in the tree stack.*

Corollary 3.8. *The termination probability of the given rPTSA is $\sum_{q\in Q}(LFP(F))_{Trips^{\gamma_{init}}_{[q_{init},q]}}$, where $LFP(F)$ is the least fixpoint of the operator F.*

Thanks to the above result, we can solve all the threshold problems associated with termination probabilities. First, using EToR, one can easily specify the existence of a fixpoint $\vec{x}$ of F (i.e. $\vec{x} = F(\vec{x})$). Then, in order to solve threshold problems for $\sim \in \{<, \leq\}$, it suffices to ask additionally whether $x_{term} \sim r$ for the given rational r, where $x_{term} = \sum_{q\in Q} x_{Trips^{\gamma_{init}}_{[q_{init},q]}}$. Note that, because $\sim \in \{<, \leq\}$, it is not necessary to insist that $\vec{x}$ be the least fixpoint. To tackle threshold problems for $\sim \in \{>, \geq\}$, we can solve the complement first (using $\sim \in \{\leq, <\}$) and reverse the answer.

Closer analysis of the equation system reveals that it can be constructed in exponential time (assuming k given in unary).

- The number of $Trips_D^\gamma$ variables is $\sum_{i=0}^{k}(|Q|^{2i}|\Gamma|) \leq |Q|^{2k+2}|\Gamma|$.
- The number of summands in eq. (1), i.e. possible choices for $\mathcal{D}$, is $(\sum_{i=0}^{k}|Q|^{2i})^{|\Gamma|} \leq |Q|^{(2k+2)|\Gamma|}$.
- Each summand involves $|\Gamma|$ variables and the computation of a hitting probability in $Markov^{\mathcal{D}}_{\gamma,D}$. This can be done in polynomial time with respect to the number of states, so we focus on this next. By definition, the number of states in $Markov^{\mathcal{D}}_{\gamma,D}$ is $|Q||\mathcal{M}|(2k+1)^{|\Gamma|+1}$.

Overall, we can conclude that the corresponding system of equations can be constructed in exponential time (wrt $(|\mathcal{P}|+k)$). Because EToR is decidable in polynomial space, this implies membership in EXPSPACE for all the problems concerned.

Theorem 3.9. *For any rPTSA $\mathcal{P}$, the threshold problems for $\mathbb{P}(\mathcal{P})$ are decidable and solvable in exponential space. In particular, the same applies to the AST problem.*

4 CHARACTERISING EXPECTED TIME TO TERMINATION

In this section we give a fixpoint characterisation of the expected time to termination of an rPTSA, which will help us establish that the PAST problem and the associated threshold problems are decidable.

Remark 4.1. Because we are interested in time to termination, we will restrict our attention to γ, D such that $wt(Trips_D^\gamma) > 0$, as only such sequences can contribute to termination. Note that γ, D such that $wt(Trips_D^\gamma) = 0$ can be identified effectively via reachability analysis of the system of equations considered in Section 3.

As in Section 3, we fix a k-restricted PTSA, and use notations introduced therein. We begin by introducing a number of random variables. Given γ and D, define the *expected length* (or *expected time to termination*) of sequences from $Trips_D^\gamma$ as

$$\mathbb{E}[Time_{\gamma,D}] := \sum_{\pi\in Trips_D^\gamma} p(\pi)\cdot \ell(\pi)$$

where $p : \pi \mapsto \frac{wt(\pi)}{\sum_\pi wt(\pi)} = \frac{wt(\pi)}{wt(Trips_D^\gamma)}$ is a probability distribution over $Trips_D^\gamma$, and $\ell(\cdot)$ is the length function, which counts the number of *flagged* horizontal transitions in π (recall from Section 2.1 that we aim to count flagged transitions only). Here $Time_{\gamma,D}$ is a $\mathbb{N}$-valued random variable defined on the sample space $Trips_D^\gamma$, with probability measure given by p.

Remark 4.2. Let $D = [q_1, \cdots, q_{2i}]$ and $\gamma \in \Gamma$. Note that p can be viewed as a conditional distribution, where the condition states that "a fresh γ-node started from state q_1 will behave consistently with the behaviour prescribed by $Trips_D^\gamma$", i.e. when started in (q_1, t_1, γ) $(t_1 = \{(\gamma, m_{init})\})$, it will terminate in (q_2, t_2, ϵ) (for some t_2), *and* when restarted from (q_3, t_2, γ) it will terminate in (q_4, t_3, ϵ) (form some t_3), *and* so on. Crucially, if $D = [q^{init}, q]$ then the condition in question states that the rPTSA has a terminating run from c_{init} to (q, t, ϵ) for some t.

Given the conditional definition of $\mathbb{E}[Time_{\gamma,D}]$, we have:

Lemma 4.3. *For any rPTSA $\mathcal{P}$,*

$$ett(\mathcal{P}) = \sum_{q\in Q}(\mathbb{E}[Time_{\gamma_{init},[q_{init},q]}]\cdot wt(Trips^{\gamma_{init}}_{[q_{init},q]})).$$

Note that, if $wt(Ter^{\gamma_{init}}_{q_{init}}) = 1$ and there is only one q such that $wt(Trips^{\gamma_{init}}_{[q_{init},q]}) > 0$, then $wt(Trips^{\gamma_{init}}_{[q_{init},q]}) = 1$ and $ett(\mathcal{P}) = \mathbb{E}[Time_{\gamma_{init},[q_{init},q]}]$.

Definition 4.4. Recall from Section 3.1 that for every $\pi \in Trips_D^\gamma$, there exists a unique tuple of even-length lists $\vec{D} = (D_1, \cdots, D_{|\Gamma|})$ where each $|D_j| \leq 2k$, such that π decomposes into $\pi \upharpoonright i \in Trips^{\gamma_i}_{D_i}$ for each $i \in [|\Gamma|]$, and a synchronising sequence, $\pi \upharpoonright \gamma \in SynP^{\vec{D}}_{\gamma,D}$. We express such a decomposition as $\pi = (\pi \upharpoonright 1, \cdots, \pi \upharpoonright |\Gamma|) \parallel \pi \upharpoonright \gamma$, and say that the ***interface*** (of the decomposition) is $\vec{D}$.

