On Probabilistic Termination of Functional Programs with Continuous Distributions

Anonymous Author(s)

Abstract

We study termination of higher-order probabilistic functional programs with recursion, stochastic conditioning and sampling from continuous distributions.

Reasoning about the termination probability of programs with continuous distributions is hard, because the enumeration of terminating executions cannot provide any nontrivial bounds. We present a new operational semantics based on *traces of intervals*, which is sound and complete with respect to the standard sampling-based semantics, in which (countable) enumeration can provide arbitrarily tight lower bounds. Consequently we obtain the first proof that deciding almost-sure termination (AST) for programs with continuous distributions is Π_2^0-complete. We also provide a compositional representation of our semantics in terms of an intersection type system.

In the second part, we present a method of proving AST for *non-affine programs*, i.e., recursive programs that can, during the evaluation of the recursive body, make *multiple* recursive calls (of a first-order function) from *distinct* call sites. Unlike in a deterministic language, the number of recursion call sites has direct consequences on the termination probability. Our framework supports a proof system that can verify AST for programs that are well beyond the scope of existing methods.

We have constructed prototype implementations of our method of computing lower bounds of termination probability, and AST verification.

1 Introduction

Probabilistic (or randomised) programs are programs that employ a degree of randomness in their control flow. Thanks to Michael Rabin's insight [52], it has been widely recognised that introducing randomisation in algorithm design can yield substantial improvements in time and space complexity. Many dozens of randomised algorithms can now solve a wide range of problems accurately, and much more efficiently than their deterministic counterparts [45, 47].

Recently, in probabilistic programming [28, 53, 59], probabilistic programs, augmented with stochastic conditioning constructs, have been used as a means of expressing generative models whose posterior probability can be computed by general-purpose inference engines. Though sampling from discrete distributions (such as binary probabilistic branching) can be considered algorithmically adequate[1] for probabilistic computation, the generation of real-world data—a basic capability expected of generative models—requires expressivity of the whole gamut of continuous distributions. For this reason, sampling from continuous distributions is an essential feature of probabilistic programming languages. (See e.g. Church [27], Stan [13], Anglican [57], Gen [19], Pyro [5], Edward [58] and Turing [26].)

In this work we study a central property of probabilistic programs: *termination*. In non-probabilistic (possibly nondeterministic) computation, termination is a purely qualitative, boolean property. However, with randomness in the control flow, termination is characterised by a scalar quantity: the *probability of termination*. We say that a program is *almost-surely terminating* (AST) if a run of it terminates with probability 1.

When a probabilistic program implements a solution to an algorithmic problem, one naturally requires the computation to terminate with a high (lower bounded) probability, usually 1. In probabilistic programming, lower bounds and guarantees of AST are equally important. Indeed, it is standard for designers and implementors of probabilistic programming systems to regard non-AST programs as defining invalid models, and hence inadmissible (see e.g. [54, §4.3.2] and [27]). Moreover [40] have recently shown that AST programs have density (a.k.a. weight) functions that are differentiable almost everywhere. This is significant, because the latter property is a precondition for the correctness of some of the most scalable inference algorithms, such as Hamiltonian Monte Carlo [48, 60] and reparameterised gradient variational inference [38].

In this paper we tackle two key questions: computation of lower bounds on the probability of termination, and AST verification. While there has been much progress in the termination analysis of probabilistic programs with discrete distributions [8, 31, 34], programs with continuous distributions have received comparatively little attention. Many

---

[1]in the sense that they are enough to make any Turing complete programming language universal for probabilistic Turing machine [36, 56]

methods and proofs hinge on the countable nature inherent to discrete distributions [9, 20, 32, 34, 42, 43, 50]. It is not at all obvious if they can be extended to systems with continuous distributions.

Using an idealised functional language with continuous samples and stochastic conditioning, we provide partial answers to these questions. On the one hand, we give a definitive answer to the lower bound problem, and precisely determine the complexity of various termination problems in the arithmetic hierarchy. On the other hand, we provide a sound (but incomplete) proof method for AST which can be seen as orthogonal to [20].

1.1 High Level Overview

Lower Bound Computation. In languages with discrete distributions, evaluation can be seen as a step-indexed probability mass on terms [20, 23, 34]. By enumerating terminating executions, we can iteratively compute arbitrarily tight lower bounds on the probability of termination. As a direct consequence, AST is a decision problem in Π_2^0 [31], the second level of the arithmetic hierarchy [33]. In languages that admit continuous distributions, we cannot assign probability mass to terms directly. Rather, by viewing a probabilistic program as a deterministic program parameterised by an *execution trace* (or simply, trace) (i.e. the sequence of random draws made during the execution), we can organise such traces into a measure space [6, 35]. The probability of termination can then be defined as the measure of all traces on which the program terminates [40]. However, in general, a single terminating execution (or even a countable set thereof) cannot be assigned any positive probability measure. This leaves open problems such as sound computation of lower bounds, and the exact complexity of deciding AST.

We approach these problems by introducing a novel operational semantics based on *interval traces*, which are a summarisation of the relevant traces. We show soundness and completeness w.r.t. the sampling-style semantics [6]. Instead of analysing a program using uncountably many traces, we work with interval traces, where only countably many such traces suffice. This yields an effective procedure to compute lower bounds on termination probability, enabling the first proof that deciding AST in the presence of continuous distributions is Π_2^0-complete (under mild assumptions on the primitive functions). Further, we show that positive almost sure termination (PAST) (i.e., finite expected time to termination) is Σ_2^0-complete, assuming the program is AST. For general PAST, we can only infer a (possibly non-tight) upper bound of Δ_3^0. This does *not* match the Σ_2^0 bounds known for discrete distributions as a proof of this bound hinges on a countable set of executions [31]. See Sec. 3.

In addition we give an alternative presentation of our semantics as an intersection type system in Sec. 4. Our system extends [9] and [23] to languages with continuous distributions; moreover, both the probability of termination *and*

the expected time to termination can be obtained as the least upper bound of all derivations. This gives a type-based, compositional method for lower bound computation.

AST Verification. While our computation of lower bounds gives a Π_2^0 decision procedure for AST, it is not really effective for AST verification. Many of the recent advances in the development of AST verification methods [1, 14, 15, 17, 18, 25, 29, 30, 43, 50] are concerned with loop-based programs. We can view such loops as tail-recursive programs that, in particular, are *affine recursive*, i.e., in each evaluation (or run) of the body of the recursion, recursive calls are made from at most one call site [20, §4.1]. By contrast, many probabilistic programming languages allow for richer recursive structures [27, 41, 57]. We propose a new verification method for probabilistic programs that are defined by *non-affine recursion*, i.e., in the evaluation of the body of the recursion, multiple recursive calls can be made from *distinct* call sites. (Note that whether a program is affine recursive cannot be checked by just counting textual occurrences of variables.)

Example 1.1 (Running Example). Consider an unreliable 3d printing company. Unfortunately, for every printing, the outcome is acceptable with only probability p; if it is unacceptable, reprinting must take place on the following day, and thus, the process is repeated. We can model this scenario, starting with a single job, as the following program

$$\left(\mu_x^\varphi.\text{if sample} \le p \text{ then } x \text{ else } \varphi(x + \underline{1})\right)\underline{1} \tag{1}$$

where $\mu_x^\varphi.(\cdot)$ is a fixpoint constructor (that binds the variable φ to the fixpoint), and sample evaluates to a random draw from the uniform distribution on $[0, 1]$. The value returned by the program is the number of days needed to complete the job. Luckily, as the program is AST for all success probabilities $p \in (0, 1]$, the company can assure its customers that it will finish the job eventually. However, in a bid to drum up business, a new quality policy is introduced. The manager advises their customers: "Each day our print attempt fails, we will print an additional copy for you." We model the situation as follows:

$$\left(\mu_x^\varphi.\text{if sample} \le p \text{ then } x \text{ else } \varphi(\varphi(x + \underline{1}))\right)\underline{1} \tag{2}$$

Soon after implementing the new policy, it was noticed that some of the print jobs could never be completed. Phrased differently: Program (2) is no longer AST for every $p \in (0, 1]$.

