



Polynomial Invariants for Probabilistic Transition Systems with Unbounded Support

Anne Schreuder¹, Lorenz Winkler², Laura Kovács², and C.-H. Luke Ong³

¹ Aalto University, Finland

`anne.schreuder@aalto.fi`

² TU Wien, Austria

`lorenz.winkler@tuwien.ac.at`, `laura.kovacs@tuwien.ac.at`

³ Nanyang Technological University, Singapore

`luke.ong@ntu.edu.sg`

Abstract

We study the synthesis of polynomial invariants for probabilistic transition systems (PTS) based on martingale theory. We present tractable methods to verify that such polynomials are indeed invariants, in the sense that their expected value upon termination is the same as their value at the start of the computation. We do this by applying the Optional Stopping Theorem (OST) in the form of a specific precondition. This precondition requires the existence of an integrable **dominating** function for the martingale expression, which implies **uniform integrability**; we refer to this condition as **dui**. For linear PTS we simplify the **dui** property to proving finiteness of the expected value of an expression depending on the update matrix, the degree of the martingale expression, and the stopping time. Specifically, if all random samples have finite moments and we can verify a moment bound on the runtime of a linear loop, then we can automatically synthesise polynomial loop invariants that satisfy the OST. Notably, **dui** allows for the sampled distributions to have unbounded support, which is a novel contribution to the field. We implemented and evaluated our method to symbolically compute upper and lower bounds for moments of program variables upon termination.

1 Introduction

Probabilistic programs are computer programs that make random choices. Their increasingly widespread use in many areas of computer science — ranging across machine learning, network protocols and robotics — has led to a recent surge of interest in the semantics of probabilistic programs and methods for reasoning about them. In this paper we present a theoretical result for invariants of probabilistic programs that sample from distributions with *unbounded support*. We then show the applicability of our result by *synthesising polynomial invariants* for a restricted class of loops and use them to *automatically derive symbolic bounds* for the expected value of monomials after the loop has terminated.

Problem setting. Following [2, 1, 15, 11, 19, 24, 6], we explore methods that automatically synthesise expressions over the program variables whose expectation is invariant over the course of the computation. Mathematically, the invariants we construct are *martingales*. Elegant and effective, martingales are commonplace in the static analysis of probabilistic programs [5, 16, 8, 21, 26, 1, 7, 22, 28, 4, 29, 18, 10].

We restrict ourselves to invariants of probabilistic programs that have the same expected value before and after the program has run. In general, the runtime of a probabilistic program is not a concrete number but a probability distribution. Consequently, the expectation of a step-wise invariant at the point of termination (*qua* stopping time) may differ from its expectation after any fixed number of steps (see Example 1). For non-probabilistic programs this effect does not exist. First observed in betting strategies, this phenomenon has been analysed extensively in probability theory, leading to Doob’s celebrated Optional Stopping Theorem (OST) e.g. [12, Theorem 5.3.1]. This theorem yields preconditions for the expectation of a martingale taken at a stopping time to coincide with the expectation taken at the beginning.

The Optional Stopping Theorem (OST) has recently been successfully applied to the static analysis of probabilistic programs [33, 32, 4, 15, 35, 9]. However the phenomenon of the necessity of an “extra step”, that is verifying a precondition of the OST, had already been observed in the analysis of nested loops [27, 21]. When reasoning about such probabilistic programs, it is common to use inductive arguments for the inner loop. For reasons similar to those described earlier, it is necessary to impose appropriate OST-type preconditions on the inner-loop invariants, in order to formulate sound proof rules. These methods have successfully been applied in [27, 15, 11, 19, 18]. In particular, for nested loop analysis it is important to handle random updates with unbounded support. Otherwise, only loops that are entered at most N times can be analysed. Although our work mainly restricts to single loop programs, our approach does not require bounds on the support of the sampled distributions.

Our approach. We work with *probabilistic transition systems* (PTS) [5, 6, 7, 11], a general model of probabilistic imperative programs. Any standard imperative program with random samples and (nested) loops can be expressed as a PTS. An example of such a program is given in Figure 1. In particular, we allow our programs to manipulate values in the field $\mathbb{R}$ of real numbers and to sample from continuous and unbounded probability distributions. Figure 1, for instance, samples from a continuous uniform and a normal distribution.

We first present a *precondition of the OST* (Theorem 1), which we call **dui**. The **dui** precondition requires the existence of an integrable dominating function (for the martingale expressions) which implies the martingale is uniformly integrable. A key advantage of **dui** is its generality and its flexibility. It does not impose a constant bound on the stopping time (runtime), the program variables, or their updates. As such, our work differs from approaches requiring a constant bound on an invariant [21, 15, 11, 19], which is a special case of the **dui** precondition, or bounded updates [32, 33, 35].

While the **dui** precondition is rather abstract, it is more concrete than the condition of

```

 $x := x_0$ 
 $y := y_0$ 
 $z := z_0$ 
 $k := 0$ 
while  $x \geq 0$  do
   $k := k + 1$ 
   $u \sim \text{Uniform}(-1, 0)$ 
   $x := x + u$ 
   $n_1 \sim \text{Normal}(1, 2)$ 
   $y := y + n_1 + u$ 
   $n_2 \sim \text{Normal}(-2, 4)$ 
   $z := z + y + n_2$ 
end while

```

Figure 1: Linear PTS with updates over distributions with unbounded support.

uniform integrability which it implies. Hence, it can be used to derive more tractable sufficient conditions for invariants satisfying the OST. We demonstrate this by deriving a sufficient condition for polynomial invariants (expressible as a martingale) of simple linear programs consisting of a single loop to satisfy the OST (Theorem 3). This novel condition is twofold: First, the distributions over which we sample must be identically distributed in every iteration, and all moments of those distributions must exist. This is a strict improvement on the state-of-the-art which requires all sample distributions to be bounded [32, 33, 35]. Second, the runtime must satisfy an expectation bound that depends on the degree of the desired polynomial invariant, the structure of the update matrix, and whether variables from distributions with unbounded support affect the runtime. We prove this theorem with the **dui** precondition. Moreover, our method can deduce upper and lower bounds on the expected values of higher moments of the program variables and of the runtime¹ at the end of the computation — potentially improving the initial bound on the runtime; bounds upon the moments of Figure 1 are listed in Table 1.

Our contributions. We bring the following main contributions².

- We focus on a very flexible precondition for the OST, which requires the existence of an integrable **dominating** function for the martingale expression implying that the martingale is **uniformly integrable** – we refer to this condition as **dui** (Section 3). To the best of our knowledge, our **dui** condition is new in the study of probabilistic programs and has the advantage that it does not impose any restrictions on the stopping time a priori³.
- We present a sufficient condition for the application of the OST for polynomial martingales of simple linear loops (Section 4). Checking the precondition then simplifies to proving finiteness of moments of the stopping time (Theorem 3).
- Taking the number of finite moments of the stopping time as an input, we construct martingale expressions, which satisfy the OST based on the **dui** condition (Section 4.1). Those martingales can then be used to symbolically compute upper and lower bounds for the moments of program variables upon termination. Notably this includes variables having negative, unbounded increments that depend on the program state.
- While we make no claims about completeness of our method and still require (potentially manual) proofs of finiteness of stopping times in advance, we give examples where our method is applicable (Section 5). We implemented our work on top of the **Polar** analyser [30] and evaluated it on programs with distributions of unbounded support (Table 1). To the best of our knowledge, existing approaches for cost analysis or bound computation in probabilistic programs cannot handle our examples with unbounded support. Our implementation and instructions for reproducing our results are publicly available⁴.

2 Probabilistic Transition Systems and Invariants

As models of imperative programs we consider probabilistic transition systems (PTS). We distinguish (deterministic) program variables $X = \{x_1, x_2, \dots, x_n\}$ which are assigned deterministically, and random sample variables $R = \{r_1, r_2, \dots, r_m\}$ which are sampled from probability

¹It is always possible to introduce a new variable that updates by one each time the loop is entered — effectively storing the runtime. This is the role of the variable k in Figure 1.

²with detailed proofs in [31]

³This precondition is more abstract. Verifying it can sometimes require bounds on the runtime as seen in Section 4.1. However, there may be other applications of this precondition where such a step is not necessary.

⁴<https://github.com/probing-lab/polar/tree/166b275>

distributions. All variables take values in $\mathbb{R}$. We assume that the random samples are stochastically independent of each other and identically, independently distributed over time; moreover we assume that the sampled distributions are known and that all of their moments are finite.

