







Computing Expected Runtimes for Constant Probability Programs

Jürgen Giesl¹ , Peter Giesl² , and Marcel Hark¹  

¹ LuFG Informatik 2, RWTH Aachen University, Aachen, Germany
`{giesl,marcel.hark}@cs.rwth-aachen.de`

² Department of Mathematics, University of Sussex, Brighton, UK
`p.a.giesl@sussex.ac.uk`

Abstract. We introduce the class of *constant probability (CP) programs* and show that classical results from probability theory directly yield a simple decision procedure for (positive) almost sure termination of programs in this class. Moreover, asymptotically tight bounds on their expected runtime can always be computed easily. Based on this, we present an algorithm to infer the *exact* expected runtime of any CP program.

Keywords: Probabilistic programs · Expected runtimes ·
(Positive) almost sure termination · Complexity · Decidability

1 Introduction

Probabilistic programs are used to describe randomized algorithms and probability distributions, with applications in many areas. As an example, consider the well-known program which models the race between a tortoise and a hare (see, e.g., [10, 24, 30]). As long as the tortoise (variable t) is not behind the hare (variable h), it does one step in each iteration. With probability $\frac{1}{2}$, the hare stays at its position and with probability $\frac{1}{2}$ it does a random number of steps uniformly chosen between 0 and 10. The race ends when the hare is in front of the tortoise. Here, the hare wins with probability one and the technique of [30] infers the upper bound $\frac{2}{3} \cdot \max(t - h + 9, 0)$ on the expected number of loop iterations. Thus, the program is positively almost surely terminating.

```
while ( $h \leq t$ ) {  
   $t = t + 1$ ;  
   $\{h = h + \text{Unif}(0, 10)\} \oplus_{\frac{1}{2}} \{h = h\}$ ;  
}
```

Section 2 recapitulates preliminaries on probabilistic programs and on the connection between their expected runtime and their corresponding recurrence equation. Then we show in Sects. 3 and 4 that classical results on random walk theory directly yield a very simple decision procedure for (positive) almost sure

Supported by the DFG Research Training Group 2236 UnRAVeL and the London Mathematical Society (Grant 41662, Research in Pairs).

termination of *CP programs* like the tortoise and hare example. In this way, we also obtain asymptotically tight bounds on the expected runtime of any CP program. Based on these bounds, in Sect. 5 we develop the first algorithm to compute closed forms for the *exact* expected runtime of such programs. In Sect. 6, we present its implementation in our tool KoAT [9] and discuss related and future work. We refer to [20] for a collection of examples to illustrate the application of our algorithm and for all proofs.

2 Expected Runtimes of Probabilistic Programs

Example 1 (Tortoise and Hare). The program $\mathcal{P}_{\text{race}}$ on the right formulates the race of the tortoise and the hare as a CP program. In the loop guard, we use the scalar product $(1, -1) \bullet (t, h)$ which stands for $t - h$. Exactly one of the instructions with numbers in brackets [...] is executed in each loop iteration and the number indicates the probability that the corresponding instruction is chosen.

```

while  $((1, -1) \bullet (t, h) > -1)$  {
   $(t, h) = (t, h) + (1, 0)$   $[\frac{6}{11}]$ ;
   $(t, h) = (t, h) + (1, 1)$   $[\frac{1}{22}]$ ;
   $(t, h) = (t, h) + (1, 2)$   $[\frac{1}{22}]$ ;
   $(t, h) = (t, h) + (1, 3)$   $[\frac{1}{22}]$ ;
   $\vdots$ 
   $(t, h) = (t, h) + (1, 10)$   $[\frac{1}{22}]$ ;
}
```

We now define the kind of probabilistic programs considered in this paper.

Definition 2 (Probabilistic Program). A program has the form on the right, where $\mathbf{x} = (x_1, \dots, x_r)$ for some $r \geq 1$ is a tuple of pairwise different program variables, $\mathbf{a}, \mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{Z}^r$ are tuples of integers, the \mathbf{c}_j are pairwise distinct, $b \in \mathbb{Z}$, \bullet is the scalar product (i.e., $(a_1, \dots, a_r) \bullet (x_1, \dots, x_r) = a_1 \cdot x_1 + \dots + a_r \cdot x_r$), and $\mathbf{d} \in \mathbb{Z}^r$ with $\mathbf{a} \bullet \mathbf{d} \leq b$.

```

while  $(\mathbf{a} \bullet \mathbf{x} > b)$  {
   $\mathbf{x} = \mathbf{x} + \mathbf{c}_1$   $[p_{\mathbf{c}_1}(\mathbf{x})]$ ;
   $\vdots$ 
   $\mathbf{x} = \mathbf{x} + \mathbf{c}_n$   $[p_{\mathbf{c}_n}(\mathbf{x})]$ ;
   $\mathbf{x} = \mathbf{d}$   $[p'(\mathbf{x})]$ ;
}
```

We require $p_{\mathbf{c}_1}(\mathbf{x}), \dots, p_{\mathbf{c}_n}(\mathbf{x}), p'(\mathbf{x}) \in \mathbb{R}_{\geq 0} = \{r \in \mathbb{R} \mid r \geq 0\}$ and $\sum_{1 \leq j \leq n} p_{\mathbf{c}_j}(\mathbf{x}) + p'(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathbb{Z}^r$. It is a program with direct termination if there is an $\mathbf{x} \in \mathbb{Z}^r$ with $\mathbf{a} \bullet \mathbf{x} > b$ and $p'(\mathbf{x}) > 0$. If all probabilities are constant, i.e., if there are $p_{\mathbf{c}_1}, \dots, p_{\mathbf{c}_n}, p' \in \mathbb{R}_{\geq 0}$ such that $p_{\mathbf{c}_j}(\mathbf{x}) = p_{\mathbf{c}_j}$ and $p'(\mathbf{x}) = p'$ for all $1 \leq j \leq n$ and all $\mathbf{x} \in \mathbb{Z}^r$, we call it a constant probability (CP) program.

Such a program means that the integer variables \mathbf{x} are changed to $\mathbf{x} + \mathbf{c}_j$ with probability $p_{\mathbf{c}_j}(\mathbf{x})$. For inputs \mathbf{x} with $\mathbf{a} \bullet \mathbf{x} \leq b$ the program terminates immediately. Note that the program in Example 1 has *no* direct termination (i.e., $p'(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbb{Z}^r$). Since the values of the program variables only depend on their values in the previous loop iteration, our programs correspond to *Markov Chains* [32] and they are related to *random walks* [16, 21, 33], cf. [20] for details.

Clearly, in general termination is undecidable and closed forms for the runtimes of programs are not computable. Thus, decidability results can only be obtained for suitably restricted forms of programs. Our class nevertheless

includes many examples that are often regarded in the literature on probabilistic programs. So while other approaches are concerned with *incomplete* techniques to analyze termination and complexity, we investigate classes of probabilistic programs where one can *decide* the termination behavior, *always* find complexity bounds, and even compute the expected runtime *exactly*. Our decision procedure could be integrated into general tools for termination and complexity analysis of probabilistic programs: As soon as one has to investigate a sub-program that falls into our class, one can use the decision procedure to compute its exact runtime. Our contributions provide a starting point for such results and the considered class of programs can be extended further in future work.

In probability theory (see, e.g., [2]), given a set Ω of possible events, the goal is to measure the probability that events are in certain subsets of Ω . To this end, one regards a set \mathfrak{F} of subsets of Ω , such that \mathfrak{F} contains the full set Ω and is closed under complement and countable unions. Such a set \mathfrak{F} is called a σ -field, and a pair of Ω and a corresponding σ -field \mathfrak{F} is called a *measurable space*.

A *probability space* $(\Omega, \mathfrak{F}, \mathbb{P})$ extends a measurable space (Ω, \mathfrak{F}) by a *probability measure* \mathbb{P} which maps every set from \mathfrak{F} to a number between 0 and 1, where $\mathbb{P}(\Omega) = 1$, $\mathbb{P}(\emptyset) = 0$, and $\mathbb{P}(\bigsqcup_{j \geq 0} A_j) = \sum_{j \geq 0} \mathbb{P}(A_j)$ for any pairwise disjoint sets $A_0, A_1, \dots \in \mathfrak{F}$. So $\mathbb{P}(A)$ is the probability that an event from Ω is in the subset A . In our setting, we use the probability space $((\mathbb{Z}^r)^\omega, \mathfrak{F}^{\mathbb{Z}^r}, \mathbb{P}_{x_0}^{\mathcal{P}})$ arising from the standard cylinder-set construction of MDP theory, cf. [20]. Here, $(\mathbb{Z}^r)^\omega$ corresponds to all infinite sequences of program states and $\mathbb{P}_{x_0}^{\mathcal{P}}$ is the probability measure induced by the program \mathcal{P} when starting in the state $x_0 \in \mathbb{Z}^r$. For example, if $A \subseteq (\mathbb{Z}^2)^\omega$ consists of all infinite sequences starting with $(5, 1)$, $(6, 1)$, $(7, 6)$, then $\mathbb{P}_{(5,1)}^{\mathcal{P}_{\text{race}}}(A) = \frac{6}{11} \cdot \frac{1}{22} = \frac{3}{121}$. So, if one starts with $(5, 1)$, then $\frac{3}{121}$ is the probability that the next two states are $(6, 1)$ and $(7, 6)$. Once a state is reached that violates the loop guard, then the probability to remain in this state is 1. Hence, if B contains all infinite sequences starting with $(7, 8)$, $(7, 8)$, then $\mathbb{P}_{(7,8)}^{\mathcal{P}_{\text{race}}}(B) = 1$. In the following, for any set of numbers M let $\bar{M} = M \cup \{\infty\}$.

