Logical Methods in Computer Science
Volume 19, Issue 4, 2023, pp. 24:1–24:30
https://lmcs.episciences.org/

Submitted Sep. 12, 2022
Published Dec. 15, 2023

MODEL CHECKING TEMPORAL PROPERTIES OF RECURSIVE PROBABILISTIC PROGRAMS

TOBIAS WINKLER, CHRISTINA GEHNEN, AND JOOST-PIETER KATOEN

RWTH Aachen University, Aachen, Germany
e-mail address: {tobias.winkler,christina.gehnen,katoen}@cs.rwth-aachen.de

Abstract. Probabilistic pushdown automata (pPDA) are a standard operational model for programming languages involving discrete random choices and recursive procedures. Temporal properties are useful for specifying the chronological order of events during program execution. Existing approaches for model checking pPDA against temporal properties have focused mostly on ω-regular and LTL properties. In this paper, we give decidability and complexity results for the model checking problem of pPDA against ω-visibly pushdown languages that can be described by specification logics such as CaRet. These logical formulae allow specifying properties that explicitly take the structured computations arising from procedural programs into account. For example, CaRet is able to match procedure calls with their corresponding future returns, and thus allows to express fundamental program properties such as total and partial correctness.

1. Introduction

Probabilistic programs extend traditional programs with the ability to flip coins or, more generally, sample values from probability distributions. These programs can be used to encode randomized algorithms and mechanisms in security [BKOB13] in a natural way. The interest in probabilistic programs has significantly increased in recent years. To a large extent, this is due to the search in AI for more expressive and succinct languages than probabilistic graphical models for Bayesian inference [GHNR14]. Probabilistic programs have many applications [vdMPYW18]. They are used in, among other areas, machine learning, systems biology, security, planning and control, quantum computing, and software–defined networks. Probabilistic variants of many programming languages exist.

Key words and phrases: Probabilistic Recursive Programs and Model Checking and Probabilistic Pushdown Automata and Visibly Pushdown Languages and CaRet.

* This is an extended version of a conference paper with the same title published at FoSSaCS 2022 [WGK22].

This work is supported by the DFG Research Training Group 2236 UnRAVeL and the ERC Advanced Grant 787914 FRAPPANT.

DOI:10.46298/LMCS-19(4:24)2023

```
proc void infectYoung() :
    y ≔ uniform(0, 3)
    repeat y times :
        infectYoung()
    e ≔ uniform(0, 2)
    repeat e times :
        f ≔ infectElder()
    return
```

```
proc bool infectElder() :
    y ≔ uniform(0, 1)
    repeat y times :
        infectYoung()
    e ≔ uniform(0, 4)
    repeat e times :
        infectElder()
    f ≔ bernoulli(0.01)
    return f
```

Figure 1: Recursive probabilistic program modeling the outbreak of an infectious disease. $\texttt{uniform}(a, b)$ stands for the *discrete* uniform distribution on $[a, b]$.

Recursion. *Procedural programs* allow for the declaration of procedures—small independent code blocks—and the ability to *call* procedures from one another, possibly in a recursive fashion. Most common programming languages such as C, Python, or Java support procedures. It is thus not surprising that recursion is also present in many modern probabilistic programming languages (PPL) such as WebPPL [GS14] or Church [SG12]. In fact, there have been numerous approaches to extend Bayesian networks with recursion even before PPL became popular [PK00, Jae01, CIRW11]. Randomized algorithms such as Hoare's quicksort (see, e.g., [Kar91]) with random pivot selection can be readily implemented using recursion. Finally, recursion in form of branching processes is an important tool to model reproduction of cells or molecules in systems biology [AK15].

Motivating example. This paper studies the automated verification of *probabilistic pushdown automata* (pPDA) [EKM04] as an explicit-state operational model of procedural probabilistic programs against temporal specifications. As a motivating example we consider a simple epidemiological model for the outbreak of an infectious disease in a large population where the number of susceptible individuals can be assumed to be infinite. Our example model distinguishes young and elderly persons. Each affected individual infects a uniformly distributed number of others, with varying rates (expected values) according to the age groups (Figure 2). The fatality rate for infected elderly and young persons is 1% and 0%, respectively. Procedure *infectElder*() returns a Boolean in order to signal to its callers whether the infection has led to fatality. Initially, we assume there is a single infected young person, i.e., the overall program is started by calling *infectYoung*(). It is an easy exercise to specify this model as a discrete probabilistic program with mutually recursive procedures (Figure 1). Note that this program can be easily amended to more realistic scenarios involving, e.g., more age or gender groups, hospitalization rate, etc.

	Y	E
Y	1.5	1
E	0.5	2

Figure 2: Example infection rates by age groups.

The behavior of such recursive probabilistic programs can be naturally described by pPDA. Roughly, the *local* states of the procedures—the values of the variables in the procedure's scope and the position of the program counter—constitute both the state space

and the stack alphabet of the automaton. Procedure calls correspond to push transitions in the pPDA in such a way that the program's *procedure stack* is simulated by the automaton's pushdown stack, i.e., the caller's current local state is saved on top of the stack. Accordingly, *returning* from a procedure corresponds to taking a pop transition in order to restore the local state of the caller. Returning a value can be handled similarly. Clearly, if the reachable local state spaces of the involved procedures are finite, then the resulting automaton will be finite as well. We refer to [ABE18] for more details.

A number of natural questions such as "Will the virus eventually become extinct?" (termination probability) or "What is the expected number of fatalities?" (expected costs) are decidable for finite pPDA (see [BEKK13] for a survey). In this work, we focus on *temporal properties*, i.e., questions that involve reasoning about the chronological order of events during the epidemic. An example are chains of infection: For instance, we might ask

> *What's the probability that eventually a young person with only young persons in their chain of infection passes the virus on to an elderly person who then dies?*

On the level of the program in Figure 1, this corresponds to the probability of reaching a *global* program configuration where the call stack only contains *infectYoung*() invocations and during execution of the current *infectYoung*(), the local variable f is eventually set to true. This requires reasoning about the nestings of calls and returns of a computation. In fact, in order to decide if $f = \mathit{true}$ in the current procedure, we must "skip" over all calls within it and only consider their local return values. This requirement (and many others) can be naturally expressed in the logic *CaRet* [AEM04], an extension of LTL:

$$\Diamond^g \left(\Box^- p_Y \ \wedge \ p_Y \ \wedge \ \Diamond^a f \right) .$$

Here, p_Y is an atomic proposition that holds at states which correspond to being in procedure *infectYoung*, and f indicates that $f = \mathit{true}$. Intuitively, the above formula states that eventually (outer $\Diamond^g$), the computation reaches a (global) state where only *infectYoung* is on the call stack and the current procedure is *infectYoung* as well ($\Box^- p_Y \ \wedge \ p_Y$), and moreover the local—aka *abstract*—path *within* in the current procedure reaches a state where f is true ($\Diamond^a f$). Such properties are in general context-free but not always regular and thus cannot be expressed in LTL [AEM04].

Contributions. The contribution of this paper is a solution to the following problem:

> *Given a (finite) pPDA Δ and an ω-visibly pushdown language (VPL) $\mathcal{L}$ in terms of either a CaRet formula or an automaton, determine the probability that a random trajectory of Δ is in $\mathcal{L}$.*

The complexity results for the associated decision problems are summarized in Table 1. As common in the literature, we consider the special case of qualitative, i.e., *almost-sure* model checking separately. To the best of our knowledge, none of the problems in Table 1 was known to be decidable before. The work of [DBB12] proved decidability of model checking against deterministic Muller *visibly pushdown automata* (VPA) which capture a strict subset of the CaRet-definable languages [AM04]. The most important technical insight of this paper is that two existing (but independently developed) constructions from the literature can be combined to enable effective model checking of pPDA against ω-VPL: The *deterministic stair-parity VPA* introduced in [LMS04], and a certain *finite Markov chain* associated with a pPDA [EKM04]. We provide some more details in the next paragraph.

Table 1: Complexity results established in this paper.

ω-VPL given in terms of ...	*qualitative*	*quantitative*
Deterministic stair-parity VPA [Theorem 4.16]	in PSPACE	in PSPACE
Non-deterministic Büchi VPA [Theorem 5.1]	EXPTIME-compl.	in EXPSPACE
CaRet formula [Theorem 5.2]	in 2EXPTIME	in 2EXPSPACE

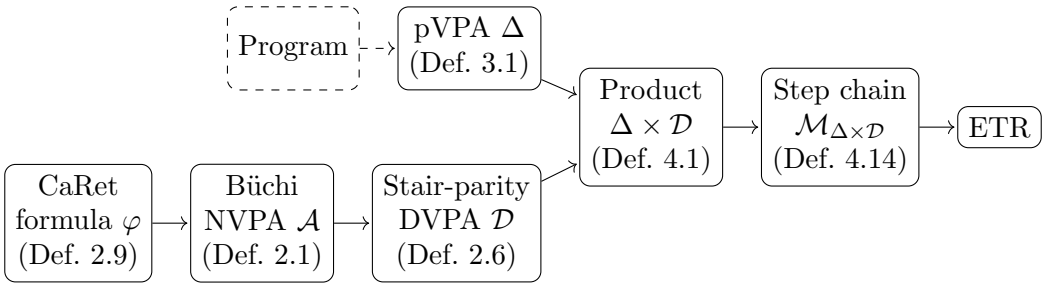


Figure 3: Chain of reductions used in this paper. ETR stands for *existential theory of the reals*, i.e., the existentially quantified fragment of the FO-theory over $(\mathbb{R}, +, \cdot, \leq)$.

Techniques and tools. We briefly outline our approach which is built on a number of existing constructions and results from the literature. In order for the model checking problems to be decidable [DBB12], we need to impose a mild *visibility* restriction on Δ, yielding a probabilistic *visibly* pushdown automaton (pVPA). Just like several previous works on model checking pPDA against ω-regular specifications [EKM04, BKS05, KEM06], we follow an automata-based approach (see Figure 3). More specifically, we first translate φ into an equivalent non-deterministic *Büchi* VPA [AM04] $\mathcal{A}$ and then determinize it using a procedure introduced by Löding et al. [LMS04]. The resulting DVPA $\mathcal{D}$ uses a so-called *stair-parity* [LMS04] acceptance condition that is strictly more expressive than standard parity or Muller DVPA [AM04]. Stair-parity differs from usual parity in that it only considers certain positions—called *steps* [LMS04]—of an infinite word where the stack height never decreases again. We then construct a standard product $\Delta \times \mathcal{D}$. Here, the visibility conditions ensure that the automata synchronize their stack actions, yielding a product automaton that uses *a single stack* instead of two independent stacks, which would lead to undecidability [DBB12]. Finally, we are left with computing a stair-parity acceptance probability in the product. This is achieved by constructing a specific finite Markov chain associated to $\Delta \times \mathcal{D}$, called *step chain* in this paper. Intuitively, the step chain "jumps" from one step of a run to the next, hence we only need to evaluate *standard parity* on the step chain. The idea of step chains was introduced by Esparza et al. [EKM04] who used them to show decidability of the model checking problem against deterministic (non-pushdown) Büchi automata. For constructing the step chain, certain *reachability probabilities* in the given pPDA need to be computed. These probabilities are algebraic numbers (i.e., solutions of polynomial equations) that may be irrational in general. However, the relevant problems are still decidable via an encoding in the existential fragment of the FO-theory of the reals (ETR) [KEM06].

Related work. We have already mentioned various works on recursion in probabilistic graphical models and PPL as well as on verifying pPDA and the equivalent model of recursive

Markov chains [EY09]. The analysis of these models focuses on reachability probabilities, ω-regular properties or (fragments of) probabilistic CTL, expected costs, and termination probabilities. The computation of termination probabilities in recursive Markov chains and variations thereof with non-determinism is supported by the software tool PReMo [WE07]. Our paper can be seen as a natural extension from checking pPDA against ω-regular properties to ω-visibly pushdown languages. In contrast to these algorithmic approaches, various deductive reasoning methods have been developed for recursive probabilistic programs. Proof rules for recursion were first provided in [Jon90], and later extended to proof rules in a weakest-precondition reasoning style [MM01, OKKM16]. The authors of [OKKM16] also address the connection to pPDA and provide proof rules for expected run-time analysis. A mechanized method for proving properties of randomized algorithms, including recursive ones, for the Coq proof assistant is presented in [AP09]. The Coq approach is based on higher-order logic using a monadic interpretation of programs as probabilistic distributions.

Conference version. A preliminary version of this article was published at FoSSaCS 2022 [WGK22]. The present journal version extends the conference paper by the full proofs as well as further examples and explanations.

Paper structure. This paper is structured as follows. We review the basics about VPA and CaRet in Section 2. Section 3 introduces probabilistic *visibly* pushdown automata (pVPA). The stair-parity DVPA model checking procedure is presented in Section 4, and the results for Büchi VPA and CaRet in Section 5. We conclude the paper in Section 6.

2. Visibly Pushdown Languages

In this section, we summarize some preliminary results on visibly pushdown languages and their corresponding automata models, and we recall the syntax and semantics of CaRet.

We use the following notation for words. Given a non-empty alphabet Σ, let Σ^* be the set of all finite words (including the empty word ϵ), and let Σ^ω be the set of all infinite words over Σ. For $i \geq 0$, the i-th symbol of a word $w \in \Sigma^* \cup \Sigma^\omega$ is denoted $w(i)$ if it exists. $|w|$ denotes the length of w. For $n \in \mathbb{N}_0$ we write $\Sigma^n = \{w \in \Sigma^* \mid |w| = n\}$. For sets of words $A \subseteq \Sigma^*$ and $B \subseteq \Sigma^* \cup \Sigma^\omega$, the concatenation of all words from A with those from B is denoted $A.B$. We also use $a.B$ and $A.b$ as shorthands for $\{a\}.B$ and $A.\{b\}$, respectively.

2.1. Visibly Pushdown Automata.

A finite alphabet Σ is called *pushdown alphabet* if it is equipped with a partition $\Sigma = \Sigma_{\mathsf{call}} \uplus \Sigma_{\mathsf{int}} \uplus \Sigma_{\mathsf{ret}}$ into three—possibly empty—subsets of *call*, *internal*, and *return* symbols. A *visibly pushdown automaton* (VPA) over Σ is like a standard pushdown automaton with the additional syntactic restriction that reading a call, internal, or return symbol triggers a push, internal, or pop transition, respectively (an internal transition is one that does not change the stack height). Formally:

Definition 2.1 (VPA [AM04])**.** Let Σ be a pushdown alphabet. A *visibly pushdown automaton (VPA)* over Σ is a tuple $\mathcal{A} = (S, s_0, \Gamma, \bot, \delta, \Sigma)$ with S a finite set of states, $s_0 \in S$ an initial state, Γ a finite stack alphabet, $\bot \in \Gamma$ a special bottom-of-stack symbol, and $\delta = (\delta_{\mathsf{call}}, \delta_{\mathsf{int}}, \delta_{\mathsf{ret}})$ a triple of relations such that

$$\delta_{\mathsf{call}} \subseteq (S \times \Sigma_{\mathsf{call}}) \times (S \times \Gamma_{\text{-}\bot}) \ , \quad \delta_{\mathsf{int}} \subseteq (S \times \Sigma_{\mathsf{int}}) \times S \ , \quad \delta_{\mathsf{ret}} \subseteq (S \times \Sigma_{\mathsf{ret}} \times \Gamma) \times S$$

where $\Gamma_{\text{-}\bot} = \Gamma \setminus \{\bot\}$. △

For $s, t \in S$, $Z \in \Gamma$, and $a \in \Sigma$, we write $s \xrightarrow{a} tZ$, $s \xrightarrow{a} t$, $sZ \xrightarrow{a} t$ to indicate that there exist transitions $(s, a, t, Z) \in \delta_{\mathsf{call}}$, $(s, a, t) \in \delta_{\mathsf{int}}$, $(s, a, Z, t) \in \delta_{\mathsf{ret}}$, respectively.