This example illustrates that non-affine recursion, as exhibited in program (2), can complicate the analysis of termination. While the affine program (1) is clearly AST for every $p > 0$, program (2) is not. It turns out that (2) is AST if and only if $p \ge \frac{1}{2}$; and in case $p = \frac{1}{2}$, while the process is AST, the expected time to termination is infinite. It is unsurprising that termination depends on the number of recursive calls, as termination itself is a quantitative property.

Termination analysis of non-affine recursive probabilistic programs does not seem to have received much attention. Methods such as those presented in [20] explicitly restrict

to affine programs and are unsound otherwise. Our method for the analysis of non-affine recursive programs can be viewed as orthogonal to [20]: while they restrict to affine programs and investigate the recursive function argument for size information, we accept the function argument without examination, and admit non-affine programs. We call our methods *counting-based*, as we over-approximate the recursive behaviour by counting recursive calls from distinct call sites, thus reducing AST analysis to the analysis of a random walk for which we show linear decidability. See Sec. 5. Our method is the basis of an AST proof system that can verify programs (including the simple example above) well beyond the reach of existing methods (Sec. 6). As a simple corollary, we obtain a functional *generalisation* of the *zero-one law* for termination of while-programs [42, §2.6][2]

Contributions. Our main contributions are as follows:

- We propose a new sound and complete interval-based semantics that enables lower bound computation. We obtain a first proof that the AST (resp. PAST) decision problem is, under mild assumptions on primitive functions, Π_2^0-complete (resp. Σ_2^0-complete) even in the presence of continuous distributions.
- We give a local representation of our semantics as an intersection type system where both the probability of termination and expected time to termination are characterised as the least upper bound over all derivations.
- We provide a new proof method for AST verification of non-affine recursive programs. We show how our proof system can be automated.

Our theoretical results give rise to practical algorithms. We provide prototype implementations for both lower bound computation and AST verification based on our novel semantics and proof system respectively (Sec. 7). Missing proofs and further discussions can be found in [2].

2 Statistical PCF (SPCF)

We begin by introducing some basics of probability theory and presenting our language of study.

2.1 Basic Probability Theory

A *σ-algebra* on a set Ω, typically written Σ_Ω, is a collection of subsets of Ω such that $\Omega \in \Sigma_\Omega$, and Σ_Ω is closed under complementation and countable unions (and hence countable intersections). A *measurable space* is a pair (Ω, Σ_Ω) where Ω is a set (of outcomes) and Σ_Ω is a σ-algebra on Ω. A function $f : \Omega_1 \to \Omega_2$ between measurable spaces, $(\Omega_1, \Sigma_{\Omega_1})$ and $(\Omega_2, \Sigma_{\Omega_2})$, is called *measurable* if for every $A \in \Sigma_{\Omega_2}$, $f^{-1}(A) \in \Sigma_{\Omega_1}$. A *measure* on (Ω, Σ_Ω) is a function $\mu : \Sigma_\Omega \to \overline{\mathbb{R}_+}$ that satisfies $\mu(\emptyset) = 0$ and is σ-additive: if $\{A_i\}_{i\in\mathbb{N}}$ is

$$
\frac{}{\Gamma \vdash \mathsf{sample} : \mathbf{R}} \qquad \frac{\Gamma \vdash M : \mathbf{R}}{\Gamma \vdash \mathsf{score}(M) : \mathbf{R}}
$$

$$
\frac{\Gamma, \varphi : \alpha \to \beta, x : \alpha \vdash M : \beta}{\Gamma \vdash \mu_x^\varphi.M : \alpha \to \beta} \qquad \frac{\{\Gamma \vdash M_i : \mathbf{R}\}_{i=1}^{|f|}}{\Gamma \vdash f(M_1, \cdots, M_{|f|}) : \mathbf{R}}
$$

Figure 1. Selection of SPCF typing Rules

a countable family of pairwise disjoint sets from Σ_Ω then $\mu(\cup_i A_i) = \sum_i \mu(A_i)$. If $\mu(\Omega) \le 1$ we call μ a subprobability measure and if $\mu(\Omega) = 1$ we call it a probability measure (or distribution). For the n-dimensional Euclidean space $\mathbb{R}^n$ we write $\Sigma_{\mathbb{R}^n}$ for the Borel σ-algebra over $\mathbb{R}^n$, which is the smallest σ-algebra that contains all open and closed n-dimensional boxes. In the special case of $n = 1$, this is the set generated by all open (and closed) intervals. The n-dimensional Lebesgue measure, denoted λ_n, is the unique measure on $(\mathbb{R}^n, \Sigma_{\mathbb{R}^n})$ that satisfies $\lambda_n([a_1, b_1] \times \cdots \times [a_n, b_n]) = \prod_{i=1}^n (b_i - a_i)$.

Discrete Sample Space. In case Ω is countable, we often work with the powerset 2^Ω as the trivial σ-algebra. Every probability measure is then uniquely determined by a *probability mass function (pmf)*, a function $p : \Omega \to \mathbb{R}_{[0,1]}$ with $\sum_{x\in\Omega} p(x) = 1$. Every pmf p gives rise to a probability measure by defining $\mu(A) \coloneqq \sum_{x\in A} p(x)$; conversely, for every probability measure μ on the powerset we can recover a generating pmf by defining $p(x) \coloneqq \mu(\{x\})$. A subprobability mass function is defined analogously.

2.2 SPCF

Statistical PCF (SPCF) is an extension of PCF [51] with support to sample[3] from the uniform distribution on $[0, 1]$ and condition executions (see [28]). Terms in SPCF are implicitly parametrised over a set $\mathbb{F}$ of *measurable* functions $f : \mathbb{R}^n \to \mathbb{R}$ that model primitive operations. Each function $f \in \mathbb{F}$ has an arity $|f| \ge 0$. The sets of terms and values are defined by the following grammar where x and φ are distinct variables (from a fixed denumerable set of symbols), $r \in \mathbb{R}$ and $f \in \mathbb{F}$:

$$
V \coloneqq x \mid \underline{r} \mid \lambda x.M \mid \mu_x^\varphi.M
$$
$$
M, N, P \coloneqq V \mid MN \mid \mathtt{if}(M, N, P) \mid f(M_1, \cdots, M_{|f|})
$$
$$
\mid \mathsf{sample} \mid \mathsf{score}(M)
$$

As usual, we identify terms modulo α-conversion. The fixpoint constructor, $\mu_x^\varphi.(\cdot)$, binds the recursively defined function φ and its argument x. We abbreviate[4] $M \oplus_P N \coloneqq \mathtt{if}(\mathsf{sample}-P, M, N)$ in the style of [42] and write $M \oplus N$ for $M \oplus_{.5} N$. We type terms using a standard simple type system with types defined by $\alpha, \beta \coloneqq \mathbf{R} \mid \alpha \to \beta$. A selection of typing rules is given in Fig. 1 and [2]. We denote the set of typable SPCF terms by Λ and its subset of closed terms by Λ_0.

---

[2]The *zero-one law* states that a while-loop is almost-surely terminating if there is a positive lower bound on the probability of exiting it.

[3]Sampling from other real-valued distributions can be obtained from sample by applying the inverse of the distribution's cumulative distribution function; see e.g. [55, §2.3.1].

[4]Our conditional statement, $\mathtt{if}(P, M, N)$, branches on whether $P \le 0$.

In this paper we consider both call-by-name (CbN) and call-by-value (CbV) evaluation strategies. We use CbN for the first part of this paper, as the results (especially those about intersection types) are cleaner this way [9, 23, 34]. (Our CbN SPCF can express CbV computation at base types, giving it a suitable algorithmic expressiveness; c.f. [22].) We switch to CbV SPCF when presenting our AST proof system, thereby enabling a more straightforward comparison to related approaches such as [20].

2.3 Operational Semantics

We give a sampling-style operational semantics for SPCF. The idea (going back to Kozen [35]) is to evaluate a term M together with a sequence of (fixed) probabilistic outcomes for each sample statement [6, 40]. We then generate a probabilistic interpretation of programs by endowing the set of traces with a measure.