Definition 1 (Probabilistic Transition Systems - PTS [5, 6, 7]). *A probabilistic transition system (PTS) is a tuple $\Pi = \langle W, X, R, \vec{x}_0, L, l_0, l_F, \mathbb{T} \rangle$ where*

- (i) $W = (\Omega, \mathcal{F}, \mathbb{P})$ is a probability space.
- (ii) $X = \{x_1, x_2, \dots, x_n\}$ are the program variables and $R = \{r_1, r_2, \dots, r_m\}$ are the random variables defined on the probability space W . Each random variable r_i represents a sampled update from a random distribution, e.g. $r_1 \sim N(0, 1)$. We assume that all x_i are updated deterministically — based on the current values of the r_i 's and x_j 's.
- (iii) $\vec{x}_0 \in \mathbb{R}^n$ is the vector of initial values of the program variables $X = \{x_1, x_2, \dots, x_n\}$.
- (iv) L is a finite set of (program) locations; $l_0 \in L$ is the initial location and $l_F \in L$ is the final location. The program terminates once l_F is reached.
- (v) $\mathbb{T} = \{\tau_1, \tau_2, \dots, \tau_p\}$ is a finite set of transitions. Each $\tau \in \mathbb{T}$ is a quadruple $\langle l_\tau^s, \phi_\tau, f_\tau, l_\tau^d \rangle$ where l_τ^s is the source location, which is never l_F , and l_τ^d the destination location; ϕ_τ is the guard assertion, which is a boolean formula over polynomial inequalities over X ; and $f_\tau : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$, the update function, which takes values of program variables and random samples $(\vec{x}, \vec{r}) \in \mathbb{R}^n \times \mathbb{R}^m$ and returns the new values $\vec{x} \in \mathbb{R}^n$ for the program variables. The update functions are polynomials.

A configuration of a PTS Π is a pair $(l, \vec{x})$ where $l \in L$ is the current program location, and $\vec{x} \in \mathbb{R}^n$ represents the contents of the program variables. Moreover, we say that a transition $\tau = \langle l_\tau^s, \phi_\tau, f_\tau, l_\tau^d \rangle \in \mathbb{T}$ is *enabled* at a configuration $(l, \vec{x})$ if $l = l_\tau^s$ and $\vec{x} \models \phi_\tau$. For $l \in L$, we write $\mathbb{T}_l := \{\tau \in \mathbb{T} \mid l_\tau^s = l\}$ for the set of transitions with l as the source location. We assume that PTSs are *non-demonic*: For all $l \in L$, $\bigwedge_{\tau \neq \tau' \in \mathbb{T}_l} (\phi_\tau \wedge \phi_{\tau'}) = \text{false}$ and $\bigvee_{\tau \in \mathbb{T}_l} \phi_\tau = \text{true}$. This amounts to *determinacy* in the sense that for all $l \in L$ and $\vec{x} \in \mathbb{R}^n$, there is a unique transition that is enabled at $(l, \vec{x})$.

PTS computation. The initial configuration of a PTS Π is $(l_0, \vec{x}_0)$ for some choice of initial values $\vec{x}_0 \in \mathbb{R}^n$. Suppose that Π is at a configuration $(l, \vec{x})$ at step k . Let τ be the unique transition that is enabled (i.e. $l = l_\tau^s$ and $\vec{x} \models \phi_\tau$), and $\vec{r} \in \mathbb{R}^m$ be a fresh sample drawn from the respective random distributions. Then the configuration of Π at step $k + 1$ is $(l_\tau^d, f_\tau(\vec{x}, \vec{r}))$.

The operational semantics of a PTS Π is given by a sequence of random variables $\{L^{[k]}, X^{[k]}\}$ on W representing the distributions of the location and the (contents of the) program variables at each computation step k . The runtime of the program is the first time the final location l_F is reached, i.e. $T = \inf\{k \in \mathbb{N} : L^{[k]} = l_F\}$. We use a very similar framework as in e.g. [5, 6, 7, 33]; details can be found in [31, Appendix D].

Probabilistic invariants. As invariants of PTS we consider functions on the program variables [27] and a step counter⁵ such that their expected value remains constant over the computation. By the pre-expectation their expected value of the next step can be expressed by the current values of the program variables [27, 19, 5, 6, 8, 7, 21, 24].

⁵In practice it is very useful to allow the runtime to be part of an invariant; see Example 7 & Figure 1.

Definition 2 (Pre-Expectation). Let Π be a PTS. Let $\tau \in \mathbb{T}$ and $h : L \times \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}$ be a measurable function over locations, valuations of the program variables, and the step counter. The *pre-expectation* $\text{preE}(h, \tau) : \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}$ is the function

$$(\vec{x}, k) \mapsto \mathbb{E}_R (h(l_\tau^d, f_\tau(\vec{x}, R), k))$$

where $\mathbb{E}_R$ is the expectation w.r.t. the distribution of the random variables $R = \{r_1, \dots, r_m\}$. The pre-expectation for the PTS, $\text{preE}(h) : L \times \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}$, is given by

$$(l, \vec{x}, k) \mapsto \sum_{\tau \in \mathbb{T}} [(l = l_\tau^s) \wedge (\vec{x} \models \phi_\tau)] \cdot \text{preE}(h, \tau)(\vec{x}, k)$$

where $[-]$ is the Iverson bracket.

Remark 1. If all the update functions f_τ are polynomials and h is polynomial for all $l \in L$, then $\text{preE}(h, \tau)(\vec{x}, k)$ is computed by replacing powers of random variables r_i^a by their respective moments $\mathbb{E}(r_i^a)$ in the expression $h(l_\tau^d, f_\tau(\vec{x}, R), k)$.

We consider *probabilistic invariants* as location-independent polynomials $p : \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}$ such that the random variables $\{p(X^{[k]}, k)\}_{k \in \mathbb{N}}$ (see [31, Definition 6]) are step-wise expectation invariant:

$$\text{preE}(p)(l, \vec{x}, k+1) = p(\vec{x}, k) \quad \text{for all } l \neq l_F, \vec{x} \in \mathbb{R}^n \text{ and } k \in \mathbb{N}. \quad (1)$$

Contrary to the deterministic case, probabilistic step-wise invariants are not necessarily global invariants. As a warning example, consider the following loop.

Example 1 (Martingale betting strategy). *The gambler starts by wagering 1 unit on a fair bet. So long as she loses, she continues by wagering twice the most recent bet on the next play. Thus, when she eventually (and inevitably) wins, the amount won will cover all her previous losses and profit her by one unit.*

```

 $x_1 := 1$ 
 $x_2 := 0$ 
while  $x_2 \leq 0$  do
   $r_1 \sim \text{Bernoulli}(\frac{1}{2})$ 
  if  $r_1 \leq 0.5$  then
     $x_2 := x_2 - x_1$ 
  else
     $x_2 := x_2 + x_1$ 
  end if
   $x_1 := 2 \cdot x_1$ 
end while

```

Consider the while loop where x_1 (current bet) and x_2 (her stake thus far) are the program variables and r_1 is the random variable.

The expression $M_k := x_2^{[k]}$ gives a step-wise invariant for this loop. In particular, $\mathbb{E}(M_k) = 0$ holds for all $k \in \mathbb{N}$.

However, note that $\mathbb{E}(M_0) = 0 < 1 = \mathbb{E}(M_T)$, where $T := \inf\{k \in \mathbb{N} \mid x_2^{[k]} > 0\}$ is the exit time of the loop. Hence, $(M_k)_{k \in \mathbb{N}}$ is not a loop invariant.

Soundness of step-wise invariants can be guaranteed by the Optional Stopping Theorem (OST) from probability theory, using the concept of martingales.

3 Martingales and the Optional Stopping Theorem

We briefly review standard notions in probability theory (see e.g. [20, 34, 13]) and fix notations along the way⁶. We fix a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. We use filtrations to represent information

⁶In [31, Appendix A] we elaborate on these abstract notions by studying some concrete examples.

known over time. Formally, a *filtration* is a nested, finite or infinite sequence $\{\mathcal{F}_k\}_{k \in \mathbb{N}}$ of σ -algebras over Ω such that $\mathcal{F}_k \subseteq \mathcal{F}_{k+1} \subseteq \mathcal{F}$ for all $k \in \mathbb{N}$. Intuitively, the σ -algebra $\mathcal{F}_k$ represents the information that is known at time $k \in \mathbb{N}$. A sequence of random variables $\{X_k\}$ is *adapted* to a filtration $\{\mathcal{F}_k\}_{k \in \mathbb{N}}$ if X_k is $\mathbb{B}(\mathbb{R})$ - $\mathcal{F}_k$ -measurable for all $k \in \mathbb{N}$, i.e. for all open $A \subseteq \mathbb{R}$ we have $X_k^{-1}(A) \in \mathcal{F}_k$. A random variable $T : \Omega \rightarrow \mathbb{N} \cup \{\infty\}$ is called a *stopping time* with respect to a filtration $\{\mathcal{F}_k\}_{k \in \mathbb{N}}$ if $\{T \leq k\} \in \mathcal{F}_k$ for all $k \in \mathbb{N}$. Moreover, we denote by $\text{supp}(X)$ the smallest closed set S such that $\mathbb{P}(X \in S) = 1$ the support of the random variable X which is bounded when $\sup\{|x| \mid x \in \text{supp}(X)\} < \infty$, and unbounded otherwise.

In this paper we consider the filtration $\{\mathcal{F}_k\}_{k \in \mathbb{N}}$ such that $\mathcal{F}_k$ contains the information of the initial value $X^{[0]}$ and the first k random samples (see [31, Appendix D]). Let X be an integrable random variable, that is $\mathbb{E}[|X|] < \infty$. Further, let $\mathcal{G}$ be a σ -subalgebra of $\mathcal{F}$. The *conditional expectation* $\mathbb{E}[X \mid \mathcal{G}]$ is a $\mathcal{G}$ -measurable function s.t. $\mathbb{E}[1_A \cdot X] = \mathbb{E}[1_A \cdot \mathbb{E}[X \mid \mathcal{G}]]$ for all $A \in \mathcal{G}$. The conditional expectation is $\mathbb{P}$ -almost surely ($\mathbb{P}$ -a.s.) unique, linear and monotone in X ⁷.