Definition 4.5. Given γ and D, let $\mathcal{D}_{\gamma,D}$ be a random variable which is defined on the probability space $Trips_D^\gamma$ whose values are tuples of the form $\vec{D} = (D_1, \cdots, D_{|\Gamma|})$. Given $\pi \in Trips_D^\gamma$, we define $\mathcal{D}_{\gamma,D}(\pi)$ to be the decomposition interface of π. We write $\mathbb{P}[\mathcal{D}_{\gamma,D} = \vec{D}]$ for the probability that $\pi \in Trips_D^\gamma$ decomposes with interface $\vec{D}$.

Thus we defined $\mathbb{P}[\mathcal{D}_{\gamma,D} = \vec{D}] := \frac{wt(\mathcal{D}_{\gamma,D}=\vec{D})}{\sum_{\vec{D}} wt(\mathcal{D}_{\gamma,D}=\vec{D})}$. Note that by Equation (1), we have $\sum_{\vec{D}} wt(\mathcal{D}_{\gamma,D} = \vec{D}) = wt(Trips_D^\gamma)$.

Lemma 4.6. *Recall that $hp^{\mathcal{D}}_{\gamma,D}$ is the hitting probability of EndStates. Then*

$$\mathbb{P}[\mathcal{D}_{\gamma,D} = \vec{D}] = \frac{\prod_{i\in[|\Gamma|]} wt(Trips^{\gamma_i}_{D_i})\cdot hp^{\vec{D}}_{\gamma,D}}{wt(Trips_D^\gamma)} \tag{3}$$

The set of subsets $\mathcal{D}^{-1}_{\gamma,D}(D_1, \cdots, D_{|\Gamma|})$, where each D_i ranges over even-length lists of states such that $|D_i| \le 2k$, is a partition of the sample space of the random variable $Time_{\gamma,D}$. Hence, by the Law of Total Expectation, we have

$$\mathbb{E}[Time_{\gamma,D}] = \sum_{\vec{D}} \mathbb{E}[Time_{\gamma,D} \mid \mathcal{D}_{\gamma,D} = \vec{D}] \cdot \mathbb{P}[\mathcal{D}_{\gamma,D} = \vec{D}]$$

Let $\pi \in Trips^{\gamma}_{D}$, define random variables $Time^{\text{inside}}_{\gamma,D}(\pi) := \ell(\pi \upharpoonright \gamma)$, and $Time^{\text{outside}}_{\gamma,D}(\pi) := \sum_{i \in [|\Gamma|]} \ell(\pi \upharpoonright i)$, and so

$$\mathbb{E}[Time^{\text{inside}}_{\gamma,D}] = \sum_{\pi \in Trips^{\gamma}_{D}} p(\pi) \cdot \ell(\pi \upharpoonright \gamma)$$

$$\mathbb{E}[Time^{\text{outside}}_{\gamma,D}] = \sum_{\pi \in Trips^{\gamma}_{D}} p(\pi) \cdot \sum_{i \in [|\Gamma|]} \ell(\pi \upharpoonright i)$$

Thus $\mathbb{E}[Time^{\text{inside}}_{\gamma,D}]$ is the expected length of preruns obtained by restricting sequences from $Trips^{\gamma}_{D}$ to the horizontal transitions inside the subject node, and up_{γ} and *down* transition from the subject node; and $\mathbb{E}[Time^{\text{outside}}_{\gamma,D}]$ is the expected length of preruns $\pi \upharpoonright i \in Trips^{\gamma_i}_{D_i}$, where $(D_1, \cdots, D_{|\Gamma|})$ is the decomposition interface of π.

As $Time_{\gamma,D} = Time^{\text{inside}}_{\gamma,D} + Time^{\text{outside}}_{\gamma,D}$, by the linearity of expectation, we have $\mathbb{E}[Time_{\gamma,D} \mid \mathcal{D}_{\gamma,D} = \vec{D}] = \mathbb{E}[Time^{\text{inside}}_{\gamma,D} \mid \mathcal{D}_{\gamma,D} = \vec{D}] + \mathbb{E}[Time^{\text{outside}}_{\gamma,D} \mid \mathcal{D}_{\gamma,D} = \vec{D}]$.

Lemma 4.7. *Recall that $mht^{\mathcal{D}}_{\gamma,D}$ is the mean hitting time of EndStates (taking only flagged transitions into account). Suppose $hp^{\vec{D}}_{\gamma,D} > 0$. Then*

1. $\mathbb{E}[Time^{inside}_{\gamma,D} \mid \mathcal{D}_{\gamma,D} = \vec{D}] = \dfrac{mht^{\vec{D}}_{\gamma,D}}{hp^{\vec{D}}_{\gamma,D}}$
2. $\mathbb{E}[Time^{outside}_{\gamma,D} \mid \mathcal{D}_{\gamma,D} = \vec{D}] = \sum_{i \in [|\Gamma|]} \mathbb{E}[Time_{\gamma_i,D_i}]$

To sum up:

$$\mathbb{E}[Time_{\gamma,D}] = \sum_{\vec{D}: hp^{\vec{D}}_{\gamma,D} > 0} X\,Y \tag{4}$$

where

$$X = \frac{mht^{\vec{D}}_{\gamma,D}}{hp^{\vec{D}}_{\gamma,D}} + \sum_{i \in [|\Gamma|]} \mathbb{E}[Time_{\gamma_i,D_i}]$$

$$Y = \frac{\prod_{i \in [|\Gamma|]} wt(Trips^{\gamma_i}_{D_i}) \cdot hp^{\vec{D}}_{\gamma,D}}{wt(Trips^{\gamma}_{D})}$$

Let $Time'_{\gamma,D} := \mathbb{E}[Time_{\gamma_i,D_i}] \cdot wt(Trips^{\gamma}_{D})$. Equation (4) then implies

$$Time'_{\gamma,D} \tag{5}$$
$$= \sum_{\substack{\vec{D} \\ hp^{\vec{D}}_{\gamma,D} > 0}} \left(mht^{\vec{D}}_{\gamma,D} + \sum_{i \in [|\Gamma|]} \left(Time'_{\gamma_i,D_i} \cdot \prod_{\substack{i \in [|\Gamma|] \\ j \ne i}} wt(Trips^{\gamma_i}_{D_i}) \cdot hp^{\vec{D}}_{\gamma,D} \right) \right)$$

Observe that $ett(\mathcal{P}) = \sum_{q \in Q} Time'_{\gamma_{init},[q_{init},q]}$.