Definition 3 (Termination Time). *For a program \mathcal{P} as in Definition 2, its termination time is the random variable $T^{\mathcal{P}} : (\mathbb{Z}^r)^\omega \rightarrow \bar{\mathbb{N}}$ that maps every infinite sequence (z_0, z_1, \dots) to the first index j where z_j violates \mathcal{P} 's loop guard.*

Thus, $T^{\mathcal{P}_{\text{race}}}(((5, 1), (6, 1), (7, 8), (7, 8), \dots)) = 2$ and $T^{\mathcal{P}_{\text{race}}}(((5, 1), (6, 1), (5, 6), (8, 6), (9, 6), \dots)) = \infty$ (i.e., this sequence always satisfies $\mathcal{P}_{\text{race}}$'s loop guard as the j th entry is $(5 + j, 6)$ for $j \geq 3$). Now we can define the different notions of termination and the expected runtime of a probabilistic program. As usual, for any random variable X on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$, $\mathbb{P}(X = j)$ stands for $\mathbb{P}(X^{-1}(\{j\}))$. So $\mathbb{P}_{x_0}^{\mathcal{P}}(T^{\mathcal{P}} = j)$ is the probability that a sequence has termination time j . Similarly, $\mathbb{P}_{x_0}^{\mathcal{P}}(T^{\mathcal{P}} < \infty) = \sum_{j \in \mathbb{N}} \mathbb{P}_{x_0}^{\mathcal{P}}(T^{\mathcal{P}} = j)$. The *expected value* $\mathbb{E}(X)$ of a random variable $X : \Omega \rightarrow \bar{\mathbb{N}}$ for a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ is the weighted average under the probability measure \mathbb{P} , i.e., $\mathbb{E}(X) = \sum_{j \in \bar{\mathbb{N}}} j \cdot \mathbb{P}(X = j)$, where $\infty \cdot 0 = 0$ and $\infty \cdot u = \infty$ for all $u \in \mathbb{N}_{>0}$.

Definition 4 (Termination and Expected Runtime). *A program \mathcal{P} as in Definition 2 is almost surely terminating (AST) if $\mathbb{P}_{x_0}^{\mathcal{P}}(T^{\mathcal{P}} < \infty) = 1$ for any*

initial value $\mathbf{x}_0 \in \mathbb{Z}^r$. For any $\mathbf{x}_0 \in \mathbb{Z}^r$, its expected runtime $rt_{\mathbf{x}_0}^{\mathcal{P}}$ (i.e., the expected number of loop iterations) is defined as the expected value of the random variable $T^{\mathcal{P}}$ under the probability measure $\mathbb{P}_{\mathbf{x}_0}^{\mathcal{P}}$, i.e., $rt_{\mathbf{x}_0}^{\mathcal{P}} = \mathbb{E}_{\mathbf{x}_0}^{\mathcal{P}}(T^{\mathcal{P}}) = \sum_{j \in \mathbb{N}} j \cdot \mathbb{P}_{\mathbf{x}_0}^{\mathcal{P}}(T^{\mathcal{P}}=j)$ if $\mathbb{P}_{\mathbf{x}_0}^{\mathcal{P}}(T^{\mathcal{P}} < \infty) = 1$, and $rt_{\mathbf{x}_0}^{\mathcal{P}} = \mathbb{E}_{\mathbf{x}_0}^{\mathcal{P}}(T^{\mathcal{P}}) = \infty$ otherwise.

The program \mathcal{P} is positively almost surely terminating (PAST) if for any initial value $\mathbf{x}_0 \in \mathbb{Z}^r$, the expected runtime of \mathcal{P} is finite, i.e., if $rt_{\mathbf{x}_0}^{\mathcal{P}} = \mathbb{E}_{\mathbf{x}_0}^{\mathcal{P}}(T^{\mathcal{P}}) < \infty$.

Example 5 (Expected Runtime for $\mathcal{P}_{\text{race}}$). By the observations in Sect. 4 we will infer that $\frac{2}{3} \cdot (t - h + 1) \leq rt_{(t,h)}^{\mathcal{P}_{\text{race}}} \leq \frac{2}{3} \cdot (t - h + 1) + \frac{16}{3}$ holds whenever $t - h > -1$, cf. Example 22. So the expected number of steps until termination is finite (and linear in the input variables) and thus, $\mathcal{P}_{\text{race}}$ is PAST. The algorithm in Sect. 5 will even be able to compute $rt_{(t,h)}^{\mathcal{P}_{\text{race}}}$ exactly, cf. Example 34.

If the initial values \mathbf{x}_0 violate the loop guard, then the runtime is trivially 0.

Corollary 6 (Expected Runtime for Violating Initial Values). For any program \mathcal{P} as in Definition 2 and any $\mathbf{x}_0 \in \mathbb{Z}^r$ with $\mathbf{a} \bullet \mathbf{x}_0 \leq b$, we have $rt_{\mathbf{x}_0}^{\mathcal{P}} = 0$.

To obtain our results, we use an alternative, well-known characterization of the expected runtime, cf. e.g., [3, 8, 15, 24–27, 32, 34]. To this end, we search for the *smallest* (or “least”) solution of the recurrence equation that describes the runtime of the program as 1 plus the sum of the runtimes in the next loop iteration, multiplied with the corresponding probabilities. Here, functions are compared pointwise, i.e., for $f, g : \mathbb{Z}^r \rightarrow \mathbb{R}_{\geq 0}$ we have $f \leq g$ if $f(\mathbf{x}) \leq g(\mathbf{x})$ holds for all $\mathbf{x} \in \mathbb{Z}^r$. So we search for the smallest function $f : \mathbb{Z}^r \rightarrow \mathbb{R}_{\geq 0}$ that satisfies

$$f(\mathbf{x}) = \sum_{1 \leq j \leq n} p_{c_j}(\mathbf{x}) \cdot f(\mathbf{x} + \mathbf{c}_j) + p'(\mathbf{x}) \cdot f(\mathbf{d}) + 1 \quad \text{for all } \mathbf{x} \text{ with } \mathbf{a} \bullet \mathbf{x} > b. \quad (1)$$

Equivalently, we can search for the least fixpoint of the “expected runtime transformer” $\mathcal{L}^{\mathcal{P}}$ which transforms the left-hand side of (1) into its right-hand side.

Definition 7 ($\mathcal{L}^{\mathcal{P}}$, cf. [32]). For \mathcal{P} as in Definition 2, we define the expected runtime transformer $\mathcal{L}^{\mathcal{P}} : (\mathbb{Z}^r \rightarrow \mathbb{R}_{\geq 0}) \rightarrow (\mathbb{Z}^r \rightarrow \mathbb{R}_{\geq 0})$, where for any $f : \mathbb{Z}^r \rightarrow \mathbb{R}_{\geq 0}$:

$$\mathcal{L}^{\mathcal{P}}(f)(\mathbf{x}) = \begin{cases} \sum_{1 \leq j \leq n} p_{c_j}(\mathbf{x}) \cdot f(\mathbf{x} + \mathbf{c}_j) + p'(\mathbf{x}) \cdot f(\mathbf{d}) + 1, & \text{if } \mathbf{a} \bullet \mathbf{x} > b \\ f(\mathbf{x}), & \text{if } \mathbf{a} \bullet \mathbf{x} \leq b \end{cases}$$

Example 8 (Expected Runtime Transformer for $\mathcal{P}_{\text{race}}$). For $\mathcal{P}_{\text{race}}$ from Example 1, $\mathcal{L}^{\mathcal{P}_{\text{race}}}$ maps any function $f : \mathbb{Z}^2 \rightarrow \mathbb{R}_{\geq 0}$ to $\mathcal{L}^{\mathcal{P}_{\text{race}}}(f)$, where $\mathcal{L}^{\mathcal{P}_{\text{race}}}(f)(t, h) =$

$$\begin{cases} \frac{6}{11} \cdot f(t + 1, h) + \frac{1}{22} \cdot \sum_{1 \leq j \leq 10} f(t + 1, h + j) + 1, & \text{if } t - h > -1 \\ f(t, h), & \text{if } t - h \leq -1 \end{cases} \quad (2)$$

Theorem 9 recapitulates that the least fixpoint of $\mathcal{L}^{\mathcal{P}}$ indeed yields an equivalent characterization of the expected runtime. In the following, let $\mathbf{o} : \mathbb{Z}^r \rightarrow \mathbb{R}_{\geq 0}$ be the function with $\mathbf{o}(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbb{Z}^r$.

Theorem 9 (Connection Between Expected Runtime and Least Fixpoint of $\mathcal{L}^{\mathcal{P}}$, cf. [32]). For any \mathcal{P} as in Definition 2, the expected runtime transformer $\mathcal{L}^{\mathcal{P}}$ is continuous. Thus, it has a least fixpoint $\text{lfp}(\mathcal{L}^{\mathcal{P}}) : \mathbb{Z}^r \rightarrow \overline{\mathbb{R}_{\geq 0}}$ with $\text{lfp}(\mathcal{L}^{\mathcal{P}}) = \sup\{\mathbf{o}, \mathcal{L}^{\mathcal{P}}(\mathbf{o}), (\mathcal{L}^{\mathcal{P}})^2(\mathbf{o}), \dots\}$. Moreover, the least fixpoint of $\mathcal{L}^{\mathcal{P}}$ is the expected runtime of \mathcal{P} , i.e., for any $\mathbf{x}_0 \in \mathbb{Z}^r$, we have $\text{lfp}(\mathcal{L}^{\mathcal{P}})(\mathbf{x}_0) = \text{rt}_{\mathbf{x}_0}^{\mathcal{P}}$.