Given a filtration $\{\mathcal{F}_k\}$, an integrable, $\{\mathcal{F}_k\}$ -adapted sequence of random variables $\{M_k\}$ is

- (i) a *martingale* if for all $k \in \mathbb{N}$, $\mathbb{E}[M_{k+1} \mid \mathcal{F}_k] = M_k$ almost surely (a.s.);
- (ii) a *supermartingale* if for all $k \in \mathbb{N}$, $\mathbb{E}[M_{k+1} \mid \mathcal{F}_k] \leq M_k$ a.s.;
- (iii) a *submartingale* if for all $k \in \mathbb{N}$, $\mathbb{E}[M_{k+1} \mid \mathcal{F}_k] \geq M_k$ a.s..

In this paper we will consider martingales as the probabilistic counterpart to (algebraic) invariants, which capture that the expected value of a given expression stays invariant throughout iterations. For a martingale $\{M_k\}$ to be an invariant, one would naturally require that the expected value after loop termination is equal to its initial expected value, i.e. $\mathbb{E}(M_T) = \mathbb{E}(M_0)$. Because the stopping time T is itself a random variable, this property unfortunately is not true in general, as shown in Example 1. Difficulties generally arise when the tails of T are too heavy, or the updates are not well-behaved. However, by imposing suitable boundedness conditions on T or the martingale, the equality $\mathbb{E}(M_T) = \mathbb{E}(M_0)$ follows from the Optional Stopping Theorem (OST).

There are numerous versions of the OST in the literature, which differ in the precondition that is imposed (e.g. [34, §10.10], [33]), for a comprehensive list see [31, Appendix B]. The OST applies, for instance, if there exists a constant upper bound for the runtime [34], or if the step-wise differences are bounded by a constant and the expected value of the runtime is finite [20]. Because bounding the steps or values by a constant is often not possible, alternative preconditions have been developed [32, 33], accommodating bounds that are expressed as functions of the iteration counter k . In [32], the authors show that if $|M_k| \leq p(k)$ almost surely (where $p(k)$ is non-decreasing) and $\mathbb{E}(p(T)) < \infty$, then the requirements for the OST are fulfilled. The almost-sure bound, however, prevents the application to martingales where the differences depend on unbounded-support distributions. Since in that case there always is a nonzero probability that a deterministic bound is exceeded, we allow the bound to be a random variable in the sense that it may depend on a specific $\omega \in \Omega$, rather than being uniform over all execution traces. This generalises previous approaches, which require a bound for the martingale(-differences) independent of ω .

A general precondition [12, Theorem 4.6.7] for the OST requires $\{M_k\}$ to be uniformly integrable. In general uniform integrability is notoriously hard to check, but in case the stopped martingale can be bounded by an integrable random variable, it holds true. We formulate this as a precondition for the OST in Theorem 1.

⁷See [31, Theorem 4] for more properties and computation rules of the conditional expectation.

Theorem 1 (Optional Stopping and **dui**). *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and $\{M_k\}_{k \in \mathbb{N}}$ be a (super)martingale adapted to a filtration $\{\mathcal{F}_k\}_{k \in \mathbb{N}}$, and $T : \Omega \rightarrow \mathbb{N} \cup \{\infty\}$ a stopping time. If the following precondition holds:*

(dui) dominating function for uniform integrability: *there is a random variable $X : \Omega \rightarrow \mathbb{R}_+$ such that $\mathbb{E}(X) < \infty$ and $\sup_{k \in \mathbb{N}} |M_{\min(k, T)}| \leq X$ almost surely,*

then M_T is integrable and $\mathbb{E}(M_T) = \mathbb{E}(M_0)$, if M is a martingale; resp. $\mathbb{E}(M_T) \leq \mathbb{E}(M_0)$, if M is a supermartingale.

We apply Theorem 1 to reason about invariants synthesised from probabilistic programs. The random variable M_k in Theorem 1 is an expression built up from the program variables and locations in the k -th step of the computation. Thus, the **dui** precondition requires that the value of $\sup_{k \in \mathbb{N}} |M_{\min(k, T)}|$ is bounded above by an integrable random variable X . The random variable X can be seen as a function mapping from the input variables and the actual values of the samples drawn to a real number. Because X is a random variable instead of a deterministic function of the iteration counter, we can overcome the limitations of [32] with respect to unbounded-support distributions.

Remark 2. *In Theorem 1 we do not assume that the stopping time T is finite. In the proof we show that $\{M_{\min(k, T)}\}_{k \in \mathbb{N}}$ is uniformly integrable. It follows from probability theory, e.g. [25], that on the event of diverging runs $\{T = \infty\}$, the martingales M_k converge to a limit value M_∞ (which is an integrable random variable). Hence, the random variable M_T is in fact defined as*

$$M_T = \lim_{k \rightarrow \infty} M_{\min(k, T)} = M_T \cdot 1_{\{T < \infty\}} + M_\infty \cdot 1_{\{T = \infty\}}.$$

We can illustrate this by the following example.

$x := 1$ while $x \geq 0$ do $x := 1$ end while	Clearly, $\mathbb{P}(T = \infty) = 1$, where $T := \inf\{k \in \mathbb{N} : x^{[k]} < 0\}$ is the exit time of the loop. Nonetheless, $\{x^{[k]}\}_{k \in \mathbb{N}}$ is a martingale, hence an invariant, for this loop and we have $\mathbb{E}(x^{[T]}) = 1 = \mathbb{E}(x^{[0]})$.
---	--

While Theorem 1 is general and does not directly give rise to automatically verifiable preconditions, we use it to derive Theorem 3. In the case of programs with linear updates without exponential growth this theorem gives novel preconditions that are much more tractable, forming the basis of our implementation.

Probabilistic martingale invariants. By the following lemma our (step-wise) probabilistic invariants are martingales. This is proven in Appendix [31, Appendix D].

Lemma 1 (Invariants and Pre-Expectations). *Let Π be a PTS with operational semantics $\{L^{[k]}, X^{[k]}\}_{k \in \mathbb{N}}$, and let $\mathcal{F}_k$ be the smallest σ -algebra such that the initial distribution and the first k random samples are $\mathcal{F}_k$ -measurable. Let $h : L \times \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}$ be such that for all $l \in L$, $h(l, -, -) : \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}$ is a polynomial.*

- (i) *Then $\mathbb{E}[h(L^{[k+1]}, X^{[k+1]}, k+1) \mid \mathcal{F}_k] = \text{pre}\mathbb{E}(h)(L^{[k]}, X^{[k]}, k+1)$ for all $k \in \mathbb{N}$.*
- (ii) *If $\text{pre}\mathbb{E}(h)(l, \vec{x}, k+1) \leq h(l, \vec{x}, k)$ for all $l \in L, \vec{x} \in \mathbb{R}^n, k \in \mathbb{N}$, then $\{h(L^{[k]}, X^{[k]}, k)\}_{k \in \mathbb{N}}$ is a supermartingale adapted to $\{\mathcal{F}_k\}_{k \in \mathbb{N}}$.*

Remark 3. *By e.g. [34, Theorem 10.9], if $(M_n)_{n \in \mathbb{N}}$ is a martingale and $T : \Omega \rightarrow \mathbb{N} \cup \{\infty\}$ is a stopping time, then $(M_{\min(n, T)})_{n \in \mathbb{N}}$ is a martingale. Hence, we do not need to consider the loop guard when computing loop invariants. We will use this fact throughout the rest of the paper.*

4 Polynomial Invariants of Linear Programs

This section focuses on the class of linear probabilistic transition systems (PTS) and shows that the **dui** precondition can be used to obtain non-linear invariants of linear PTS. We provide a sufficient condition under which polynomial martingale expressions satisfy the Optional Stopping Theorem (OST) for programs consisting of a single loop (Theorem 3). For the remainder of this paper Definition 3 imposes additional restrictions on the updates.

Definition 3 (Single-Loop Linear Programs). *A simple single-loop linear program consists of a vector of variables $\vec{x}$, which is updated in the loop body through the following update function:*

$$\vec{x}^{[k+1]} = A\vec{x}^{[k]} + B\vec{r}^{[k+1]} \quad (2)$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ and $\vec{x}^{[k]}$ denotes the value of $\vec{x}$ in iteration k . We impose that all eigenvalues λ of A satisfy $|\lambda| \leq 1$. Moreover, $\vec{r}^{[k+1]} \sim \vec{R}$ is a vector of fresh draws from random distributions, where all moments are finite, i.e. $\forall i \in \{1, 2, \dots, m\}, \forall N \in \mathbb{N}. \mathbb{E}(|R_i|^N) < \infty$. Samples are assumed to be independent.

The interactions between variables must be represented through a matrix with eigenvalues with absolute value at most 1. Current techniques [32] for verifying the applicability of the OST work for those programs, when random updates sample only from distributions with bounded support. Importantly, we allow sampling from distributions with unbounded support, only requiring that all moments are finite for those distributions. Examples for such programs are given in Figures 1 & 2. While the program class is restrictive, our approach can be adapted to loops with branching, by overapproximating the updates made in each iteration.