Using Equations (1) and (5), one can now define an operator $G : \mathbb{R}^n_\infty \to \mathbb{R}^n_\infty$ that combines equations for $Trips^{\gamma}_{D}$ and $Time'_{\gamma,D}$. We have $n \le 2\sum_{i=0}^{k}(|Q|^{2i}|\Gamma|)$ (recall that we discard γ, D with $wt(Trips^{\gamma}_{D}) = 0$). Note that G will consist of multivariate polynomials with positive coefficients, i.e. it will be monotone and, thus, have a non-negative least fixpoint $\vec{x}$ in $\mathbb{R}^n_\infty$ [27]. To make sure that the least fixpoint contains ∞ only if $ett(\mathcal{P}) = \infty$, we will discard equations for variables $x_{Time'_{\gamma,D}}$ that are irrelevant to $ett(\mathcal{P})$, i.e. those that no $Time'_{\gamma_{init},[q_{init},q]}$ depends on according to Equation (5) (this can be determined statically by analysing dependencies in Equation (5)).

Lemma 4.8. *$(G^h(\vec{0}))_{Trips^{\gamma}_{D}}$ and $(G^h(\vec{0}))_{Time'_{\gamma,D}}$ correspond to $wt(Trips^{\gamma}_{D})$ and $Time'_{\gamma,D}$ calculated using sequences from $Trips^{\gamma}_{D}$ that can visit up to h levels in the tree stack. Hence, $ett(\mathcal{P}) = \sum_{q \in Q} LFP(G)_{Time'_{\gamma,[q_{init},q]}}$.*

Then the PAST problem can be solved in EToR by a query that simply asks about the existence of a fixpoint of G. The threshold problems for $\sim \in \{<, \le\}$ can be solved similarly by asking about the existence of a fixpoint $\vec{x}$ and checking $\sum_{q \in Q} x_{Time'_{\gamma_{init},[q_{init},q]}} \sim r$. For $\sim \in \{>, \ge\}$, we can appeal to the complement problem and then reverse the answer.

By inspecting the shape of the equations involved (similarly to the previous analysis for $Trips^{\gamma}_{D}$) one can confirm that the final EToR formula is of exponential size. Note that in order to prepare the final EToR query corresponding to G, we need to perform dependency/reachability analysis on Equations (1) and (5), which requires exponential time. Overall, this implies solvability in exponential space.

Theorem 4.9. *For any rPTSA $\mathcal{P}$, the PAST problem and threshold problems for $ett(\mathcal{P})$ are decidable and in EXPSPACE.*

5 MODEL CHECKING ω-REGULAR PROPERTIES

Let $\mathcal{P} = \langle Q, \mathcal{M}, \Gamma, \Delta, q_{init}, m_{init}, \gamma_{init} \rangle$ be an rPTSA. Our earlier analyses all concerned finite runs. In this section we investigate infinite ones. In order to measure the probability of sets containing such runs, we follow the standard approach based on deriving σ-algebras from a Markov chain [1]. In our case, this is the Markov chain on configurations of $\mathcal{P}$ implied by Δ, in which terminating configurations are assumed to be absorbing states. We use deterministic Rabin automata as the specification formalism.

Definition 5.1. A *deterministic Rabin automaton* (DRA) is a tuple $\mathcal{U} = \langle U, \Sigma, \to, u_{init}, R \rangle$,where U is a finite set of control states, Σ is a finite input alphabet, $\to \subseteq U \times \Sigma \times U$ is a deterministic and total transition relation, $u_{init} \in U$ is an initial state and $R = \{(E_1, F_1), \cdots, (E_n, F_n)\}$ is a Rabin acceptance condition, where $E_i, F_i \subseteq U$.

Let $\boldsymbol{x} = x_0 x_1 \cdots \in \Sigma^\omega$. A computation of $\mathcal{U}$ over w is an infinite sequence $\boldsymbol{u} = u_0 u_1 \cdots \in U^\omega$ such that $u_0 = u_{init}$, $(u_i, x_i, u_{i+1}) \in \to$ for all $i \ge 0$. Let $Inf(\boldsymbol{u}) \subseteq U$ be the set of states that appear infinitely often in $\boldsymbol{u}$. We say that $\boldsymbol{x}$ is accepted by $\mathcal{U}$ if there exists a computation $\boldsymbol{u}$ of $\mathcal{U}$ over $\boldsymbol{x}$ such that $Inf(\boldsymbol{u}) \cap E_i = \emptyset$ and $Inf(\boldsymbol{u}) \cap F_i \neq \emptyset$ for some i that $1 \leq i \leq n$.

To specify properties of infinite runs, we use DRA over the input alphabet $\Sigma = Q \times \Gamma$. We will assume that DRA process sequences of (non-terminating) configurations by reading the state and the current stack symbol of each configuration. To make sure that the current stack symbol is always available, we assume that $\mathcal{P}$ generates infinite sequences almost surely, i.e. $wt(Ter^{\gamma_{init}}_{q_{init}}) = 0$. Let $Accept_{\mathcal{P},\mathcal{U}}$ be the set of infinite runs from generated by $\mathcal{P}$ that start in c_{init} and are accepted by $\mathcal{U}$. We will consider the following verification problems.

- *Qualitative*: given an rPTSA $\mathcal{P}$ and a DRA $\mathcal{U}$ over $Q \times \Gamma$, is it true that $P(Accept_{\mathcal{P},\mathcal{U}}) = r$ for $r \in \{0, 1\}$?
- *Quantitative*: given a rational r, does $P(Accept_{\mathcal{P},\mathcal{U}}) \sim r$ hold, where $\sim \in \{<, \leq, >, \geq\}$?

Both will turn out to be decidable.

5.1 Automata constructions

As a first step towards a decision procedure, we apply a product construction to $\mathcal{P}$ and $\mathcal{U}$, to obtain another rPTSA, called $\mathcal{P}_{\mathcal{U}}$. The rPTSA operates in the same way as $\mathcal{P}$, running $\mathcal{U}$ concurrently. Accordingly, its set of states is $Q \times U$, whereas the stack alphabet Γ remains the same.

To analyze infinite runs of an rPTSA, it is convenient to assume that it is given in a normal form, in which there are no horizontal transitions. Due to the restriction condition, only a finite number of transitions can then occur in any node and the restriction of an infinite run to any node will be finite. This will simplify the decomposition of infinite runs in what follows.