So the expected runtime $\text{rt}_{(t,h)}^{\mathcal{P}_{\text{race}}}$ can also be characterized as the smallest function $f : \mathbb{Z}^2 \rightarrow \overline{\mathbb{R}_{\geq 0}}$ satisfying $f(t, h) = (2)$, i.e., as the least fixpoint of $\mathcal{L}^{\mathcal{P}_{\text{race}}}$.

3 Expected Runtime of Programs with Direct Termination

We start with stating a decidability result for the case where for all \mathbf{x} with $\mathbf{a} \bullet \mathbf{x} > b$, the probability $p'(\mathbf{x})$ for direct termination is at least p' for some $p' > 0$. Intuitively, these programs have a termination time whose distribution is closely related to the geometric distribution with parameter p' (which has expected value $\frac{1}{p'}$). By using the alternative characterization of $\text{rt}_{\mathbf{x}_0}^{\mathcal{P}}$ from Theorem 9, one obtains that such programs are *always PAST* and their expected runtime is indeed *bounded by the constant $\frac{1}{p'}$* . This result will be used in Sect. 5 when computing the *exact* expected runtime of such programs. The more involved case where $p'(\mathbf{x}) = 0$ is considered in Sect. 4.

Theorem 10 (PAST and Expected Runtime for Programs With Direct Termination). Let \mathcal{P} be a program as in Definition 2 where there is a $p' > 0$ such that $p'(\mathbf{x}) \geq p'$ for all $\mathbf{x} \in \mathbb{Z}^r$ with $\mathbf{a} \bullet \mathbf{x} > b$. Then \mathcal{P} is PAST and its expected runtime is at most $\frac{1}{p'}$, i.e., $\text{rt}_{\mathbf{x}_0}^{\mathcal{P}} \leq \frac{1}{p'}$ if $\mathbf{a} \bullet \mathbf{x}_0 > b$, and $\text{rt}_{\mathbf{x}_0}^{\mathcal{P}} = 0$ if $\mathbf{a} \bullet \mathbf{x}_0 \leq b$.

Example 11 (Example 1 with Direct Termination). Consider the variant $\mathcal{P}_{\text{direct}}$ of $\mathcal{P}_{\text{race}}$ on the right, where in each iteration, the hare either does nothing with probability $\frac{9}{10}$ or one directly reaches a configuration where the hare is ahead of the tortoise.

```

while  $((1, -1) \bullet (t, h) > -1)$  {
   $(t, h) = (t, h) + (1, 0)$     $[\frac{9}{10}]$ ;
   $(t, h) = (7, 8)$           $[\frac{1}{10}]$ ;
}
```

By Theorem 10 the program is PAST and its expected runtime is at most $\frac{1}{\frac{1}{10}} = 10$, i.e., independent of the initial state it takes at most 10 loop iterations on average. In Sect. 5 it will turn out that 10 is indeed the exact expected runtime, cf. Example 32.

4 Expected Runtimes of Constant Probability Programs

Now we present a very simple decision procedure for termination of CP programs (Sect. 4.2) and show how to infer their asymptotic expected runtimes (Sect. 4.3). This will be needed for the computation of exact expected runtimes in Sect. 5.

4.1 Reduction to Random Walk Programs

As a first step, we show that we can restrict ourselves to *random walk programs*, i.e., programs with a single program variable x and the loop condition $x > 0$.

Definition 12 (Random Walk Program). A CP program \mathcal{P} is called a random walk program if there exist $m, k \in \mathbb{N}$ and $d \in \mathbb{Z}$ with $d \leq 0$ such that \mathcal{P} has the form on the right. Here, we require that $m > 0$ implies $p_m > 0$ and that $k > 0$ implies $p_{-k} > 0$.

```

while (x > 0) {
  x = x + m   [p_m];
  ⋮
  x = x + 1   [p_1];
  x = x       [p_0];
  x = x - 1   [p_{-1}];
  ⋮
  x = x - k   [p_{-k}];
  x = d       [p'];
}
    
```

Definition 13 shows how to transform any CP program as in Definition 2 into a random walk program. The idea is to replace the tuple \mathbf{x} by a single variable x that stands for $\mathbf{a} \bullet \mathbf{x} - b$. Thus, the loop condition $\mathbf{a} \bullet \mathbf{x} > b$ now becomes $x > 0$. Moreover, a change from \mathbf{x} to $\mathbf{x} + \mathbf{c}_j$ now becomes a change from x to $x + \mathbf{a} \bullet \mathbf{c}_j$.

```

while (a • x > b) {
  x = x + c_1 [p_{c_1}];
  ⋮
  x = x + c_n [p_{c_n}];
  x = d       [p'];
}
    
```

Definition 13 (Transforming CP Programs to Random Walk Programs). Let \mathcal{P} be the CP program on the left with $\mathbf{x} = (x_1, \dots, x_r)$ and $\mathbf{a} \bullet \mathbf{d} \leq b$. Let $\text{rdw}_{\mathcal{P}}$ denote the affine map $\text{rdw}_{\mathcal{P}}: \mathbb{Z}^r \rightarrow \mathbb{Z}$ with $\text{rdw}_{\mathcal{P}}(\mathbf{z}) = \mathbf{a} \bullet \mathbf{z} - b$ for all $\mathbf{z} \in \mathbb{Z}^r$. Thus, $\text{rdw}_{\mathcal{P}}(\mathbf{d}) \leq 0$. Let $k_{\mathcal{P}}, m_{\mathcal{P}} \in \mathbb{N}$ be minimal such

```

while (x > 0) {
  x = x + m_{\mathcal{P}} [p_{m_{\mathcal{P}}}^{\text{rdw}}];
  ⋮
  x = x - k_{\mathcal{P}} [p_{-k_{\mathcal{P}}}^{\text{rdw}}];
  x = \text{rdw}_{\mathcal{P}}(\mathbf{d}) [p'];
}
    
```

that $-k_{\mathcal{P}} \leq \mathbf{a} \bullet \mathbf{c}_j \leq m_{\mathcal{P}}$ holds for all $1 \leq j \leq n$. For $-k_{\mathcal{P}} \leq j \leq m_{\mathcal{P}}$, we define $p_j^{\text{rdw}} = \sum_{1 \leq u \leq n, \mathbf{a} \bullet \mathbf{c}_u = j} p_{\mathbf{c}_u}$. This results in the random walk program \mathcal{P}^{rdw} on the right.

Example 14 (Transforming $\mathcal{P}_{\text{race}}$). For the program $\mathcal{P}_{\text{race}}$ of Example 1, the mapping $\text{rdw}_{\mathcal{P}_{\text{race}}}: \mathbb{Z}^2 \rightarrow \mathbb{Z}$ is $\text{rdw}_{\mathcal{P}_{\text{race}}}(t, h) = (1, -1) \bullet (t, h) + 1 = t - h + 1$. Hence we obtain the random walk program $\mathcal{P}_{\text{race}}^{\text{rdw}}$ on the right, where $x = \text{rdw}_{\mathcal{P}_{\text{race}}}(t, h)$ represents the distance between the tortoise and the hare.

```

while (x > 0) {
  x = x + 1   [6/11];
  x = x       [1/22];
  x = x - 1   [1/22];
  x = x - 2   [1/22];
  ⋮
  x = x - 9   [1/22];
}
    
```

Approaches based on supermartingales (e.g., [1, 4, 10, 12, 13, 17]) use mappings similar to $\text{rdw}_{\mathcal{P}}$ in order to infer a real-valued term which over-approximates the expected runtime. However, in the following (non-trivial) theorem we show that our transformation is not only an over- or under-approximation, but the termination behavior and the expected runtime of \mathcal{P} and \mathcal{P}^{rdw} are identical.

Theorem 15 (Transformation Preserves Termination & Expected Runtime). Let \mathcal{P} be a CP program as in Definition 2. Then the termination times $T^{\mathcal{P}}$ and $T^{\mathcal{P}^{\text{rdw}}}$ are identically distributed w.r.t. $\text{rdw}_{\mathcal{P}}$, i.e., for all $\mathbf{x}_0 \in \mathbb{Z}^r$

with $x_0 = rdw_{\mathcal{P}}(\mathbf{x}_0)$ and all $j \in \bar{\mathbb{N}}$ we have $\mathbb{P}_{\mathbf{x}_0}^{\mathcal{P}}(T^{\mathcal{P}}=j) = \mathbb{P}_{\mathbf{x}_0}^{\mathcal{P}^{rdw}}(T^{\mathcal{P}^{rdw}}=j)$. So in particular, $\mathbb{P}_{\mathbf{x}_0}^{\mathcal{P}}(T^{\mathcal{P}}<\infty) = \mathbb{P}_{\mathbf{x}_0}^{\mathcal{P}^{rdw}}(T^{\mathcal{P}^{rdw}}<\infty)$ and $rt_{\mathbf{x}_0}^{\mathcal{P}} = \mathbb{E}_{\mathbf{x}_0}^{\mathcal{P}}(T^{\mathcal{P}}) = \mathbb{E}_{\mathbf{x}_0}^{\mathcal{P}^{rdw}}(T^{\mathcal{P}^{rdw}}) = rt_{\mathbf{x}_0}^{\mathcal{P}^{rdw}}$. Thus, the expected runtimes of \mathcal{P} on the input \mathbf{x}_0 and of \mathcal{P}^{rdw} on x_0 coincide.