The semantics of a VPA is defined as usual via configurations and runs. A *configuration* of VPA $\mathcal{A}$ is a tuple $(s, \gamma) \in S \times \Gamma^*_{\text{-}\perp}.\perp$, written more succinctly as $s\gamma$ in the sequel. Intuitively, being in configuration $s\gamma$ means that the automaton is currently in state s and has the word γ on its stack. If $s\gamma = sZ\gamma'$ for some $Z \in \Gamma$, then sZ is called the *head* of $s\gamma$. A *bottom configuration* is a configuration with head $s\perp$ for some $s \in S$. Let $w \in \Sigma^\omega$ be an infinite input word. An infinite sequence $\rho = s_0\gamma_0, s_1\gamma_1 \ldots$ of configurations is called a *run* of $\mathcal{A}$ on w if $s_0\gamma_0 = s_0\perp$ and for all $i \geq 0$, exactly one of the following cases applies:

- $w(i) \in \Sigma_{\mathsf{call}}$ and $\gamma_{i+1} = Z\gamma_i$ for some $Z \in \Gamma_{\text{-}\perp}$ such that $s_i \xrightarrow{w(i)} s_{i+1}Z$; or
- $w(i) \in \Sigma_{\mathsf{int}}$ and $\gamma_{i+1} = \gamma_i$ and $s_i \xrightarrow{w(i)} s_{i+1}$; or
- $w(i) \in \Sigma_{\mathsf{ret}}$ and $Z\gamma_{i+1} = \gamma_i$ for some $Z \in \Gamma_{\text{-}\perp}$ such that $s_iZ \xrightarrow{w(i)} s_{i+1}$; or
- $w(i) \in \Sigma_{\mathsf{ret}}$ and $\gamma_i = \gamma_{i+1} = \perp$ and $s_i\perp \xrightarrow{w(i)} s_{i+1}$.

In other words, if $\mathcal{A}$ reads a call (or internal) symbol a while being in configuration $s\gamma$ and there exists a suitable call transition $s \xrightarrow{a} tZ$ (or internal transition $s \xrightarrow{a} t$), then a run of $\mathcal{A}$ may evolve from configuration $s\gamma$ to $tZ\gamma$ (or $t\gamma$, respectively). Similarly, if $\mathcal{A}$ reads a return symbol a in configuration $sZ\gamma$ where $Z \neq \perp$ and there is a transition $sZ \xrightarrow{a} t$, then a run can move from $sZ\gamma$ to $t\gamma$. Note that invoking a return transition in a bottom configuration $s\perp$ does not remove the topmost symbol $\perp$ from the stack.

A *Büchi acceptance condition* for $\mathcal{A}$ is a subset $F \subseteq S$. A VPA equipped with a Büchi condition is called a *Büchi VPA*. An infinite word $w \in \Sigma^\omega$ is accepted by a Büchi VPA if there exists a run $s_0\gamma_0, s_1\gamma_1, \ldots$ of $\mathcal{A}$ on w such that $s_i \in F$ for infinitely many $i \geq 0$. The ω-language of words accepted by a Büchi VPA $\mathcal{A}$ is denoted $\mathcal{L}(\mathcal{A}) \subseteq \Sigma^\omega$.

Definition 2.2 (ω-VPL [AM04])**.** Let Σ be a pushdown alphabet. $L \subseteq \Sigma^\omega$ is an *ω-visibly pushdown language (ω-VPL)* if $L = \mathcal{L}(\mathcal{A})$ for a Büchi VPA $\mathcal{A}$ over Σ. △

A VPA is *deterministic* (DVPA) if the relations $\delta_{\mathsf{call}}, \delta_{\mathsf{int}}$, and δ_{ret} are total functions, i.e., $\delta_{\mathsf{call}} \colon (S \times \Sigma_{\mathsf{call}}) \to (S \times \Gamma_{\text{-}\perp})$, $\delta_{\mathsf{int}} \colon (S \times \Sigma_{\mathsf{int}}) \to S$, and $\delta_{\mathsf{ret}} \colon (S \times \Sigma_{\mathsf{ret}} \times \Gamma) \to S$. Note that DVPA have exactly one run on each input word. As for standard NBA, the class of languages recognized by Büchi DVPA is a strict subset of the languages recognized by non-deterministic Büchi VPA. Unlike in the non-pushdown case, DVPA with Muller or parity conditions are also strictly less expressive than non-deterministic Büchi VPA [AM04]. A deterministic automaton model for ω-VPL was given in [LMS04]. It uses a so-called *stair-parity* acceptance condition which we explain in the upcoming Section 2.2.

2.2. **Steps and Stair-parity Conditions.** In the remainder of this section, Σ denotes a pushdown alphabet and $\mathcal{A}$ a VPA over Σ. Consider a run $\rho = s_0\gamma_0, s_1\gamma_1, \ldots$ of $\mathcal{A}$ on an infinite word $w \in \Sigma^\omega$. We define the *stack height* of the i-th configuration as $sh(\rho(i)) = |\gamma_i| - 1$ (i.e., the bottom symbol $\perp$ does not count to the stack height). The stair-parity condition relies on the notion of *steps*:

Definition 2.3 (Step)**.** Let ρ be a run of $\mathcal{A}$. Position $i \geq 0$ is a *step* of ρ if

$$\forall n \geq i: \quad sh(\rho(n)) \;\geq\; sh(\rho(i)) \;. \qquad \triangle$$

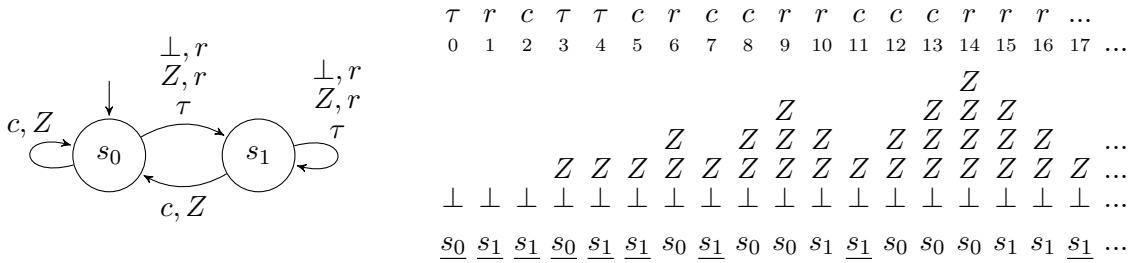


Figure 4: Left: An example VPA (in fact, a DVPA) with $\Gamma = \{Z, \perp\}$ over input alphabet $\Sigma = \{c\} \uplus \{\tau\} \uplus \{r\}$. Transitions labeled c, Z are *call transitions* that push Z on the stack. The transitions labeled τ are *internal transitions*; they ignore the stack completely. Transitions labeled Z, r and $\perp, r$ are *return transitions* that are only enabled if Z ($\perp$, respectively) is on top of the stack. When executing Z, r, the symbol Z is popped from the stack. However, the special bottom-of-stack symbol $\perp$ can never be popped (e.g., position 1 in the run). Right: The *unique* run of the DVPA on input word $\tau\, r\, c\, \tau\, \tau\, c\, r\, c^2\, r^2\, c^3\, r^3 \ldots$. Steps are underlined.

Intuitively, a step is a position of a run such that there is no future position where the stack height is strictly smaller. Slightly abusing terminology, we also say that a *configuration* $s_i\gamma_i$ of a given run $\rho = s_0\gamma_0, s_1\gamma_1, \ldots$ is a step if position i is a step.

Example 2.4. Figure 4 depicts a DVPA and the initial fragment of its unique run ρ on the input word $\tau\, r\, c\, \tau\, \tau\, c\, r\, c^2\, r^2\, c^3\, r^3 \ldots$. The step positions are underlined, i.e., positions 0-5, 7, 11, and 17 are steps. Note that if $\rho(i) = s\perp$ for some $s \in S$ then i is a step, i.e., bottom configurations are always steps.

Steps play a central role in the rest of the paper. We therefore explain some of their fundamental properties. Suppose that ρ is a run of $\mathcal{A}$ on an infinite word $w \in \Sigma^*$.

- If positions $i < j$ are *adjacent* steps, i.e., there is no step k strictly in between i and j, then $sh(\rho(j)) - sh(\rho(i)) \in \{0, 1\}$, i.e., the stack height from one step to the next increases by either zero or one.
- Each step i has a *next* step $j > i$: If the symbol at step i is internal (e.g., $i = 0, 3, 4$ in Figure 4) or a return (e.g., $i = 1$) then the next step is simply the next position $j = i + 1$ and the stack height does not increase. If the symbol at position i is a call, then one of two cases occurs: Either the call has no matching future return (e.g., $i = 2$); in this case, the next step is the next position $j = i + 1$. Otherwise the call is eventually matched (e.g., $i = 5, 7, 11$) and the next step $j > i + 1$ occurs after the corresponding matching return is read and has the same stack height.
- As a consequence, each infinite run has infinitely many steps. Notice though that the difference between two adjacent step positions may grow unboundedly as in Figure 4.
- The stack height at the steps either grows unboundedly or eventually stabilizes (the latter occurs in Figure 4; the stack heights at the steps induce the sequence $0, 0, 0, 1^\omega$).

Remark 2.5. We can also define the steps of a *word* $w \in \Sigma^\omega$ as the positions where a run of *any arbitrary* VPA on w has a step. Due to the visibility restriction, the actual behavior of the VPA does not influence the step positions [LMS04]. In other words, the step positions are predetermined by the input word. Thus, we can also speak of the stack height $sh(w(i))$ of word w at position i.

We need one last notion before defining stair-parity. The *footprint* of an infinite run $\rho = s_0\gamma_0, s_1\gamma_1, \ldots$ is the infinite sequence $\rho{\downarrow_{Steps}} = s_{n_0}s_{n_1}\ldots \in S^\omega$ where for all $i \geq 0$ the position n_i is the i-th step of ρ. In words, $\rho{\downarrow_{Steps}}$ is the projection of the run ρ onto the *states* occurring at its steps. For the example run in Figure 4, $\rho{\downarrow_{Steps}} = s_0s_1s_1s_0s_1^\omega$.

Definition 2.6 (Stair-parity [LMS04])**.** Let $\mathcal{A}$ be a VPA over pushdown alphabet Σ. A *stair-parity* acceptance condition for $\mathcal{A}$ is defined in terms of a priority function $\Omega\colon S \to \mathbb{N}_0$. A word $w \in \Sigma^\omega$ is accepted if $\mathcal{A}$ has a run ρ on w such that

$$\min\{\, k \mid \overset{\infty}{\exists} i \geq 0\colon \Omega(\,\rho{\downarrow_{Steps}}(i)\,) = k \,\} \quad \text{is even.}$$

The language accepted by $\mathcal{A}$ is denoted $\mathcal{L}(\mathcal{A})$. △

Intuitively, $\mathcal{L}(\mathcal{A})$ contains all words that have a run ρ on $\mathcal{A}$ such that the minimum priority occurring infinitely often at states in the *footprint* $\rho{\downarrow_{Steps}}$ is even.

Example 2.7. The DVPA in Figure 4 with $\Omega(s_0) = 1$ and $\Omega(s_1) = 2$ accepts

$$\mathcal{L}_{repbdd} \;=\; \{\, w \in \Sigma^\omega \mid \exists B \geq 0,\ \overset{\infty}{\exists} i \geq 0\colon sh(w(i)) = B \,\}\,,$$

the language of *repeatedly bounded* words [LMS04], i.e., words whose stack height (cf. Remark 2.5) is infinitely often equal to some constant B. The example word from Figure 4 satisfies this property with $B = 1$. To see why the automaton accepts $\mathcal{L}_{repbdd}$, note that a word is repeatedly bounded iff the stack height at the steps stabilizes eventually. The latter occurs iff in just finitely many cases, the transition before reaching a step was a call. The DVPA in Figure 4 detects this behavior; when reading a call symbol, it always moves to state s_0 which has odd priority, and it accepts iff s_0 is visited finitely often *at call positions*. It is known that $\mathcal{L}_{repbdd}$ is not expressible through DVPA with usual parity conditions [AM04].

Theorem 2.8 [LMS04, Theorem 1]**.** *For every non-deterministic Büchi VPA $\mathcal{A}$ there exists a deterministic stair-parity DVPA $\mathcal{D}$ with $2^{\mathcal{O}(|S|^2)}$ states such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{D})$. Moreover, $\mathcal{D}$ can be constructed in exponential time in the size of $\mathcal{A}$.*

It was also shown in [LMS04] that stair-parity DVPA characterize exactly the class of ω-VPL (and are thus not more expressive than non-deterministic Büchi VPA).

2.3. **CaRet, a Temporal Logic of Calls and Returns.** Specifying requirements directly in terms of automata is tedious in practice. *CaRet* [AEM04] is an extension of Linear Temporal Logic (LTL) [Pnu77] that can be used to describe certain ω-VPL.

Definition 2.9 (Syntax of CaRet [AEM04])**.** Let AP be a finite set of atomic propositions. The logic CaRet adheres to the grammar

$$\varphi \;\coloneqq\; p \mid \varphi \vee \varphi \mid \neg\varphi \mid \bigcirc^g\varphi \mid \varphi\,\mathcal{U}^g\varphi \mid \bigcirc^a\varphi \mid \varphi\,\mathcal{U}^a\varphi \mid \bigcirc^-\varphi \mid \varphi\,\mathcal{U}^-\varphi\,,$$

where $p \in AP \cup \{\mathsf{call}, \mathsf{int}, \mathsf{ret}\}$. △

Other common modalities such as $\Diamond^b$ and $\Box^b$ for $b \in \{g, a, -\}$ are defined as usual via $\Diamond^b\varphi = \mathit{true}\,\mathcal{U}^b\,\varphi$, and $\Box^b\varphi = \neg\Diamond^b\neg\varphi$. We now explain the intuitive semantics of CaRet, the formal definition is stated further below in Definition 2.10. We assume familiarity with LTL (see, e.g., [BK08, Ch. 5] for an introduction). CaRet formulae are interpreted over infinite words from the pushdown alphabet $\Sigma = 2^{AP} \times \{\mathsf{call}, \mathsf{int}, \mathsf{ret}\}$. $\bigcirc^g$ and $\mathcal{U}^g$ are the standard next and until modalities from LTL (called *global* next and until in CaRet). CaRet

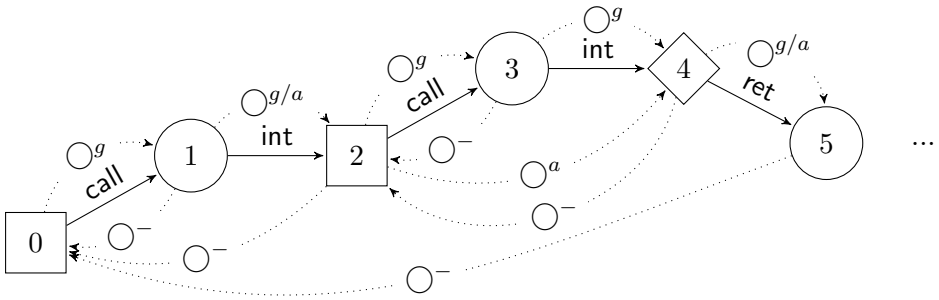


Figure 5: CaRet's various next modalities applied to the initial fragment of an example word. Call, internal, and return positions are depicted as boxes, circles, and rhombs, respectively. Note that $\bigcirc^a$ of position 3 is undefined because $\bigcirc^g$ is a return. Whether or not $\bigcirc^a$ of position 0 is defined depends on how the words continues after position 5; more specifically, it is defined iff there occurs a return position on the same height as position 5. In this case, $\bigcirc^a$ of position 0 will point to the first such occurrence.

extends LTL by two key operators, the *caller* modality $\bigcirc^-$ and the *abstract successor* $\bigcirc^a$. The semantics of these operators is visually explained in Figure 5. The caller $\bigcirc^-$ is a *past* modality that points to the position of the last pending call (if such a call exists). For internal and return symbols, the abstract successor $\bigcirc^a$ behaves like $\bigcirc^g$ unless the latter is a return, in which case $\bigcirc^a$ is undefined (e.g., position 3 in Figure 5). On the other hand, the abstract successor of a call symbol is its *matching return* if it exists, or undefined otherwise. The caller and abstract successor modalities induce sequences of positions which we call *caller path* and *abstract path*, respectively. The caller path is always finite and the abstract path can be either finite or infinite. The until modalities $\mathcal{U}^-$ and $\mathcal{U}^a$ are then defined analogously to the standard until $\mathcal{U}^g$ with the difference that they are interpreted over the caller and abstract path, respectively.