CbN SPCF. We define the set of traces $\mathbb{S}$ as all finite sequences of real numbers from $\mathbb{R}_{[0,1]} := \{r \in \mathbb{R} \mid 0 \leq r \leq 1\}$), i.e., $\mathbb{S} := \mathbb{R}^*_{[0,1]} = \bigcup_{n \in \mathbb{N}} \mathbb{R}^n_{[0,1]}$. We let s range over elements in $\mathbb{S}$, denote the empty trace with ϵ; for $r \in \mathbb{R}_{[0,1]}$ write $\underline{r}$ for the one element trace; and s_1, s_2 for concatenation. The set of CbN redexes and evaluation contexts is defined by:

$$R := (\lambda x.M)N \mid (\mu^\varphi_x.M)N \mid \mathtt{if}(\underline{r}, N, P)$$
$$\mid f(\underline{r_1}, \cdots, \underline{r_{|f|}}) \mid \mathtt{sample} \mid \mathtt{score}(\underline{r})$$
$$E := [\cdot] \mid EM \mid \mathtt{if}(E, N, P) \mid \mathtt{score}(E)$$
$$\mid f(\underline{r_1}, \cdots, \underline{r_{k-1}}, E, M_{k+1}, \cdots, M_{|f|})$$

Given a context E and a term M the (capture-permitting) substitution $E[M]$ is defined in the obvious way. An easy induction establishes that every $M \in \Lambda_0$ is either a value or there are *unique* E and R, s.t., $M = E[R]$ (see e.g. [6]). The small-step reduction relation has the from $\langle M, s \rangle \rightarrow \langle M', s' \rangle$ where M, M' are terms and s, s' are traces. It is defined inductively by the rules given in Fig. 2 where $M[N_i/x_i]_i$ denotes standard capture-avoiding substitution [4]. Note that our reduction does not enjoy *progress*, as e.g. redex $\mathtt{score}(\underline{r})$ cannot reduce if $r < 0$. As this work is a study of termination properties, we elide the weight parameter used for stochastic conditioning[5] (see e.g. [6, 28, 40]).

A Measure on Traces. To interpret probabilistic programs using traces, we first need to endow the set of traces with a measure. We cannot assign probability mass to individual traces directly, as there are uncountably many traces. Instead we define a suitable measurable space of program traces following [6]. Let $\Sigma_{\mathbb{R}^n_{[0,1]}}$ be the Borel σ-algebra on $\mathbb{R}^n_{[0,1]}$ (We set $\Sigma_{\mathbb{R}^0_{[0,1]}} := \{\emptyset, \{\epsilon\}\}$). We can then define a σ-algebra

---

[5]The score-construct which is responsible for stochastic conditioning has, nevertheless, a subtle effect on termination as we require the conditioned value to be positive.

on traces ($\Sigma_{\mathbb{S}}$) and a measure ($\mu_{\mathbb{S}}$) by:

$$\Sigma_{\mathbb{S}} := \{\biguplus_{n \in \mathbb{N}} B_n \mid B_n \in \Sigma_{\mathbb{R}^n_{[0,1]}}\}$$
$$\mu_{\mathbb{S}}\left(\biguplus_{n \in \mathbb{N}} B_n\right) := \sum_{n \in \mathbb{N}} \lambda_n(B_n)$$

As shown in [6, Lem. 7 & 8], $(\mathbb{S}, \Sigma_{\mathbb{S}})$ is a measurable space and $\mu_{\mathbb{S}}$ a (σ-finite) measure on $(\mathbb{S}, \Sigma_{\mathbb{S}})$.

2.4 Probabilistic Termination

With $\rightarrow^n$ we denote the n-fold self-composition, and with $\rightarrow^*$ the reflexive-transitive closure, of $\rightarrow$. We define

$$\mathbb{T}_{M,\text{term}} := \left\{ s \in \mathbb{S} \mid \exists V : \langle M, s \rangle \rightarrow^* \langle V, \epsilon \rangle \right\}$$

as the set of traces on which a term M terminates, which is measurable (similar to [6, Lem. 9]). As shown in [40, Lem. 7], $\mu_{\mathbb{S}}(\mathbb{T}_{M,\text{term}}) \leq 1$; we are therefore justified in calling the interpretation $\mu_{\mathbb{S}}(\mathbb{T}_{M,\text{term}})$ a "probability".

Definition 2.1. The *probability of termination* of $M \in \Lambda_0$ is defined by $\mathbb{P}_{\text{term}}(M) := \mu_{\mathbb{S}}(\mathbb{T}_{M,\text{term}})$. M is called *almost-surely terminating* (AST) if $\mathbb{P}_{\text{term}}(M) = 1$.

Positive Almost-Sure Termination. An even stronger property than AST is finiteness of the expected time to termination. For any trace $s \in \mathbb{T}_{M,\text{term}}$ we define $\#^s_{\downarrow}(M) \in \mathbb{N}$ as the unique number n such that $\langle M, s \rangle \rightarrow^n \langle V, \epsilon \rangle$ for some value V. For any $n \in \mathbb{N}$ we define

$$\mathbb{T}^{\leq n}_{M,\text{term}} := \left\{ s \in \mathbb{T}_{M,\text{term}} \mid \#^s_{\downarrow}(M) \leq n \right\}$$

as the set of traces on which termination occurs within n steps, which is measurable. We define $\mathbb{T}^n_{M,\text{term}}$ analogously.

Definition 2.2. For $M \in \Lambda_0$ we define the *expected time to termination*, $\mathbb{E}_{\text{term}}(M) \in \overline{\mathbb{R}_+}$, by

$$\mathbb{E}_{\text{term}}(M) := \sum_{n=0}^{\infty} \left(1 - \mu_{\mathbb{S}}\left(\mathbb{T}^{\leq n}_{M,\text{term}}\right)\right)$$

M is *positive almost-surely terminating* if $\mathbb{E}_{\text{term}}(M) < \infty$.

It is easy to see that any program that is PAST is also AST. Following [31], $\mathbb{E}_{\text{term}}(M)$ can be phrased as $\sum_{n=0}^{\infty} \mathbb{P}(\text{"}M \text{ runs for more than } n \text{ steps"}) = \sum_{n=0}^{\infty} \left(1 - \mathbb{P}(\text{"}M \text{ terminates within } n \text{ steps"})\right)$, with the latter expressed in Def. 2.2. We can show that, provided M is AST, the expected time to termination is the expected value of the random variable that gives the number of reduction steps:

Lemma 2.3. *If M is AST,* $\mathbb{E}_{term}(M) = \sum_{n=0}^{\infty} \mu_{\mathbb{S}}\left(\mathbb{T}^n_{M,term}\right) \cdot n$

CbV SPCF. Our CbV SPCF is essentially the system of [40], except that we use a simpler CbV fixpoint reduction rule.

3 Interval-based Semantics

It is impractical to use the standard trace-based (or sampling-style) semantics to reason about termination properties of SPCF programs, because the trace measure $\mu_{\mathbb{S}}$ is continuous. Suppose we are interested in the decidability of the *lower bound question*: does a term terminate with probability

$$\frac{}{\langle (\lambda x.M)N, s\rangle \to \langle M[N/x], s\rangle} \qquad \frac{}{\langle (\mu_x^\varphi.M)N, s\rangle \to \langle M[N/x, (\mu_x^\varphi.M)/\varphi], s\rangle} \qquad \frac{}{\langle \mathtt{sample}, r\, s\rangle \to \langle \underline{r}, s\rangle}$$

$$\frac{r \le 0}{\langle \mathtt{if}(\underline{r}, N, P), s\rangle \to \langle N, s\rangle} \qquad \frac{r > 0}{\langle \mathtt{if}(\underline{r}, N, P), s\rangle \to \langle P, s\rangle} \qquad \frac{r \ge 0}{\langle \mathtt{score}(\underline{r}), s\rangle \to \langle \underline{r}, s\rangle}$$

$$\frac{}{\langle f(\underline{r_1}, \cdots, \underline{r_{|f|}}), s\rangle \to \langle \underline{f(r_1, \cdots, r_{|f|})}, s\rangle} \qquad \frac{\langle R, s\rangle \to \langle M, s'\rangle}{\langle E[R], s\rangle \to \langle E[M], s'\rangle}$$

Figure 2. Call by Name small-step reduction for SPCF.

strictly greater than p? For discrete distributions, this problem is r.e. (in Σ_1^0) as we can enumerate terminating paths until the sum of the weight of those paths exceeds p [31, 34]. In the presence of continuous distributions, this is no longer possible. A well-known property of the Lebesgue measure on $\mathbb{R}_{[0,1]}^n$ (inherited by the trace measure $\mu_\mathbb{S}$) is that every countable set of elements is a null set. So even if we can identify a countably infinite set of traces $A \subseteq \mathbb{T}_{M,\mathrm{term}}$, we cannot obtain any non-trivial lower bound on $\mathbb{P}_{\mathrm{term}}(M)$. Thus the semantics itself cannot be used to settle such complexity questions as whether the lower bound problem for SPCF is in Σ_1^0, or whether the AST problem is in Π_2^0, or whether the PAST problem in Σ_2^0. In this section, we introduce a novel operational semantics for SPCF by executing terms parameterised by a trace of *intervals*. We demonstrate that this semantics, which is complete w.r.t. the trace-based semantics, is well-suited to the derivation of lower bounds. The completeness hinges on the observation that, under mild restrictions on primitive functions, interval-based reasoning can effectively abstract actual traces. This is the basis of our positive answer to the questions above.