The following definition identifies pathological programs that can be disregarded.

Definition 16 (Trivial Program). Let \mathcal{P} be a CP program as in Definition 2. We call \mathcal{P} trivial if $\mathbf{a} = \mathbf{0} = (0, 0, \dots, 0)$ or if \mathcal{P}^{rdw} is the program on the right.

```

while (x > 0) {
    x = x    [1];
}
    
```

Note that a random walk program \mathcal{P} is trivial iff it has the form `while(x > 0){x = x [1];}`, since $\mathcal{P} = \mathcal{P}^{rdw}$ holds for random walk programs \mathcal{P} . From now on, we will exclude trivial programs \mathcal{P} as their termination behavior is obvious: for inputs \mathbf{x}_0 that satisfy the loop condition $\mathbf{a} \bullet \mathbf{x}_0 > b$, the program never terminates (i.e., $rt_{\mathbf{x}_0}^{\mathcal{P}} = \infty$) and for inputs \mathbf{x}_0 with $\mathbf{a} \bullet \mathbf{x}_0 \leq b$ we have $rt_{\mathbf{x}_0}^{\mathcal{P}} = 0$. Note that if $\mathbf{a} = \mathbf{0}$, then the termination behavior just depends on b : if $b < 0$, then $rt_{\mathbf{x}_0}^{\mathcal{P}} = \infty$ for all \mathbf{x}_0 and if $b \geq 0$, then $rt_{\mathbf{x}_0}^{\mathcal{P}} = 0$ for all \mathbf{x}_0 .

4.2 Deciding Termination

We now present a simple decision procedure for (P)AST of random walk programs \mathcal{P} . By the results of Sect. 4.1, this also yields a decision procedure for arbitrary CP programs. If $p' > 0$, then Theorem 10 already shows that \mathcal{P} is PAST and its expected runtime is bounded by the constant $\frac{1}{p'}$. Thus, in the rest of Sect. 4 we regard *random walk programs without direct termination*, i.e., $p' = 0$.

Definition 17 introduces the *drift* of a random walk program, i.e., the expected value of the change of the program variable in one loop iteration, cf. [4].

Definition 17 (Drift). Let \mathcal{P} be a random walk program \mathcal{P} as in Definition 12. Then its drift is $\mu_{\mathcal{P}} = \sum_{-k \leq j \leq m} j \cdot p_j$.

Theorem 18 shows that to decide (P)AST, one just has to compute the drift.

Theorem 18 (Decision Procedure for (P)AST of Random Walk Programs). Let \mathcal{P} be a non-trivial random walk program without direct termination.

- If $\mu_{\mathcal{P}} > 0$, then the program is not AST.
- If $\mu_{\mathcal{P}} = 0$, then the program is AST but not PAST.
- If $\mu_{\mathcal{P}} < 0$, then the program is PAST.

Example 19 (\mathcal{P}_{race} is PAST). The drift of \mathcal{P}_{race}^{rdw} in Example 14 is $\mu_{\mathcal{P}_{race}^{rdw}} = 1 \cdot \frac{6}{11} + \frac{1}{22} \cdot \sum_{-9 \leq j \leq 0} j = -\frac{3}{2} < 0$. So on average the distance x between the tortoise and the hare decreases in each loop iteration. Hence by Theorem 18, \mathcal{P}_{race}^{rdw} is PAST and the following Corollary 20 implies that \mathcal{P}_{race} is PAST as well.

Corollary 20 (Decision Procedure for (P)AST of CP programs). *For a non-trivial CP program \mathcal{P} , \mathcal{P} is (P)AST iff \mathcal{P}^{rdw} is (P)AST. Hence, Theorems 15 and 18 yield a decision procedure for AST and PAST of CP programs.*

In [20], we show that Theorem 18 follows from classical results on random walks [33]. Alternatively, Theorem 18 could also be proved by combining several recent results on probabilistic programs: The approach of [28] could be used to show that $\mu_{\mathcal{P}} = 0$ implies AST. Moreover, one could prove that $\mu_{\mathcal{P}} < 0$ implies PAST by showing that x is a *ranking supermartingale* of the program [4, 10, 13, 17]. That the program is not PAST if $\mu_{\mathcal{P}} \geq 0$ and not AST if $\mu_{\mathcal{P}} > 0$ could be proved by showing that $-x$ is a $\mu_{\mathcal{P}}$ -*repulsing supermartingale* [12].

While the proof of Theorem 18 is based on known results, the formulation of Theorem 18 shows that there is an extremely *simple* decision procedure for (P)AST of CP programs, i.e., checking the sign of the drift is much simpler than applying existing (general) techniques for termination analysis of probabilistic programs.

4.3 Computing Asymptotic Expected Runtimes

It turns out that for random walk programs (and thus by Theorem 15, also for CP programs), one can not only decide termination, but one can also infer tight bounds on the expected runtime. Theorem 21 shows that the computation of the bounds is again very *simple*.

Theorem 21 (Bounds on the Expected Runtime of CP Programs).

Let \mathcal{P} be a non-trivial CP program as in Definition 2 without direct termination which is PAST (i.e., $\mu_{\mathcal{P}^{rdw}} < 0$). Moreover, let $k_{\mathcal{P}}$ be obtained according to the transformation from Definition 13. If $rdw_{\mathcal{P}}(\mathbf{x}_0) \leq 0$, then $rt_{\mathbf{x}_0}^{\mathcal{P}} = 0$. If $rdw_{\mathcal{P}}(\mathbf{x}_0) > 0$, then \mathcal{P} 's expected runtime is asymptotically linear and we have

$$-\frac{1}{\mu_{\mathcal{P}^{rdw}}} \cdot rdw_{\mathcal{P}}(\mathbf{x}_0) \leq rt_{\mathbf{x}_0}^{\mathcal{P}} \leq -\frac{1}{\mu_{\mathcal{P}^{rdw}}} \cdot rdw_{\mathcal{P}}(\mathbf{x}_0) + \frac{1-k_{\mathcal{P}}}{\mu_{\mathcal{P}^{rdw}}}.$$

Example 22 (Bounds on the Runtime of \mathcal{P}_{race}). In Example 19 we saw that the program \mathcal{P}_{race}^{rdw} from Example 14 is PAST as it has the drift $\mu_{\mathcal{P}_{race}^{rdw}} = -\frac{3}{2} < 0$. Note that here $k = 9$. Hence by Theorem 21 we get that whenever $rdw_{\mathcal{P}_{race}}(t, h) = t - h + 1$ is positive, the expected runtime $rt_{(t,h)}^{\mathcal{P}_{race}}$ is between $-\frac{1}{\mu_{\mathcal{P}_{race}^{rdw}}} \cdot rdw_{\mathcal{P}_{race}}(t, h) = \frac{2}{3} \cdot (t - h + 1)$ and $-\frac{1}{\mu_{\mathcal{P}_{race}^{rdw}}} \cdot rdw_{\mathcal{P}_{race}}(t, h) + \frac{1-k}{\mu_{\mathcal{P}_{race}^{rdw}}} = \frac{2}{3} \cdot (t - h + 1) + \frac{16}{3}$. The same upper bound $\frac{2}{3} \cdot (t - h + 1) + \frac{16}{3}$ was inferred in [30] by an incomplete technique based on several inference rules and linear programming solvers. In contrast, Theorem 21 allows us to read off such bounds directly from the program.

Our proof of Theorem 21 in [20] again uses the connection to random walks and shows that the classical Lemma of Wald [21, Lemma 10.2(9)] directly yields both the upper and the lower bound for the expected runtime. Alternatively, the upper bound in Theorem 21 could also be proved by considering that $rdw_{\mathcal{P}}(\mathbf{x}_0) +$

$(1 - k_{\mathcal{P}})$ is a *ranking supermartingale* [1, 4, 10, 13, 17] whose expected decrease in each loop iteration is $\mu_{\mathcal{P}}$. The lower bound could also be inferred by considering the *difference-bounded submartingale* $-rdw_{\mathcal{P}}(\mathbf{x}_0)$ [7, 19].

5 Computing Exact Expected Runtimes

While Theorems 10 and 21 state how to deduce the *asymptotic* expected runtime, we now show that based on these results one can compute the runtime of CP programs *exactly*. In general, whenever it is possible, then inferring the exact runtimes of programs is preferable to asymptotic runtimes which ignore the “coefficients” of the runtime.