Example 2. *The loop in Figure 1 is a program as in Definition 3 with an update matrix with eigenvalues 1. Concretely we can formulate the following recurrence:*

$$\begin{pmatrix} z^{[k+1]} \\ y^{[k+1]} \\ x^{[k+1]} \\ k^{[k+1]} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} z^{[k]} \\ y^{[k]} \\ x^{[k]} \\ k^{[k]} \\ 1 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u^{[k+1]} \\ n_1^{[k+1]} \\ n_2^{[k+1]} \end{pmatrix}.$$

The variables $u^{[k+1]}, n_1^{[k+1]}$ and $n_2^{[k+1]}$ are fresh samples from $\text{Uniform}(-1, 0)$, $\text{Normal}(1, 2)$, and $\text{Normal}(-2, 4)$ respectively.

Moreover, linear loops are guarded by a loop guard ϕ and as such, in what follows, the runtime $T := \inf\{k \in \mathbb{N} \mid \neg\phi(\vec{x}^{[k]})\}$ is defined as the first time this condition is violated. We furthermore write $x \in \phi$, when the program variable x occurs in ϕ syntactically.

We next show how dominating functions (up to the runtime T) can be constructed for monomials over the program variables. As integration is a linear operator, a polynomial martingale candidate is uniformly integrable when the same is true for its individual monomials (e.g. [25, Theorem 6.18]). Since the construction of the dominating function for a monomial depends on the structure of the Jordan decomposition of the update (sub)matrix of the program variables it depends on, we first make the notion of dependence precise in Definition 4.

Definition 4 (Variable Dependence). *For two program variables x_i, x_j , we write $x_i \preceq x_j$ when x_i depends on x_j , which is the case when $A_{ij} \neq 0$. Similarly, a program variable x_i depends on a probability distribution R_j , when $B_{ij} \neq 0$. Furthermore, $\preceq$ is transitive and reflexive.*

Let $I_i = \{j \mid x_i \preceq x_j\}$ be the indices of the variables that the variable x_i depends on. We denote by A_i the sub-matrix of A , consisting only of the rows and columns with indices in I_i . Similarly, B_i is the sub-matrix of B consisting only of the rows with index in I_i . By definition of dependence, only considering A_i, B_i instead of A, B has no effect on the value of x_i .

In order to apply the **dui** precondition, for a fixed program variable x_i we first need to construct an (integrable) random variable Z_i that is greater than $|x_i^{\lfloor \min(k, T) \rfloor}|$ almost surely for all $k \in \mathbb{N}$ simultaneously. The following example illustrates how such a random variable can be constructed, while Theorem 2 establishes that for our programming model (Definition 3) this is always possible.

Example 3. Consider the program in Figure 1. We denote by $u^{[k]}$, $n_1^{[k]}$ and $n_2^{[k]}$ the random samples drawn from $\text{Uniform}(-1, 0)$, $\text{Normal}(1, 2)$ and $\text{Normal}(-2, 4)$ in the k -th iteration. We denote by $u^{[k]}(\omega)$ and $n_i^{[k]}(\omega)$ the value for a given element from the sample space $\omega \in \Omega$. Then we construct a random variable $Z_y : \Omega \rightarrow \mathbb{R}$, such that $\forall \omega \in \Omega, \forall k \in \mathbb{N} : |y^{\lfloor \min(k, T) \rfloor}(\omega)| \leq Z_y(\omega)$.

By Figure 1, $y^{[k]}(\omega) = y^{[k-1]}(\omega) + n_1^{[k]}(\omega) + u^{[k]}(\omega)$. Hence, we get the closed form expression $y^{[k]}(\omega) = y^{[0]} + \sum_{i=1}^k (n_1^{[i]}(\omega) + u^{[i]}(\omega))$. Because the absolute value is sub-additive, $|y^{[k]}(\omega)| \leq |y^{[0]}| + \sum_{i=1}^k (|n_1^{[i]}(\omega)| + |u^{[i]}(\omega)|)$. Since that bound is monotonically increasing, we can choose $Z_y(\omega) := |y^{[0]}| + \sum_{i=1}^{T(\omega)} (|n_1^{[i]}(\omega)| + |u^{[i]}(\omega)|)$ and so $|y^{\lfloor \min(k, T(\omega)) \rfloor}(\omega)| \leq Z_y(\omega)$ for all $k \in \mathbb{N}$.

The construction of bounds similar to Example 3 can be done in general — as stated in Theorem 2 and detailed next. This plays a crucial role in proving Theorem 3 using Theorem 1. We will prove Theorem 2 in [31, Theorem 14].

Theorem 2 (Almost Sure Bounds). Assume that all eigenvalues λ of A satisfy $|\lambda| \leq 1$. For a program variable x_i , let A_i, B_i be the update (sub-)matrices for the program variables that x_i depends on. Let m_i be the size of the largest Jordan block of the Jordan normal form of A_i . Then there are constants $c_{i,1}, c_{i,2} \in \mathbb{R}$ that depend only on the matrices A_i, B_i , s.t. if we define

$$Z_i := c_{i,1}(T^{m_i-1} + 1)\|\vec{x}^{[0]}\|_2 + c_{i,2}(T^{m_i-1} + 1) \sum_{l=1}^T \|\vec{r}^{[l]}\|_2,$$

then $0 \leq |x_i^{\lfloor \min(k, T) \rfloor}| \leq Z_i$ holds almost surely for all $k \in \mathbb{N}$.

In the following, we use a multi-index notation for monomials over the program variables, i.e. $\alpha \in \{(\alpha_1, \alpha_2, \dots, \alpha_n) : \alpha_i \in \mathbb{N}\}$ and $\vec{x}^\alpha = \prod_{\alpha_i \in \alpha} x_i^{\alpha_i}$. Moreover, $(\vec{x}^\alpha)^{[k]}$ denotes the value of $\vec{x}^\alpha$ in iteration k .

Theorem 3 (Random Variables with Unbounded Support). Assume the setting of Definition 3. For all program variables x_i , let A_i, B_i be the update (sub-)matrices for the variables that x_i depends on. Let m_i be the size of the largest Jordan block of the matrix A_i and let $\beta_i = \|B_i\|_2$. Fix a multi-index $\alpha = (\alpha_1, \dots, \alpha_n)$ and set

$$N := N(\alpha) = \sum_{i=1}^n (m_i - 1 + 1_{\{\beta_i > 0\}}) \alpha_i.$$

Then the random variable sequence $\{(\vec{x}^\alpha)^{\lfloor \min(k, T) \rfloor}\}_{k \in \mathbb{N}}$ is uniformly integrable if

- (i) $\mathbb{E}(T^N) < \infty$ and $x \in \phi \wedge x \preceq R_j \implies \text{supp}(R_j)$ is bounded, or
- (ii) $\mathbb{E}(T^{N+1}) < \infty$.

For a fixed multi-index α , we use the variables Z_i from Theorem 2 to construct an almost sure upper bound for the monomial $(\vec{x}^\alpha)^{[\min(k,T)]}$ over all $k \in \mathbb{N}$. In Theorem 3 we show that those monomials are uniformly integrable, when the expected value of the N -th power of the stopping time is finite, where N depends on the update matrices, and T does not depend on distributions with unbounded support. Naturally this is the case, when variables in the loop guard ϕ depend (Definition 4) only on random variables drawn from bounded support distributions. If T does depend on distributions with unbounded support, instead finiteness of the $(N + 1)$ -th moment is required. We prove Theorem 3 in [31, Appendix E].

Example 4. *The loop in Figure 1 is a simple single-loop linear program as in Definition 3 with an update matrix with eigenvalue 1, which we have written out in Example 2.*

The update matrix A is already in Jordan normal form. The variables x, y are only affected by Jordan blocks with size 1, and have random updates, while z is affected by a Jordan block of size 2 and has random updates. The variable k is affected by a Jordan block of size 2, but is not updated by the random samples. Because the runtime depends only on bounded support distributions, by Theorem 3 through ensuring that $E(T^N) < \infty$, every monomial $x^a y^b z^c k^d$ is uniformly integrable, when $a + b + 2c + d \leq N$.

In conclusion, a polynomial over random variables is uniformly integrable, if every monomial in it is. If each monomial of a polynomial martingale satisfies the prerequisites of Theorem 3, then it satisfies the OST by the **dui** precondition (Theorem 1).

4.1 Constructing Martingales for Linear Programs

We use Theorem 3 to identify uniformly integrable monomials up to the stopping time, which can be used for *constructing martingales that enable bounding loop variables with unbounded support*. By Definition 3 we represent the loop body by a linear update $\vec{x}^{[k+1]} = A\vec{x}^{[k]} + B\vec{r}^{[k+1]}$. Given a finite N , for which $\mathbb{E}(T^N) < \infty$ holds, we check which monomials $\vec{x}^\alpha$ are uniformly integrable. Thereby we consider the size of the Jordan blocks affecting its variables and potential dependencies of the runtime from unbounded support distributions.

By (1) our goal is to find polynomials $m : \mathbb{R}^n \times \mathbb{N} \rightarrow \mathbb{R}$ such that $\text{pre}\mathbb{E}(m)(\cdot, \vec{x}, k + 1) = m(\vec{x}, k)$ holds for all $\vec{x}$ and k . It then follows by Lemma 1 that $\{m(\vec{x}^{[k]}, k)\}_{k \in \mathbb{N}}$ is a martingale⁸.