A prime application of CaRet is to express *total correctness* of a procedure F [AEM04]:

$$\varphi_{total} \quad = \quad \Box^g \left(\mathsf{call} \wedge p \wedge p_F \;\rightarrow\; \bigcirc^a q \right)$$

where p and q are atomic propositions that hold at the states where the pre- and post-condition is satisfied, respectively, and p_F is an atomic proposition marking the calls to F. Another example is the language of repeatedly bounded words from Example 2.7; it is described by the formula $\varphi_{repbdd} = \Diamond^g \Box^g (\mathsf{call} \rightarrow \bigcirc^a \mathsf{ret})$ which states that all but finitely many calls have a matching return. Further examples are given in [AEM04].

We now define the semantics of CaRet formally. Let $\Sigma = 2^{AP} \times \{\mathsf{call}, \mathsf{int}, \mathsf{ret}\}$ be the pushdown alphabet and $w \in \Sigma^\omega$. If $w(i) \in \Sigma_{\mathsf{call}}$, then let $MR_w(i)$ be the position of the matching return of $w(i)$, or *undef* if there is no matching return. The *abstract successor* $succ^a_w(i)$ of position i in word w is defined as follows:

$$succ^a_w(i) \;=\; \begin{cases} MR_w(i) & \text{if } w(i) \in \Sigma_{\mathsf{call}} \\ i+1 & \text{if } w(i) \notin \Sigma_{\mathsf{call}} \;\wedge\; w(i{+}1) \notin \Sigma_{\mathsf{ret}} \\ \mathit{undef} & \text{if } w(i) \notin \Sigma_{\mathsf{call}} \;\wedge\; w(i{+}1) \in \Sigma_{\mathsf{ret}} \end{cases}$$

The *caller* $succ_w^-(i)$ of position i in word w is defined as the innermost ("closest") unmatched call position $j < i$, or *undef* if there was no previous open call.

Definition 2.10 (Semantics of CaRet)**.** Let $\Sigma = 2^{AP} \times \{\mathsf{call}, \mathsf{int}, \mathsf{ret}\}$ and $w \in \Sigma^\omega$. For all CaRet formulae φ and $i \in \mathbb{N}_0$, we define $w(i) \models \varphi$ by induction over the structure of φ as follows:

- $w(i) \models p$ iff $w(i) = (X, \mathsf{type})$ and either $p \in X$ or $p = \mathsf{type}$
- $w(i) \models \varphi_1 \vee \varphi_2$ iff $w(i) \models \varphi_1$ or $w(i) \models \varphi_2$
- $w(i) \models \neg\varphi$ iff $w(i) \not\models \varphi$
- $w(i) \models \bigcirc^g \varphi_1$ iff $w(i+1) \models \varphi_1$
- $w(i) \models \bigcirc^a \varphi_1$ iff $succ_w^a(i) = j \in \mathbb{N}_0$ is defined and $w(j) \models \varphi_1$
- $w(i) \models \bigcirc^- \varphi_1$ iff $succ_w^-(i) = j \in \mathbb{N}_0$ is defined and $w(j) \models \varphi_1$
- $w(i) \models \varphi_1 \mathcal{U}^b \varphi_2$ for $b \in \{a, g, -\}$ if there exist positions $i = i_0, i_1, \ldots, i_k$, $k \in \mathbb{N}_0$, such that
  (1) $w(i_k) \models \varphi_2$, and
  (2) for all $0 \leq j < k$, $w(i_j) \models \varphi_1$ and $i_{j+1} = succ_w^b(i_j)$.

Furthermore, we write $w \models \varphi$ if $w(0) \models \varphi$. The language of all words satisfying a CaRet formula φ is denoted $\mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid w \models \varphi\}$. △

Theorem 2.11 [AAB+07, Theorem 5.1]**.** *CaRet-definable languages are ω-VPL: For each CaRet formula φ there exists a (non-deterministic) Büchi VPA $\mathcal{A}$ such that $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$, and $\mathcal{A}$ can be constructed in time $2^{\mathcal{O}(|\varphi|)}$.*

The above theorem is well-known in the literature [AAB+07, ABE18] even though it is usually stated for *Nested Word Automata* (NWA) which are equivalent to VPA, and it is more common to state a space bound on $\mathcal{A}$ rather than a time bound for the construction. The theorem even applies to the logic NWTL$^+$, an FO-complete extension of CaRet [AAB+07] which we do not consider here for the sake of simplicity.

Theorems 2.8 and 2.11 together imply that each CaRet formula can be translated to a *deterministic* stair-parity VPA of doubly-exponential size.

3. Probabilistic Visibly Pushdown Automata

As explained in the introduction, we employ *probabilistic pushdown automata* [EKM04] (pPDA) as an operational model for procedural probabilistic programs. pPDA thus play a fundamentally different role in this paper than VPA (cf. Definition 2.1): pPDA are used to model the *system*, while VPA encode the *specification*. Consequently, our pPDA do *not* read an input word like VPA do, but instead take their transitions randomly, according to fixed probability distributions. In this way, they define a probability space over their possible traces, i.e., runs projected on their labeling sequence. These traces constitute the input words of the VPA. In order for the model checking problems to be decidable [DBB12], a syntactic visibility restriction needs to be imposed on pPDA. In a nutshell, the condition is that each state only has outgoing transitions of one *type*, i.e., push, internal, or pop. This means that the stack operation is *visible in the states* (recall that for VPA, the stack operation is visible in the input symbol). This restriction is not severe in the context of modeling programs (see Remark 3.4 further below) and leads to our notion of probabilistic *visibly* pushdown automata (pVPA) which we now define formally.

Given a finite set X, we write $Dist(X) = \{\, f \colon X \to [0,1] \mid \sum_{a \in X} f(a) = 1 \,\}$ for the set of probability distributions over X.

Definition 3.1 (pVPA)**.** A *probabilistic visibly pushdown automaton* (pVPA) is a tuple $\Delta = (Q, q_0, \Gamma, \bot, P, \Sigma, \lambda)$ where Q is a finite set of states partitioned into $Q = Q_{\mathsf{call}} \uplus Q_{\mathsf{int}} \uplus Q_{\mathsf{ret}}$, $q_0 \in Q$ is an initial state, Γ is a finite stack alphabet, $\bot \in \Gamma$ is a special bottom-of-stack symbol, $P = (P_{\mathsf{call}}, P_{\mathsf{int}}, P_{\mathsf{ret}})$ is a triple of functions

$$P_{\mathsf{call}} \colon Q_{\mathsf{call}} \to \mathit{Dist}(Q \times \Gamma_{\text{-}\bot}) \ , \quad P_{\mathsf{int}} \colon Q_{\mathsf{int}} \to \mathit{Dist}(Q) \ , \quad P_{\mathsf{ret}} \colon Q_{\mathsf{ret}} \times \Gamma \to \mathit{Dist}(Q) \ ,$$

$\Sigma = \Sigma_{\mathsf{call}} \uplus \Sigma_{\mathsf{int}} \uplus \Sigma_{\mathsf{ret}}$ is a pushdown alphabet, and $\lambda \colon Q \to \Sigma$ is a state labeling function consistent with the visibility condition, i.e., for all $\mathsf{type} \in \{\mathsf{call}, \mathsf{int}, \mathsf{ret}\}$ and all $q \in Q$, it is required that $q \in Q_{\mathsf{type}}$ iff $\lambda(q) \in \Sigma_{\mathsf{type}}$. △

Similar to VPA, we use the notation $q \xrightarrow{p} rZ$, $q \xrightarrow{p} r$, and $qZ \xrightarrow{p} r$ to indicate that $P_{\mathsf{call}}(q)(r, Z) = p$, $P_{\mathsf{int}}(q)(r) = p$, and $P_{\mathsf{ret}}(q, Z)(r) = p$, respectively.

Intuitively, the behavior of a pVPA Δ is as follows. If the current state q is a call state, then the probability distribution $P_{\mathsf{call}}(q)$ determines a random successor state and stack symbol to be pushed on the stack ($\bot$ cannot be pushed). Similarly, if the current state q is internal, then $P_{\mathsf{int}}(q)$ is the distribution over possible successor states and no stack operation is performed. Lastly, if the current state q is a return state and symbol $Z \in \Gamma_{\text{-}\bot}$ is on top of the stack, then Z is popped and Δ moves to a successor state with probability according to $P_{\mathsf{ret}}(q, Z)$. As in VPA, the special symbol $\bot$ is *not* popped from the stack if a return transition occurs in a bottom configuration.

The formal semantics of a pVPA is defined in terms of a countably infinite discrete-time *Markov chain*. A (labeled) Markov chain is essentially the special case of a pVPA where $Q = Q_{\mathsf{int}}$, with the only difference that we allow for countably infinite Q and do not impose the restriction on the labeling function λ. A Markov chain can thus be specified as a 5-tuple $(Q, q_0, P, \Sigma, \lambda)$, i.e., we omit $\bot$ and Γ from the definition of pVPA because a Markov chain does not use a stack. A *run* of a Markov chain is an infinite sequence of states, i.e., an element from Q^ω. Note that in our definition, runs do not necessarily start in q_0; this is just for technical convenience—impossible runs starting in a state other than q_0 will simply have probability 0. We extend the labeling function λ from states to runs in the natural way.

We define the *Markov chain generated by a pVPA* $\Delta = (Q, q_0, \Gamma, \bot, P, \Sigma, \lambda)$ as

$$\mathcal{D}_\Delta \ = \ (\, Q \times \Gamma^*_{\text{-}\bot}.\bot,\, q_0\bot,\, P_\Delta,\, \Sigma,\, \lambda_\Delta \,) \ ,$$

i.e., the state space of $\mathcal{D}_\Delta$ is the set of configurations of Δ, and the transition probability function P_Δ is defined as follows. $P_\Delta(q\gamma)(r\gamma') = p > 0$ iff exactly one of the following cases applies:

- $q \in Q_{\mathsf{call}}$ and $\gamma' = Z\gamma$ for some $Z \in \Gamma_{\text{-}\bot}$ and $q \xrightarrow{p} rZ$; or
- $q \in Q_{\mathsf{int}}$ and $\gamma' = \gamma$ and $q \xrightarrow{p} r$; or
- $q \in Q_{\mathsf{ret}}$ and $Z\gamma' = \gamma$ for some $Z \in \Gamma_{\text{-}\bot}$ and $qZ \xrightarrow{p} q$; or
- $q \in Q_{\mathsf{ret}}$ and $\gamma' = \gamma = \bot$ and $q\bot \xrightarrow{p} r$.

Moreover, the labeling function of $\mathcal{D}_\Delta$ is $\lambda_\Delta(q\gamma) = \lambda(q)$ for all $q\gamma \in Q \times (\Gamma_{\text{-}\bot})^*.\bot$.

Example 3.2. Figure 6 depicts a pVPA Δ and a fragment of its generated Markov chain $\mathcal{D}_\Delta$. Even though $\mathcal{D}_\Delta$ is infinite, many problems remain decidable, including in particular questions about reachability probabilities which can be characterized as the least solution of a system of polynomial equations [EKM04]. We will use this extensively in Section 4.

We define the set Runs_Δ of a pVPA Δ as the runs of the Markov chain $\mathcal{D}_\Delta$, i.e., $\mathit{Runs}_\Delta = (Q \times \Gamma^*_{\text{-}\bot}.\bot)^\omega$. *Steps* of pVPA runs are defined as in Definition 2.3. A further example pVPA and its possible runs are depicted in Figure 7 on page 16.

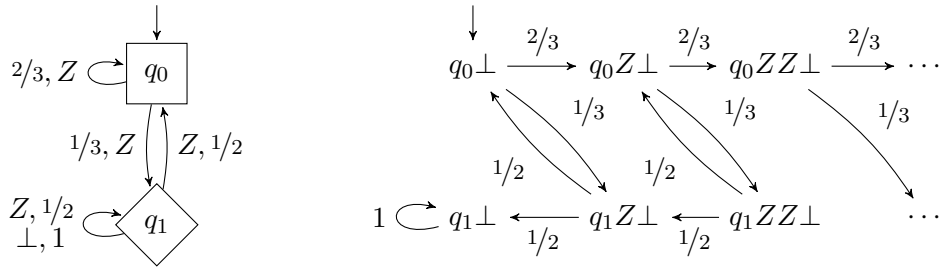


Figure 6: Left: A pVPA Δ (labeling function omitted) with $\Gamma = \{\bot, Z\}$. Call and return states are drawn as squares and rhombs, respectively. Transitions labeled p, Z for $p \in [0,1]$ mean that with probability p symbol Z is pushed. Transitions labeled X, p, for $X \in \Gamma$ and $p \in [0,1]$, indicate that *if* X is on top of the stack, then the transition is taken with probability p (and X is popped if $X \neq \bot$). Note that the visibility restriction on pVPA enforces that a state may not have both outgoing push and outgoing pop transitions. Right: The infinite-state Markov chain $\mathcal{D}_\Delta$ generated by Δ.

We associate a probability space with $Runs_\Delta$ in the usual way (see, e.g., [BK08, Ch. 10]). To this end, we define the σ-algebra $\mathcal{F} \subseteq 2^{Runs_\Delta}$ as the smallest set that contains all the *cylinder sets* $\rho.Runs_\Delta$, where ρ is an arbitrary *finite* prefix $\rho \in (Q \times \Gamma^*_{\text{-}\bot}.\bot)^*$ of a run, and that is closed under complement and countable union. The sets in $\mathcal{F}$ are called *measurable* and there is a unique probability measure $\mathbb{P}_\Delta \colon \mathcal{F} \to [0,1]$ satisfying

$$\mathbb{P}_\Delta(\rho.Runs_\Delta) \;=\; \begin{cases} \prod_{i=0}^{|\rho|-2} P_\Delta(\rho(i), \rho(i{+}1)) & \text{if } |\rho| = 0 \text{ or } (|\rho| > 0 \text{ and } \rho(0) = q_0\bot), \\ 0 & \text{otherwise,} \end{cases}$$

where an empty product (which occurs if $|\rho| \leq 1$) is defined to be equal to 1. We omit the subscript in $\mathbb{P}_\Delta$ whenever Δ is given by the context.

In the following two remarks, we summarize the technical differences between our pVPA model and existing models in the literature.