Again, we first consider *random walk* programs and generalize our technique to CP programs using Theorem 15 afterwards. Throughout Sect. 5, for any random walk program \mathcal{P} as in Definition 12, we require that \mathcal{P} is PAST, i.e., that $p' > 0$ (cf. Theorem 10) or that the drift $\mu_{\mathcal{P}}$ is negative if $p' = 0$ (cf. Theorem 18). Note that whenever $k = 0$ and \mathcal{P} is PAST, then $p' > 0$.¹

To compute \mathcal{P} 's expected runtime exactly, we use its characterization as the least fixpoint of the expected runtime transformer $\mathcal{L}^{\mathcal{P}}$ (cf. Theorem 9), i.e., $rt_x^{\mathcal{P}}$ is the smallest function $f : \mathbb{Z} \rightarrow \overline{\mathbb{R}}_{\geq 0}$ satisfying the constraint

$$f(x) = \sum_{-k \leq j \leq m} p_j \cdot f(x + j) + p' \cdot f(d) + 1 \quad \text{for all } x > 0, \quad (3)$$

cf. (1). Since \mathcal{P} is PAST, f never returns ∞ , i.e., $f : \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$. Note that the smallest function $f : \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$ that satisfies (3) also satisfies

$$f(x) = 0 \quad \text{for all } x \leq 0. \quad (4)$$

Therefore, as $d \leq 0$, the constraint (3) can be simplified to

$$f(x) = \sum_{-k \leq j \leq m} p_j \cdot f(x + j) + 1 \quad \text{for all } x > 0. \quad (5)$$

In Sect. 5.1 we recapitulate how to compute all solutions of such inhomogeneous recurrence equations (cf., e.g., [14, Ch. 2]). However, to compute $rt_x^{\mathcal{P}}$, the challenge is to find the *smallest* solution $f : \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$ of the recurrence equation (5). Therefore, in Sect. 5.2 we will exploit the knowledge gained in Theorems 10 and 21 to show that there is only a *single* function f that satisfies both (4) and (5) and is bounded by a constant (if $p' > 0$, cf. Theorem 10) resp. by a linear function (if $p' = 0$, cf. Theorem 21). This observation then allows us to compute $rt_x^{\mathcal{P}}$ exactly. So the crucial prerequisites for this result are Theorem 9 (which characterizes the expected runtime as the smallest solution of the recurrence equation (5)), Theorem 18 (which allows the restriction to negative drift if $p' = 0$), and in particular Theorems 10 and 21 (since Sect. 5.2 will show that the results of Theorems 10 and 21 on the asymptotic runtime can be translated into suitable conditions on the solutions of (5)).

¹ If $p' = 0$ and $k = 0$ then $\mu_{\mathcal{P}} \geq 0$.

5.1 Finding All Solutions of the Recurrence Equation

Example 23 (Modification of $\mathcal{P}_{\text{race}}^{\text{rdw}}$). To illustrate our approach, we use a modified version of $\mathcal{P}_{\text{race}}^{\text{rdw}}$ from Example 14 to ease readability. In Sect. 6, we will consider the original program $\mathcal{P}_{\text{race}}^{\text{rdw}}$ resp. $\mathcal{P}_{\text{race}}$ from Example 14 resp. Example 1 again and show its exact expected runtime inferred by the implementation of our approach. In the modified program $\mathcal{P}_{\text{race}}^{\text{mod}}$ on the right, the distance between the tortoise and the hare still increases with probability $\frac{6}{11}$, but the probability of decreasing by more than two is distributed to the cases where it stays the same and where it decreases by two. We have $p' = 0$ and the drift is $\mu_{\mathcal{P}_{\text{race}}^{\text{mod}}} = 1 \cdot \frac{6}{11} + 0 \cdot \frac{1}{11} - 1 \cdot \frac{1}{22} - 2 \cdot \frac{7}{22} = -\frac{3}{22} < 0$. So by Theorem 18, $\mathcal{P}_{\text{race}}^{\text{mod}}$ is PAST. By Theorem 9, $\text{rt}_x^{\mathcal{P}_{\text{race}}^{\text{mod}}}$ is the smallest function $f : \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$ satisfying

```

while (x > 0) {
  x = x + 1  [  $\frac{6}{11}$  ];
  x = x      [  $\frac{1}{11}$  ];
  x = x - 1  [  $\frac{1}{22}$  ];
  x = x - 2  [  $\frac{7}{22}$  ];
}
    
```

$$f(x) = \frac{6}{11} \cdot f(x+1) + \frac{1}{11} \cdot f(x) + \frac{1}{22} \cdot f(x-1) + \frac{7}{22} \cdot f(x-2) + 1 \text{ for all } x > 0. \quad (6)$$

Instead of searching for the *smallest* $f : \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$ satisfying (5), we first calculate the set of *all* functions $f : \mathbb{Z} \rightarrow \mathbb{C}$ that satisfy (5), i.e., we also consider functions returning negative or complex numbers. Clearly, (5) is equivalent to

$$0 = p_m \cdot f(x+m) + \dots + p_1 \cdot f(x+1) + (p_0 - 1) \cdot f(x) + p_{-1} \cdot f(x-1) + \dots + p_{-k} \cdot f(x-k) + 1 \text{ for all } x > 0. \quad (7)$$

The set of solutions on $\mathbb{Z} \rightarrow \mathbb{C}$ of this linear, inhomogeneous recurrence equation is an affine space which can be written as an arbitrary particular solution of the inhomogeneous equation plus any linear combination of $k + m$ linearly independent solutions of the corresponding homogeneous recurrence equation.

We start with computing a solution to the inhomogeneous equation (7). To this end, we use the bounds for $\text{rt}_x^{\mathcal{P}}$ from Theorems 10 and 21 (where we take the upper bound $\frac{1}{p'}$ if $p' > 0$ and the lower bound $-\frac{1}{\mu_{\mathcal{P}}} \cdot x$ if $p' = 0$). So we define

$$C_{\text{const}} = \frac{1}{p'}, \text{ if } p' > 0 \quad \text{and} \quad C_{\text{lin}} = -\frac{1}{\mu_{\mathcal{P}}}, \text{ if } p' = 0.$$

One easily shows that if $p' > 0$, then $f(x) = C_{\text{const}}$ is a solution of the inhomogeneous recurrence equation (7) and if $p' = 0$, then $f(x) = C_{\text{lin}} \cdot x$ solves (7).

Example 24 (Example 23 cont.). In the program $\mathcal{P}_{\text{race}}^{\text{mod}}$ of Example 23, we have $p' = 0$ and $\mu_{\mathcal{P}_{\text{race}}^{\text{mod}}} = -\frac{3}{22}$. Hence $C_{\text{lin}} = \frac{22}{3}$ and $C_{\text{lin}} \cdot x$ is a solution of (6).

After having determined one particular solution of the inhomogeneous recurrence equation (7), now we compute the solutions of the *homogeneous* recurrence equation which results from (7) by replacing the add-on “+ 1” with 0. To this end, we consider the corresponding *characteristic polynomial* $\chi_{\mathcal{P}}$:²

$$\chi_{\mathcal{P}}(\lambda) = p_m \cdot \lambda^{k+m} + \dots + p_1 \cdot \lambda^{k+1} + (p_0 - 1) \cdot \lambda^k + p_{-1} \cdot \lambda^{k-1} + \dots + p_{-k} \quad (8)$$

² If $m = 0$ then $\chi_{\mathcal{P}}(\lambda) = (p_0 - 1) \cdot \lambda^k + p_{-1} \cdot \lambda^{k-1} + \dots + p_{-k}$, and if $k = 0$ then $\chi_{\mathcal{P}}(\lambda) = p_m \cdot \lambda^m + \dots + p_1 \cdot \lambda + (p_0 - 1)$. Note that $p_0 \neq 1$ since \mathcal{P} is PAST and in Definition 12 we required that $m > 0$ implies $p_m > 0$ and $k > 0$ implies $p_{-k} > 0$. Hence, the characteristic polynomial has exactly the degree $k + m$, even if $m = 0$ or $k = 0$.

Let $\lambda_1, \dots, \lambda_c$ denote the pairwise different (possibly complex) roots of the characteristic polynomial $\chi_{\mathcal{P}}$. For all $1 \leq j \leq c$, let $v_j \in \mathbb{N} \setminus \{0\}$ be the multiplicity of the root λ_j . Thus, we have $v_1 + \dots + v_c = k + m$.

Then we obtain the following $k + m$ linearly independent solutions of the homogeneous recurrence equation resulting from (7):

$$\lambda_j^x \cdot x^u \quad \text{for all } 1 \leq j \leq c \text{ and all } 0 \leq u \leq v_j - 1$$

So $f: \mathbb{Z} \rightarrow \mathbb{C}$ is a solution of (5) (resp. (7)) iff there exist coefficients $a_{j,u} \in \mathbb{C}$ with

$$f(x) = C(x) + \sum_{1 \leq j \leq c} \sum_{0 \leq u \leq v_j - 1} a_{j,u} \cdot \lambda_j^x \cdot x^u \quad \text{for all } x > -k, \quad (9)$$

where $C(x) = C_{const} = \frac{1}{p'}$ if $p' > 0$ and $C(x) = C_{lin} \cdot x = -\frac{1}{\mu_{\mathcal{P}}} \cdot x$ if $p' = 0$. The reason for requiring (9) for all $x > -k$ is that $-k + 1$ is the smallest argument where f 's value is taken into account in (5).