As is shown in Example 2, the update matrix A of a PTS resembles a recurrence relation for every variable $x_i^{[k+1]} = p_{x_i}(\vec{x}^{[k]}, \vec{r}^{[k+1]})$, where p_{x_i} is a (linear) polynomial over the current variable values and the current samples. Hence, $q_{x_i} := \text{pre}\mathbb{E}(x_i)(\cdot, \vec{x}, k + 1)$ is a (linear) polynomial over the variable values and the step counter⁹.

Example 5. *Consider again the example in Figure 1 and the monomial kx . We get $p_x = x^{[k+1]} = x^{[k]} + u^{[k+1]}$ and $p_k = k^{[k+1]} = k^{[k]} + 1$. Then $(kx)^{[k+1]} = p_k p_x$, and so*

$$\begin{aligned} q_{kx} &:= \text{pre}\mathbb{E}(kx)(\cdot, \vec{x}, k + 1) = \mathbb{E}_u((x + u)(k + 1)) = xk + x + \mathbb{E}(u)k + \mathbb{E}(u) \\ &= kx - (1/2)k + x - (1/2), \end{aligned}$$

where $u \sim \text{Uniform}(-1, 0)$ denotes a fresh random sample. Similarly, by allowing more complex

⁸Note that by Remark 3 we do not need to consider the loop guard at this stage.

⁹Often it is easier to create a program variable that stores the step counter, i.e. k in Figure 1.

monomials and by replacing the moments of the random samples, we get

$$\begin{aligned} q_k &= k + 1 & q_{kx} &= kx - (1/2)k + x - (1/2) \\ q_x &= x - (1/2) & q_{ky} &= ky + (1/2)k + y + (1/2) \\ q_y &= y + (1/2) & q_{x^2} &= x^2 - x + (1/3) \\ q_z &= z + y - (3/2) & q_{xy} &= xy + (1/2)x - (1/2)y - (1/6) \\ q_{k^2} &= k^2 + 2k + 1 & q_{y^2} &= y^2 + y + (7/3). \end{aligned}$$

By linearity and as the moments of the random samples are known, this allows us to compute the pre-expectation for any monomial $\vec{x}^\alpha$, which is a polynomial to which we also refer as q_α .

Martingales of linear PTS. Recall that $\{M^{[k]}\}_{k \in \mathbb{N}}$ is a martingale, whenever for all $k \in \mathbb{N}$ we have $\mathbb{E}(M^{[k+1]} \mid \mathcal{F}_k) = M^{[k]}$. For a polynomial $m(x_1, \dots, x_n)$ over the program variables to be a martingale, it must hence hold that $m(\vec{x}^{[k]}) = \mathbb{E}(m(\vec{x}^{[k+1]}) \mid \mathcal{F}_k) = \text{pre}\mathbb{E}(m)(\cdot, \vec{x}^{[k]}, k+1)$ where we have used Lemma 1. This can be captured by a system of linear equations.

Example 6. Consider the loop in Figure 1. If we choose the template $m(\vec{x}) = a_x x + a_{kx} kx + a_{k^2} k^2 + a_{x^2} x^2$, then by the computations from Example 5 and by Lemma 1,

$$\mathbb{E}(m(\vec{x}^{[k+1]}) \mid \mathcal{F}_k) = m(\vec{x}^{[k]}) + (a_{kx} - a_{x^2})x^{[k]} + \left(2a_{k^2} - \frac{a_{kx}}{2}\right)k^{[k]} - \frac{a_x}{2} - \frac{a_{kx}}{2} + a_{k^2} + \frac{a_{x^2}}{3}.$$

The equation $0 = a_{kx} - a_{x^2}$ ensures that the coefficient of $x^{[k]}$ in $m(\vec{x}^{[k]})$, and the coefficient of $x^{[k+1]}$ in $\mathbb{E}(m(\vec{x}^{[k+1]}) \mid \mathcal{F}_k)$ are the same. We repeat this for all coefficients and get a system of linear equations. One solution is $m(\vec{x}) = 3k^2 + 12kx + 12x^2 + 2x$, which is thus a martingale.

To construct martingales of linear PTS, we proceed as follows. We fix a set of monomials $\mathcal{M}$ which are uniformly integrable, and denote by $\mathcal{A} \subseteq \{(\alpha_1, \alpha_2, \dots, \alpha_n) : \alpha_i \in \mathbb{N}\}$ the corresponding set of multi-indices, i.e. $\mathcal{M} = \{\vec{x}^\alpha \mid \alpha \in \mathcal{A}\}$. Moreover, each monomial $\vec{x}^\alpha$ gets assigned a coefficient variable $a_\alpha \in \mathbb{R}$. We define the following set of linear equations, with $[\vec{x}^\alpha]q(\vec{x})$ denoting the coefficient of $\vec{x}^\alpha$ in the polynomial $q(\vec{x})$:

$$\left\{ a_\alpha = \sum_{\beta \in \mathcal{A}} a_\beta [\vec{x}^\beta] q_\beta(\vec{x}) \mid \alpha \in \mathcal{A} \right\}. \quad (3)$$

Every solution of Equation (3) corresponds to a polynomial $m(\vec{x}) := \sum_{\alpha \in \mathcal{A}} a_\alpha \vec{x}^\alpha$, which satisfies the martingale property. Intuitively, this equation enforces that for all monomials, the coefficient in $m(\vec{x}^{[k]})$ is the same as in $\mathbb{E}(m(\vec{x}^{[k+1]}) \mid \mathcal{F}_k)$.

Combined with Theorem 3, when these martingales satisfy the OST, the expected value after loop termination equals its initial value. Hence, throughout we will use $\vec{x}_T^\alpha = (\vec{x}^\alpha)^{[T]}$ to refer to the value of a monomial after the loop.

Example 7. In Figure 1 we have $\mathbb{E}(T^2) < \infty$. Hence, by Example 4 we can consider the set of uniformly integrable monomials $\{x, k, kx, k^2, x^2\}$, for which we have computed $q_\alpha = \text{pre}\mathbb{E}(\vec{x}^\alpha)$ in Example 5. So, by Example 6, $m(\vec{x}) = 3k^2 + 12kx + 12x^2 + 2x$ is a martingale for this loop. In particular, this martingale satisfies the OST by Theorem 3 and by Example 4, i.e.

$$\mathbb{E}(3k_T^2 + 12k_T x_T + 12x_T^2 + 2x_T) = 12x_0^2 + 2x_0. \quad (4)$$

Similarly by considering $\mathcal{M} = \{k, x, y, z, k^2, kx, ky, x^2, xy, y^2\}$ we get:

$$\mathbb{E}(k_T + 2x_T) = 2x_0 \quad (5)$$

$$\mathbb{E}(x_T + y_T) = x_0 + y_0 \quad (6)$$

$$\mathbb{E}(14x_T + 3x_T^2 + 6x_T y_T + 3y_T^2) = 14x_0 + 3x_0^2 + 6x_0 y_0 + 3y_0^2 \quad (7)$$

4.2 Deriving Bounds for Moments

In Section 4.1 we derived martingale expressions for linear PTS with a single loop that satisfy the OST. These martingales, together with the negated loop guard and a bound on its value after termination, can be used to derive bounds for moments of program variables after termination. For that, we implement a saturation algorithm that systematically applies predefined rules in order to *automatically derive bounds on the expected value of monomials after PTS termination*, as follows. While our rules exploit well-known mathematical properties, to the best of our knowledge our symbolic propagation approach has not been implemented elsewhere.

The computation of bounds works by maintaining a set of bounds of monomials $\vec{x}_T^\alpha$ and their expected value $\mathbb{E}(\vec{x}_T^\alpha)$ at the stopping time, and using a set of derivation rules to derive new bounds. The set of derivation rules consists of predefined rules, which apply in general to (random) variables, as well as rules which are obtained from the previously generated martingales. We implemented such derivation of bounds in extension to the probabilistic loop analysis tool **Polar** [30], allowing us to automatically derive bounds for linear PTS, including Figure 1. We note that our work for bounding (expected) values of random variables is not specific to linear loops, or to loop analysis in general, and can further be extended with new derivation rules. Similarly, Theorem 1 is not bound to the application to linear programs. It may be possible to derive analogous results to Theorem 3 for different classes of programs.