Remark 3.3. Unlike the pPDA from [EKM04], our pVPA only generate *infinite* runs, i.e., they do not "terminate" when reaching the empty stack. Indeed, in our pVPA, the stack can never be empty because the special bottom symbol $\bot$ cannot be popped. We have chosen this semantics for compatibility with CaRet which describes ω-languages by definition. Nonetheless, terminating behavior can be easily simulated in our framework by moving to a dedicated sink state once the pVPA attempts to pop $\bot$ for the first time. Another technical difference between our pVPA and the pPDA introduced in [EKM04] is that in pVPA, only pop transitions can read the stack, whereas in pPDA, all types of transitions can read, and possibly exchange, the current topmost stack symbol. We have chosen this definition (which is not a true restriction) for compatibility with VPA as defined in [AM04].

Remark 3.4. The visibility restriction of our pVPA is slightly different from the definition given in [DBB12] which requires all *incoming* transitions to a state to be of the same type, i.e., call, internal, or return. Our definition, on the other hand, imposes the same requirement on the states' *outgoing* transitions. We believe that our condition is more natural for pVPA obtained from procedural programs, such as the one in Figure 1. In fact,

programs where randomness is restricted to *internal* statements such as $x := \texttt{bernoulli}(0.5)$ or $x := \texttt{uniform}(0, 3)$ naturally comply with our visibility condition because all call and return states of such programs are deterministic and thus cannot violate visibility. However, the alternative condition of [DBB12] is not necessarily fulfilled for such programs.

We can now formally state our main problem of interest:

Definition 3.5 (Probabilistic CaRet Model Checking)**.** Let AP be a finite set of atomic propositions, φ be a CaRet formula over AP, Δ be a pVPA with labels from the pushdown alphabet $\Sigma = 2^{AP} \times \{\mathsf{call}, \mathsf{int}, \mathsf{ret}\}$, and $\theta \in [0, 1] \cap \mathbb{Q}$. The *quantitative CaRet Model Checking problem* is to decide whether

$$\mathbb{P}(\{\, \rho \in \mathit{Runs}_\Delta \mid \lambda(\rho) \in \mathcal{L}(\varphi) \,\}) \;\geq_?\; \theta \;.$$

The *qualitative* CaRet Model Checking problem is the special case where $\theta = 1$. △

The probabilities in Definition 3.5 are well-defined as ω-VPL are measurable [LMS04].

4. Model Checking pVPA against Stair-parity DVPA

In this section, we show that model checking pVPA (Definition 3.1) against VPL given in terms of a stair-parity DVPA (Definition 2.6) is decidable. This is achieved by first computing an automata-theoretic product of the pVPA and the DVPA and then evaluating the acceptance condition in the product automaton.

4.1. Products of Visibly Pushdown Automata.

In general, pushdown automata are not closed under taking products as this would require *two* independent stacks. However, the visibility conditions on VPA and pVPA ensure that their product is again an automaton with just a single stack because the stack operations (push, internal, or pop) are forced to synchronize.

We now define the product formally. An *unlabeled* pVPA is a pVPA where the labeling function λ and alphabet Σ are omitted.

Definition 4.1 (Product $\Delta \times \mathcal{D}$)**.** Let $\Delta = (Q, q_0, \Gamma, \bot, P, \Sigma, \lambda)$ be a pVPA, and $\mathcal{D} = (S, s_0, \Gamma', \bot, \delta, \Sigma)$ be a DVPA over pushdown alphabet Σ. The product of Δ and $\mathcal{D}$ is the unlabeled pVPA

$$\Delta \times \mathcal{D} \;=\; (\, Q \times S,\; (q_0, s_0),\; \Gamma \times \Gamma',\; \langle \bot, \bot \rangle,\; P_{\Delta \times \mathcal{D}} \,)\;,$$

where $P_{\Delta \times \mathcal{D}}$ is the smallest set of transitions satisfying the following rules for all $q, r \in Q$, $Z \in \Gamma$, $s, t \in S$, and $Y \in \Gamma'$:

$$\text{(call)}\ \frac{q \xrightarrow{p}_\Delta r Z \;\wedge\; s \xrightarrow{\lambda(q)}_\mathcal{D} t Y}{(q, s) \xrightarrow{p}_{\Delta \times \mathcal{D}} (r, t)\langle Z, Y \rangle} \qquad \text{(return)}\ \frac{q Z \xrightarrow{p}_\Delta r \;\wedge\; s Y \xrightarrow{\lambda(q)}_\mathcal{D} t}{(q, s)\langle Z, Y \rangle \xrightarrow{p}_{\Delta \times \mathcal{D}} (r, t)}$$

$$\text{(internal)}\ \frac{q \xrightarrow{p}_\Delta r \;\wedge\; s \xrightarrow{\lambda(q)}_\mathcal{D} t}{(q, s) \xrightarrow{p}_{\Delta \times \mathcal{D}} (r, t)}\;.$$

If the DVPA $\mathcal{D}$ is equipped with a priority function $\Omega \colon S \to \mathbb{N}_0$, then we extend Ω to $\Omega' \colon Q \times S \to \mathbb{N}_0$ via $\Omega'(q, s) = \Omega(s)$. △

It is not difficult to show that $\Delta \times \mathcal{D}$ is indeed a well-defined pVPA and moreover satisfies the following property (the proof is standard, see [WGK21, Appendix B.1]):

Lemma 4.2 (Soundness of $\Delta \times \mathcal{D}$)**.** *Let Δ be a pVPA and $\mathcal{D}$ be a stair-parity DVPA with priority function Ω, both over pushdown alphabet Σ. Then the product pVPA $\Delta \times \mathcal{D}$ with priority function Ω' as in Definition 4.1 satisfies*

$$\mathbb{P}(\{\, \rho \in Runs_\Delta \mid \lambda(\rho) \in \mathcal{L}(\mathcal{D}) \,\}) \;=\; \mathbb{P}(\{\, \rho \in Runs_{\Delta\times\mathcal{D}} \mid \rho{\downarrow_{Steps}} \in Parity_{\Omega'} \,\}),$$

where $Parity_{\Omega'}$ denotes the set of words in $(Q \times S)^\omega$ satisfying the standard *parity condition induced by Ω'. Moreover, $\Delta \times \mathcal{D}$ can be constructed in polynomial time.*

Remark 4.3. It is *not* actually important that the product pVPA again satisfies the visibility condition, even though this happens to be the case. All techniques we apply to the product also work for general pPDA.

4.2. **Stair-parity Acceptance Probabilities in pVPA.** Lemma 4.2 effectively reduces model checking pVPA against stair-parity DVPA to computing stair-parity acceptance in the product, which is again an (unlabeled) pVPA. We therefore focus on pVPA in this section and do not consider DVPA.

Throughout the rest of this section, let $\Delta = (\, Q,\, q_0,\, \Gamma,\, \bot,\, P\,)$ be an unlabeled pVPA. On the next pages we describe the construction of a *finite* Markov chain $\mathcal{M}_\Delta$ that we call the *step chain* of Δ. Loosely speaking, $\mathcal{M}_\Delta$ simulates jumping from one *step* (see Definition 2.3) of a run of Δ to the next.

Remark 4.4. The idea of the step Markov chain $\mathcal{M}_\Delta$ first appeared in [EKM04]. However, the step chain as presented here differs from the original definition in [EKM04] in at least two important aspects. First, we have to take the semantics of our special bottom symbol $\bot$ into account. This is why our chain uses a subset of $Q \cup Q\bot$ as states—it must distinguish whether a step occurs at a bottom configuration. The pPDA in [EKM04], on the other hand, may have both finite and infinite runs, and this needs to be handled differently in the step chain. Second, we use step chains for a different purpose than [EKM04], namely to show that general measurable properties defined on steps—this includes stair-parity—can be evaluated on pVPA (Lemma 4.15).

4.2.1. *Steps as events.* For all $n \in \mathbb{N}_0$, we define a *random* variable $V^{(n)}$ on $Runs_\Delta$ whose value is either the state q of Δ at the n-th step, or the *extended state* $q\bot$ in the special case where the n-th step occurs at a bottom configuration $q\bot$, for some $q \in Q$. We denote the set of all such extended states with $Q\bot = \{\, q\bot \mid q \in Q \,\}$. Formally, $V^{(n)} \colon Runs_\Delta \to Q \cup Q\bot$ is defined as

$$V^{(n)}(\rho) \quad = \quad \begin{cases} q & \text{if } step_n(\rho) = q\gamma \text{ and } \gamma \neq \bot \\ q\bot & \text{if } step_n(\rho) = q\bot\ , \end{cases}$$

where $step_n(\rho)$ denotes the configuration at the n-th step of ρ.

Lemma 4.5. *For all $n \in \mathbb{N}_0$ and $v \in Q \cup Q\bot$, the event $V^{(n)} = v$ is measurable, and thus $V^{(n)}$ is a well-defined random variable.*

Proof. This was proved more generally in [EKM04]. Here we give an alternative proof using the fact that all ω-VPL are measurable [LMS04]. We view $Q \cup Q\bot$ as a pushdown alphabet (the partition is induced by the partition on Q). We can construct a non-deterministic Büchi VPA that accepts a word $w \in (Q \cup Q\bot)^\omega$ iff the n-th step of w is v (the size of this

automaton depends on n). To this end, the VPA guesses the first n positions that are steps and goes to an accepting state s if the n-th step was v. The automaton can verify that it guessed correctly as follows. If it believes a call symbol is a step, it pushes a special marker on the stack; if this marker is ever popped, then the call was no step and the guess was wrong. If it detects a wrong guess in this way, then it leaves the accepting state s, otherwise it loops there forever. The claim follows because the function $f\colon \mathit{Runs}_\Delta \to (Q \cup Q\bot)^\omega$ that maps runs to the sequence of their (extended) states is measurable (indeed, the preimage $f^{-1}(w)$ of every $w \in (Q \cup Q\bot)^\omega$ is

$$f^{-1}(w) \;=\; \bigcap_{i\geq 0} \; \bigcup_{\gamma_0 \ldots \gamma_i \in \Gamma^*_{\not\bot}.\bot} w(0)\gamma_0 \ldots w(i)\gamma_i.\mathit{Runs}_\Delta$$

which is a countable intersection of countable unions of basic cylinder sets). □

We can view the sequence $V^{(0)}, V^{(1)} \ldots$ of random variables as a *stochastic process*. It is intuitively clear that for all $n \in \mathbb{N}_0$, the value of $V^{(n+1)}$ depends only on $V^{(n)}$, but not on $V^{(i)}$ for $i < n$. This is due to the more general observation that only the *state* q at any step configuration $q\gamma$ (with $\gamma \neq \bot$) *fully determines the future of the run* because being a step already implies that no symbol in γ can ever be read, as reading it implies popping it from the stack. In particular, q determines the probability distribution over possible next steps. A similar observation applies to bottom configurations of the form $q\bot$. Phrased in the language of probability theory, the process $V^{(0)}, V^{(1)} \ldots$ has the *Markov property*, i.e.,

$$\mathbb{P}(V^{(n)} = v_n \mid V^{(n-1)} = v_{n-1} \wedge \ldots \wedge V^{(0)} = v_0) \;=\; \mathbb{P}(V^{(n)} = v_n \mid V^{(n-1)} = v_{n-1}) \quad (4.1)$$

holds for all values of $v_0, \ldots, v_n$ such that the conditional probabilities are well-defined[1]. This was proved in detail in [EKM04]. It is also clear that the Markov process is time-homogeneous in the sense that

$$\mathbb{P}(V^{(n+1)} = v' \mid V^{(n)} = v) \quad = \quad \mathbb{P}(V^{(m+1)} = v' \mid V^{(m)} = v) \quad (4.2)$$

holds for all $n, m \in \mathbb{N}_0$ for which the two conditional probabilities are well-defined. The following example provides some intuition on these facts.

Example 4.6. Consider the pVPA in Figure 7 (left). The initial fragments of its two equiprobable runs are depicted in the middle. In this example, it is easy to read off the next-step probabilities $\mathbb{P}(V^{(n)} = v_n \mid V^{(n-1)} = v_{n-1})$ for all $n \in \mathbb{N}_0$ and $v_n, v_{n-1} \in Q \cup Q\bot$. They are summarized in the Markov chain on the right. For example, $V^{(0)} = q_0\bot$ holds with probability 1, and $V^{(1)} = q_1$ and $V^{(1)} = q_3\bot$ hold with probability $1/2$ each because the second step occurs either at position 1 with configuration $q_1\bot Z$ or at position 3 with configuration $q_3\bot$, and both options are equally likely. The case $\mathbb{P}(V^{(2)} = q_2 \mid V^{(1)} = q_1) = 1$ is slightly more interesting: Given that a configuration $q_1\gamma$ with $\gamma \neq \bot$ is a step, we know that the next state must be q_2 (which is then also a step). Even though there is a transition from q_1 to q_3 in Δ, the next state cannot be q_3 because the latter is a return state which would immediately decrease the stack height of γ. This shows that, intuitively speaking, *conditioning on being a step influences the probabilities of a state's outgoing transitions.*

[1]A conditional probability is well-defined if the *condition*, i.e., the event on the right hand side of the vertical bar, has positive probability. Expressions like the one in (4.1) are thus not necessarily well-defined because the probability that $V^{(n-1)} = v_{n-1}$ might be zero for certain values of n and v_{n-1}.

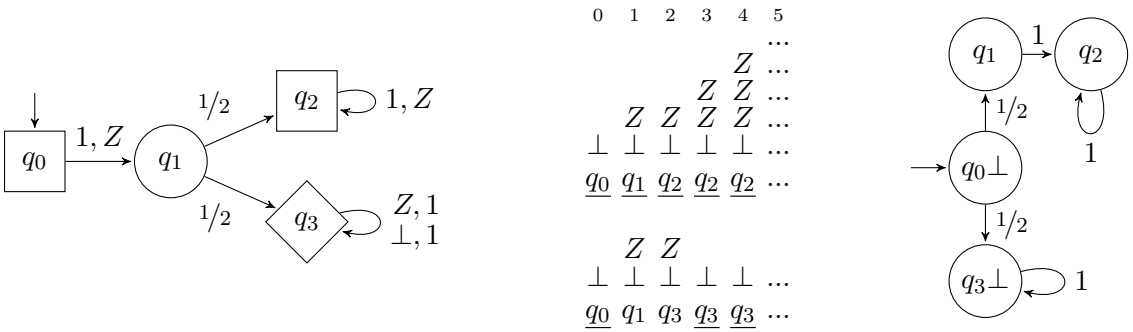


Figure 7: Left: An example (unlabeled) pVPA Δ. Recall that call and return states are drawn as squares and rhombs, respectively, whereas internal states are depicted as circles. Middle: Initial fragments of the two possible runs of Δ. Steps are underlined. Right: The step Markov chain $\mathcal{M}_\Delta$ (Definition 4.14, page 22).

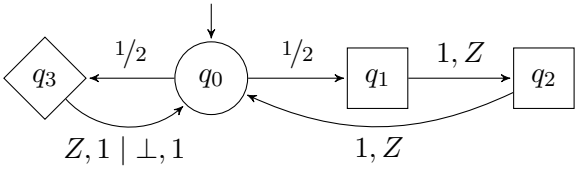
4.2.2. *Probabilities of next steps, returns, and diverges.* Our next goal is to provide expressions for the next-step probabilities $\mathbb{P}(V^{(n+1)} = v' \mid V^{(n)} = v)$ as we did in Example 4.6. It turns out that those can be stated in terms of the *return* and *diverge* probabilities of Δ.