Example 25 (Example 24 cont.). The characteristic polynomial for the program \mathcal{P}_{race}^{mod} of Example 23 has the degree $k + m = 2 + 1 = 3$ and is given by

$$\chi_{\mathcal{P}_{race}^{mod}}(\lambda) = \frac{6}{11} \cdot \lambda^3 - \frac{10}{11} \cdot \lambda^2 + \frac{1}{22} \cdot \lambda + \frac{7}{22}.$$

Its roots are $\lambda_1 = 1$, $\lambda_2 = -\frac{1}{2}$, and $\lambda_3 = \frac{7}{6}$. So here, all roots are real numbers and they all have the multiplicity 1. Hence, three linearly independent solutions of the homogeneous part of (6) are the functions $1^x = 1$, $(-\frac{1}{2})^x$, and $(\frac{7}{6})^x$. Therefore, a function $f: \mathbb{Z} \rightarrow \mathbb{C}$ satisfies (6) iff there are $a_1, a_2, a_3 \in \mathbb{C}$ such that

$$\begin{aligned} f(x) &= C_{lin} \cdot x + a_1 \cdot 1^x + a_2 \cdot (-\frac{1}{2})^x + a_3 \cdot (\frac{7}{6})^x \\ &= \frac{22}{3} \cdot x + a_1 + a_2 \cdot (-\frac{1}{2})^x + a_3 \cdot (\frac{7}{6})^x \quad \text{for } x > -2. \end{aligned} \quad (10)$$

5.2 Finding the Smallest Solution of the Recurrence Equation

In Sect. 5.1, we recapitulated the standard approach for solving inhomogeneous recurrence equations which shows that any function $f: \mathbb{Z} \rightarrow \mathbb{C}$ that satisfies the constraint (5) is of the form (9). Now we will present a novel technique to compute $rt_x^{\mathcal{P}}$, i.e., the *smallest non-negative* solution $f: \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$ of (5). By Theorems 10 and 21, this function f is bounded by a constant (if $p' > 0$) resp. linear (if $p' = 0$). So, when representing f in the form (9), we must have $a_{j,u} = 0$ whenever $|\lambda_j| > 1$. The following lemma shows how many roots with absolute value less or equal to 1 there are (i.e., these are the only roots that we have to consider). It is proved using Rouché's Theorem which allows us to infer the number of roots whose absolute value is below a certain bound. Note that 1 is a root of the characteristic polynomial iff $p' = 0$, since $\sum_{-k \leq j \leq m} p_j = 1 - p'$.

Lemma 26 (Number of Roots With Absolute Value ≤ 1). *Let \mathcal{P} be a random walk program as in Definition 12 that is PAST. Then the characteristic polynomial $\chi_{\mathcal{P}}$ has k roots $\lambda \in \mathbb{C}$ (counted with multiplicity) with $|\lambda| \leq 1$.*

Example 27 (Example 25 cont.). In $\mathcal{P}_{\text{race}}^{\text{mod}}$ of Example 23 we have $k = 2$. So by Lemma 26, $\chi_{\mathcal{P}}$ has exactly two roots with absolute value ≤ 1 . Indeed, the roots of $\chi_{\mathcal{P}}$ are $\lambda_1 = 1$, $\lambda_2 = -\frac{1}{2}$, and $\lambda_3 = \frac{7}{6}$, cf. Example 25. So $|\lambda_3| > 1$, but $|\lambda_1| \leq 1$ and $|\lambda_2| \leq 1$.

Based on Lemma 26, the following lemma shows that when imposing the restriction that $a_{j,u} = 0$ whenever $|\lambda_j| > 1$, then there is only a *single* function of the form (9) that also satisfies the constraint (4). Hence, this must be the function that we are searching for, because the desired smallest solution $f : \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$ of (5) also satisfies (4).

Lemma 28 (Unique Solution of (4) and (5) when Disregarding Roots With Absolute Value > 1). *Let \mathcal{P} be a random walk program as in Definition 12 that is PAST. Then there is exactly one function $f : \mathbb{Z} \rightarrow \mathbb{C}$ which satisfies both (4) and (5) (thus, it has the form (9)) and has $a_{j,u} = 0$ whenever $|\lambda_j| > 1$.*

The main theorem of Sect. 5 now shows how to compute the expected runtime exactly. By Theorems 10 and 21 on the bounds for the expected runtime and by Lemma 28, we no longer have to search for the *smallest* function that satisfies (4) and (5), but we just search for *any* solution of (4) and (5) which has $a_{j,u} = 0$ whenever $|\lambda_j| > 1$ (because there is just a single such solution). So one only has to determine the values of the remaining k coefficients $a_{j,u}$ for $|\lambda_j| \leq 1$, which can be done by exploiting that $f(x)$ has to satisfy both (4) for all $x \leq 0$ and it has to be of the form (9) for all $x > -k$. In other words, the function in (9) must be 0 for $-k + 1 \leq x \leq 0$.

Theorem 29 (Exact Expected Runtime for Random Walk Programs).

Let \mathcal{P} be a random walk program as in Definition 12 that is PAST and let $\lambda_1, \dots, \lambda_c$ be the roots of its characteristic polynomial with multiplicities v_1, \dots, v_c . Moreover, let $C(x) = C_{\text{const}} = \frac{1}{p'}$ if $p' > 0$ and $C(x) = C_{\text{lin}} \cdot x = -\frac{1}{\mu_{\mathcal{P}}} \cdot x$ if $p' = 0$. Then the expected runtime of \mathcal{P} is $rt_x^{\mathcal{P}} = 0$ for $x \leq 0$ and

$$rt_x^{\mathcal{P}} = C(x) + \sum_{1 \leq j \leq c, |\lambda_j| \leq 1} \sum_{0 \leq u \leq v_j - 1} a_{j,u} \cdot \lambda_j^x \cdot x^u \quad \text{for } x > 0,$$

where the coefficients $a_{j,u}$ are the unique solution of the k linear equations:

$$0 = C(x) + \sum_{1 \leq j \leq c, |\lambda_j| \leq 1} \sum_{0 \leq u \leq v_j - 1} a_{j,u} \cdot \lambda_j^x \cdot x^u \quad \text{for } -k + 1 \leq x \leq 0 \quad (11)$$

So in the special case where $k = 0$, we have $rt_x^{\mathcal{P}} = C(x) = C_{\text{const}} = \frac{1}{p'}$ for $x > 0$.

Thus for $x > 0$, the expected runtime $rt_x^{\mathcal{P}}$ can be computed by summing up the bound $C(x)$ and an add-on $\sum_{1 \leq j \leq c, |\lambda_j| \leq 1} \sum_{0 \leq u \leq v_j - 1} \dots$. Since $C(x)$ is an upper bound for $rt_x^{\mathcal{P}}$ if $p' > 0$ and a lower bound for $rt_x^{\mathcal{P}}$ if $p' = 0$, this add-on is non-positive if $p' > 0$ and non-negative if $p' = 0$.

Example 30 (Example 27 cont.). By Theorem 29, the expected runtime of the program $\mathcal{P}_{\text{race}}^{\text{mod}}$ from Example 23 is $rt_x^{\mathcal{P}_{\text{race}}^{\text{mod}}} = 0$ for $x \leq 0$ and

$$rt_x^{\mathcal{P}_{\text{race}}^{\text{mod}}} = \frac{22}{3} \cdot x + a_1 + a_2 \cdot \left(-\frac{1}{2}\right)^x \quad \text{for } x > 0, \text{ cf. (10).}$$

The coefficients a_1 and a_2 are the unique solution of the $k = 2$ linear equations

$$\begin{aligned} 0 &= \frac{22}{3} \cdot 0 + a_1 + a_2 \cdot \left(-\frac{1}{2}\right)^0 = a_1 + a_2 \\ 0 &= \frac{22}{3} \cdot (-1) + a_1 + a_2 \cdot \left(-\frac{1}{2}\right)^{-1} = -\frac{22}{3} + a_1 - 2 \cdot a_2 \end{aligned}$$

So $a_1 = \frac{22}{9}$, $a_2 = -\frac{22}{9}$, and hence $rt_x^{\mathcal{P}_{\text{race}}^{\text{mod}}} = \frac{22}{3} \cdot x + \frac{22}{9} - \frac{22}{9} \cdot \left(-\frac{1}{2}\right)^x$ for $x > 0$.

By Theorem 15, we can lift Theorem 29 to arbitrary CP programs \mathcal{P} immediately.

Corollary 31 (Exact Expected Runtime for CP Programs). *For any CP program, its expected runtime can be computed exactly.*

Note that irrespective of the degree of the characteristic polynomial, its roots can always be approximated numerically with any chosen precision. Thus, “exact computation” of the expected runtime in the corollary above means that a closed form for $rt_x^{\mathcal{P}}$ can also be computed with any desired precision.

Example 32 (Exact Expected Runtime of $\mathcal{P}_{\text{direct}}$). Reconsider the program $\mathcal{P}_{\text{direct}}$ of Example 11 with the probability $p' = \frac{1}{10}$ for direct termination. $\mathcal{P}_{\text{direct}}$ is PAST and its expected runtime is at most $\frac{1}{p'} = 10$, cf. Example 11.

```

while (x > 0) {
  x = x + 1  [  $\frac{9}{10}$  ];
  x = 0     [  $\frac{1}{10}$  ];
}
```

The random walk program $\mathcal{P}_{\text{direct}}^{\text{rdw}}$ on the right is obtained by the transformation of Definition 13. As $k = 0$, by Theorem 29 we obtain $rt_x^{\mathcal{P}_{\text{direct}}^{\text{rdw}}} = \frac{1}{p'} = 10$ for $x > 0$. By Theorem 15, this implies $rt_{(t,h)}^{\mathcal{P}_{\text{direct}}^{\text{rdw}}} = rt_{\text{rdw}\mathcal{P}_{\text{direct}}^{\text{rdw}}(t,h)}^{\mathcal{P}_{\text{direct}}^{\text{rdw}}} = 10$ if $\text{rdw}_{\mathcal{P}_{\text{direct}}^{\text{rdw}}}(t,h) = t - h + 1 > 0$, i.e., 10 is indeed the exact expected runtime of $\mathcal{P}_{\text{direct}}$.