Multiplication of monomial bounds. For bounds of monomials, we use algebraic properties based on the signs of existing bounds. In total, there are 5 rules for deriving new upper bounds, and 7 rules for lower bounds, some of which are shown below.

$$\frac{}{\vec{x}^{2\alpha} \geq 0} \text{ (SQUARE-POSITIVE)} \quad \frac{a \leq \vec{x}^\alpha \leq b \quad c \leq \vec{x}^\beta \leq d \quad a \geq 0 \quad c \geq 0}{\vec{x}^{\alpha+\beta} \leq bd} \text{ (MUL-UB-1)}$$

Bounds for moments. Bounds for moments can either be derived directly from a bound of the value of a monomial, or through Jensen's inequality [25]. Additionally, we provide three multiplication rules. Those rules exploit, in addition to bounds of moments, the signs of the random variables in question. In [31, Appendix F] we show the soundness of our derivation rules.

$$\frac{a \leq \vec{x}^\alpha}{a \leq \mathbb{E}(\vec{x}^\alpha)} \text{ (MOMENT-LB)} \quad \frac{\vec{x}^\alpha \leq a}{\mathbb{E}(\vec{x}^\alpha) \leq a} \text{ (MOMENT-UB)} \quad \frac{a \leq \mathbb{E}(\vec{x}^\alpha) \quad 0 \leq a}{a^2 \leq \mathbb{E}(\vec{x}^{2\alpha})} \text{ (JENSEN-LB-1)}$$

$$\frac{a \leq \vec{x}^\alpha \quad a \leq 0 \quad 0 \leq \vec{x}^\beta \quad \mathbb{E}(\vec{x}^\beta) \leq b}{ab \leq \mathbb{E}(\vec{x}^{\alpha+\beta})} \text{ (RV-MUL-1)}$$

Derivation through martingales. Derivation rules are also created based on the synthesised polynomial martingales which satisfy the OST via the **dui** precondition. For such a martingale $m = \sum_{\alpha \in \mathcal{A}} a_\alpha \vec{x}^\alpha$, we set $\mathcal{A}^+ := \{\alpha \in \mathcal{A} \mid a_\alpha > 0\}$ and $\mathcal{A}^- := \{\alpha \in \mathcal{A} \mid a_\alpha < 0\}$. We denote by $m_0 := \sum_{\alpha \in \mathcal{A}} a_\alpha \vec{x}_0^\alpha$ the (potentially symbolic) initial value of the polynomial. As m satisfies the OST, $\mathbb{E}(m_T) = m_0$. From upper/lower bounds of the expected value of all but one monomial $\vec{x}_T^\alpha$ of the polynomial, we derive a new bound for $\mathbb{E}(\vec{x}_T^\alpha)$. With u_α and l_α being existing upper and lower bounds on monomials, we use the following rules (ub-martingale) and (lb-martingale), assuming w.l.o.g. $\alpha \in \mathcal{A}^+$ (otherwise we multiply the martingale by -1).

$$\frac{\alpha \in \mathcal{A}^+ \quad \forall \beta \in \mathcal{A}^+ \setminus \{\alpha\} : l_\beta \leq \mathbb{E}(\vec{x}^\beta) \quad \forall \beta \in \mathcal{A}^- : \mathbb{E}(\vec{x}^\beta) \leq u_\beta}{\mathbb{E}(\vec{x}^\alpha) \leq \frac{1}{a_\alpha} \left(m_0 - \sum_{\beta \in \mathcal{A}^+ \setminus \{\alpha\}} l_\beta a_\beta + \sum_{\beta \in \mathcal{A}^-} u_\beta a_\beta \right)} \text{ (UB-MARTINGALE)}$$

$$\frac{\alpha \in \mathcal{A}^+ \quad \forall \beta \in \mathcal{A}^+ \setminus \{\alpha\} : \mathbb{E}(\vec{x}^\beta) \leq u_\beta \quad \forall \beta \in \mathcal{A}^- : l_\beta \leq \mathbb{E}(\vec{x}^\beta)}{\frac{1}{a_\alpha} \left(m_0 - \sum_{\beta \in \mathcal{A}^+ \setminus \{\alpha\}} u_\beta a_\beta + \sum_{\beta \in \mathcal{A}^-} l_\beta a_\beta \right) \leq \mathbb{E}(\vec{x}^\alpha)} \text{ (LB-MARTINGALE)}$$

Example 8. In Figure 1, by the negated loop guard and by knowing the support of $\text{Uniform}(-1, 0)$, we have $-1 \leq x_T \leq 0$. So, by using the (moment-lb) and (moment-ub) rules, $-1 \leq \mathbb{E}(x_T) \leq 0$. Also by standard interval propagation $0 \leq x_T^2 \leq 1$, and thus, $0 \leq \mathbb{E}(x_T^2) \leq 1$. Because the loop is entered at least once, we know that $k_T \geq 1$. Moreover, by using the (ub-martingale) and (lb-martingale) rules for the martingale in Eq. (5), we derive that $2x_0 \leq \mathbb{E}(k_T) \leq 2x_0 + 2$. Similarly, by using Eq. (6) and the same rules, we obtain that $x_0 + y_0 \leq \mathbb{E}(y_T) \leq x_0 + y_0 + 1$.

Furthermore, using the (rv-mul-1) rule we derive the bound $-2x_0 - 2 \leq \mathbb{E}(k_T x_T)$, since we know that $-1 \leq x_T$ and $1 \leq k_T$ as well as $\mathbb{E}(k_T) \leq 2x_0 + 2$. This is useful, as it enables us to derive the upper bound $\mathbb{E}(k_T^2) \leq 4x_0^2 + \frac{26}{3}x_0 + \frac{26}{3}$ through the (ub-martingale) rule and the martingale in Eq. (4).

Cauchy-Schwarz. Since the polynomial martingales are not capable of tracking all relations between variables, purely linear propagation of bounds via the rules (ub-martingale) and (lb-martingale) is usually insufficient. In some cases, instead of obtaining a bound on $\mathbb{E}(\vec{x}^\alpha)$ directly, it is however possible to obtain one for $\mathbb{E}((a\vec{x}^\beta + b\vec{x}^\gamma)^2)$, such that $\alpha = \gamma + \beta$. In that case, we can use the rules (cs-lb) and (cs-ub), to derive bounds for $\mathbb{E}(\vec{x}^\alpha)$. Those rules are direct consequences of the Cauchy-Schwarz inequality [13].

$$\frac{\mathbb{E}((a\vec{x}^\alpha + b\vec{x}^\beta)^2) \leq g \quad \mathbb{E}(\vec{x}^{2\alpha}) \leq h}{-\frac{\sqrt{gh}}{|b|} - \frac{h}{2} \left(\left| \frac{a}{b} \right| + \frac{a}{b} \right) \leq \mathbb{E}(\vec{x}^{\alpha+\beta})} \text{ (CS-LB)} \quad \frac{\mathbb{E}((a\vec{x}^\alpha + b\vec{x}^\beta)^2) \leq g \quad \mathbb{E}(\vec{x}^{2\alpha}) \leq h}{\mathbb{E}(\vec{x}^{\alpha+\beta}) \leq \frac{\sqrt{gh}}{|b|} + \frac{h}{2} \left(\left| \frac{a}{b} \right| - \frac{a}{b} \right)} \text{ (CS-UB)}$$

To exploit this, we perform square completion by pattern matching, to find expressions of the form $\mathbb{E}((a\vec{x}^\alpha + b\vec{x}^\beta)^2)$ that yield martingales with fewer summands. The other rules are then also instantiated with those expressions.

Example 9. We aim to derive an upper bound for the expression $\mathbb{E}(x_T y_T)$. Such a bound cannot be directly derived from the martingale in Eq. (7), since we are lacking an appropriate bound for $\mathbb{E}(y_T^2)$. We can, however, rewrite this martingale as $14\mathbb{E}(x_T) + 3\mathbb{E}((x_T + y_T)^2) = 14x_0 + 3x_0^2 + 6x_0 y_0 + 3y_0^2$. Since we have a lower bound for $\mathbb{E}(x_T)$ (Example 8), we get that $\mathbb{E}((x_T + y_T)^2) \leq x_0^2 + 2x_0 y_0 + y_0^2 + \frac{14}{3}x_0 + \frac{14}{3}$. Then we can use the (cs-ub) rule, since we already obtained a bound for $\mathbb{E}(x_T^2)$ in Example 8: $\mathbb{E}(x_T y_T) \leq \sqrt{x_0^2 + 2x_0 y_0 + y_0^2 + \frac{14}{3}x_0 + \frac{14}{3}}$.

Limitations of automatic bound generation. The martingale rule is applied for all possible combinations of bounds. To avoid such computationally expensive tasks, we employ aggressive subsumption checking. To facilitate this, we restrict square roots to only contain one single initial variable or numeric values, and only compare the highest degree monomials. After

applying the rules based on Cauchy-Schwarz, we use the sub-additivity of square roots. The equation system defined in Eq. (3) is usually under-determined, and enumeration of all rules is too costly. We therefore select a number of sparsest solutions of the equation system (see [31, Appendix F.1]).

5 Examples of Bound Derivation

We show that our approach is suitable for computing symbolic bounds for the expected value of monomials after loop termination. To this end, we implement the automatic bound derivation in **Polar** and analyse Figures 1 & 2, for both of which the computation of tight symbolic bounds for moments after termination is out of reach for existing tools.

In Table 1 we list symbolic bounds computed for different inputs. Column 1 specifies which moment of the stopping time of the analysed program must be finite in order for the derivation to be sound. For the analysis to be sound, we must verify that $\mathbb{E}(T^2)$ is finite for the bounds derived for Figure 1, and $\mathbb{E}(T^3) < \infty$ for the bounds for Figure 2, since the stopping time depends on variables with unbounded support (see Theorem 3). We currently require those conditions to be verified externally. The condition $\mathbb{E}(T^2) < \infty$ for Figure 1 can be verified using state-of-the-art tools [32], since the variable in the loop guard is not affected by unbounded support updates. To the best of our knowledge, this is however not the case for Figure 2, as existing methods struggle with unbounded updates. We show, through an application of tail bounds in [31, Appendix G], that the required bounds for the moments of the stopping times hold.