The construction that eliminates horizontal transitions replaces each of them with an $up_\top$ ($\top \notin \Gamma$) and records information about the current state, memory current stack symbol using new states of the form $\ulcorner((q, u), m, \gamma)\urcorner$. This continues until a non-horizontal transition, *op* (say), is to be fired, in which case the automaton will return to the original node via a cascade of *down* operations that remember—in new state q_{op}—what non-horizontal transition should be executed, once we reach the original node. Because of the restriction condition the number of $up_\top$ operations executed in a single node will also be restricted.

More formally, we add $\top$ to Γ and modify the rules of $\mathcal{P}_{\mathcal{U}}$ as follows.

- Any transition of the form $((q, u), m, \gamma, (q', u'), p)$ is *replaced* by $((q, u), m, \gamma, (\ulcorner((q', u'), m, \gamma)\urcorner, m, up_\top), p)$ and $(\ulcorner((q, u), m, \gamma)\urcorner, m_{init}, \top, (\ulcorner((q', u'), m, \gamma)\urcorner, m_{init}, up_\top), p)$.
- For any transition $((q, u), m, \gamma, op, p)$, where *op* is non-horizontal, *add* $(\ulcorner((q, u), m, \gamma)\urcorner, m_{init}, \top, (q_{op}, m_{init}, down), p)$, $(q_{op}, m_{init}, \top, (q_{op}, m_{init}, down), 1)$ and $(q_{op}, m, \gamma, op, 1)$.

We refer to the normalised automaton as $\mathcal{P}^{norm}_{\mathcal{U}}$. Note that $P(Accept_{\mathcal{P},\mathcal{U}})$ now corresponds to the probability of infinite runs of $\mathcal{P}^{norm}_{\mathcal{U}}$ in which, for some $1 \leq i \leq n$, all states from E_i occur finitely often, and some state from F_i does so infinitely often. For the purpose of identifying occurrences of a $\mathcal{U}$-state u in the states of $\mathcal{P}^{norm}_{\mathcal{U}}$, we deem u to occur not only in the original states of the form (q, u), but also in states of the form $\ulcorner((q, u), m, \gamma)\urcorner$ as well as q_{op} (provided u occurs in *op*).

5.2 Decomposition equation

Here we discuss how to decompose infinite runs. Recall that in the decomposition equation for $Trips^{\gamma}_{D}$, $|D|$ was even, because we considered scenarios in which each *up* had a matching *down*. When working with infinite runs, we will also need to consider scenarios with unmatched *up*'s.

Definition 5.2. Suppose $D = [q_1, \cdots, q_{2i}, q_{2i+1}]$ and $i \geq 0$.

- For $i = 0$, we let $Up^{\gamma}_{[q_1]}$ be the set of infinite runs π from the configuration $(q_1, \{(\gamma, m_{init})\}, \gamma)$.
- For $i > 0$, we let Up^{γ}_{D} consist of sequences $\xi\pi$, where $\xi \in Trips^{\gamma}_{[q_1,\cdots,q_{2i}]}$ and π is an infinite run such that if ξ ends in $(q_{2i}, t_{i+1}, \epsilon)$ then π begins from $c_\xi = (q_{2i+1}, t_{i+1}, \gamma)$. Writing R^{inf}_{ξ} for the set of infinite runs starting at c_ξ, we define $wt(Up^{\gamma}_{D}) := \sum_{\xi \in Trips^{\gamma}_{D}} (wt(\xi) \cdot wt(R^{inf}_{\xi}))$.

Overall, Up^{γ}_{D} captures the history of a node that has been visited $i + 1$ times (from below), and ultimately left via one of its children, along with the history of all nodes above it.

Lemma 5.3 (Subtraction). *Let $i \geq 0$, $D = [q_1, \cdots, q_{2i}]$ and $q_{2i+1} \in Q$. Then*

$$wt(Up^{\gamma}_{D +\!\!+ [q_{2i+1}]}) = wt(Trips^{\gamma}_{D}) - \sum_{q \in Q} wt(Trips^{\gamma}_{D +\!\!+ [q_{2i+1}, q]})$$

where $+\!\!+$ stands for list concatenation. For $i = 0$, we assume $Trips^{\gamma}_{[]} = \{\epsilon\}$ and $wt(\epsilon) = 1$.

Proof. Suppose $\xi \in Trips^{\gamma}_{D}$. Let $c_\xi := (q_{2+1}, t_{i+1}, \gamma)$ if ξ ends with $(q_{2i}, t_{i+1}, \epsilon)$. Let R^{fin}_{ξ} be the set of terminating runs from c_ξ (i.e. finite runs from c_ξ ending in a configuration with empty stack pointer). Observe that $wt(R^{fin}_{\xi}) + wt(R^{inf}_{\xi}) = 1$. Then we have

$$\begin{aligned}
&wt(Trips^{\gamma}_{D}) \\
&= \sum_{\xi \in Trips^{\gamma}_{D}} wt(\xi) = \sum_{\xi \in Trips^{\gamma}_{D}} wt(\xi)(wt(R^{fin}_{\xi}) + wt(R^{inf}_{\xi})) \\
&= \sum_{\xi \in Trips^{\gamma}_{D}} (wt(\xi) wt(R^{fin}_{\xi})) + \sum_{\xi \in Trips^{\gamma}_{D}} (wt(\xi) wt(R^{inf}_{\xi})) \\
&= \sum_{q} \sum_{\xi \in Trips^{\gamma}_{D +\!\!+ [q_{2i+1}, q]}} wt(\xi) + wt(Up^{\gamma}_{D +\!\!+ [q_{2i+1}]}) \\
&= \sum_{q} wt(Trips^{\gamma}_{D +\!\!+ [q_{2i+1}, q]}) + wt(Up^{\gamma}_{D +\!\!+ [q_{2i+1}]})
\end{aligned}$$

□

Note that our assumption $wt(Ter^{\gamma_{init}}_{q_{init}}) = 0$ is equivalent to $wt(Up^{\gamma_{init}}_{[q_{init}]}) = 1$.

A typical sequence from Up^{γ}_{D} will enter the subject node, make several return trips to all but one child and then leave the node by moving on to that child and never returning again. This is captured by the following.