Syntax of Interval Terms. We adjust the syntax of terms slightly and treat intervals as constant symbols of type **R**. We define interval values and interval terms as follows where $a \le b \in \mathbb{R}$.

$$\mathcal{V} := x \mid \underline{[a,b]} \mid \lambda x.\mathcal{M} \mid \mu_x^\varphi.\mathcal{M}$$
$$\mathcal{M}, \mathcal{N}, \mathcal{P} := \mathcal{V} \mid \mathcal{M}\mathcal{N} \mid \mathtt{if}(\mathcal{M}, \mathcal{N}, \mathcal{P}) \mid f(\mathcal{M}_1, \cdots, \mathcal{M}_{|f|})$$
$$\mid \mathtt{sample} \mid \mathtt{score}(\mathcal{M})$$

Our simple type system extends naturally. We denote the set of (closed, bounded) intervals by $\mathfrak{I}$, and write $\mathfrak{I}_{0,1} := \{[a,b] \mid a, b \in \mathbb{R}, 0 \le a \le b \le 1\}$ as the set of intervals with endpoints between 0 and 1. With $\mathfrak{I}^\mathbb{Q}$ and $\mathfrak{I}_{0,1}^\mathbb{Q}$ we denote the sets $\mathfrak{I}$ and $\mathfrak{I}_{0,1}$ respectively, restricted to *rational endpoints*.

Definition 3.1. We call $f : \mathbb{R}^n \to \mathbb{R}$ *interval preserving* (resp. $\mathbb{Q}$-*interval preserving*) if there is a function $\hat{f} : \mathbb{R}^{2n} \to \mathfrak{I}$ (resp. $\hat{f} : \mathbb{Q}^{2n} \to \mathfrak{I}^\mathbb{Q}$) such that for every sequence of intervals $[a_1, b_1], \cdots, [a_n, b_n] \in \mathfrak{I}$ (resp. $\in \mathfrak{I}^\mathbb{Q}$) we have $f([a_1, b_1] \times \cdots \times [a_n, b_n]) = \hat{f}(a_1, b_1, \cdots, a_n, b_n)$, i.e., the image of every n-dimensional box (resp. with rational endpoints) is an interval (resp. with rational endpoints).

$$\frac{b \le 0}{\langle \mathtt{if}(\underline{[a,b]}, \mathcal{N}, \mathcal{P}), \wp\rangle \rightsquigarrow \langle \mathcal{N}, \wp\rangle} \qquad \frac{a > 0}{\langle \mathtt{if}(\underline{[a,b]}, \mathcal{N}, \mathcal{P}), \wp\rangle \rightsquigarrow \langle \mathcal{P}, \wp\rangle}$$

$$\frac{}{\langle \mathtt{sample}, [a,b] :: \wp\rangle \rightsquigarrow \langle \underline{[a,b]}, \wp\rangle} \qquad \frac{0 \le a}{\langle \mathtt{score}(\underline{[a,b]}), \wp\rangle \rightsquigarrow \langle \underline{[a,b]}, \wp\rangle}$$

$$\frac{}{\langle f(\underline{[a_1, b_1]}, \cdots, \underline{[a_{|f|}, b_{|f|}]}), \wp\rangle \rightsquigarrow \langle \underline{\hat{f}(a_1, b_1, \cdots, a_{|f|}, b_{|f|})}, \wp\rangle}$$

Figure 3. Selection of interval-based reduction rules

We restrict the primitive functions to those that are interval preserving to ensure that interval-based reasoning is compatible with primitive operations. As the following shows, most interesting functions (including e.g. $+, \cdot, -, \exp, |\cdot|, \cdots$) are interval preserving.

Lemma 3.2. *If $f : \mathbb{R}^n \to \mathbb{R}$ is continuous then f is interval preserving.*

3.1 Interval-based Syntax and Semantics

We define the set of interval traces by $\mathbb{S}_\mathfrak{I} := \bigcup_{n \in \mathbb{N}} \mathfrak{I}_{0,1}^n$, i.e., finite sequences of intervals with endpoints between 0 and 1 (inclusive). We let $\wp$ range over elements in $\mathbb{S}_\mathfrak{I}$. To avoid confusion, we shall refer to elements of $\mathbb{S}_\mathfrak{I}$ as *interval traces*, and elements of $\mathbb{S}$ as *standard traces*.

Redexes and evaluation contexts of interval terms are defined as expected. As we only replace real-valued numerals with interval-valued, our standard small-step semantics (Fig. 2) mostly extends to interval terms. The specific reduction rules concerning the control flow and primitive functions are given in Fig. 9. As a useful intuition, it is helpful to view an interval numeral $\underline{[a,b]}$ as an unknown value within that interval. As in the standard semantics, we are interested in the interval traces that lead to a normal form.

$$\mathbb{T}_{\mathcal{M},\mathrm{term}}^\mathfrak{I} := \{\wp \in \mathbb{S}_\mathfrak{I} \mid \exists \mathcal{V} : \langle \mathcal{M}, \wp\rangle \rightsquigarrow^* \langle \mathcal{V}, \epsilon\rangle\}$$

For any $\wp \in \mathbb{T}_{\mathcal{M},\mathrm{term}}^\mathfrak{I}$, we define $\#_\downarrow^\wp(\mathcal{M})$ as the number of reduction steps to termination.

Embedding Into Intervals. While we want to analyse the termination probability of standard terms, our interval-based semantics builds on interval terms. We define a natural embedding $(\cdot)^{2\mathfrak{I}}$ that maps every standard term M to the interval term $M^{2\mathfrak{I}}$ obtained by replacing every numeral $\underline{r}$ by the interval numeral $\underline{[r,r]}$. Our soundness and completeness

results are now based on the operational behavior of $M^{2\Im}$ (in the interval semantics), and they allow us to draw conclusions about the behavior of M (in the standard semantics).

3.2 Soundness

We now show that the interval-based semantics gives lower bounds on the probability of termination in the standard semantics. We define the *weight of an interval trace* $\wp$, denoted by $\omega(\wp)$, in the obvious way:

$$\omega([a_1, b_1], \cdots, [a_n, b_n]) := \prod_{i=1}^{n} (b_i - a_i)$$

To combine the weight of multiple terminating interval traces we need to ensure that the interval traces are disjoint, i.e., we do not account twice for the same standard trace.

Definition 3.3. Two interval traces $\wp = [a_1, b_1], \cdots, [a_n, b_n]$ and $\wp' = [a'_1, b'_1], \cdots, [a'_m, b'_m]$ are *compatible* if $n \neq m$ or there exists i such that $b_i \leq a'_i$ or $b'_i \leq a_i$.

For example, the four interval traces, $[0, 1][0, \frac{1}{3}]$, $[0, 1][\frac{1}{3}, \frac{1}{2}]$, $[0, 1][\frac{3}{4}, 1]$ and $[0, 1]$, are pairwise compatible. For a *countable* set of interval traces A we define $\omega(A) := \sum_{\wp \in A} \omega(\wp)$; if $A \subseteq \mathbb{T}^{\Im}_{M,\text{term}}$ we also define the expected value of A, denoted $\mathbb{E}(\mathcal{M}, A)$, by

$$\mathbb{E}(\mathcal{M}, A) := \sum_{\wp \in A} \omega(\wp) \cdot \#^{\wp}_{\downarrow}(\mathcal{M})$$

We can now state soundness as follows:

Theorem 3.4. *For every countable set of* pairwise compatible *traces* $A \subseteq \mathbb{T}^{\Im}_{M^{2\Im},\text{term}}$ *the following holds:*

- $\omega(A) \leq \mathbb{P}_{term}(M)$
- $\mathbb{E}(M^{2\Im}, A) \leq \mathbb{E}_{term}(M)$

This (perhaps unsurprising) soundness result is the basis of an effective tool to verify lower bounds on $\mathbb{P}_{\text{term}}(M)$ and $\mathbb{E}_{\text{term}}(M)$. The real force of the interval-based semantics lies in its completeness.