While we still require manual intervention, our implementation demonstrates the benefits of our two-step approach, as the tail bounds in the manual proof are powerful for asymptotic analysis but usually do not yield useful symbolic bounds. Those symbolic bounds can then be computed as a second step by our approach¹⁰.

Column 1 of Table 1 states the set of initially known inequalities. The bound for $\mathbb{E}(x_T)$ in Figure 2 is obtained by reasoning about the last step: $\mathbb{E}(x_T) = \mathbb{E}(x + Y \mid x > 0 \wedge x + Y < 0)$, where Y is the distribution of the last update. This update Y is lower bounded in expectation by $Y' = \text{Normal}(-1, 1)$, hence $\mathbb{E}(x_T) \geq \mathbb{E}(Y' \mid Y' < 0)$, which is equal to the mean of a truncated random variable, which for these parameters is lower bounded by -1.3 . By analogous reasoning using a truncated random variable, we get that $\mathbb{E}(x_T^2) \leq \mathbb{E}((Y')^2 \mid Y' < 0) \leq 2.3$. We additionally assume that the variable x_0 is positive in both examples; this is equivalent to assuming that the loop is executed at least once.

Column 3 of Table 1 shows that, in many cases, the obtained bounds are asymptotically sharp, in a sense that the leading terms of upper and lower bounds are equal. This holds also for the bounds for $\mathbb{E}(z_T)$ in Figure 1, although the increments of z are not bounded from below, and are affected by the value of y . To the best of our knowledge, no other approach for cost analysis or bound computation is able to handle those properties simultaneously.

```

 $z := z_0$ 
 $x := x_0$ 
 $k := 0$ 
while  $x \geq 0$  do
   $k := k + 1$ 
   $c \sim \text{Bernoulli}(\frac{7}{10})$ 
  if  $c = 1$  then
     $z \sim \text{Normal}(-1, 1)$ 
     $x := x + z$ 
  else
     $z \sim \text{Normal}(1, 1)$ 
     $x := x + z$ 
  end if
end while

```

Figure 2: Linear loop with unbounded update affecting the loop guard.

¹⁰It is always possible to introduce a program variable that simply updates by one in every step of the program, effectively storing the runtime. Hence, our approach is equally capable of deriving upper and lower bounds on moments of the runtime. This is the variable k in Figures 1 & 2.

[program] assumptions	moment	bound	
[Fig. 1] $\mathbb{E}(T^2) < \infty$ $x_T \leq 0$ $x_T \geq -1$ $x_0 > 0$	$\mathbb{E}(k_T)$	$\geq 2x_0$	$\leq 2x_0 + 2$
	$\mathbb{E}(y_T)$	$\geq x_0 + y_0$	$\leq x_0 + y_0 + 1$
	$\mathbb{E}(z_T)$	$\geq x_0^2 + 2x_0y_0 - 5x_0 + z_0 - 2 y_0 - \sqrt{\frac{56}{3}}\sqrt{x_0} - \sqrt{\frac{56}{3}} - 4$	
		$\leq x_0^2 + 2x_0y_0 - x_0 + z_0 + 2 y_0 + \sqrt{\frac{56}{3}}\sqrt{x_0} + \sqrt{\frac{56}{3}} + 2$	
	$\mathbb{E}(k_T x_T)$	$\geq -2x_0 - 2$	≤ 0
	$\mathbb{E}(k_T y_T)$	$\geq 2x_0^2 + 2x_0y_0 - \frac{4}{3}x_0 - 2 y_0 - \sqrt{\frac{56}{3}}\sqrt{x_0} - \sqrt{\frac{56}{3}} - 2$	
		$\leq 2x_0^2 + 2x_0y_0 + \frac{14}{3}x_0 + 2 y_0 + \sqrt{\frac{56}{3}}\sqrt{x_0} + \sqrt{\frac{56}{3}} + \frac{14}{3}$	
	$\mathbb{E}(x_T y_T)$	$\geq -x_0 - y_0 - \sqrt{\frac{14}{3}}\sqrt{x_0} - \sqrt{\frac{14}{3}} - 1$	
		$\leq x_0 + y_0 + \sqrt{\frac{14}{3}}\sqrt{x_0} + \sqrt{\frac{14}{3}}$	
	$\mathbb{E}(k_T^2)$	$\geq 4x_0^2 + \frac{2}{3}x_0 - 4$	$\leq 4x_0^2 + \frac{26}{3}x_0 + \frac{26}{3}$
[Fig. 2] $\mathbb{E}(T^3) < \infty$ $\mathbb{E}(x_T) \geq -1.3$ $\mathbb{E}(x_T^2) \leq 2.3$ $x_0 > 0$	$\mathbb{E}(y_T^2)$	$\geq x_0^2 + 2x_0y_0 + y_0^2 + \frac{8}{3}x_0 - 2 y_0 - \sqrt{\frac{56}{3}}\sqrt{x_0} - \sqrt{\frac{56}{3}} - 1$	
		$\leq x_0^2 + 2x_0y_0 + y_0^2 + \frac{20}{3}x_0 + 2 y_0 + \sqrt{\frac{56}{3}}\sqrt{x_0} + \sqrt{\frac{56}{3}} + \frac{20}{3}$	
	$\mathbb{E}(k_T)$	$\geq \frac{5}{2}x_0$	$\leq \frac{5}{2}x_0 + \frac{13}{4}$
	$\mathbb{E}(k_T^2)$	$\geq \frac{25}{4}x_0^2 + \frac{115}{4}x_0 - \frac{115}{8}$	
		$\leq \frac{25}{4}x_0^2 + \sqrt{\frac{2875}{8}}x_0 + \frac{115}{4}x_0 + \sqrt{\frac{13225}{8}}\sqrt{x_0} + \sqrt{\frac{34385}{16}} + \frac{529}{8}$	
	$\mathbb{E}(k_T x_T)$	$\geq -\sqrt{\frac{115}{8}}x_0 - \sqrt{\frac{529}{8}}\sqrt{x_0} - \sqrt{\frac{6877}{80}} - \frac{23}{4}$	
		≤ 0	

Table 1: Symbolic bounds on moments of PTS.

6 Related Work

Recently, there have been significant advances in two related problems: *synthesis of probabilistic invariants* and *reasoning about (positive) almost sure termination*. A deep link between these two problems is evident in the preceding, especially in our setting of linear PTS from Section 4. A telling perspective is provided by the Optional Stopping Theorem; the OST typically requires the runtime, the martingale expression or both to be bounded in some sense. Other examples of a mixed precondition for the OST are given in [33, 32, 35, 9]; for an overview, see [31, Appendix B]. The precondition of [32] uses martingales $\{Y_n\}_{n \in \mathbb{N}}$ that are polynomially bounded in n and expects that sufficiently high moments of the runtime are finite. Hence, the setting of [32] only supports sampling from bounded distributions. Similarly, in [33] such a polynomial bound is required for the step-wise difference of a martingale, together with exponential tail decay of the stopping time. In [35] a similar bound for the stopping time is required, but the step-wise difference can grow exponentially. Nonetheless, all of these (almost sure) step-wise difference bounds are incompatible with sampling from distributions with unbounded support. Moreover,

in [9] such a step-wise difference bound is not required, but the martingales are required to be non-negative, which restricts their applicability for unbounded support distributions. Unlike this, our result from Theorem 3 handles unbounded distributions and martingale invariants without a sign constraint for a (restricted) class of PTS programs.

This link between probabilistic invariants and runtime bounds is further investigated in [21]. Here, it is shown that for computing lower bounds on the runtime of probabilistic programs, it is necessary to satisfy the OST. As a natural counterpart to upper bounds, the synthesis of lower bounds of the expected runtime has been studied in [17, 18, 14].

Conversely, upper bounds on the expected runtime can be improved via probabilistic invariants. Upper bounds of the expected runtime can be found by finding suitable non-negative supermartingales [23, 26, 8, 5, 16, 32]. Once the OST is satisfied, our probabilistic invariants can take both negative and positive values, extending thus [9] where moments of polynomials of program costs are derived provided a fixed lower bound exists.

Higher-order moments of program variables are inferred in [2, 3, 33]. Importantly, in [14] the authors provide lower bounds for expected program quantities, even if the program in question is not almost surely terminating. This is achieved by reducing the question to a well-chosen almost surely terminating program. The general approach of checking preconditions of the OST for polynomial invariants of higher-order is further studied in [27], providing sufficient conditions that have been successfully implemented in [15, 11, 19].

7 Conclusion

We study the invariant synthesis problem for probabilistic transition systems, by leveraging the Optional Stopping Theorem e.g. [12], for which we present a general sufficient precondition, coined as **dui**. Our **dui** yields a criterion under which a given polynomial expression qualifies as an invariant. We use this criterion, to automatically synthesise polynomial probabilistic invariants for a certain class of linear loops, and use the invariants to automatically derive bounds for higher- and mixed-moments of program variables and the runtime after loop termination. Notably, our approach handles distributions with unbounded support, enabling thus the analysis of loops that cannot be handled by other techniques. Future work includes handling nonlinear loops for which our **dui** precondition can be used or adjusted, thus further improving the automatic derivation of moment bounds. These methods are especially promising regarding nested loops. Such an approach would necessarily involve unbounded updates, as both termination time and variable updates of inner loops can scarcely be expected to be bounded by constants. The flexibility of the **dui** precondition could provide tractable sufficient conditions for the applicability of the OST under compositional reasoning.