Decomposition formula 2: Up^{γ}_{D}

$$wt(Up^{\gamma}_{D}) = \sum_{\mathcal{D}} \Big(\big(\prod_{i \in [|\Gamma|]} wt(TripsOrUp^{\gamma_i}_{\mathcal{D}_i}) \big) \cdot \sum_{\pi \in SynP^{\mathcal{D}}_{\gamma,D}} wt(\pi) \Big) \tag{6}$$

where $\mathcal{D}$ ranges over $(\mathcal{D}_1, \cdots, \mathcal{D}_{|\Gamma|})$ such that all but one $\mathcal{D}_i$ are of even length and

$$TripsOrUp^{\gamma_i}_{\mathcal{D}_i} = \begin{cases} Trips^{\gamma_i}_{\mathcal{D}_i} & |\mathcal{D}_i| \text{ is even} \\ Up^{\gamma_i}_{\mathcal{D}_i} & |\mathcal{D}_i| \text{ is odd} \end{cases}$$

In this case $SynP^{\mathcal{D}}_{\gamma,D}$ ranges over finitely many finite trajectories that reach $(q, m, (|D|, |D_{\gamma_1}|, \cdots, |D_{\gamma_{|\Gamma|}}|))$, where q is the last element of the unique odd-length $\mathcal{D}_i$ (this is the node to which the trace moves eventually).

5.3 Auxiliary Markov chain

Next we construct a Markov chain that will capture dependencies between the sets Up^{γ}_{D}. Its states will be of the form Up^{γ}_{D}, where $wt(Up^{\gamma}_{D}) > 0$. Similarly, below we only mean to refer to $Trips^{\gamma}_{D}$ such that $wt(Trips^{\gamma}_{D}) > 0$. Such cases are decidable, thanks to the results in Section 3. We do this to make sure that the probabilities involved are non-zero. We set $Up^{\gamma}_{D} \xrightarrow{p} Up^{\gamma_i}_{D_i}$, where p is taken to be

$$\frac{\sum_{\substack{\mathcal{D} \\ \mathcal{D}_i = D_i}} \Big(\big(\prod_{j \neq i} wt(Trips^{\gamma_k}_{\mathcal{D}_j}) \big) \cdot \sum_{\pi \in SynP^{\mathcal{D}}_{\gamma,D}} wt(\pi) \Big) \cdot wt(Up^{\gamma_i}_{D_i})}{wt(Up^{\gamma}_{D})}.$$

This expresses the conditional probability that an infinite trace, on entering a γ-node, will leave a "footprint" D_i in γ_i (in particular it will leave the node via γ_i) under the condition that its γ "footprint" is D. We obtain a Markov chain this way, because when an infinite trace starts a γ-node leaving an odd-length footprint D, it must produce an odd-length footprint in one of its upper nodes. Consequently, the weights of outgoing transitions from Up^{γ}_{D} add up to 1.

5.4 Strongly connected components

Let C be a bottom strongly connected component (BSCC) of the Markov chain defined above. Note that, in order to determine such components, we can ignore probabilities and only use the underlying directed graph. We are interested in BSCCs, because this is where infinite runs will (almost surely) end up. Consequently, the BSCC where an infinite run ends up determines the acceptance status of that run.

Recall that all states of C have the form Up^{γ}_{D}. Pick Up^{γ}_{D} from C. We will be interested in computing the set C_{inf} of states of U that occur inside sequences from Up^{γ}_{D} (recall that states of the converted rPTSA $\mathcal{P}^{norm}_{\mathcal{U}}$ contain occurrences of states from U, which were initially introduced by the product construction and then preserved by the normalisation routine).

C_{inf} can be calculated via reachability analysis on the decomposition equations for *Up* and *Trips*: we need to record all the states from U occurring on the relevant synchronising paths and those corresponding to the associated *Trips*- and *Up*-sets. The latter can be calculated recursively using the decomposition equations. We stop the recursion when we encounter the same set in a recursive branch, as no new states can be found by inspecting the same equations. A BSCC C is accepting if $C_{inf} \cap E_i = \emptyset$ and $C_{inf} \cap F_i \neq \emptyset$ for some i.

After accepting BSCC have been identified, calculating $P(Accept_{\mathcal{P},\mathcal{U}})$ boils down to calculating the probability of reaching such an accepting BSCC from $Up^{\gamma_{init}}_{[(q_{init}, u_{init})]}$ in the Markov chain defined in Section 5.3. This can be done by solving a system of linear equations, as the relevant probability corresponds to the probability of hitting states from good BSCCs.

5.5 Solution via EToR

Ultimately, we will solve both the qualitative and quantitative model-checking problems via an EToR query. In earlier sections, it sufficed to build EToR queries that referred to any fixpoint of the system of questions defining $wt(Trips^{\gamma}_{\mathcal{D}})$. However, in order to express the above methodology in EToR, it is necessary to specify the exact values of $wt(Trips^{\gamma}_{\mathcal{D}})$ (i.e. the LFP) in EToR, so that we can construct the directed graph behind the auxiliary Markov chain, identify the relevant BSCCs and be able to specify the weights from the auxiliary Markov in EToR. It is clear that the LFP can be expressed in the first-order theory of the reals but, to do the same in EToR, we will augment the system of equations with additional constraints, which will deliver the LFP as the *unique* fixpoint. To this end, we follow ideas from [4, 9] and, in addition to fixpoint equations for x, include constraints of the form $x_{Trips^{\gamma}_{D}} = 0$ or $x_{Trips^{\gamma}_{D}} = 1$ or $0 < x_{Trips^{\gamma}_{D}} < 1$, depending on whether $wt(Trips^{\gamma}_{D}) = 0$, $wt(Trips^{\gamma}_{D}) = 1$ or $0 < wt(Trips^{\gamma}_{D}) < 1$ respectively. Note that we can determine which is the case for any given γ, D using techniques from Section 3, i.e. auxiliary EToR queries.