Note that Theorem 29 and Corollary 31 imply that for any $\mathbf{x}_0 \in \mathbb{Z}^r$, the expected runtime $rt_{\mathbf{x}_0}^{\mathcal{P}}$ of a CP program \mathcal{P} that is PAST and has only rational probabilities $p_{c_1}, \dots, p_{c_n}, p' \in \mathbb{Q}$ is always an algebraic number. Thus, one could also compute a closed form for the exact expected runtime $rt_{\mathbf{x}}^{\mathcal{P}}$ using a representation with algebraic numbers instead of numerical approximations.

Nevertheless, Theorem 29 may yield a representation of $rt_{\mathbf{x}}^{\mathcal{P}}$ which contains complex numbers $a_{j,u}$ and λ_j , although $rt_{\mathbf{x}}^{\mathcal{P}}$ is always real. However, one can easily obtain a more intuitive representation of $rt_{\mathbf{x}}^{\mathcal{P}}$ without complex numbers:

Since the characteristic polynomial $\chi_{\mathcal{P}}$ only has real coefficients, whenever $\chi_{\mathcal{P}}$ has a complex root λ of multiplicity v , its conjugate $\bar{\lambda}$ is also a root of $\chi_{\mathcal{P}}$ with the same multiplicity v . So the pairwise different roots $\lambda_1, \dots, \lambda_c$ can

be distinguished into pairwise different real roots $\lambda_1, \dots, \lambda_s$, and into pairwise different non-real complex roots $\lambda_{s+1}, \overline{\lambda_{s+1}}, \dots, \lambda_{s+t}, \overline{\lambda_{s+t}}$, where $c = s + 2 \cdot t$.

For any coefficients $a_{j,u}, a'_{j,u} \in \mathbb{C}$ with $j \in \{s+1, \dots, s+t\}$ and $u \in \{0, \dots, v_j - 1\}$ let $b_{j,u} = 2 \cdot \operatorname{Re}(a_{j,u}) \in \mathbb{R}$ and $b'_{j,u} = -2 \cdot \operatorname{Im}(a_{j,u}) \in \mathbb{R}$. Then $a_{j,u} \cdot \lambda_j^x + a'_{j,u} \cdot \overline{\lambda_j^x} = b_{j,u} \cdot \operatorname{Re}(\lambda_j^x) + b'_{j,u} \cdot \operatorname{Im}(\lambda_j^x)$. Hence, by Theorem 29 we get the following representation of the expected runtime which only uses *real* numbers:

$$rt_x^{\mathcal{P}} = \begin{cases} C(x) + \sum_{1 \leq j \leq s, |\lambda_j| \leq 1} \sum_{0 \leq u \leq v_j - 1} a_{j,u} \cdot \lambda_j^x \cdot x^u \\ \quad + \sum_{s+1 \leq j \leq s+t, |\lambda_j| \leq 1} \sum_{0 \leq u \leq v_j - 1} (b_{j,u} \cdot \operatorname{Re}(\lambda_j^x) + b'_{j,u} \cdot \operatorname{Im}(\lambda_j^x)) \cdot x^u, & \text{for } x > 0 \\ 0, & \text{for } x \leq 0 \end{cases} \quad (12)$$

To compute $\operatorname{Re}(\lambda_j^x)$ and $\operatorname{Im}(\lambda_j^x)$, take the polar representation of the non-real roots $\lambda_j = w_j \cdot e^{\theta_j \cdot i}$. Then $\operatorname{Re}(\lambda_j^x) = w_j^x \cdot \cos(\theta_j \cdot x)$ and $\operatorname{Im}(\lambda_j^x) = w_j^x \cdot \sin(\theta_j \cdot x)$.

Therefore, we obtain the following algorithm to deduce the exact expected runtime automatically.

Algorithm 33 (Computing the Exact Expected Runtime). *To infer the runtime of a CP program \mathcal{P} as in Definition 12 that is PAST, we proceed as follows:*

1. Transform \mathcal{P} into \mathcal{P}^{rdw} by the transformation of Definition 13. Thus, \mathcal{P}^{rdw} is a random walk program as in Definition 12.
2. Compute the solution $C(x) = C_{\text{const}} = \frac{1}{p'}$ resp. $C(x) = C_{\text{lin}} \cdot x = -\frac{1}{\mu_{\mathcal{P}^{\text{rdw}}}} \cdot x$ of the inhomogeneous recurrence equation (7).
3. Compute the $k + m$ (possibly complex) roots of the characteristic polynomial $\chi_{\mathcal{P}^{\text{rdw}}}$ (cf. (8)) and keep the k roots λ with $|\lambda| \leq 1$.
4. Determine the coefficients $a_{j,u}$ by solving the k linear equations in (11).
5. Return the solution (12) where $b_{j,u} = 2 \cdot \operatorname{Re}(a_{j,u})$, $b'_{j,u} = -2 \cdot \operatorname{Im}(a_{j,u})$, and for $\lambda_j = w_j \cdot e^{\theta_j \cdot i}$ we have $\operatorname{Re}(\lambda_j^x) = w_j^x \cdot \cos(\theta_j \cdot x)$ and $\operatorname{Im}(\lambda_j^x) = w_j^x \cdot \sin(\theta_j \cdot x)$. Moreover, x must be replaced by $\text{rdw}_{\mathcal{P}}(x)$.

6 Conclusion, Implementation, and Related Work

We presented decision procedures for termination and complexity of classes of probabilistic programs. They are based on the connection between the expected runtime of a program and the smallest solution of its corresponding recurrence equation, cf. Sect. 2. For our notion of probabilistic programs, if the probability for leaving the loop directly is at least p' for some $p' > 0$, then the program is always PAST and its expected runtime is asymptotically constant, cf. Sect. 3. In Sect. 4 we showed that a very simple decision procedure for AST and PAST of CP programs can be obtained by classical results from random walk theory and that the expected runtime is asymptotically linear if the program is PAST. Based on these results, in Sect. 5 we presented our algorithm to automatically infer a closed form for the *exact* expected runtime of CP programs (i.e., with arbitrarily high precision). All proofs and a collection of examples to demonstrate our algorithm can be found in [20].

Implementation. We implemented Algorithm 33 in our tool KoAT [9], which was already one of the leading tools for complexity analysis of (non-probabilistic) integer programs. The implementation is written in OCaml and uses the Python libraries MpMath [22] and SymPy [29] for solving linear equations and for finding the roots of the characteristic polynomial. In addition to the closed form for the exact expected runtime, our implementation can also compute the concrete number of expected loop iterations if the user specifies the initial values of the variables. For further details, a set of benchmarks, and to download our implementation, we refer to <https://aprove-developers.github.io/recurrence/>.

Example 34 (Computing the Exact Expected Runtime of $\mathcal{P}_{\text{race}}$ Automatically). For the tortoise and hare program $\mathcal{P}_{\text{race}}$ from Example 1, our implementation in KoAT computes the following expected runtime within 0.49 s on an Intel Core i7-6500 with 8 GB memory (when selecting a precision of 2 decimal places):

$$\begin{aligned} rt_{(t,h)}^{\mathcal{P}_{\text{race}}} = & 0.049 \cdot 0.65^{(t-h+1)} \cdot \sin(2.8 \cdot (t-h+1)) - 0.35 \cdot 0.65^{(t-h+1)} \cdot \cos(2.8 \cdot (t-h+1)) \\ & + 0.15 \cdot 0.66^{(t-h+1)} \cdot \sin(2.2 \cdot (t-h+1)) - 0.35 \cdot 0.66^{(t-h+1)} \cdot \cos(2.2 \cdot (t-h+1)) \\ & + 0.3 \cdot 0.7^{(t-h+1)} \cdot \sin(1.5 \cdot (t-h+1)) - 0.39 \cdot 0.7^{(t-h+1)} \cdot \cos(1.5 \cdot (t-h+1)) \\ & + 0.62 \cdot 0.75^{(t-h+1)} \cdot \sin(0.83 \cdot (t-h+1)) - 0.49 \cdot 0.75^{(t-h+1)} \cdot \cos(0.83 \cdot (t-h+1)) \\ & + \frac{2}{3} \cdot (t-h) + 2.3 \end{aligned}$$

So when starting in a state with $t = 1000$ and $h = 0$, according to our implementation the number of expected loop iterations is $rt_{(1000,0)}^{\mathcal{P}_{\text{race}}} = 670$.

Related Work. Many techniques to analyze (P)AST have been developed, which mostly rely on ranking supermartingales, e.g., [1, 4, 10, 12, 13, 17, 19, 28, 30]. Indeed, several of these works (e.g., [1, 4, 17, 19]) present complete criteria for (P)AST, although (P)AST is undecidable. However, the corresponding automation of these techniques is of course incomplete. In [13] it is shown that for affine probabilistic programs, a superclass of our CP programs, the existence of a linear ranking supermartingale is decidable. However, the existence of a linear ranking supermartingale is sufficient but not necessary for PAST or an at most linear expected runtime.