Acknowledgements. Anne Schreuder is supported by the Cantab Capital Institute for the Mathematics of Information and by the Academy of Finland Centre of Excellence Programme grant number 346315 “Finnish centre of excellence in Randomness and Structures (FiRST)”.

This research was also partially funded by the Austrian Science Fund (FWF) [Grant ID 10.55776/DOC1345324] and the Vienna Science and Technology Fund (WWTF), State of Lower Austria [Grant ID 10.47379/ICT25017].

References

- [1] Barthe, G., Espitau, T., Ferrer Fioriti, L.M., Hsu, J.: Synthesizing probabilistic invariants via doob’s decomposition. In: Chaudhuri, S., Farzan, A. (eds.) *Computer Aided Verification*. pp. 43–61. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-41528-4_3
- [2] Bartocci, E., Kovács, L., Stankovič, M.: Automatic generation of moment-based invariants for prob-solvable loops. In: Chen, Y.F., Cheng, C.H., Esparza, J. (eds.) *Automated Technology for Verification and Analysis*. pp. 255–276. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-31784-3_15
- [3] Bartocci, E., Kovács, L., Stankovič, M.: Mora - automatic generation of moment-based invariants. In: Biere, A., Parker, D. (eds.) *Tools and Algorithms for the Construction and Analysis of Systems*. pp. 492–498. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-45190-5_28
- [4] Batz, K., Kaminski, B.L., Katoen, J., Matheja, C.: Relatively complete verification of probabilistic programs: an expressive language for expectation-based reasoning. *Proc. ACM Program. Lang.* **5**(POPL), 1–30 (2021). <https://doi.org/10.1145/3434320>
- [5] Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: Sharygina, N., Veith, H. (eds.) *Computer Aided Verification*. pp. 511–526. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_34
- [6] Chakarov, A., Sankaranarayanan, S.: Expectation invariants for probabilistic program loops as fixed points. In: Müller-Olm, M., Seidl, H. (eds.) *Static Analysis*. pp. 85–100. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-10936-7_6
- [7] Chakarov, A., Voronin, Y.L., Sankaranarayanan, S.: Deductive proofs of almost sure persistence and recurrence properties. In: Chechik, M., Raskin, J.F. (eds.) *Tools and Algorithms for the Construction and Analysis of Systems*. pp. 260–279. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_15
- [8] Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In: *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. pp. 327–342. POPL ’16, Association for Computing Machinery, New York, NY, USA (2016). <https://doi.org/10.1145/2837614.2837639>
- [9] Chatterjee, K., Goharshady, A.K., Meggendorfer, T., Žikelić, Đ.: Quantitative bounds on resource usage of probabilistic programs. *Proc. ACM Program. Lang.* **8**(OOPSLA1) (Apr 2024). <https://doi.org/10.1145/3649824>
- [10] Chatterjee, K., Novotný, P., Žikelić, Đ.: Stochastic invariants for probabilistic termination. In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. pp. 145–160. POPL ’17, Association for Computing Machinery, New York, NY, USA (2017). <https://doi.org/10.1145/3009837.3009873>
- [11] Chen, Y.F., Hong, C.D., Wang, B.Y., Zhang, L.: Counterexample-guided polynomial loop invariant generation by lagrange interpolation. In: Kroening, D., Păsăreanu, C.S. (eds.) *Computer Aided Verification*. pp. 658–674. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_44
- [12] Cohen, S.N., Elliott, R.J.: *Stochastic Calculus and Applications*. Springer New York Heidelberg Dordrecht London, second edn. (2015)
- [13] Durrett, R.: *Probability: theory and examples*. Belmont, Calif. : Thomson Brooks/ Cole (2005)
- [14] Feng, S., Chen, M., Su, H., Kaminski, B.L., Katoen, J.P., Zhan, N.: Lower bounds for possibly divergent probabilistic programs. *Proc. ACM Program. Lang.* **7**(OOPSLA1) (Apr 2023). <https://doi.org/10.1145/3586051>
- [15] Feng, Y., Zhang, L., Jansen, D.N., Zhan, N., Xia, B.: Finding polynomial loop invariants for probabilistic programs. In: D’Souza, D., Narayan Kumar, K. (eds.) *Automated Technol-*

- ogy for Verification and Analysis. pp. 400–416. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_26
- [16] Ferrer Fioriti, L.M., Hermanns, H.: Probabilistic termination: Soundness, completeness, and compositionality. In: Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 489–501. POPL ’15, Association for Computing Machinery, New York, NY, USA (2015). <https://doi.org/10.1145/2676726.2677001>
 - [17] Frohn, F., Naaf, M., Brockschmidt, M., Giesl, J.: Inferring lower runtime bounds for integer programs. *ACM Trans. Program. Lang. Syst.* **42**(3), 13:1–13:50 (2020). <https://doi.org/10.1145/3410331>
 - [18] Fu, H., Chatterjee, K.: Termination of nondeterministic probabilistic programs. In: Enea, C., Piskac, R. (eds.) *Verification, Model Checking, and Abstract Interpretation*. pp. 468–490. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-11245-5_22
 - [19] Gretz, F., Katoen, J.P., McIver, A.: Prinsys—on a quest for probabilistic loop invariants. In: Joshi, K., Siegle, M., Stoelinga, M., D’Argenio, P.R. (eds.) *Quantitative Evaluation of Systems*. pp. 193–208. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40196-1_17
 - [20] Grimmett, G.R., Stirzaker, D.R.: *Probability and random processes*. Oxford University Press (2001)
 - [21] Hark, M., Kaminski, B.L., Giesl, J., Katoen, J.: Aiming low is harder: induction for lower bounds in probabilistic program verification. *Proc. ACM Program. Lang.* **4**(POPL), 37:1–37:28 (2020). <https://doi.org/10.1145/3371105>
 - [22] Huang, M., Fu, H., Chatterjee, K., Goharshady, A.K.: Modular verification for almost-sure termination of probabilistic programs. *Proc. ACM Program. Lang.* **3**(OOPSLA) (Oct 2019). <https://doi.org/10.1145/3360555>
 - [23] Kaminski, B.L., Katoen, J.P., Matheja, C., Olmedo, F.: Weakest precondition reasoning for expected run-times of probabilistic programs. In: Thiemann, P. (ed.) *Programming Languages and Systems*. pp. 364–389. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49498-1_15
 - [24] Katoen, J.P., McIver, A.K., Meinicke, L.A., Morgan, C.C.: Linear-invariant generation for probabilistic programs. In: Cousot, R., Martel, M. (eds.) *Static Analysis*. pp. 390–406. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15769-1_24
 - [25] Klenke, A.: *Probability Theory: A comprehensive course*. Springer Verlag Berlin Heidelberg (2006)
 - [26] Kura, S., Urabe, N., Hasuo, I.: Tail probabilities for randomized program runtimes via martingales for higher moments. In: Vojnar, T., Zhang, L. (eds.) *Tools and Algorithms for the Construction and Analysis of Systems*. pp. 135–153. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-17465-1_8
 - [27] McIver, A., Morgan, C.: *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science, Springer (2005). <https://doi.org/10.1007/b138392>
 - [28] Meyer, F., Hark, M., Giesl, J.: Inferring expected runtimes of probabilistic integer programs using expected sizes. In: Groote, J.F., Larsen, K.G. (eds.) *Tools and Algorithms for the Construction and Analysis of Systems*. pp. 250–269. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-72016-2_14
 - [29] Moosbrugger, M., Bartocci, E., Katoen, J.P., Kovács, L.: Automated termination analysis of polynomial probabilistic programs. In: Yoshida, N. (ed.) *Programming Languages and Systems*. pp. 491–518. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-72019-3_18
 - [30] Moosbrugger, M., Stankovic, M., Bartocci, E., Kovács, L.: This is the moment for probabilistic loops. *Proc. ACM Program. Lang.* **6**(OOPSLA2), 1497–1525 (2022). <https://doi.org/10.1145/3563341>
 - [31] Schreuder, A., Winkler, L., Kovács, L., Ong, C.H.L.: Polynomial invariants for probabilistic tran-

- sition systems with unbounded support (2026), <https://arxiv.org/abs/2609.02446>
- [32] Wang, D., Hoffmann, J., Reps, T.: Central moment analysis for cost accumulators in probabilistic programs. In: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. pp. 559–573. PLDI 2021, Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3453483.3454062>
 - [33] Wang, P., Fu, H., Goharshady, A.K., Chatterjee, K., Qin, X., Shi, W.: Cost analysis of non-deterministic probabilistic programs. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 204–220. PLDI 2019, Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3314221.3314581>
 - [34] Williams, D.: Probability with Martingales. Cambridge University Press (1991)
 - [35] Yang, T., Feng, S., Fu, H., Zhan, N., Ke, J., Wu, S.: Piecewise analysis of probabilistic programs via k -induction. Proc. ACM Program. Lang. **10**(POPL) (Jan 2026). <https://doi.org/10.1145/3776709>