Theorem 5.4. *Both the qualitative and quantitative DRA model-checking problems for rPTSA are decidable.*

Proof. To solve both problems, we start off with the automata-theoretic constructions. Then, using the methodology of Sections 3 and 4 (decidable EToR queries), one can effectively construct the directed graph underlying the auxiliary Markov chain (Section 5.3). Using standard graph-theoretic algorithms, one can then decompose it into strongly connected components and identify the bottom ones. For each BSCC C, we then verify whether it is accepting with the help of the decomposition formula, as discussed above. Next the probability of reaching an accepting BSCC can be specified in EToR as a solution to a system of linear equations, since the coefficients (probabilities from the auxiliary Markov chain) can be. This is because they are arithmetic expressions built from either $wt(Trips^{\gamma}_{D})$ (whose specification is discussed above), $wt(Up^{\gamma}_{D})$ (which can be derived from $wt(Trips^{\gamma}_{D})$'s) or hitting probabilities from the synchronising Markov chain, which is finite. Finally, both the qualitative and quantitative verification problems can be specified in EToR. □

One can show that the final EToR query will be of exponential size and that it can be produced using exponential space, due to the auxiliary EToR queries and graph construction. This implies membership in EXPSPACE.

Theorem 5.5. *The qualitative and quantitative DRA model-checking problems for rPTSA are in EXPSPACE.*

6 PROBABILISTIC ADDITIVE HORS (PAHORS)

In this section we return to PAHORS, show how to relate them to rPTSA, which will enable us to derive decidability results for PAHORS.

Let $\mathcal{G} = \langle \mathcal{N}, \mathcal{R}, S\rangle$ be a PAHORS. The connection to rPTSA is based on a relationship between the deterministic version of rPTSA and a class of recursion schemes called MAHORS [6]. MAHORS are tuples $\langle \Sigma, \mathcal{N}, \mathcal{R}, S\rangle$, where Σ is a finite alphabet of terminals $a : o^n \multimap o$ and $\mathcal{N}, \mathcal{R}, S$ are defined as for PAHORS except that probabilistic choice is disallowed. S is then rewritten using (unweighted versions of) rules for unfolding non-terminals and projecting, which can be applied in contexts of the form

$$C ::= [\,] \mid a\langle t_1, \cdots, t_i, C, t_{i+1}, \cdots, t_n\rangle.$$

Thus, MAHORS generate a potentially infinite tree over a finite ranked alphabet of terminals, whereas PAHORS do not generate any structure at all (indeed termination is the only thing observable about PAHORS computation). In what follows we recast the termination probability of a PAHORS as a summation over terminating branches of a tree generated by a MAHORS.

6.1 From PAHORS to MAHORS

The translation is based on replacing probabilistic branching $\oplus_p$ with a terminal $a_p : o\&o \multimap o$. Let $\mathbb{Q}_{\mathcal{G}}$ be the set of rational numbers that occur in the rules of $\mathcal{G}$. Define the alphabet $\Sigma_{\mathcal{G}}$ to consist of terminals $a_p : o\&o \multimap o$ for each $p \in \mathbb{Q}_{\mathcal{G}}$. We write $\bar{t}$ for the term t in which each subterm of the form $u_1 \oplus_p u_2$ is replaced by $a_p\langle u_1, u_2\rangle$ inductively. Let $\overline{\mathcal{R}}(F)$ be $\overline{\mathcal{R}(F)}$. Then $\overline{\mathcal{G}} = \langle \Sigma_{\mathcal{G}} \cup \{\top\}, \mathcal{N}, \overline{\mathcal{R}}, S\rangle$ is a MAHORS. Accordingly, one can view $\mathcal{T}_{\overline{\mathcal{G}}}$ as a partial function from $\{L, R\}^*$ to $\Sigma_{\mathcal{G}} \cup \{\top\}$ with a prefix-closed domain. Let $Top(\mathcal{T}_{\overline{\mathcal{G}}}) = \{\pi \in \{L, R\}^* \mid \mathcal{T}_{\overline{\mathcal{G}}}(\pi) = \top\}$. Given $\pi = \ell_1 \cdots \ell_k \in Top(\mathcal{T}_{\overline{\mathcal{G}}})$, with $\ell_i \in \{L, R\}$, let $a_{p_1} \cdots a_{p_k}\top$ be the corresponding branch of $\mathcal{T}_{\mathcal{G}'}$. Let the corresponding weight $\mathbb{P}(\overline{\mathcal{G}}, \pi)$ be $\prod_{i=1}^{k} p'_i$, where $p'_i = p_i$ (if $\ell_i = L$) and $p'_i = 1 - p_i$ (if $\ell_i = R$). Notice that there is a 1-1 correspondence between maximal reduction sequences of the PAHORS $\mathcal{G}$ and maximal branches of the tree $\mathcal{T}_{\overline{\mathcal{G}}}$. Moreover we have $\mathbb{P}(\mathcal{G}) = \sum_{\pi \in Top(\mathcal{T}_{\overline{\mathcal{G}}})} \mathbb{P}(\overline{\mathcal{G}}, \pi)$.

Lemma 6.1. *For any PAHORS $\mathcal{G}$, there exists an rPTSA whose termination probability is $\mathbb{P}(\mathcal{G})$.*

Proof. We take advantage of the correspondence between PAHORS and MAHORS. Consider the MAHORS $\overline{\mathcal{G}}$. In [6], it was shown how to construct a restricted *tree-generating* tree-stack automaton that generates the same tree as $\overline{\mathcal{G}}$. In particular, the automaton will have transitions of the form $\Delta(q, m, \gamma) = a_p(q_1, q_2)$ and $\Delta(q, m, \gamma) = \top$ associated with terminals of $\overline{\mathcal{G}}$. To obtain an rPTSA, we replace $\Delta(q, m, \gamma) = a_p(q_1, q_2)$ with (q, m, γ, q_1, p) and $(q, m, \gamma, q_2, 1 - p)$. To handle $\Delta(q, m, \gamma) = \top$, we add a new state q_{term} and transitions that will trigger a cascade of *down*'s: $(q, m, \gamma, q_{term}, 1)$ and $(q_{term}, m', \gamma', (q_{term}, m', down), 1)$ for any stack symbol γ' and memory m'. Because $\mathbb{P}(\mathcal{G}) = \sum_{\pi \in Top(\mathcal{T}_{\overline{\mathcal{G}}})} \mathbb{P}(\overline{\mathcal{G}}, \pi)$, the resultant rPTSA has the same termination probability as $\mathbb{P}(\mathcal{G})$. □

The result makes it possible to apply techniques developed in Section 3 for rPTSA to PAHORS. In particular, we have:

Theorem 6.2. *The AST problem and the threshold problem for PAHORS are decidable.*

6.2 From PAHORS to MAHORS with steps

In order to investigate the expected time to termination for PAHORS, we will consider another conversion to MAHORS. In addition to generating a tree corresponding to the structure of probabilistic branching, it will also keep track of the number of reduction steps related to unfolding non-terminals and the projection rules. For short, we shall call them *u/p rules*.