Definition 4.7. Let $q, r \in Q$ be states, and $Z \in \Gamma_{\text{-}\bot}$ be a stack symbol of pVPA Δ. We define the following probabilities:

- The *return* probability $[qZ{\downarrow}r]$ is the probability to reach configuration $r\bot$ from $qZ\bot$ without visiting another bottom configuration $t\bot$ for some $t \neq r$ in between. Formally,
$$[qZ{\downarrow}r] \;=\; \mathbb{P}_{qZ\bot}(\,\{\, q'\gamma\bot \mid q' \in Q, \gamma \in \Gamma^+_{\text{-}\bot} \,\}\; \mathcal{U}\; \{r\bot\}\,)$$
where $\mathbb{P}_{qZ\bot}$ is the probability measure associated with the infinite Markov chain $\mathcal{D}_\Delta$ assuming initial state $qZ\bot$, and $\mathcal{U}$ is the standard until operator from LTL.
- The *diverge* probability $[q{\uparrow}] = 1 - \sum_{t\in Q}[qZ{\downarrow}t]$, i.e., the probability to never pop Z from the stack when starting in $qZ\bot$. Note that $[q{\uparrow}]$ is indeed independent of Z because the only way to read Z is by popping it from the stack. Recall that this is due to our specific definition of pVPA (Definition 3.1) in which only pop transitions can read from the stack just like in VPA (Definition 2.1); we remark that in traditional (p)PDA, all types of transition can read —and possibly replace— the topmost stack symbol [EKM04]. △

The diverge probabilities are closely related to steps. In fact, the probability that a non-bottom configuration with head qZ is a step is equal to $[q{\uparrow}]$. For example, in the pVPA in Figure 7 the configuration $q_1Z\bot$ is a step with probability $[q_1{\uparrow}] = 1/2$.

Example 4.8. It is well known that the return and diverge probabilities are not necessarily rational. We give a minimal example to illustrate this fact. Consider the following pVPA:



Intuitively, this pVPA either pops the topmost symbol with probability 1/2, or it pushes two times the symbol Z. Note that all return probabilities of the form $[\ldots{\downarrow}q_i]$ for $i \neq 0$

are equal to zero. In [EKM04] it was shown that the remaining return probabilities are the *component-wise least non-negative* solution of the *polynomial system*:

$$\begin{aligned}
[q_0 Z{\downarrow}q_0] &= \frac{1}{2} \cdot [q_1 Z{\downarrow}q_0] + \frac{1}{2} \cdot [q_3 Z{\downarrow}q_0] \\
[q_1 Z{\downarrow}q_0] &= [q_2 Z{\downarrow}q_0] \cdot [q_0 Z{\downarrow}q_0] \\
[q_2 Z{\downarrow}q_0] &= [q_0 Z{\downarrow}q_0] \cdot [q_0 Z{\downarrow}q_0] \\
[q_3 Z{\downarrow}q_0] &= 1 \ .
\end{aligned}$$

It follows that $[q_0 Z{\downarrow}q_0]$ must be the least non-negative solution of

$$[q_0 Z{\downarrow}q_0] = \frac{1}{2} \cdot [q_0 Z{\downarrow}q_0]^3 + \frac{1}{2}$$

which is $[q_0 Z{\downarrow}q_0] = \frac{\sqrt{5}-1}{2} \approx 0.618$, the reciprocal of the golden ratio. △

The probabilities in Example 4.8 can still be expressed by radicals (square roots, cubic roots, and so on) which allows for certain effective computations. However, in general, the probabilities cannot even be expressed in this way. For example, consider a pVPA that repeats the following steps until emptying its stack or getting stuck in a sink state: (i) It pushes four symbols with probability $\frac{1}{6}$, or (ii) pops one symbol with probability $\frac{1}{2}$, or (iii) gets stuck otherwise. The resulting return probability is the least $x \geq 0$ with $x = \frac{1}{6}x^5 + \frac{1}{2}$, which is an algebraic number not solvable by radicals [EY09, Theorem 3.2(1)].

Remark 4.9. The probabilities $[qZ{\downarrow}r]$ from Definition 4.7 were called *termination probabilities* in previous work [BEKK13]. However, we believe that *return* probability is more appropriate. When modeling procedural probabilistic programs as pVPA, $[qZ{\downarrow}r]$ is just the probability to eventually *return* from local state q of the current procedure to local state r of the calling procedure (the return address is stored on the stack in Z). We believe that the term *termination* probability is more adequate for referring to the quantity $\sum_{r \in Q} [q_0 Z_0 {\downarrow} r]$, where Z_0 is some initial stack symbol, i.e., the probability that some initial procedure indentified by Z_0 returns at all when started in local state q_0.

We now state the technical key lemma of this section, the characterization of the next step probabilities $\mathbb{P}(V^{(n+1)} = v' \mid V^{(n)} = v)$ as given in Table 2. The upcoming section is devoted to proving that the entries in the table are correct.

Lemma 4.10. *The conditional next-step probabilities in Table 2 are correct in the sense that if $\mathbb{P}(V^{(n+1)} = v' \mid V^{(n)} = v)$ is defined for $n \in \mathbb{N}_0$ and $v, v' \in Q \cup Q\bot$, then it is equal to the probability in the respective column $v \to v'$.*

4.2.3. *Proof of Lemma 4.10.* We first explain the trivial entries in Table 2. Further below, we give a self-contained proof of the two non-trivial expressions in the left-most column of Table 2. Throughout the whole proof we fix an (unlabed) pVPA $\Delta = (Q, q_0, \Gamma, \bot, P)$, with $P = (P_{\mathsf{call}}, P_{\mathsf{int}}, P_{\mathsf{ret}})$ the call, internal, and return transition functions, respectively. The following items correspond to the trivial entries in Table 2 and are ordered column-by-column, from left to right:

- The probability $\mathbb{P}(V^{(n+1)} = r \mid V^{(n)} = q)$ with $q \in Q_{\mathsf{ret}}$ is never well-defined because it is impossible that steps occur at non-bottom configurations with a return state.

Table 2: Next-step probabilities of the step Markov chain. P_{type} for $\mathsf{type} \in \{\mathsf{call}, \mathsf{int}, \mathsf{ret}\}$ are the probabilities of the pVPA's call, internal, and return transitions, respectively. The values $[r'Z{\downarrow}r]$ and $[q{\uparrow}]$ are the return and diverge probabilities from Definition 4.7.

	$q \to r$	$q\bot \to r$	$q\bot \to r\bot$	$q \to r\bot$
$q \in Q_{\mathsf{call}}$	$\frac{[r\uparrow]}{[q\uparrow]}\Big(\sum_{r',Z} P_{\mathsf{call}}(q,r'Z)[r'Z{\downarrow}r] + \sum_{Z} P_{\mathsf{call}}(q,rZ)\Big)$	$\sum_{Z} P_{\mathsf{call}}(q,rZ)[r\uparrow]$	$\sum_{r',Z} P_{\mathsf{call}}(q,r'Z)[r'Z{\downarrow}r]$	0
$q \in Q_{\mathsf{int}}$	$\frac{[r\uparrow]}{[q\uparrow]} P_{\mathsf{int}}(q,r)$	0	$P_{\mathsf{int}}(q,r)$	0
$q \in Q_{\mathsf{ret}}$	undef.	0	$P_{\mathsf{ret}}(q\bot,r)$	undef.

- The probability $\mathbb{P}(V^{(n+1)} = r \mid V^{(n)} = q\bot)$ with $q \in Q_{\mathsf{int}}$ is trivially zero because if q is internal then the next step after a bottom configuration $q\bot$ is necessarily also a bottom configuration.
- The probability $\mathbb{P}(V^{(n+1)} = r \mid V^{(n)} = q\bot)$ with $q \in Q_{\mathsf{ret}}$ is trivially zero because if q is a return state at a bottom configuration then the next step occurs at the immediate successor configuration which is a bottom configuration as well.
- The probability $\mathbb{P}(V^{(n+1)} = r\bot \mid V^{(n)} = q\bot)$ with $q \in Q_{\mathsf{int}}$ is straightforward because $q\bot$ and $r\bot$ are both steps and the probability that the next state after $q\bot$ is $r\bot$ is $P_{\mathsf{int}}(q,r)$.
- The probability $\mathbb{P}(V^{(n+1)} = r\bot \mid V^{(n)} = q\bot)$ with $q \in Q_{\mathsf{ret}}$ is simply $P_{\mathsf{ret}}(q\bot,r)$ for the same reason as in the previous case (recall that a return state at a bottom-configuration behaves exactly like an internal one).
- All the remaining probabilities in the rightmost column "$q \to r\bot$" are trivially zero or ill-defined because if a step occurs at non-bottom configuration, then the next step can never occur at a bottom configuration.

We now focus on the following non-trivial cases. Let $r \in Q$ and $n \in \mathbb{N}_0$ be arbitrary.

(1) If $q \in Q_{\mathsf{int}}$ then,

$$\mathbb{P}(V^{(n+1)} = r \mid V^{(n)} = q) \;=\; \frac{[r\uparrow]}{[q\uparrow]} P_{\mathsf{int}}(q,r)\ .$$

(2) If $q \in Q_{\mathsf{call}}$ then,

$$\mathbb{P}(V^{(n+1)} = r \mid V^{(n)} = q) \;=\; \frac{[r\uparrow]}{[q\uparrow]} \left(\sum_{r',Z} P_{\mathsf{call}}(q,r'Z)[r'Z{\downarrow}r] + \sum_{Z} P_{\mathsf{call}}(q,rZ)\right)\ .$$

The other two non-trivial cases are easier variants of case (2), hence we omit them here (see [WGK21, p. 30] for details). Next we provide some intuition about cases (1) and (2):

- For (1), suppose that the n-th step is at position i of the run. Since the n-th step occurs at an *internal* state $q \in Q_{\mathsf{int}}$, the $n{+}1$-st step must necessarily occur immediately at position $i{+}1$. The factor $P_{\mathsf{int}}(q,r)[r\uparrow]$ is proportional to the probability to take an (internal) transition from q to r and then diverge in r, which is necessary in order for the next configuration to be a step. However, the values $\{\, P_{\mathsf{int}}(q,r)[r\uparrow] \mid r \in Q \,\}$ do not form a probability distribution in general. Therefore we divide by the normalizing constant $[q\uparrow] = \sum_{r\in Q} P_{\mathsf{int}}(q,r)[r\uparrow]$.

- In (2), the two summands correspond to the following case distinction: Since the n-th step occurs at the *call* state $q \in Q_{\mathsf{call}}$, the $n{+}1$-st step either (i) occurs at the same stack height as the current step n (which means that the current call has a matching return), or (ii) the stack height at the next step $n{+}1$ is incremented by 1 compared to the stack height at step n. In case (ii), the next step occurs immediately at the next position, which is why the second summand is just the 1-step probability to go from q to r. In case (i), the symbols pushed by the outgoing transitions of q must be eventually popped. For instance, if we assume that q takes a transition to an immediate successor r' and pushes Z on the stack, then the probability that the next step occurs at r is precisely the *return probability* $[r'Z{\downarrow}r]$ (see Definition 4.7). The factor $\frac{[r\uparrow]}{[q\uparrow]}$ is needed for similar reasons as in (1).

We now give the formal proofs for (1) and (2). In the following, we often use equations involving conditional probabilities such as $\mathbb{P}(A \mid B) = \mathbb{P}(C \mid D)$. These conditional probabilities are not necessarily well-defined in all cases. Therefore, the meaning of our equations is that they hold only if all probabilities involved are well-defined. We need some definitions and (simple) lemmas first.

Definition 4.11. Let $i \in \mathbb{N}_0$ and $q \in Q$. We introduce the following events:

- $q@i$ is the set of all runs $\rho \in \mathit{Runs}_\Delta$ such that $\rho(i) = q\gamma$ with $\gamma \neq \bot$, i.e., the runs whose i-th configuration has state q and stack unequal to $\bot$.
- Similarly, $q\bot@i$ denotes the set of all runs $\rho \in \mathit{Runs}_\Delta$ such that $\rho(i) = q\bot$, i.e., the runs whose i-th configuration is a *bottom* configuration with state q.
- $\mathit{step}@i$ denotes the set of all runs such that the i-th configuration is a step.
- We define $\mathit{sh}@i = \mathit{sh}(\rho(i)) \in \mathbb{N}_0$, i.e., the stack height of the i-th configuration. Strictly speaking, $\mathit{sh}@i$ is a random variable, not an event. Note that $\mathit{step}@i$ is by definition equivalent to $\forall j > i\colon \mathit{sh}@j \geq \mathit{sh}@i$.

These events are all measurable. △

Lemma 4.12. *For all $i \in \mathbb{N}_0$, and $q \in Q$, the following identities hold:*

$$\mathbb{P}(\mathit{step}@i \mid q@i) \;=\; [q\uparrow] \tag{4.3}$$

$$\mathbb{P}(\mathit{step}@i \mid q\bot@i) \;=\; 1 \tag{4.4}$$

Further, for $q \in Q_{\mathsf{int}}$ and $r \in Q$,

$$\mathbb{P}(r@(i{+}1) \mid q@i) \;=\; P_{\mathsf{int}}(q,r) \tag{4.5}$$

$$\mathbb{P}(\mathit{step}@(i{+}1) \wedge \mathit{step}@i \mid r@(i{+}1) \wedge q@i) \;=\; \mathbb{P}(\mathit{step}@(i{+}1) \mid r@(i{+}1)) \tag{4.6}$$

Proof. The first three equations follow immediately from the definitions. For (4.6) we have:

$$\begin{aligned} & \mathbb{P}(\mathit{step}@(i{+}1) \wedge \mathit{step}@i \mid r@(i{+}1) \wedge q@i) \\ = \;& \mathbb{P}(\mathit{step}@(i{+}1) \mid r@(i{+}1) \wedge q@i) \\ = \;& \mathbb{P}(\mathit{step}@(i{+}1) \mid r@(i{+}1)) \end{aligned}$$

The first equation holds because if $q \in Q_{\mathsf{int}}$ and $q@i$, then $\mathit{step}@(i{+}1) \wedge \mathit{step}@i$ is already implied by $\mathit{step}@(i{+}1)$, and the second equation holds because the probability that $\mathit{step}@(i{+}1)$ depends only on the state at position $i{+}1$, not on the state at position i. □

To prove equation for case (1) we argue as follows. By time-homogeneity (see (4.2)) and the definition of $V^{(n)}$, we have for all $i, n \in \mathbb{N}_0$, $q \in Q_{\mathsf{int}}$ and $r \in Q$ that

$$\mathbb{P}(V^{(n+1)} = r \mid V^{(n)} = q) \;=\; \mathbb{P}(r@(i{+}1) \wedge \mathit{step}@(i{+}1) \mid q@i \wedge \mathit{step}@i) \tag{4.7}$$

Now:

$$\begin{aligned}
& \mathbb{P}(V^{(n+1)} = r \mid V^{(n)} = q) \\
= \;& \mathbb{P}(r@(i{+}1) \wedge \mathit{step}@(i{+}1) \mid q@i \wedge \mathit{step}@i) && \text{(by (4.7))} \\
= \;& \frac{\mathbb{P}(r@(i{+}1) \wedge \mathit{step}@(i{+}1) \wedge q@i \wedge \mathit{step}@i)}{\mathbb{P}(q@i \wedge \mathit{step}@i)} && \text{(cond. probability)} \\
= \;& \frac{\mathbb{P}(r@(i{+}1) \wedge \mathit{step}@(i{+}1) \wedge \mathit{step}@i \mid q@i) \cdot \mathbb{P}(q@i)}{\mathbb{P}(\mathit{step}@i \mid q@i) \cdot \mathbb{P}(q@i)} && \text{(cond. probability)} \\
= \;& \frac{\mathbb{P}(r@(i{+}1) \wedge \mathit{step}@(i{+}1) \wedge \mathit{step}@i \mid q@i)}{\mathbb{P}(\mathit{step}@i \mid q@i)} && \text{(simplification)} \\
= \;& \frac{\mathbb{P}(\mathit{step}@(i{+}1) \wedge \mathit{step}@i \mid r@(i{+}1) \wedge q@i) \cdot \mathbb{P}(r@(i{+}1) \mid q@i)}{\mathbb{P}(\mathit{step}@i \mid q@i)} && \text{(cond. probability)} \\
= \;& \frac{\mathbb{P}(\mathit{step}@(i{+}1) \mid r@(i{+}1)) \cdot \mathbb{P}(r@(i{+}1) \mid q@i)}{\mathbb{P}(\mathit{step}@i \mid q@i)} && \text{(by (4.6))} \\
= \;& \frac{[r\uparrow] \cdot P_{\mathsf{int}}(q, r)}{[q\uparrow]} \,. && \text{(by (4.3), (4.5))}
\end{aligned}$$

This concludes the proof for case (1).