Classes of programs where termination is decidable have already been studied for deterministic programs. In [35] it was shown that for a class of linear loop programs over the reals, the halting problem is decidable. This result was transferred to the rationals [5] and under certain conditions to integer programs [5, 18, 31]. Termination analysis for probabilistic programs is substantially harder than for non-probabilistic ones [23]. Nevertheless, there is some previous work on classes of probabilistic programs where termination is decidable and asymptotic bounds on the expected runtime are computable. For instance, in [6] it was shown that AST is decidable for certain stochastic games and [11] presents an automatic approach for inferring asymptotic upper bounds on the expected runtime by considering uni- and bivariate recurrence equations.

However, our algorithm is the first which computes a general formula (i.e., a closed form) for the *exact* expected runtime of arbitrary CP programs. To our knowledge, up to now such a formula was only known for the very restricted special case of *bounded* simple random walks (cf. [16]), i.e., programs of the

form on the right for some $1 \geq p \geq 0$ and some $b \in \mathbb{Z}$. Note that due to the *two* boundary conditions $x > 0$ and $b > x$, the resulting recurrence equation for the expected runtime of the program only has a *single* solution $f : \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$ that also satisfies $f(0) = 0$ and $f(b) = 0$. Hence, standard techniques for solving

```

while (b > x > 0) {
  x = x + 1 [p];
  x = x - 1 [1 - p];
}

```

recurrence equations suffice to compute this solution. In contrast, we developed an algorithm to compute the exact expected runtime of *unbounded arbitrary* CP programs where the loop condition only has *one* boundary condition $x > 0$, i.e., x can grow infinitely large. For that reason, here the challenge is to find an algorithm which computes the *smallest* solution $f : \mathbb{Z} \rightarrow \mathbb{R}_{\geq 0}$ of the resulting recurrence equation. We showed that this can be done using the information on the asymptotic bounds of the expected runtime from Sects. 3 and 4.

Future Work. There are several directions for future work. In Sect. 4.1 we reduced CP programs to random walk programs. In future work, we will consider more advanced reductions in order to extend the class of probabilistic programs where termination and complexity are decidable. Moreover, we want to develop techniques to automatically *over- or under-approximate* the runtime of a program \mathcal{P} by the runtimes of corresponding CP programs \mathcal{P}_1 and \mathcal{P}_2 such that $rt_x^{\mathcal{P}_1} \leq rt_x^{\mathcal{P}} \leq rt_x^{\mathcal{P}_2}$ holds for all $x \in \mathbb{Z}^T$. Furthermore, we will integrate the easy inference of runtime bounds for CP programs into existing techniques for analyzing more general probabilistic programs.

Acknowledgments. We would like to thank Nicos Georgiou and Vladislav Vysotskiy for drawing our attention to Wald's Lemma and to the work of Frank Spitzer on random walks, and Benjamin Lucien Kaminski and Christoph Matheja for many helpful discussions. Furthermore, we thank Tom Küspert who helped with the implementation of our technique in our tool KoAT.

References

1. Agrawal, S., Chatterjee, K., Novotný, P.: Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. Proc. ACM Program. Lang. (POPL) **2**, 34:1–34:32 (2018). <https://doi.org/10.1145/3158122>
2. Ash, R.B., Doleans-Dade, C.A.: Probability and Measure Theory. Elsevier/Academic Press (2000)
3. Bazzi, L., Mitter, S.: The solution of linear probabilistic recurrence relations. Algorithmica **36**(1), 41–57 (2003). <https://doi.org/10.1007/s00453-002-1003-4>
4. Bournez, O., Garnier, F.: Proving positive almost-sure termination. In: Giesl, J. (ed.) RTA 2005. LNCS, vol. 3467, pp. 323–337. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32033-3_24
5. Braverman, M.: Termination of integer linear programs. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 372–385. Springer, Heidelberg (2006). https://doi.org/10.1007/11817963_34
6. Brázdil, T., Brozek, V., Etessami, K.: One-counter stochastic games. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2010, LIPIcs, vol. 8, pp. 108–119 (2010). <https://doi.org/10.4230/LIPIcs.FSTTCS.2010.108>

7. Brázdil, T., Kučera, A., Novotný, P., Wojtczak, D.: Minimizing expected termination time in one-counter Markov decision processes. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012. LNCS, vol. 7392, pp. 141–152. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31585-5_16
8. Brázdil, T., Esparza, J., Kiefer, S., Kucera, A.: Analyzing probabilistic pushdown automata. *Formal Methods Syst. Des.* **43**(2), 124–163 (2013). <https://doi.org/10.1007/s10703-012-0166-0>
9. Brockschmidt, M., Emmes, F., Falke, S., Fuhs, C., Giesl, J.: Analyzing runtime and size complexity of integer programs. *ACM Trans. Program. Lang. Syst.* **38**(4), 13:1–13:50 (2016). <https://doi.org/10.1145/2866575>
10. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 511–526. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_34
11. Chatterjee, K., Fu, H., Murhekar, A.: Automated recurrence analysis for almost-linear expected-runtime bounds. In: Majumdar, R., Kunčák, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 118–139. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_6
12. Chatterjee, K., Novotný, P., Zikelic, D.: Stochastic invariants for probabilistic termination. In: Castagna, G., Gordon, A.D. (eds.) POPL 2017, pp. 145–160 (2017). <https://doi.org/10.1145/3093333.3009873>
13. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. *ACM Trans. Program. Lang. Syst.* **40**(2), 7:1–7:45 (2018). <https://doi.org/10.1145/3174800>
14. Elaydi, S.: *An Introduction to Difference Equations*. Springer, New York (2005). <https://doi.org/10.1007/0-387-27602-5>
15. Esparza, J., Kucera, A., Mayr, R.: Quantitative analysis of probabilistic pushdown automata: expectations and variances. In: Panangaden, P. (ed.) LICS 2005, pp. 117–126 (2005). <https://doi.org/10.1109/LICS.2005.39>
16. Feller, W.: *An Introduction to Probability Theory and Its Applications, Probability and Mathematical Statistics*, vol. 1. Wiley, Hoboken (1950)
17. Fioriti, L.M.F., Hermanns, H.: Probabilistic termination: soundness, completeness, and compositionality. In: Rajamani, S.K., Walker, D. (eds.) POPL 2015, pp. 489–501 (2015). <https://doi.org/10.1145/2676726.2677001>
18. Frohn, F., Giesl, J.: Termination of triangular integer loops is decidable. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11562, pp. 426–444. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25543-5_24
19. Fu, H., Chatterjee, K.: Termination of nondeterministic probabilistic programs. In: Enea, C., Piskac, R. (eds.) VMCAI 2019. LNCS, vol. 11388, pp. 468–490. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11245-5_22
20. Giesl, J., Giesl, P., Hark, M.: Computing expected runtimes for constant probability programs. *CoRR* abs/1905.09544 (2019). <https://arxiv.org/abs/1905.09544>
21. Grimmett, G., Stirzaker, D.: *Probability and Random Processes*. Oxford University Press, Oxford (2001)
22. Johansson, F., et al.: MpMath: a Python library for arbitrary-precision floating-point arithmetic. <http://mpmath.org/>
23. Kaminski, B.L., Katoen, J.: On the hardness of almost-sure termination. In: Italiano, G.F., Pighizzini, G., Sannella, D. (eds.) MFCS 2015. LNCS, vol. 9234, pp. 307–318. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48057-1_24

24. Kaminski, B.L., Katoen, J.-P., Matheja, C., Olmedo, F.: Weakest precondition reasoning for expected run-times of probabilistic programs. In: Thiemann, P. (ed.) ESOP 2016. LNCS, vol. 9632, pp. 364–389. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49498-1_15
25. Karp, R.M.: Probabilistic recurrence relations. *J. ACM* **41**(6), 1136–1150 (1994). <https://doi.org/10.1145/195613.195632>
26. Kozen, D.: Semantics of probabilistic programs. In: Kosaraju, S.R. (ed.) FOCS 1979, pp. 101–114 (1979). <https://doi.org/10.1109/SFCS.1979.38>
27. McIver, A., Morgan, C.: *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, New York (2005). <https://doi.org/10.1007/b138392>
28. McIver, A., Morgan, C., Kaminski, B.L., Katoen, J.: A new proof rule for almost-sure termination. *Proc. ACM Program. Lang.* (POPL) **2**, 33:1–33:28 (2018). <https://doi.org/10.1145/3158121>
29. Meurer, A., et al.: SymPy: symbolic computing in Python. *Peer J Comput. Sci.* **3**, e103 (2017). <https://doi.org/10.7717/peerj-cs.103>
30. Ngo, V.C., Carbonneaux, Q., Hoffmann, J.: Bounded expectations: resource analysis for probabilistic programs. In: Foster, J.S., Grossman, D. (eds.) PLDI 2018, pp. 496–512 (2018). <https://doi.org/10.1145/3192366.3192394>. Extended Version available at <https://arxiv.org/abs/1711.08847>
31. Ouaknine, J., Pinto, J.S., Worrell, J.: On termination of integer linear loops. In: Indyk, P. (ed.) SODA 2015, pp. 957–969 (2015). <https://doi.org/10.1137/1.9781611973730.65>
32. Puterman, M.L.: *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York (1994)
33. Spitzer, F.: *Principles of Random Walk*. Springer, New York (1964). <https://doi.org/10.1007/978-1-4757-4229-9>
34. Tassarotti, J., Harper, R.: Verified tail bounds for randomized programs. In: Avigad, J., Mahboubi, A. (eds.) ITP 2018. LNCS, vol. 10895, pp. 560–578. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94821-8_33
35. Tiwari, A.: Termination of linear programs. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 70–82. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_6