This will be done with the help of a dedicated terminal $step : o \multimap o$. Given a term $\mathcal{N} \mid \emptyset \vdash t : o$, let $\hat{t}$ be t in which each subterm of the form $u_1 \oplus_p u_2$ is replaced with $a_p\langle u_1, u_2\rangle$ (as in $\bar{t}$) and each subterm of the form $\pi_i\, u$ is replaced with $step(\pi_i u)$ (all in an inductive manner).

Definition 6.3. Given a PAHORS $\mathcal{G} = \langle \mathcal{N}, \mathcal{R}, S\rangle$, let $\hat{\mathcal{G}} = \langle \Sigma_{\mathcal{G}} \cup \{\top, step\}, \hat{\mathcal{R}}, S\rangle$ be a MAHORS in which, given $\mathcal{R}(F) = \lambda x_1 \cdots x_k.t$, we set $\hat{\mathcal{R}}(F) = \lambda x_1 \cdots x_k.step(\hat{t})$.

Let us write $\longrightarrow_H$ for a u/p step of $\hat{\mathcal{G}}$ executed in an evaluation context H, defined by $H ::= [\,] \mid step\, H$. Then $\hat{\mathcal{G}}$ turns out to track the number of u/p steps in $\mathcal{G}$ in the following sense.

Lemma 6.4. *Suppose $\mathcal{G} = \langle \mathcal{N}, \mathcal{R}, S\rangle$ is a PAHORS and $\mathcal{N} \vdash t : o$. We have $t \xrightarrow{\epsilon,1}{}^{n} t'$ via u/p steps if and only if $\hat{t} \longrightarrow_H^n step^n(\hat{t'})$.*

Proof. Observe that $t \xrightarrow{\epsilon,1} t'$ via a u/p step if and only if $\hat{t} \longrightarrow_H step(\hat{t'})$. To conclude the Lemma, it suffices to iterate the above observation, because $\longrightarrow_H$ permits reductions under *step*. □

Lemma 6.4 shows that u/p steps in $\mathcal{G}$ are faithfully represented in $\hat{\mathcal{G}}$. In particular, an infinite sequence of u/p steps in $\mathcal{G}$ will generate infinitely many occurrences of *step*. On the other hand, if a sequence of u/p steps in $\mathcal{G}$ cannot be extended with another u/p step then the reduction must have encountered $\top$ or $u_1 \oplus_p u_2$. In this case, by Lemma 6.4, the corresponding $\longrightarrow_H^*$ reduction in $\hat{\mathcal{G}}$ will get stuck at $H[\top]$ or $H[a_p\langle \hat{u_1}, \hat{u_2}\rangle]$ respectively, for some context H. In the former case, both $\mathcal{G}$ and $\hat{\mathcal{G}}$ are stuck. In the latter case, in $\mathcal{G}$ the reduction will continue from u_1 or u_2. Correspondingly, in $\hat{\mathcal{G}}$ we can start reducing $\hat{u_1}$ or $\hat{u_2}$ (using $\longrightarrow_H$) respectively. Consequently, reduction sequences in $\mathcal{G}$ are in 1-1 correspondence with branches of $\hat{\mathcal{G}}$. We will rely on $\hat{\mathcal{G}}$ when calculating $ett(\mathcal{G})$. Note that the definitions of $ett(\cdot)$ are slightly different for rPTSA and PAHORS, but they coincide if $\mathbb{P}(\mathcal{G}) = 1$ (Remark 2.14). Recall that $\mathbb{P}(\mathcal{G}) < 1$ implies $ett(\mathcal{G}) = \infty$.

Lemma 6.5. *For any PAHORS $\mathcal{G}$ with $\mathbb{P}(\mathcal{G}) = 1$, there exists an rPTSA whose expected time to termination is $ett(\mathcal{G})$.*

Proof. This time, we exploit the MAHORS $\hat{\mathcal{G}}$. Via [6], we can obtain a restricted *tree-generating* tree-stack automaton that generates the same tree as $\hat{\mathcal{G}}$. In particular, the automaton will have transitions of the form $\Delta(q,m,\gamma) = \mathit{step}(q')$, $\Delta(q,m,\gamma) = a_p(q_1,q_2)$ and $\Delta(q,m,\gamma) = \top$.

To obtain an rPTSA, we replace them with horizontal transitions. To handle $\Delta(q,m,\gamma) = \mathit{step}(q')$, we add $(q,m,\gamma,q',1)$. For $\Delta(q,m,\gamma) = a_p(q_1,q_2)$, we add (q,m,γ,q_1,p) and $(q,m,\gamma,q_2,1-p)$; for $\Delta(q,m,\gamma) = \top$, we add a new state q_{term} along with $(q,m,\gamma,q_{term},1)$ and $(q_{term},m',\gamma',(q_{term},m',\mathit{down}),1)$ for any stack symbol γ' and memory m'.

Note that all terminals have been replaced with horizontal transitions but only the first two cases should be flagged as those to be counted. Note also that the extra *down* transitions related to termination will not be counted. By Lemma 6.4 and the following discussion, the expected time to termination in the resultant rPTSA will coincide with $\mathit{ett}(\mathcal{G})$. □

The above result enables us to bring the results developed in Section 4 to bear on PAHORS. Recall that before referring to rPTSA, it is necessary to check $\mathbb{P}(\mathcal{G}) = 1$, which can be done using Theorem 6.2. If $\mathbb{P}(\mathcal{G}) < 1$, $\mathcal{G}$ can be classified immediately as a negative instance of PAST.

Theorem 6.6. *The PAST problem for PAHORS is decidable.*

Using similar conversions to the above-mentioned PAHORS-to-rPTSA translations, one can also deploy our results on DRA model-checking (Section 5) in the setting of PAHORS. In order to specify interesting properties of PAHORS, we can reintroduce terminals (in the spirit of *step*), which will map to special states using the translation from [6]. One can then use these states in DRA specifications. In turn, access to Γ gives DRA the ability to talk about the current active procedure (non-terminal).