3.3 Completeness

We show that, under mild assumptions on the primitive functions, a countable number of traces for $M^{2\Im}$ already gives the exact probability of termination $\mathbb{P}_{\text{term}}(M)$. Consequently, by an incremental search of terminating interval-traces, we can compute arbitrarily tight lower bounds on $\mathbb{P}_{\text{term}}(M)$.

Example 3.5. Consider the term

$$M = \left(\mu^{\varphi}_x . \text{if sample} + \text{sample} - \underline{1} \text{ else } x \text{ else } \varphi \, x \right) \underline{0}.$$

For the moment we focus on the set of traces on which this term terminates *without* making a single recursive call which is $T = \{r_1 \, r_2 \in \mathbb{R}^2_{[0,1]} \mid r_1 + r_2 \leq 1\}$. This set cannot be described by a countable union of interval traces, i.e., there are no interval traces $\{\wp_i\}_{i \in \mathbb{N}}$ such that $s \in T \Leftrightarrow \exists i \in \mathbb{N} : s \triangleleft \wp_i$, where $s \triangleleft \wp_i$ means that s refines $\wp_i$ (see [2]). Nevertheless, as M is AST, our completeness result states that we can find a countable family of (pairwise compatible) interval traces, $A \subseteq \mathbb{T}^{\Im}_{M^{2\Im},\text{term}}$, whose cumulative weight (i.e. $\omega(\mathcal{A})$) equals $\mathbb{P}_{\text{term}}(M) = 1$.

To achieve completeness we need the concept of *interval separable* primitive functions. For measurable $A, B \subseteq \mathbb{R}^n$ we write $A \Subset B$ if $A \subseteq B$ and $\lambda_n(B \setminus A) = 0$, i.e., A is contained in, and, up to a null set, equal to, B. Interval separability now states that the preimage of every interval can be written, up to a null set, as a countable union of boxes. Precisely:

Definition 3.6. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *interval separable* if for every interval $[a, b] \in \Im$, there exists a family of *boxes* $\{B_i\}_{i \in \mathbb{N}}$ with $B_i \subseteq \mathbb{R}^n$ such that $\cup_i B_i \Subset f^{-1}([a, b])$.

Most interesting functions such as $+$, $\cdot$, $\exp$, etc. are interval separable.

Lemma 3.7. *If* $f : \mathbb{R}^n \to \mathbb{R}$ *is continuous, and for all* $y \in \mathbb{R}$, $f^{-1}(\{y\})$ *is a Lebesgue null set, then* f *is interval separable.*

Theorem 3.8. *If every* $f \in \mathbb{F}$ *is interval separable, then for every* $M \in \Lambda_0$ *there exists a countable set of pairwise-compatible interval traces* $A \subseteq \mathbb{T}^{\Im}_{M^{2\Im},\text{term}}$ *such that* $\omega(A) = \mathbb{P}_{term}(M)$; *and if* M *is AST then* $\mathbb{E}(M^{2\Im}, A) = \mathbb{E}_{term}(M)$.

Proof Sketch. We first partition $\mathbb{T}_{M,\text{term}}$ according to the branching behaviour, i.e., sequences in $\{0, 1\}^*$ indicating if the left or the right branch of conditionals was taken. We then fix a branching behaviour (notice that $\{0, 1\}^*$ is countable) and employ *stochastic symbolic execution* (in the sense of [40]) by executing a term on a trace of variables, while collecting symbolic constraints along the way. As primitive functions are interval separable, we show that the corresponding constraints can be exhausted via interval traces. □

Incompleteness. While the collection of primitive functions with respect to which our semantics is complete is very broad (c.f. Lem. 3.7), interval-based reasoning is incomplete in the presence of arbitrary continuous functions.

Example 3.9. Let $C \subseteq \mathbb{R}$ be any Smith-Volterra-Cantor set, i.e., C has positive Lebesgue measure but is nowhere dense, i.e., there are no $a < b$ with $[a, b] \in C$. Now construct function $f_C : \mathbb{R} \to \mathbb{R}$ by $f_C(x) := d(x, C)$, the distance of x to C. As C is a closed set, the function is well-defined and obviously continuous; and the roots of f_C coincide with C. Then $M := \text{if } f_c(\text{sample}) \text{ then } \underline{0} \text{ else } \underline{1}$ is clearly AST. However, in the interval-based semantics, we can never derive a termination probability of more than $1 - \lambda_1(C) < 1$ as there is no non-trivial interval trace taking the left branch.

3.4 AST and PAST in the Arithmetic Hierarchy

If we only consider functions that are $\mathbb{Q}$-interval preserving we can restrict the previous reasoning to intervals and boxes with rational endpoints. This has direct recursion-theoretic consequences.

Theorem 3.10. *Assume that every* $f \in \mathbb{F}$ *is* $\mathbb{Q}$-*interval preserving and interval separable, and* $\hat{f}$ *is computable. For any term* M *(containing only rational numerals), deciding AST is*

in Π_2^0. If M is AST, deciding PAST is in Σ_2^0. In general, deciding PAST is in Δ_3^0.

Proof. Thanks to Thm. 3.4 and Thm. 3.8 we can express "M is AST" by the following $\forall\exists$-formula:

$$\forall\epsilon > 0 \in \mathbb{Q}.\ \exists A.\ A \subseteq \mathbb{T}_{M^{2\mathfrak{I}},\text{term}}^{\mathfrak{I}} \wedge \omega(A) \geq 1 - \epsilon$$

where A ranges over (encodings of) *finite, pairwise compatible* sets of interval traces with *rational* endpoints. If M is AST we can express PAST (i.e. $\mathbb{E}_{\text{term}}(M) < \infty$) as this $\exists\forall$-formula:

$$\exists c \in \mathbb{Q}.\ \forall A.\ A \subseteq \mathbb{T}_{M^{2\mathfrak{I}},\text{term}}^{\mathfrak{I}} \Rightarrow \mathbb{E}(M^{2\mathfrak{I}}, A) \leq c$$

In general, M-is-PAST $\Leftrightarrow M$-is-AST $\wedge\ \mathbb{E}_{\text{term}}(M) < \infty$, so the general PAST decision problem is in Δ_3^0. □

If addition is definable, then—thanks to the hardness results in [31]—deciding AST in the presence of continuous distributions (and suitable primitive functions) is Π_2^0-complete; and deciding PAST (assuming AST) is Σ_2^0-complete. We remark that the Δ_3^0 upper bound for the general PAST problem does not match the corresponding bound for discrete distributions [31]. The approach in [31] uses the fact that there are finitely many traces of a given length; this property obviously does not hold in the presence of continuous distributions.

4 Intersection Type System

Intersection types have long been studied in termination analysis as they can give a complete characterisation of termination: A λ-term is typable in a (suitable) intersection type system iff it is strongly normalising. A first study of the quantitative notion of AST, and whether the intriguing completeness of intersection types can be extended to a probabilistic language, was conducted in [9]. Owing to the intrinsic Π_2^0-hardness of AST [31], we cannot hope for a semi-decidable type system in which a term is typable iff it is AST. Instead [9] presented two approaches to termination analysis, where the probability of termination is either a sum over all (countably many) typing derivation (called the oracle system) or the least upper bound (lub) thereof. We show that completeness of intersection types w.r.t. termination can also be established for a language with continuous samples (where a program admits uncountably many distinct runs), thereby giving a *local representation* of our interval-based semantics. In our system the lub over countably many derivations gives the probability of termination *and* the expected number of computation steps. Thus we obtain a complete, compositional and recursion-theoretically optimal method for computing lower bounds on both the probability of termination and the expected time to termination.