We now turn to case (2). For all $i, n \in \mathbb{N}_0$, $q \in Q_{\mathsf{call}}$ and $r \in Q$, it holds that

$$\begin{aligned}
& \mathbb{P}(V^{(n+1)} = r \mid V^{(n)} = q) \\
= \;& \mathbb{P}(\exists k > i : \mathit{step}@k \wedge r@k \wedge \forall i < j < k : \neg \mathit{step}@j \mid \mathit{step}@i \wedge q@i) \\
& \qquad \text{(by time homogeneity and definition of } V^{(n)}) \\
= \;& \mathbb{P}(\exists k > i{+}1 : \mathit{step}@k \wedge r@k \wedge \forall i < j < k : \neg \mathit{step}@j \mid \mathit{step}@i \wedge q@i) \\
& + \; \mathbb{P}(r@(i{+}1) \wedge \mathit{step}@(i{+}1) \mid q@i \wedge \mathit{step}@i)
\end{aligned} \tag{4.8}$$

The last equality results from a split in two disjoint events. For the second summand in (4.8) it can be shown that

$$\mathbb{P}(r@(i{+}1) \wedge \mathit{step}@(i{+}1) \mid q@i \wedge \mathit{step}@i) \;=\; \frac{[r\uparrow]}{[q\uparrow]} \sum_{Z} P_{\mathsf{call}}(q, rZ)$$

by a similar derivation as in case (1) (the sum over all stack symbols Z is because $q \in Q_{\mathsf{call}}$, so that there may be multiple —up to $|\Gamma_{\text{-}\perp}|$ many— direct transitions from q to r).

We need a couple of lemmas before deriving an equation for the first summand in (4.8).

Lemma 4.13. *For all $q \in Q_{\mathsf{call}}$, $r \in Q$, and $i \in \mathbb{N}_0$ it holds that:*

$$\sum_{r', Z} P_{\mathsf{call}}(q, r'Z)[r'Z{\downarrow}r] \;=\; \sum_{k > i+1} \mathbb{P}(r@k \wedge \forall i{<}j{<}k : \mathit{sh}@j > \mathit{sh}@k = \mathit{sh}@i \mid q@i) \tag{4.9}$$

Moreoever, for all $q \in Q_{\mathsf{call}}$, $i \in \mathbb{N}_0$, and $k \in \mathbb{N}_0$, we have:

$$\begin{aligned}
& \mathbb{P}(\mathit{step}@k \wedge \forall i{<}j{<}k : \neg \mathit{step}@j \wedge \mathit{step}@i \mid q@i) \\
= \; & \mathbb{P}(\mathit{step}@k \wedge \forall i{<}j{<}k : \mathit{sh}@j > \mathit{sh}@i = \mathit{sh}@k \mid q@i)
\end{aligned} \tag{4.10}$$

and

$$\begin{aligned}
& \mathbb{P}(\forall i{<}j{<}k : \mathit{sh}@j > \mathit{sh}@i = \mathit{sh}@k \mid q@i \wedge r@k \wedge \mathit{step}@k) \\
= \; & \mathbb{P}(\forall i{<}j{<}k : \mathit{sh}@j > \mathit{sh}@i = \mathit{sh}@k \mid q@i \wedge r@k)
\end{aligned} \tag{4.11}$$

Proof. For (4.9), note that $\sum_{r',Z} P_{\mathsf{call}}(q, r'Z)[r'Z{\downarrow}r]$ is the probability to go from q (with empty stack) to a successor state r', push $Z \in \Gamma$ and then later reach r with empty stack within finitely many steps. If we assume that $q@i$, then this is the same as summing over all positions $k > i+1$ (we can exclude $k = i+1$ because is not possible because to push and pop Z within one transition) such that $r@k$ and for all $i < j < k$ the stack height is greater than at position i and k. Positions i and k have the same stack height because in the transition from i the symbol Z is pushed, and k is the position directly after Z is popped. In between those two transitions, the stack below Z cannot change, so the stack is the same at both positions.

For (4.10) we argue as follows:

$$\begin{aligned}
&\mathbb{P}(\mathit{step}@k \wedge \forall i{<}j{<}k : \neg \mathit{step}@j \wedge \mathit{step}@i \mid q@i)\\
&= \mathbb{P}(\mathit{step}@k \wedge \forall i{<}j{<}k : \mathit{sh}@j > \mathit{sh}@i = \mathit{sh}@k \wedge \mathit{step}@i \mid q@i)\\
&= \mathbb{P}(\mathit{step}@k \wedge \forall i{<}j{<}k : \mathit{sh}@j{>}\mathit{sh}@i = \mathit{sh}@k \mid q@i)
\end{aligned}$$

The first equality holds because if no position between i and k is a step, then the stack at those positions must be higher than at position k. Furthermore, since i is a step, we have $\mathit{sh}@i \leq \mathit{sh}@k$; and moreover, since $i{+}1$ is not a step and q is a call state, we even have $\mathit{sh}@i = \mathit{sh}@k$. The second equality holds because if i and k have the same stack height and all positions between them have a higher stack, then i is a step if and only of k is a step.

Equation (4.11) is somewhat counter-intuitive because conditioning on $\mathit{step}@k$ is like "conditioning on the future": The stack height *after* position k should never be smaller than at position k. Knowing that $\mathit{step}@k$ gives information about the (extended) state at position k. However, in (4.11) we also condition on the fact that $r@k$, i.e., at position k, the run is in step r and the topmost stack symbol is not $\bot$. Hence, in the context of (4.11), we can drop $\mathit{step}@k$ from the condition. □

We conclude the proof of case (2) and thus of the whole Lemma 4.10 by deriving the desired equation for the first summand in (4.8):

$$\begin{aligned}
&\mathbb{P}(\exists k{>}i{+}1 : \mathit{step}@k \wedge r@k \wedge \forall i{<}j{<}k : \neg \mathit{step}@j \mid \mathit{step}@i \wedge q@i)\\
=\ & \frac{\mathbb{P}(\exists k{>}i{+}1 : \mathit{step}@k \wedge r@k \wedge \forall i{<}j{<}k : \neg \mathit{step}@j \wedge \mathit{step}@i \mid q@i) \cdot \mathbb{P}(q@i)}{\mathbb{P}(\mathit{step}@i \mid q@i) \cdot \mathbb{P}(q@i)} && \text{(cond. probability twice)}\\
=\ & \frac{\mathbb{P}(\exists k{>}i{+}1 : \mathit{step}@k \wedge r@k \wedge \forall i{<}j{<}k : \neg \mathit{step}@j \wedge \mathit{step}@i \mid q@i)}{\mathbb{P}(\mathit{step}@i \mid q@i)} && \text{(simplification)}\\
=\ & \sum_{k>i+1} \frac{\mathbb{P}(\mathit{step}@k \wedge r@k \wedge \forall i{<}j{<}k : \neg \mathit{step}@j \wedge \mathit{step}@i \mid q@i)}{\mathbb{P}(\mathit{step}@i \mid q@i)} && \text{(split disjoint events)}\\
=\ & \sum_{k>i+1} \frac{\mathbb{P}(\mathit{step}@k \wedge r@k \wedge \forall i{<}j{<}k : \mathit{sh}@j > \mathit{sh}@i = \mathit{sh}@k \mid q@i)}{\mathbb{P}(\mathit{step}@i \mid q@i)} && \text{(by (4.10))}\\
=\ & \sum_{k>i+1} \Bigg(\mathbb{P}(\mathit{step}@k \wedge r@k \mid q@i)\\
& \qquad \cdot \frac{\mathbb{P}(\forall i{<}j{<}k : \mathit{sh}@j > \mathit{sh}@i = \mathit{sh}@k \mid q@i \wedge \mathit{step}@k \wedge r@k)}{\mathbb{P}(\mathit{step}@i \mid q@i)} \Bigg) && \text{(cond. probability)}
\end{aligned}$$

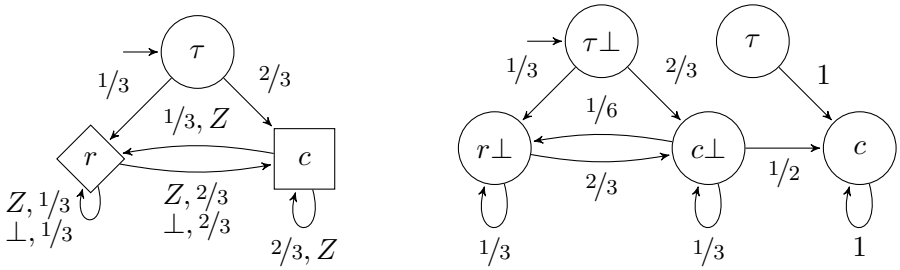


Figure 8: Left: Example pVPA Δ with the following return and diverge probabilities: $[cZ{\downarrow}c] = 1/6$, $[cZ{\downarrow}r] = 1/12$, $[rZ{\downarrow}r] = 1/3$, $[rZ{\downarrow}c] = 2/3$, and $[c{\uparrow}] = 3/4$, $[\tau{\uparrow}] = 1/2$, $[r{\uparrow}] = 0$. In general, these probabilities may be irrational numbers [EY09]. Right: The step chain $\mathcal{M}_\Delta$ according to Definition 4.14. The transition probabilities can be computed using the return and diverge probabilities and Table 2.

$$= \sum_{k>i+1} \mathbb{P}(step@k \wedge r@k \mid q@i) \cdot \frac{\mathbb{P}(\forall i{<}j{<}k : sh@j > sh@i = sh@k \mid q@i \wedge r@k)}{\mathbb{P}(step@i \mid q@i)}$$

(by (4.11))

$$= \sum_{k>i+1} \frac{\mathbb{P}(step@k \mid q@i \wedge r@k)}{\mathbb{P}(step@i \mid q@i)} \cdot \mathbb{P}(\forall i{<}j{<}k : sh@j > sh@i = sh@k \wedge r@k \mid q@i)$$

(rewriting)

$$= \frac{[r{\uparrow}]}{[q{\uparrow}]} \sum_{k>i+1} \mathbb{P}(r@k \wedge \forall i{<}j{<}k : sh@j > sh@i = sh@k \mid q@i)$$

(by (4.3) and noticing that $\mathbb{P}(step@k \mid q@i \wedge r@k) = \mathbb{P}(step@k \mid r@k)$)

$$= \frac{[r{\uparrow}]}{[q{\uparrow}]} \sum_{r',Z} P_{\mathsf{call}}(q, r'Z)[r'Z{\downarrow}r]$$

(by (4.9))

This concludes the proof of Lemma 4.10. □

4.2.4. *The step chain.* Recall from (4.1) that the stochastic process $V^{(0)}, V^{(1)} \ldots$, where $V^{(i)} \in Q \cup Q\bot$ is the extended state at the ith step, has the Markov property. Since $Q \cup Q\bot$ is a finite set, we can now use Lemma 4.10 to construct the underlying finite Markov chain explicitly.

Definition 4.14 (The Step Chain $\mathcal{M}_\Delta$)**.** $\mathcal{M}_\Delta$ is the Markov chain with states

$$M \quad = \quad \{\, q \in Q_{\mathsf{call}} \cup Q_{\mathsf{int}} \mid [q{\uparrow}] > 0 \,\} \cup Q\bot \,,$$

initial state $q_0\bot$, and for all $v, v' \in M$, the probability of transition $v \to v'$ is defined according to Table 2. △

Figure 8 depicts a non-trivial pVPA and its step chain. In this example, all return and diverge probabilities are rational. In general, however, the return and diverge probabilities (Definition 4.7) are algebraic numbers that are not always rational or even expressible by radicals [EY09] (cf. Example 4.8). As a consequence, one cannot easily perform numerical computations on the step chain. However, the probabilities can be encoded implicitly as the unique solution of an *existential theory of the reals* (ETR) formula, i.e., an existentially

quantified FO-formula over $(\mathbb{R}, +, \cdot, \leq)$ [EKM04]. Since the ETR is decidable in PSPACE, many questions about the step chain are in PSPACE as well. We will make use of this in Theorem 4.16 below.

The property of $\mathcal{M}_\Delta$ that is most relevant to us is given by the following Lemma 4.15. For $\rho \in Runs_\Delta$, we let $\rho\Downarrow_{Steps} = V^{(0)}(\rho)V^{(1)}(\rho)\ldots \in (Q \cup Q\bot)^\omega$ (note that this is a slightly different "footprint" than the one introduced in Section 2.2).

Lemma 4.15 (Soundness of $\mathcal{M}_\Delta$)**.** *Let Δ be a pVPA with step chain $\mathcal{M}_\Delta$. Let M be the states of the step chain and let $R \subseteq M^\omega$ be measurable. Then*

$$\mathbb{P}_\Delta(\{\, \rho \in Runs_\Delta \mid \rho\Downarrow_{Steps} \in R \,\}) \quad = \quad \mathbb{P}_{\mathcal{M}_\Delta}(R)$$

where $\mathbb{P}_\Delta$ and $\mathbb{P}_{\mathcal{M}_\Delta}$ are the probability measures associated with Δ and $\mathcal{M}_\Delta$, respectively.

Proof. The formal proof requires some basic notions from measure theory. In fact, Lemma 4.15 is actually an instance of the following more general statement:

($\star$) *Let $(X, \mathcal{A}, \mu)$ and $(Y, \mathcal{B}, \nu)$ be probability spaces such that $\mathcal{B} = \sigma(\mathcal{G})$, where $\mathcal{G} \subseteq 2^Y$, i.e., $\mathcal{B}$ is the σ-algebra generated by the sets in $\mathcal{G}$. Assume furthermore that $\mathcal{G}$ is a π-system, i.e., $\mathcal{G}$ is non-empty and closed under finite intersections. Let $f\colon X \to Y$ be such that for all $G \in \mathcal{G}$, $f^{-1}(G) \in \mathcal{A}$ and $\mu(f^{-1}(G)) = \nu(G)$. Then f is a measurable function and the pushforward measure $\nu' = \mu \circ f^{-1}$ coincides with ν.*

We now explain how to prove ($\star$) using fundamental measure theory. The fact that f is measurable follows because the inverse image f^{-1} preserves set operations (see, e.g., [ADD00, below Definition 1.5.1]). For the claim that $\nu' = \nu$ it suffices to note that by assumption we have for all $G \in \mathcal{G}$ that $\nu'(G) = \nu(G)$, and since $\mathcal{G}$ is a π-system, an application of the π-λ theorem (see, e.g., [Pan09, Proposition 2.10]) implies that $\nu' = \nu$.