The above translations to rPTSA can be implemented in polynomial time, because the underpinning translation in the tree-generating case has this complexity [6, Theorem 7]. Also, the restriction parameter k is polynomial in the size of $\mathcal{G}$. Consequently, the exponential-space complexity for rPTSA (wrt automaton size and k) extends to decision problems for PAHORS.

Theorem 6.7. *The AST, PAST and threshold problems for PAHORS are in EXPSPACE.*

7 CONCLUSION

To our knowledge, PAHORS is the first non-trivial class of higher-order programs with decidable AST and PAST problems. What is more, we have developed decision procedures that help analyse the associated termination probability, expected termination time, and omega-regular properties. They can be reduced to statements in the existential fragment of the first-order theory of the reals.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for helpful comments and suggestions. This work was supported by the Engineering and Physical Sciences Research Council (EP/T006579/1).

REFERENCES

[1] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking.* MIT Press.

[2] Raven Beutner and Luke Ong. 2021. On Probabilistic Termination of Functional Programs with Continuous Distributions. In *Proceedings of the 42st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2021.*

[3] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. 2019. Pyro: Deep Universal Probabilistic Programming. *J. Mach. Learn. Res.* 20 (2019), 28:1–28:6.

[4] Tomás Brázdil, Javier Esparza, Stefan Kiefer, and Antonín Kucera. 2013. Analyzing probabilistic pushdown automata. *Formal Methods Syst. Des.* 43, 2 (2013), 124–163.

[5] John F. Canny. 1988. Some Algebraic and Geometric Computations in PSPACE. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, Janos Simon (Ed.). ACM, 460–467.

[6] Pierre Clairambault and Andrzej S. Murawski. 2019. On the expressivity of linear recursion schemes. In *Proceedings of 44th International Symposium on Mathematical Foundations of Computer Science (MFCS) (LIPIcs, Vol. 138).* Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 50:1–50:14.

[7] Marco F. Cusumano-Towner, Feras A. Saad, Alexander K. Lew, and Vikash K. Mansinghka. 2019. Gen: a general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22-26, 2019.* ACM, 221–236.

[8] Tobias Denkinger. 2016. An Automata Characterisation for Multiple Context-Free Languages. In *Developments in Language Theory - 20th International Conference, DLT 2016, Montréal, Canada, July 25-28, 2016, Proceedings (Lecture Notes in Computer Science, Vol. 9840)*, Srecko Brlek and Christophe Reutenauer (Eds.). Springer, 138–150.

[9] Kousha Etessami and Mihalis Yannakakis. 2005. Algorithmic verification of recursive probabilistic state machines. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems.* Springer, 253–270.

[10] Hong Ge, Kai Xu, and Zoubin Ghahramani. 2018. Turing: A Language for Flexible Probabilistic Inference. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 84).* PMLR, 1682–1690.

[11] Jean-Yves Girard. 1987. Linear Logic. *Theoretical Computer Science* 50 (1987), 1–102.

[12] Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Keith Bonawitz, and Joshua B. Tenenbaum. 2008. Church: a language for generative models. In *UAI 2008, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence, Helsinki, Finland, July 9-12, 2008.* AUAI Press, 220–229.

[13] Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. 2014. Probabilistic programming. In *Proceedings of the on Future of Software Engineering, FOSE 2014, Hyderabad, India, May 31 - June 7, 2014.* ACM, 167–181.

[14] Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong, and Olivier Serre. 2008. Collapsible Pushdown Automata and Recursion Schemes. In *Proceedings of IEEE Symposium on Logic in Computer Science.* Computer Society Press, 452–461.

[15] Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2019. On the hardness of analyzing probabilistic programs. *Acta Informatica* 56, 3 (2019), 255–285.

[16] Naoki Kobayashi. 2009. Types and higher-order recursion schemes for verification of higher-order programs. In *Proceedings of POPL.* 416–428.

[17] Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. 2020. On the Termination Problem for Probabilistic Higher-Order Recursive Programs. *Log. Methods Comput. Sci.* 16, 4 (2020). https://lmcs.episciences.org/6817

[18] Ugo Dal Lago, Claudia Faggian, and Simona Ronchi Della Rocca. 2021. Intersection types and (positive) almost-sure termination. *Proc. ACM Program. Lang.* 5, POPL (2021), 1–32. https://doi.org/10.1145/3434313

[19] Carol Mak, C.-H. Luke Ong, Hugo Paquet, and Dominik Wagner. 2021. Densities of Almost Surely Terminating Probabilistic Programs are Differentiable Almost Everywhere. In *Programming Languages and Systems - 30th European Symposium on Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12648)*, Nobuko Yoshida (Ed.). Springer, 432–461.

[20] James R. Norris. 1998. *Markov Chains.* Cambridge University Press.

[21] C.-H. Luke Ong. 2006. On Model-Checking Trees Generated by Higher-Order Recursion Schemes. In *Proceedings of LICS.* Computer Society Press, 81–90.

[22] Michael O. Rabin. 1976. Probabilistic algorithms. In *Algorithms and Complexity: New Directions and Results.* Academic Press, 21–39.

[23] Tom Rainforth. 2017. *Automating Inference, Learning, and Design Using Probabilistic Programming.* Ph. D. Dissertation. University of Oxford.

[24] Sylvain Salvati. 2015. MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is captured by the IO and the OI hierarchies. *J. Comput. Syst. Sci.* 81, 7 (2015), 1252–1277. https://doi.org/10.1016/j.jcss.2015.03.004

[25] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On Multiple Context-Free Grammars. *Theor. Comput. Sci.* 88, 2 (1991), 191–229.
[26] Alfred Tarski. 1948. *A Decision Method for Elementary Algebra and Geometry.* Technical Report. RAND Corporation, Santa Monica, California. 63 pp. pages.
[27] Alfred Tarski. 1955. A Lattice-theoretical Fixpoint Theorem and its Applications. *Pacific J. Mathematics* 5 (1955), 285–309.
[28] David Tolpin, Jan-Willem van de Meent, and Frank D. Wood. 2015. Probabilistic Programming in Anglican. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 9286).* Springer, 308–311.
[29] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja R. Rudolph, Dawen Liang, and David M. Blei. 2016. Edward: A library for probabilistic modeling, inference, and criticism. *CoRR* abs/1610.09787 (2016).
[30] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. 2018. An Introduction to Probabilistic Programming. *CoRR* abs/1809.10756 (2018).