4.1 Intersection Type System For SPCF

Our system conceptually lies between the two approaches of [9] (alluded to above): we reason about the lub, and at the same type explicitly enumerate terminating (interval) traces as in the oracle system of [9]. The system in [9] relies on

the countable nature of the execution tree and can exhibit subject reduction by taking the weighted (finite) sum over the reduction relation. This approach does not work for SPCF because of the uncountable nature of the latter. Instead, our proofs hinge on the soundness and completeness of the interval-based semantics (Sec. 3).

Set Types. We define *set types* by the following grammar:

$$\alpha := [a, b] \mid \sigma \to \mathcal{A} \qquad \sigma := \{\mathcal{A}_1, \cdots, \mathcal{A}_n\}$$

$$\mathcal{A} := \left\{(\alpha_1, \wp_1, \tau_1), \cdots, (\alpha_m, \wp_m, \tau_m)\right\}$$

where each $\wp_i$ is an interval trace, and τ_i a natural number. We refer to elements σ as *intersections* and $\mathcal{A}$ as *set types*. To effectively type conditionals we need to integrate first-order data, in our case intervals, in the types themselves. This is similar to the type system in [23]. For a set type $\mathcal{A} = \left\{(\alpha_i, \wp_i, \tau_i)\right\}_i$ we write $\mathcal{A}^{(\uparrow\wp, \tau)}$ for the set type $\left\{(\alpha_i, \wp\,\wp_i, \tau_i + \tau)\right\}_i$, i.e., the set obtained by prepending $\wp$ to every trace and adding τ to every count.

Type System. Typing judgments are of the form $\Gamma \vdash \mathcal{M} : \mathcal{A}$. Valid judgments are defined by induction over the rules in Fig. 4. Intuitively, if $\vdash \mathcal{M} : \left\{(\alpha_i, \wp_i, \tau_i)\right\}_i$ then $\wp_i$ are all terminating traces for M on which exactly τ_i steps are made until a value is reached. We advise the reader to compare this system with the monadic system given in [9, §6.1]. Note, in particular, that the type of an application is determined by the left argument, matching the CbN β-reduction where arguments are passed unevaluated. While the (if)-rule looks complicated at first sight, the subscript ($[a, b], \wp, \tau$) for each set type is merely used as an index, i.e., if $\Gamma \vdash \mathcal{M} : \mathcal{A}$ we can combine a different type derivation for every element in $\mathcal{A}$. Although we restrict primitive functions to have arity 2, the rules can easily be extended to handle higher arities. We omitted the general rule as it gets chaotic. For set type $\mathcal{A} = \left\{(\alpha_i, \wp_i, \tau_i) \mid i \in [n]\right\}$ we define $\omega(\mathcal{A}) := \sum_{i\in[n]} \omega(\wp_i)$ and $\mathbb{E}(\mathcal{A}) := \sum_{i\in[n]} \omega(\wp_i)\cdot\tau_i$. We can then show correctness.

Theorem 4.1. *For every term $M \in \Lambda_0$,*

1. $\displaystyle\bigvee_{\vdash M^{2\mathfrak{I}}:\mathcal{A}} \omega(\mathcal{A}) = \mathbb{P}_{term}(M)$, and
2. *If M is AST,* $\displaystyle\bigvee_{\vdash M^{2\mathfrak{I}}:\mathcal{A}} \mathbb{E}(\mathcal{A}) = \mathbb{E}_{term}(M)$

This gives a recursion-theoretically optimal characterisation of AST that is purely based on the type system (c.f. [9, §5.3]). Thus we can computationally analyse termination, not just by evaluation (c.f. Sec. 3), but also via a local typing system. By incrementally searching for typing derivations, we can compute arbitrarily tight bounds. Compared to [9], a novel feature of our work lies in the fact that we can explicitly reason about execution time, thus enabling a type-based characterisation of PAST (for terms that are AST). While our system as a whole may look a little intimidating, each rule is actually simple by itself, requiring no complex operations. The idea of annotating types by a step count is also

$$\frac{\mathcal{A} \in \sigma}{\Gamma, x : \sigma \vdash x : \mathcal{A}} \text{ (var)} \qquad \frac{}{\Gamma \vdash \underline{[a,b]} : \left\{([a,b], \epsilon, 0)\right\}} \text{ (num)} \qquad \frac{\{[a_i, b_i]\}_{i \in [n]} \text{ are almost disjoint}}{\Gamma \vdash \mathtt{sample} : \left\{([a_i, b_i], [a_i, b_i], 1) \mid i \in [n]\right\}} \text{ (sample)}$$

$$\frac{\Gamma, x : \sigma, \varphi : \gamma \vdash \mathcal{M} : \mathcal{A} \qquad \left\{\Gamma \vdash \mu_x^\varphi.\mathcal{M} : \mathcal{B} \mid \forall \mathcal{B} \in \gamma\right\}}{\Gamma \vdash \mu_x^\varphi.\mathcal{M} : \left\{(\sigma \to \mathcal{A}, \epsilon, 0)\right\}} \text{ (fix)} \qquad \frac{\Gamma \vdash \mathcal{M} : \mathcal{A} \qquad \{\Gamma \vdash \mathcal{N} : C \mid (\sigma \to \mathcal{B}, \wp, \tau) \in \mathcal{A}, C \in \sigma\}}{\Gamma \vdash \mathcal{M}\mathcal{N} : \bigcup_{(\sigma \to \mathcal{B}, \wp, \tau) \in \mathcal{A}} \mathcal{B}^{(\uparrow \wp, \tau+1)}} \text{ (app)}$$

$$\frac{}{\Gamma \vdash \mathcal{M} : \{\}} \text{ ($\{\!\!\{\}\!\!\}$)} \qquad \frac{\Gamma, x : \sigma \vdash \mathcal{M} : \mathcal{A}}{\Gamma \vdash \lambda x.\mathcal{M} : \left\{(\sigma \to \mathcal{A}, \epsilon, 0)\right\}} \text{ (abs)} \qquad \frac{\Gamma \vdash \mathcal{M} : \mathcal{A}}{\Gamma \vdash \mathtt{score}(\mathcal{M}) : \left\{([a,b], \wp, \tau+1) \mid ([a,b], \wp, \tau) \in \mathcal{A}, a \geq 0\right\}} \text{ (score)}$$

$$\frac{\Gamma \vdash \mathcal{M} : \mathcal{A} \qquad \{\Gamma \vdash \mathcal{N} : \mathcal{B}_{([a,b], \wp, \tau)} \mid ([a,b], \wp, \tau) \in \mathcal{A}, b \leq 0\} \qquad \{\Gamma \vdash \mathcal{P} : C_{([a,b], \wp, \tau)} \mid ([a,b], \wp, \tau) \in \mathcal{A}, a > 0\}}{\Gamma \vdash \mathtt{if}(\mathcal{M}, \mathcal{N}, \mathcal{P}) : \bigcup_{([a,b], \wp, \tau) \in \mathcal{A} \mid b \leq 0} \mathcal{B}^{(\uparrow \wp, \tau+1)}_{([a,b], \wp, \tau)} \cup \bigcup_{([a,b], \wp, \tau) \in \mathcal{A} \mid a > 0} C^{(\uparrow \wp, \tau+1)}_{([a,b], \wp, \tau)}} \text{ (if)}$$

$$\frac{\Gamma \vdash \mathcal{M} : \mathcal{A} \qquad \{\Gamma \vdash \mathcal{N} : \mathcal{B}_{([a,b], \wp, \tau)} \mid ([a,b], \wp, \tau) \in \mathcal{A}\}}{\Gamma \vdash f(\mathcal{M}, \mathcal{N}) : \bigcup_{([a,b], \wp, \tau) \in \mathcal{A}} \bigcup_{([c,d], \wp', \tau') \in \mathcal{B}_{([a,b], \wp, \tau)}} \left\{(\hat{f}(a,b,c,d), \wp\wp', \tau + \tau' + 1)\right\}} \text{ (f_2)}$$

Figure 4. Intersection Type System for SPCF.

applicable to the setting of [9], giving a strict generalisation of their system. Our system can also be easily generalised to the untyped λ-calculus considered in [6]. Lastly, while our correctness proof hinges on the completeness of the interval-based semantics, we can present the system without referring to interval traces directly, and instead consider probability mass functions on types (as done in e.g. [9]).