We instantiate ($\star$) as follows to prove Lemma 4.15: The probability spaces are the ones associated with the measures $\mathbb{P}_\Delta$ and $\mathbb{P}_{\mathcal{M}_\Delta}$. In particular, the σ-algebra on which $\mathbb{P}_{\mathcal{M}_\Delta}$ is defined is the one generated by the cylinder sets $\mathcal{C} = \{w.M^\omega \mid w \in M^*\} \subseteq 2^{M^\omega}$. It is easy to see that $\mathcal{G} = \mathcal{C} \cup \{\emptyset\}$ is a π-system, and $\sigma(\mathcal{C}) = \sigma(\mathcal{G})$. We define $f\colon (Q \times \Gamma^*_{\bot}.\bot)^\omega \to M^\omega, f(\rho) = \rho\Downarrow_{Steps}$, i.e., f projects a run from Δ to its footprint of steps (which is a run in $\mathcal{M}_\Delta$). To apply ($\star$) it remains to show that for all cylinder sets $w.M^\omega$, $w \in M^*$, we have that (i) $f^{-1}(w.M^\omega)$ is measurable, and (ii) $\mathbb{P}_\Delta(f^{-1}(w.M^\omega)) = \mathbb{P}_{\mathcal{M}_\Delta}(w.M^\omega)$. For (i) notice that $f^{-1}(w.M^\omega) = \bigwedge_{i=0}^{|w|-1}(V^{(i)} = w(i))$ is indeed measurable because it is a finite intersection of measurable events by Lemma 4.5; recall that $V^{(i)}$ denotes the (extended) state at the ith step. (ii) is trivial in the case where $|w| = 0$, so we let $|w| = n + 1$, $n \geq 0$, and exploit the properties of the step chain $\mathcal{M}_\Delta$. If $w(0) \neq q_0\bot$ (the initial state of $\mathcal{M}_\Delta$) then $\mathbb{P}_\Delta(V^{(0)} = w(0)) = \mathbb{P}_{\mathcal{M}_\Delta}(w.M^\omega) = 0$. Otherwise $w(0) = q_0\bot$. In this case, if $n = 0$ (i.e., $|w| = 1$), then $\mathbb{P}_\Delta(V^{(0)} = w(0)) = \mathbb{P}_{\mathcal{M}_\Delta}(w.M^\omega) = 1$. Else, if $n > 0$ ($|w| > 1$), by the Markov property from equation (4.1), we have

$$\begin{aligned}
& \mathbb{P}_\Delta(V^{(n)} = w(n) \wedge \ldots \wedge V^{(0)} = w(0)) \\
= \; & \mathbb{P}_\Delta(V^{(n)} = w(n) \mid V^{(n-1)} = w(n-1)) \cdot \ldots \cdot \mathbb{P}_\Delta(V^{(1)} = w(1) \mid V^{(0)} = w(0)) \\
= \; & P(w(n-1), w(n)) \cdot \ldots \cdot P(w(0), w(1)) \qquad \text{(By Lemma 4.10)} \\
= \; & \mathbb{P}_{\mathcal{M}_\Delta}(w.M^\omega)
\end{aligned}$$

where P is the transition probability function in the Markov chain $\mathcal{M}_\Delta$ and the last equality holds by definition of the probability measure $\mathbb{P}_{\mathcal{M}_\Delta}$. □

Table 3: The underlying graph of the step chain. The condition in each cell is true iff the corresponding transition probability in Table 2 is non-zero.

	$q \to r$	$q\bot \to r$	$q\bot \to r\bot$	$q \to r\bot$
$q \in Q_{\mathsf{call}}$	$[r\uparrow] > 0 \ \wedge\ ($ $(\exists r', Z\colon P_{\mathsf{call}}(q, r'Z) > 0 \wedge [r'Z{\downarrow}r]) > 0)$ $\vee\ \exists Z\colon P_{\mathsf{call}}(q, rZ) > 0)$	$\exists Z\colon P_{\mathsf{call}}(q, rZ) > 0$ $\wedge\ [r\uparrow] > 0$	$\exists r', Z\colon P_{\mathsf{call}}(q, rZ) > 0$ $\wedge\ [r'Z{\downarrow}r] > 0$	false
$q \in Q_{\mathsf{int}}$	$[r\uparrow] > 0 \ \wedge\ P_{\mathsf{int}}(q, r) > 0$	false	$P_{\mathsf{int}}(q, r) > 0$	false
$q \in Q_{\mathsf{ret}}$	false	false	$P_{\mathsf{ret}}(q\bot, r) > 0$	false

4.3. **Main Result of Section 4.** The following is the main result of Section 4:

Theorem 4.16. *Let Δ be a pVPA and $\mathcal{D}$ be a stair-parity DVPA, both over the same pushdown alphabet Σ. Then for all $\theta \in [0,1] \cap \mathbb{Q}$, the following is decidable in* PSPACE*:*

$$\mathbb{P}(\{\rho \in \mathit{Runs}_\Delta \mid \lambda(\rho) \in \mathcal{L}(\mathcal{D})\}) \geq_? \theta \ .$$

The rest of Section 4.3 is devoted to the proof of Theorem 4.16. We first provide a brief overview. The first step is to construct the product $\Delta \times \mathcal{D}$ according to Definition 4.1. By Lemma 4.2 we need to compute the stair-parity acceptance probability of $\Delta \times \mathcal{D}$. Lemma 4.15 reduces this to computing a usual parity acceptance probability in the step chain $\mathcal{M}_{\Delta\times\mathcal{D}}$. This can be achieved through finding the bottom strongly connected components (BSCC) of $\mathcal{M}_{\Delta\times\mathcal{D}}$, classifying them as *good* (the minimum priority of a BSCC state is even) or otherwise as *bad*, and running a standard reachability analysis w.r.t. the good BSCCs (see Figure 9 for an example). The remaining technical difficulty is that the transition probabilities of $\mathcal{M}_{\Delta\times\mathcal{D}}$ are not rational in general. We handle this using the fact that these probabilities are expressible in the ETR [EKM04].

We now present the formal proof. We use the following result about return probabilities of pPDA, which is originally due to [EKM04]:

Lemma 4.17 (as stated in [BEKK13, Theorem 2])**.** *The return probabilities $[pZ{\downarrow}q]$ are expressible in ETR. More specifically, there exists an FO-formula Φ over $(\mathbb{R}, +, \cdot, \leq)$ which uses just existential quantifiers and free variables $\langle pZ{\downarrow}q\rangle_{p,q\in Q, Z\in\Gamma}$ such that Φ becomes a true FO-sentence iff each free variable $\langle pZ{\downarrow}q\rangle$ is substituted by the actual return probability $[pZ{\downarrow}q]$. Moreover, Φ can be effectively constructed in polynomial space.*

Lemma 4.18. *The next-step probabilities (i.e., the transition probabilities of the step chain) in Table 2 are expressible in ETR.*

Proof. With Lemma 4.17 it suffices to note that ETR expressible numbers are closed under addition, multiplication and division. Let $x, y \in \mathbb{R}$ be expressed by ETR formulae $\Phi(x)$ and $\Phi'(y)$, respectively. Then the formula $\Phi''(z) \coloneqq \exists x, y\colon \Phi(x) \wedge \Phi'(y) \wedge z = x + y$ where z is a fresh variable expresses the sum of x and y, and similar for multiplication. For division, we have that $\Phi''(z) \coloneqq \exists x, y\colon \Phi(x) \wedge \Phi'(y) \wedge z \cdot y = x$ expresses x/y provided that $y \neq 0$ (if $y = 0$ then Φ'' does not express a unique real number as it is then either unsatisfiable or trivial). □

We now describe our PSPACE algorithm to prove Theorem 4.16.

Step 1. We first construct the product pVPA $\tilde{\Delta} = \Delta \times \mathcal{D}$ with priority function $\Omega'\colon Q \to \mathbb{N}_0$ where Q are the states of $\tilde{\Delta}$ as in Definition 4.1. By Lemma 4.2 it suffices to

compute the probability

$$\mathbb{P}(\{\, \pi \in \mathit{Runs}_{\tilde{\Delta}} \mid \rho{\downarrow_{\mathit{Steps}}} \in \mathit{Parity}_{\Omega'} \,\}) \tag{4.12}$$

that the footprint of a run in the product $\tilde{\Delta}$ satisfies the parity condition induced by Ω'. $\tilde{\Delta}$ can be constructed in polynomial time.

Step 2. We express (4.12) using the step chain $\mathcal{M}_{\tilde{\Delta}}$. Let $M \subseteq Q \cup Q\bot$ be the states of the step chain $\mathcal{M}_{\tilde{\Delta}}$. Let $\Omega''\colon M \to \mathbb{N}_0$ be the extension of Ω' to the states of M via $\Omega''(q) = \Omega''(q\bot) = \Omega'(q)$ for all $q \in Q$. That is, Ω'' induces a parity acceptance set $\mathit{Parity}_{\Omega''} \subseteq M^\omega$ which is ω-regular and thus measurable. Let $\rho \in \mathit{Runs}_{\tilde{\Delta}}$. Clearly, $\rho{\downarrow_{\mathit{Steps}}} \in \mathit{Parity}_{\Omega'}$ iff $\rho{\Downarrow_{\mathit{Steps}}} \in \mathit{Parity}_{\Omega''}$. Thus by Lemma 4.15, (4.12) is equal to

$$\mathbb{P}(\{\, \pi \in \mathit{Runs}_{\tilde{\Delta}} \mid \rho{\downarrow_{\mathit{Steps}}} \in \mathit{Parity}_{\Omega'} \,\}) \quad = \quad \mathbb{P}(\mathit{Parity}_{\Omega''})$$

where the right hand side is the probability that a run of the step chain $\mathcal{M}_{\tilde{\Delta}}$ satisfies the parity condition induced by Ω''. We have thus reduced the problem of computing a stair-parity acceptance probability in the product pPDA $\tilde{\Delta}$ to computing a standard parity acceptance probability in the finite Markov chain $\mathcal{M}_{\tilde{\Delta}}$.

The rest of the proof uses standard techniques and is similar to the proof of [EKM04, Theorem 5.15]. The main technical difficulty is that the transition probabilities of $\mathcal{M}_{\tilde{\Delta}}$ cannot be written in an explicit numerical form since they are in general algebraic numbers.

Step 3. We construct the *underlying graph* $G_{\tilde{\Delta}}$ of the step chain $\mathcal{M}_{\tilde{\Delta}}$, i.e., we determine the set M of states and include a directed edge between states $v, v' \in M$ iff the probability of transition $v \to v'$ is positive. Table 3 gives sufficient and necessary conditions for this in all cases (Table 3 can be seen as the "qualitative" version of Table 2). The conditions in Table 3 (as well as constructing the state space of $\mathcal{M}_{\tilde{\Delta}}$) require checking if $[q\uparrow] > 0$ for states $q \in Q$ of $\tilde{\Delta}$. The latter is equivalent to checking if $\sum_{r \in Q}[qZ{\downarrow}r] < 1$, which is reducible to ETR by Lemma 4.17 and hence decidable in PSPACE.

Step 4. We determine the bottom strongly connected components (BSCC) of $G_{\tilde{\Delta}}$ from the previous step by a standard (efficient) graph analysis. We mark the BSCCs $B \subseteq M$ such that $\min_{v \in B} \Omega''(v)$ is even as "good", the others as "bad". It is well-known that in the finite Markov chain $\mathcal{M}_{\tilde{\Delta}}$ it holds that

$$\mathbb{P}(\rho \in \mathit{Parity}_{\Omega''} \mid \rho \text{ reaches a good BSCC}) = 1$$

due to the Long-Run Theorem of finite Markov chains: Each state of a BSCC is visited infinitely often almost-surely, provided that this BSCC is reached at all. Moreover, if a run ρ reaches a bad BSCC, than the probability to satisfy the parity condition is zero and thus

$$\begin{aligned}\mathbb{P}(\rho \in \mathit{Parity}_{\Omega''}) &= \mathbb{P}(\rho \in \mathit{Parity}_{\Omega''} \mid \rho \text{ reaches a good BSCC}) \cdot \mathbb{P}(\rho \text{ reaches a good BSCC}) \\ &= \mathbb{P}(\rho \text{ reaches a good BSCC})\end{aligned}$$

Thus it only remains to compute the probability to reach a good BSCC in $\mathcal{M}_{\tilde{\Delta}}$.

Step 5. We use the previous step to classify the states M of the step chain $\mathcal{M}_{\tilde{\Delta}}$ into three categories: $M_{=0}$ contains all states from which no good BSCC is reachable in the graph $G_{\tilde{\Delta}}$, $M_{=1}$ contains all good BSCCs, and $M_?$ contains all other states. We recall that the probabilities to reach $M_{=1}$ are the *unique* solution of the following linear equation system

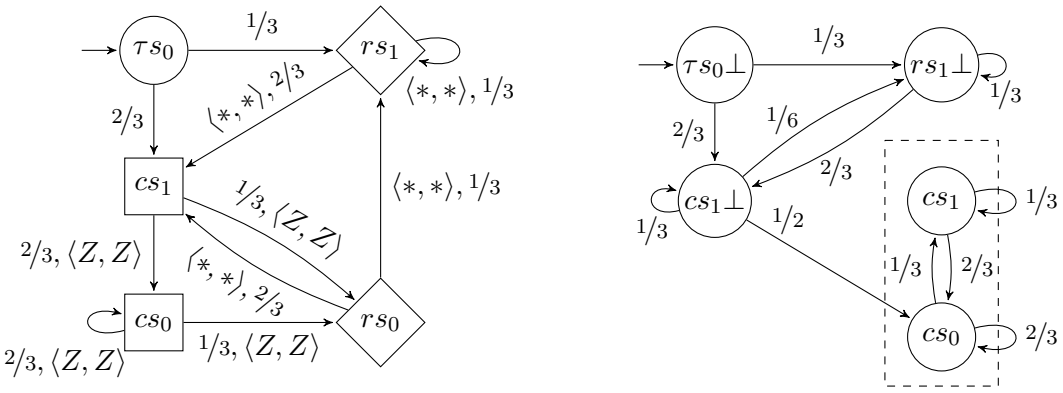


Figure 9: Left: The product of the pVPA from Figure 8 (left) and the DVPA from Figure 4 on page 7. Right: Its step chain according to Definition 4.14. The dashed region is the only BSCC. It violates the parity condition $\Omega(s_0) = 1$ and $\Omega(s_1) = 2$ inherited from the DVPA (see Example 2.7 on page 8) since every run reaching the BSCC visits cs_0 infinitely often with probability 1. Only reachable states are depicted. $*$ is a placeholder that stands for an arbitrary stack symbol.

(see, e.g., [BK08, Ch. 10]):

$$\begin{aligned} & x_v = 0 && \text{if } v \in M_{=0} \\ \wedge \quad & x_v = 1 && \text{if } v \in M_{=1} \\ \wedge \quad & x_v = \sum_{v \xrightarrow{p} v'} p x_{v'} && \text{if } v \in M_? \ . \end{aligned}$$

We can treat the vectors of probabilities $\vec{p}$ and the variables $\vec{x}$ in this equation system as free variables of an ETR formula $\mathcal{R}(\vec{p}, \vec{x})$. By Lemma 4.18, there is an ETR formula $\Phi(\vec{p})$ expressing $\vec{p}$. The ETR formula

$$\exists \vec{p}, \vec{x}\colon \quad \Phi(\vec{p}) \quad \wedge \quad \mathcal{R}(\vec{p}, \vec{x}) \quad \wedge \quad x_{v_0} \geq \theta$$

is thus true iff the probability to reach a good BSCC from initial state $v_0 \in M$ is at least θ. Truth of this formula can be decided in PSPACE, which concludes the proof of Theorem 4.16. □

4.4. **Implications for Probabilistic One-counter Automata.** A probabilistic visibly *one-counter automaton* (pVOC) is the special case of a pVPA with unary stack alphabet, i.e., $|\Gamma_{\text{-}\bot}| = 1$. For example, the pVPA in Figure 8 (left) is a pVOC. For many problems, better complexity bounds are known for pVOC than for the general case. In particular, $[p\uparrow] >_? 0$, i.e., the question whether a pVOC started in state p never reaches counter value (or stack height) zero with positive probability, can be decided in P [BEKK13, Theorem 4]. We can exploit this to improve Theorem 4.16 in the pVOC case:

Corollary 4.19. *Let Δ be a pVOC and $\mathcal{D}$ be a stair-parity DVPA over pushdown alphabet Σ. The problem $\mathbb{P}(\{\rho \in Runs_\Delta \mid \lambda(\rho) \in \mathcal{L}(\mathcal{D})\}) =_? 1$ is decidable in* P.