5 Counting-based Recursion Analysis

We now turn our attention to devising a method that, unlike the preceding approach, can prove AST efficiently. We focus on programs that can make multiple recursive calls from distinct call sites (during evaluation of the recursive body); we call such recursion *non-affine*. As evident from Ex. 1.1, non-affine recursion complicates AST analysis considerably. Intuitive results such as the *zero-one law of termination*[2] [42] are only valid for affine recursion.

Our framework builds on the idea of counting. We show that analysis of the resulting distributions on natural numbers suffices for proving AST of the program. As a corollary we obtain a functional generalisation of the *zero-one law*, which specialises to the original law in case the recursion is affine. Our approach to proving non-affine recursion can be viewed as orthogonal to [20] (which is restricted to affine recursion): rather than using the size-related information of the recursive function argument, we count the number of recursive calls from distinct call sites in the evaluation of the body of the recursion. Moreover, compared to [20], our approach supports continuous distributions, and it is not restricted to binary probabilistic choice. Compared to techniques based on ranking functions – a dominant approach to AST verification [1, 14, 17, 18, 25, 42, 43], our method is fully automatic and easy to implement, as we show in Sec. 6.

Example 5.1. Let's revisit the 3d printing company (Ex. 1.1). The new situation is that the staff gets tired over time and prints an incorrect number of copies. In case the print is faulty, there is a probability $sig(x)$ of the operator becoming tired and making mistakes, where sig is the sigmoid function. (Thus with increasing time (x), the probability of making mistakes approaches 1.) The operator's mistake takes the form of printing 3 instead of the intended 2 copies with probability .5. We model this scenario by the following term:

$$\mu_x^\varphi.x \oplus_p \left(\left(\varphi^3(x+1) \oplus \varphi^2(x+1)\right) \oplus_{sig(x)} \varphi^2(x+1)\right).$$

The question now becomes: for which p is this term AST?

We keep track of the number of calls by extracting a *counting distribution*, a (sub) pmf on $\mathbb{N}$, that models the distribution on new calls made. To account for the fact that a recursive function that is called n times (inclusive of the original call) contributes $n - 1$ to the total number of *pending calls*[6], we shift the counting pattern by -1, obtaining a (sub) pmf on $\mathbb{Z}$. The counting distribution is analysed via a random walk whose current value can be seen as the number of pending calls. In this section, we first introduce the necessary tools to analyse a random walk (Sec. 5.1), then present the extraction of the counting distribution from programs (Sec. 5.2), and the soundness theorem that relates termination behavior of the shifted random walk with that of the non-affine recursive program in question (Sec. 5.3).

5.1 Random Walk on $\mathbb{N}$

Assume a countable state space X. A *stochastic matrix on X* is a function $\mathfrak{P} : X \times X \to \mathbb{R}_{[0,1]}$ such that $\sum_{y \in X} \mathfrak{P}(x, y) = 1$ for every $x \in X$ (see e.g. [3, §10.1] or [44]). $\mathfrak{P}(x, y)$ gives

---

[6]equivalently, the maximum number of stack frames on the function's call stack

the probability of transitioning from x to y. Given stochastic matrices $\mathfrak{P}$ and $\mathfrak{P}'$, we write $\mathfrak{P}\,\mathfrak{P}'$ for their product, which is a stochastic matrix; and $\mathfrak{P}^n$ for the n-fold product of $\mathfrak{P}$.

We consider Markov chains whose step behaviour is definable in terms of relative change, independently of the current state. The relative change in each step is given by a *step distribution* which is a (sub)probability mass function $s : \mathbb{Z} \to \mathbb{R}_{[0,1]}$. We call s *finite* if it has finite support. We interpret the "missing probability", $1 - \sum_{i \in \mathbb{Z}} s(i)$, as failure.

Definition 5.2. Given a step distribution $s : \mathbb{Z} \to \mathbb{R}_{[0,1]}$ we define a stochastic matrix $\mathfrak{P}_s$ on $\mathbb{N}_\bot := \mathbb{N} \cup \{\bot\}$ by:

	$\bot$	0	$m > 0$
$\bot$	1	0	0
0	0	1	0
$n > 0$	$1 - \sum_{i \in \mathbb{Z}} s(i)$	$\sum_{i \le -n} s(i)$	$s(m - n)$

Note that s gives the relative change in each step, the walk is truncated (trapped) at 0, and the probability mass deficit in s (if any) is balanced by the probability of transitioning (from a good state) to the failure state $\bot$. We call s AST if the associated walk reaches state 0 a.s.

Definition 5.3. A step distribution s is called *AST* if for every $m \in \mathbb{N}$, $\lim_{n \to \infty} \mathfrak{P}_s^n(m, 0) = 1$.

Note that the limit in the definition above always exists (and lies between 0 and 1) as the sequence $(\mathfrak{P}_s^n(m, 0))_n$ is monotone *increasing* and bounded. A step distribution can be shown AST by reduction to a one-counter Markov decision process (MDP) (following [20]), for which a.s. termination can be decided in polynomial time [7]. We present a new proof that avoids the detour to MDPs, giving a tighter (in fact optimal) complexity upper bound than that in [20]. The crux lies in a simple, and decidable—if s is finite and rational valued—characterisation of AST which directly gives *linear-time decidability*.

Theorem 5.4. *A finite step distribution s is AST if and only if all of the following hold*

a) $\sum_{i \in \mathbb{Z}} s(i) = 1$ b) $s \ne \delta_0$ c) $\sum_{i \in \mathbb{Z}} i \cdot s(i) \le 0$

Uniform AST. We also model the case where in each time step, a different distribution can be chosen from an available set of step distributions, similar to a MDP [3].

Definition 5.5. A family of step distributions $\{s_i\}_{i \in I}$ is *uniform AST* if for every $m \in \mathbb{N}$

$$\lim_{n \to \infty} \left(\inf_{i_1, \cdots, i_n} \mathfrak{P}_{s_{i_1}} \cdots \mathfrak{P}_{s_{i_n}}(m, 0) \right) = 1$$

Informally it reads that as step count n tends to ∞, no matter which step distribution from $\{s_i\}_{i \in I}$ is chosen at each step, the walk eventually reaches 0 almost surely. Obviously, uniform AST implies AST for each of the s_i but, in general, not conversely. However we can show:

Lemma 5.6. *If $\{s_i\}_{i \in I}$ is a finite family of step distributions and each s_i is AST then $\{s_i\}_{i \in I}$ is uniform AST.*

5.2 Counting-based Extraction of Random Walks

Let's fix a *1st-order* program $\mu_x^\varphi.M$ with *no* nested recursion. To extract the counting pattern of $\mu_x^\varphi.M$, we instrument a counting-based reduction relation $\overset{\star}{\to}$, and use it to analyse a related term $\mathbf{body}_{\mu_x^\varphi.M}(r) := M[\underline{r}/x, \boxed{\mu}/\varphi]$, i.e., the body of the program $\mu_x^\varphi.M$, with x instantiated to a fixed actual argument $\underline{r}$, and a special symbol $\boxed{\mu}$ in place of all recursive calls. The counting-based reduction relation $\overset{\star}{\to}$ is presented in Fig. 5 and acts on configuration of the form $\langle N, s, n \rangle$ where $n \in \mathbb{N}$ counts recursive calls. The main idea is to replace outcomes of recursive calls by a distinguished value $\star$ of type **R** which stands for an unknown numeral. Note that the unknown numeral $\star$ can end up in the guard of a conditional if recursive outcomes affect the control flow of the program. This is, however, unavoidable if we want to count recursive calls (without reference to the program denotation) as the number of function call sites can depend on the (probabilistic) outcome of a prior call. We define

$$\mathbb{T}_{N;n}^\star := \{ s \in \mathbb{S} \mid \exists V : \langle N, s, 0 \rangle \overset{\star}{\to}{}^* \langle V, \epsilon, n \rangle \}$$

the set of traces on which recursive calls from exactly n *distinct* call sites are made. As the reduction relation is deterministic we get that $\{\mathbb{T}_{N;n}^\star\}_{n \in \mathbb{N}}$ are pairwise disjoint. Using similar arguments in [6], it is easy to see that $\mathbb{T}_{N;n}^\star$ is a measurable set of traces.