Proof. The key observation is that, since we can efficiently decide $[p\uparrow] >_? 0$, we can efficiently (in polynomial time) construct the underlying graph $G_{\Delta \times \mathcal{D}}$ of the step chain of $\Delta \times \mathcal{D}$ (as

in the proof of Theorem 4.16), and then apply polynomial-time graph analysis algorithms to check if only good BSCCs are reachable in $G_{\Delta\times\mathcal{D}}$. □

Corollary 4.19 implies that there exist efficient algorithms for many properties of pVOC-expressible random walks on $\mathbb{N}_0$. In fact, almost-sure satisfaction of each *fixed* visibly-pushdown property can be decided in P. For instance, using the DVPA from Figure 4 we can decide if a random walk is repeatedly bounded with probability 1.

5. Model Checking against Büchi VPA and CaRet

With Theorem 2.8 and theorem 4.16 it follows immediately that quantitative model checking of pVPA against non-deterministic Büchi VPA is decidable in EXPSPACE. We can improve the complexity in the qualitative case:

Theorem 5.1. *Let Δ be a pVPA and $\mathcal{A}$ be a (non-deterministic) Büchi VPA over the same pushdown alphabet. The problem $\mathbb{P}(\{\rho \in Runs_\Delta \mid \lambda(\rho) \in \mathcal{L}(\mathcal{A})\}) =_? 1$ is* EXPTIME*-complete.*

Proof. The lower bound is due to [EY05, Theorem 8] and already holds for non-pushdown Büchi automata. We now describe an EXPTIME decision procedure:

- We first determinize $\mathcal{A}$ using Theorem 2.8 which is possible in time exponential in $|\mathcal{A}|$. Let $\mathcal{D}$ be the resulting stair-parity DVPA and consider the product $\Delta \times \mathcal{D}$ (Definition 4.1). Note that the product can be constructed in polynomial time in $|\mathcal{D}|$ and $|\Delta|$, and thus in exponential time in the overall size of the input.
- The crucial observation for the next step is that $[q\uparrow] = [(q,s)\uparrow]$ for all states (q,s) of $\Delta \times \mathcal{D}$. This holds because by definition of the product, $\mathcal{D}$ merely *observes* the runs of Δ, and thus the diverge probabilities of $\Delta \times \mathcal{D}$ and Δ are essentially the same. We compute the set $Div_\Delta = \{q \mid [q\uparrow] > 0\} \subseteq Q$, where Q are the states of Δ, in *exponential time in* $|\Delta|$ using a PSPACE decision procedure for the ETR [EKM04]. Note that computing $Div_{\Delta\times\mathcal{D}} = \{(q,s) \mid [(q,s)\uparrow] > 0\}$ directly would take *doubly-exponential* time in $|\mathcal{A}|$; the proposed "optimization" is thus essential for obtaining the EXPTIME upper bound.
- We now determine the set of triples $Ret_{\Delta\times\mathcal{D}} = \{(q,Z,p) \mid [qZ{\downarrow}p] > 0\}$ in $\Delta \times \mathcal{D}$. Unlike the diverge probabilities, this set can be computed in *polynomial* time in the size of $\Delta \times \mathcal{D}$ (hence exponential in the size of the input) because we may disregard the exact transition probabilities and conduct a standard reachability analysis in a non-deterministic pushdown automaton [ABE18], also see [BEKK13, p. 136].
- The next step is to construct the underlying *graph* $G_{\Delta\times\mathcal{D}}$ of the step chain $\mathcal{M}_{\Delta\times\mathcal{D}}$, i.e., the directed graph that has the same vertices as $\mathcal{M}_{\Delta\times\mathcal{D}}$ and includes an edge (u,v) iff the 1-step transition probability from u to v is positive in the Markov chain. This can be done in polynomial time in $|\Delta \times \mathcal{D}|$ using the sets Div_Δ and $Ret_{\Delta\times\mathcal{D}}$ defined above and Table 3.
- The final step is, as in Theorem 4.16, to determine the BSCCs of $G_{\Delta\times\mathcal{D}}$, classify them as good or bad according to whether they satisfy the (standard) parity condition inherited from $\mathcal{D}$, and then check if there is a bad BSCC reachable from the initial state. All these steps can be done in polynomial time in $|\Delta \times \mathcal{D}|$. □

In the above result, membership in EXPTIME relies on the fact that one can construct the underlying *graph* of a step chain $\mathcal{M}_{\Delta\times\mathcal{D}}$ in time exponential in the size of Δ but *polynomial* in the size of $\mathcal{D}$. EXPTIME-hardness follows from [EY05, Theorem 8]. In fact, qualitative model checking of pPDA against *non-pushdown* Büchi automata is also EXPTIME-complete [EY05].

With Theorems 2.8 and 2.11 and theorems 4.16 and 5.1 we immediately obtain the following complexity results for CaRet model checking:

Theorem 5.2. *The quantitative and qualitative probabilistic CaRet model checking problems (Definition 3.5) are decidable in* 2EXPSPACE *and* 2EXPTIME*, respectively.*

Both problems are known to be EXPTIME-hard [YE05].

6. Conclusion

We have presented the first decidability result for model checking probabilistic pushdown automata—an operational model of recursive discrete probabilistic programs—against CaRet, or more generally, against the class of ω-VPL. We heavily rely on the determinization procedure from [LMS04, Theorem 1] and the notion of a step chain used in previous works [EKM04, KEM06]. These two constructions turn out to be a natural match.

We conjecture that the upper bounds from Theorem 5.2 are not tight due to the exponential blow up incurred by applying the VPA determinization from [LMS04, Theorem 1]. Future work is thus to investigate whether the doubly-exponential complexity can be lowered to singly-exponential, e.g., by generalizing the automata-less algorithm from [YE05]. Another open question is whether existing results [EY09, EKL10, SEY15] for *approximately* computing the probabilities $[qZ{\downarrow}r]$ can be used for approximate quantitative CaRet and ω-VPL model checking. We also plan to extend our recent work on *certificates* [WK23a, WK23b] to temporal and other logical properties. Such certificates can be approximate as well.

Other future work includes exploring to what extent algorithms for probabilistic CTL can be generalized to the branching-time variant of CaReT [GMN18], considering more expressive logics such as visibly LTL [BS18] or OPTL [CMP20], and studying the interplay of *conditioning* and recursion [SG12] through the lens of pPDA.

Acknowledgement. The authors thank Christof Löding for his pointer to stair-parity VPA, and the anonymous reviewers for their useful suggestions and feedback. We also thank Darion Haase for helpful discussions regarding the proof of Lemma 4.15.

References

[AAB+07] Rajeev Alur, Marcelo Arenas, Pablo Barceló, Kousha Etessami, Neil Immerman, and Leonid Libkin. First-Order and Temporal Logics for Nested Words. In *LICS*, pages 151–160. IEEE Computer Society, 2007. doi:10.1109/LICS.2007.19.

[ABE18] Rajeev Alur, Ahmed Bouajjani, and Javier Esparza. Model Checking Procedural Programs. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 541–572. Springer, 2018. doi:10.1007/978-3-319-10575-8_17.

[ADD00] Robert B Ash and Catherine A Doléans-Dade. *Probability and Measure Theory*. Academic press, 2000.

[AEM04] Rajeev Alur, Kousha Etessami, and P. Madhusudan. A Temporal Logic of Nested Calls and Returns. In *TACAS*, volume 2988 of *Lecture Notes in Computer Science*, pages 467–481. Springer, 2004. doi:10.1007/978-3-540-24730-2_35.

[AK15] David Axelrod and Marek Kimmel. *Branching Processes in Biology*. Springer-Verlag, 2015. doi:10.1007/978-1-4939-1559-0.

[AM04] Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In *STOC*, pages 202–211. ACM, 2004. doi:10.1145/1007352.1007390.

[AP09] Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. *Sci. Comput. Program.*, 74(8):568–589, 2009. doi:10.1016/j.scico.2007.09.002.

[BEKK13] Tomás Brázdil, Javier Esparza, Stefan Kiefer, and Antonín Kucera. Analyzing probabilistic pushdown automata. *Formal Methods Syst. Des.*, 43(2):124–163, 2013. doi:10.1007/s10703-012-0166-0.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BKOB13] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Béguelin. Probabilistic Relational Reasoning for Differential Privacy. *ACM Trans. Program. Lang. Syst.*, 35(3):9:1–9:49, 2013. doi:10.1145/2492061.

[BKS05] Tomás Brázdil, Antonín Kucera, and Oldrich Strazovský. On the Decidability of Temporal Properties of Probabilistic Pushdown Automata. In *STACS*, volume 3404 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 2005. doi:10.1007/978-3-540-31856-9_12.

[BS18] Laura Bozzelli and César Sánchez. Visibly Linear Temporal Logic. *J. Autom. Reason.*, 60(2):177–220, 2018. doi:10.1007/s10817-017-9410-z.

[CIRW11] Lorenzo Casini, Phyllis McKay Illari, Federica Russo, and Jon Williamson. Models for prediction, explanation and control: recursive Bayesian networks. *THEORIA. Revista de Teoría, Historia y Fundamentos de la Ciencia*, 26(1):5–33, 2011.

[CMP20] Michele Chiari, Dino Mandrioli, and Matteo Pradella. Operator precedence temporal logic and model checking. *Theor. Comput. Sci.*, 848:47–81, 2020. doi:10.1016/j.tcs.2020.08.034.

[DBB12] Clemens Dubslaff, Christel Baier, and Manuela Berg. Model checking probabilistic systems against pushdown specifications. *Inf. Process. Lett.*, 112(8-9):320–328, 2012. doi:10.1016/j.ipl.2012.01.006.

[EKL10] Javier Esparza, Stefan Kiefer, and Michael Luttenberger. Computing the Least Fixed Point of Positive Polynomial Systems. *SIAM J. Comput.*, 39(6):2282–2335, 2010. doi:10.1137/090749591.

[EKM04] Javier Esparza, Antonín Kucera, and Richard Mayr. Model checking probabilistic pushdown automata. In *LICS*, pages 12–21. IEEE Computer Society, 2004. doi:10.1109/LICS.2004.1319596.

[EY05] Kousha Etessami and Mihalis Yannakakis. Algorithmic Verification of Recursive Probabilistic State Machines. In *TACAS*, volume 3440 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2005. doi:10.1007/978-3-540-31980-1_17.

[EY09] Kousha Etessami and Mihalis Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM*, 56(1):1:1–1:66, 2009. doi:10.1145/1462153.1462154.

[GHNR14] Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. In *FOSE*, pages 167–181. ACM, 2014. doi:10.1145/2593882.2593900.

[GMN18] Jens Oliver Gutsfeld, Markus Müller-Olm, and Benedikt Nordhoff. A Branching Time Variant of CaRet. In *SPIN*, volume 10869 of *Lecture Notes in Computer Science*, pages 153–170. Springer, 2018. doi:10.1007/978-3-319-94111-0_9.

[GS14] Noah D Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org, 2014. Accessed: 2023-7-25.

[Jae01] Manfred Jaeger. Complex Probabilistic Modeling with Recursive Relational Bayesian Networks. *Ann. Math. Artif. Intell.*, 32(1-4):179–220, 2001. doi:10.1023/A:1016713501153.

[Jon90] Claire Jones. *Probabilistic non-determinism*. PhD thesis, University of Edinburgh, UK, 1990. URL: http://hdl.handle.net/1842/413.

[Kar91] Richard M. Karp. An introduction to randomized algorithms. *Discret. Appl. Math.*, 34(1-3):165–201, 1991. doi:10.1016/0166-218X(91)90086-C.

[KEM06] Antonín Kucera, Javier Esparza, and Richard Mayr. Model Checking Probabilistic Pushdown Automata. *Log. Methods Comput. Sci.*, 2(1), 2006. doi:10.2168/LMCS-2(1:2)2006.

[LMS04] Christof Löding, P. Madhusudan, and Olivier Serre. Visibly Pushdown Games. In *FSTTCS*, volume 3328 of *Lecture Notes in Computer Science*, pages 408–420. Springer, 2004. doi:10.1007/978-3-540-30538-5_34.

[MM01] Annabelle McIver and Carroll Morgan. Partial correctness for probabilistic demonic programs. *Theor. Comput. Sci.*, 266(1-2):513–541, 2001. doi:10.1016/S0304-3975(00)00208-5.

[OKKM16] Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Reasoning about Recursive Probabilistic Programs. In *LICS*, pages 672–681. ACM, 2016. doi:10.1145/2933575.2935317.

[Pan09] Prakash Panangaden. *Labelled Markov Processes*. World Scientific, 2009.

[PK00] Avi Pfeffer and Daphne Koller. Semantics and Inference for Recursive Probability Models. In *AAAI/IAAI*, pages 538–544. AAAI Press / The MIT Press, 2000. URL: http://www.aaai.org/Library/AAAI/2000/aaai00-082.php.

[Pnu77] Amir Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.32.

[SEY15] Alistair Stewart, Kousha Etessami, and Mihalis Yannakakis. Upper bounds for newton's method on monotone polynomial systems, and p-time model checking of probabilistic one-counter automata. *J. ACM*, 62(4):30:1–30:33, 2015. doi:10.1145/2789208.

[SG12] Andreas Stuhlmüller and Noah D. Goodman. A Dynamic Programming Algorithm for Inference in Recursive Probabilistic Programs. In *StarAI@UAI*, 2012. URL: https://starai.cs.kuleuven.be/2012/accepted/stuhlmuller.pdf.

[vdMPYW18] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An Introduction to Probabilistic Programming. *CoRR*, abs/1809.10756, 2018. URL: http://arxiv.org/abs/1809.10756, arXiv:1809.10756.

[WE07] Dominik Wojtczak and Kousha Etessami. PReMo: An Analyzer for Probabilistic Recursive Models. In *TACAS*, volume 4424 of *Lecture Notes in Computer Science*, pages 66–71. Springer, 2007. doi:10.1007/978-3-540-71209-1_7.

[WGK21] Tobias Winkler, Christina Gehnen, and Joost-Pieter Katoen. Model Checking Temporal Properties of Recursive Probabilistic Programs. *CoRR*, abs/2111.03501v2, 2021. arXiv:2111.03501v2.

[WGK22] Tobias Winkler, Christina Gehnen, and Joost-Pieter Katoen. Model checking temporal properties of recursive probabilistic programs. In *FoSSaCS*, volume 13242 of *Lecture Notes in Computer Science*, pages 449–469. Springer, 2022. doi:10.1007/978-3-030-99253-8_23.

[WK23a] Tobias Winkler and Joost-Pieter Katoen. Certificates for Probabilistic Pushdown Automata via Optimistic Value Iteration. In *TACAS (2)*, volume 13994 of *Lecture Notes in Computer Science*, pages 391–409. Springer, 2023. doi:10.1007/978-3-031-30820-8_24.

[WK23b] Tobias Winkler and Joost-Pieter Katoen. On certificates, expected runtimes, and termination in probabilistic pushdown automata. In *LICS*, pages 1–13, 2023. doi:10.1109/LICS56636.2023.10175714.

[YE05] Mihalis Yannakakis and Kousha Etessami. Checking LTL Properties of Recursive Markov Chains. In *QEST*, pages 155–165. IEEE Computer Society, 2005. doi:10.1109/QEST.2005.8.