Definition 5.7. Given a term $\mu_x^\varphi.M$ we define the r-indexed family $\{[\![\mu_x^\varphi.M \mid r]\!] : \mathbb{N} \to \mathbb{R}_{[0,1]}\}_{r \in \mathbb{R}}$, called the *counting pattern of $\mu_x^\varphi.M$*, whereby

$$[\![\mu_x^\varphi.M \mid r]\!](n) := \mu_{\mathbb{S}}\big(\mathbb{T}^\star_{\mathbf{body}_{\mu_x^\varphi.M}(r);n}\big)$$

In words, $[\![\mu_x^\varphi.M \mid r]\!](n)$ gives the probability of *a run of* $\mu_x^\varphi.M$, on the actual argument r, making recursive calls from n distinct call sites. It is straightforward to see that for every r we have $\sum_n [\![\mu_x^\varphi.M \mid r]\!](n) \le 1$, by the same argument as in [40, Lem. 7].

Example 5.8. Consider the term $\mu_x^\varphi.M$ from Ex. 5.1. We get $[\![\mu_x^\varphi.M \mid r]\!](0) = p$, $[\![\mu_x^\varphi.M \mid r]\!](1) = 0$, $[\![\mu_x^\varphi.M \mid r]\!](2) = (1 - p) \cdot \frac{1}{2} \cdot (2 - sig(r))$, $[\![\mu_x^\varphi.M \mid r]\!](3) = (1 - p) \cdot \frac{1}{2} \cdot sig(r)$ and $[\![\mu_x^\varphi.M \mid r]\!](n) = 0$ for all other n.

5.3 Termination via Counting Patterns

Our main result of this section is that we can use the counting pattern of a program to soundly reason about its termination property. For any *counting distribution*, i.e., (sub) pmf $s : \mathbb{N} \to \mathbb{R}_{[0,1]}$, we define the shifted step distribution $\bar{s} : \mathbb{Z} \to \mathbb{R}_{[0,1]}$ by $\bar{s}(z) = s(z + 1)$ for $z \ge -1$ and $\bar{s}(z) = 0$ otherwise.

Theorem 5.9. *If $\{\overline{[\![\mu_x^\varphi.M \mid r]\!]}\}_{r \in \mathbb{R}}$ (qua family of step distributions) is uniform AST then $\mu_x^\varphi.M$ is AST on every actual argument.*

$$\frac{}{\langle (\lambda x.M)V, s, n\rangle \xrightarrow{\star} \langle M[V/x], s, n\rangle} \qquad \frac{}{\langle \boxed{\mu}V, s, n\rangle \xrightarrow{\star} \langle \star, s, n+1\rangle} \qquad \frac{}{\langle \mathtt{sample}, r :: s, n\rangle \xrightarrow{\star} \langle \underline{r}, s, n\rangle}$$

$$\frac{r \leq 0}{\langle \mathtt{if}(\underline{r}, N, P), s, n\rangle \xrightarrow{\star} \langle N, s, n\rangle} \qquad \frac{r > 0}{\langle \mathtt{if}(\underline{r}, N, P), s, n\rangle \xrightarrow{\star} \langle P, s, n\rangle} \qquad \frac{}{\langle f(\underline{r_1}, \cdots, \underline{r_{|f|}}), s, n\rangle \xrightarrow{\star} \langle \underline{f(r_1, \cdots, r_{|f|})}, s, n\rangle}$$

$$\frac{}{\langle f(V_1, \cdots, \star, \cdots, V_{|f|}), s, n\rangle \xrightarrow{\star} \langle \star, s, n\rangle} \qquad \frac{r \geq 0}{\langle \mathtt{score}(\underline{r}), s, n\rangle \xrightarrow{\star} \langle \underline{r}, s, n\rangle} \qquad \frac{\langle R, s, n\rangle \xrightarrow{\star} \langle M, s', n'\rangle}{\langle E[R], s, n\rangle \xrightarrow{\star} \langle E[M], s', n'\rangle}$$

Figure 5. Small-step reduction rules for $\xrightarrow{\star}$.

Proof Sketch. We decompose the set of terminating traces on a fixed argument according to the arguments of recursive calls arranged in a tree. We can lower bound the probability of each partition in terms of $\{\overline{[\![\mu_x^\varphi.M \mid r]\!]}\}_{r \in \mathbb{R}}$ and show that uniform AST implies that the cumulative weight over every decomposed part equals 1, i.e., the program is AST. □

A Partial Order For Counting Distributions. We can equip the set of counting distributions (i.e. (sub)pmfs $s, t :$ $\mathbb{N} \to \mathbb{R}_{[0,1]}$) with a partial order that is compatible with the termination behavior. We define

$$s \sqsubseteq t \Leftrightarrow \forall n \in \mathbb{N}.\ \textstyle\sum_{m \leq n} s(m) \leq \sum_{m \leq n} t(m)$$

i.e., $s \sqsubseteq t$ if the cumulative weight of s is no greater than that of t at any point. It is easy to see that $\sqsubseteq$ is a partial order. Furthermore, we can show compatibility w.r.t. AST (using Thm. 5.4):

Lemma 5.10. *If s, $\{t_i\}_{i \in \mathcal{I}}$ are counting distributions and for all $i \in \mathcal{I}$, $s \sqsubseteq t_i$ and $\overline{s}$ is AST then $\{\overline{t_i}\}_{i \in \mathcal{I}}$ is uniform AST.*

Example 5.11. The counting pattern presented in Ex. 5.8 for the term from Ex. 5.1 satisfies the preconditions of Lem. 5.10 for $s := p\delta_0 + (1-p)\frac{1}{2}\delta_2 + (1-p)\frac{1}{2}\delta_3$ (where δ_i denotes the Dirac-distribution). For $p \geq \frac{3}{5}$, we can deduce that the counting pattern is uniform AST (via Lem. 5.10 and Thm. 5.4) and thus the example is AST on every input (via Thm. 5.9).

5.4 ϵ-Recursion Avoiding Fixpoint Terms

An interesting quantity that arises from analysing Thm. 5.9 is $[\![\mu_x^\varphi.M \mid r]\!]\,(0)$, i.e., the probability of a run of $\mu_x^\varphi.M$ (on argument $\underline{r}$) making no further recursive calls. Let's consider programs where this probability has a positive lower bound.

Definition 5.12. *A recursive program $\mu_x^\varphi.M$ is ϵ-recursion avoiding (ϵ-RA) if for all $r \in \mathbb{R}$, $[\![\mu_x^\varphi.M \mid r]\!]\,(0) \geq \epsilon$.*

Lets assume $\sum_n [\![\mu_x^\varphi.M \mid r]\!]\,(n) = 1$, i.e., the $\xrightarrow{\star}$-reduction is never stuck. In [2] we show how this can be statically ensured via a type system. Note that a program may be ϵ-RA for a positive ϵ, and yet not AST (as evident from Ex. 1.1). To ensure AST, the positive probability ϵ must be "large enough", in relation to the number of recursive calls. We define the *recursive rank* of $\mu_x^\varphi.M$ to be the minimal m such that for

all $n > m$, and r, $[\![\mu_x^\varphi.M \mid r]\!]\,(n) = 0$ (or, equivalently, the maximal number of call sites from which recursive calls are made in a run of $(\mu_x^\varphi.M)\,\underline{r}$, for any r). (In [2], we show that the recursive rank can be upper bounded via a decidable non-idempotent intersection type system.) Now, using Thm. 5.4 combined with Thm. 5.9, we can get an easy corollary:

Corollary 5.13. *If $\mu_x^\varphi.M$ has recursive rank m and is ϵ-RA for some $\epsilon > 0$ that satisfies $m(1 - \epsilon) \leq 1$ then $\mu_x^\varphi.M$ is AST on every argument.*

Example 5.14. The program **(2)** in Ex. 1.1 has recursive rank 2, and is p-RA. So Cor. 5.13 is applicable whenever $2(1 - p) \leq 1 \Leftrightarrow p \geq \frac{1}{2}$. Note that Cor. 5.13 is weaker than Thm. 5.9; for example, Cor. 5.13 on Ex. 5.1 is only applicable for $p \geq \frac{2}{3}$ whereas Thm. 5.9 is applicable for $p \geq \frac{3}{5}$ (Ex. 5.11).

Example 5.15. As a further example, consider yet another variation to our 3d-printing program from Ex. 5.1:

$$\mu_x^\varphi.\mathtt{let}\ e = \mathtt{sample}\ \mathtt{in}\ \mathtt{if}\ e \leq p\ \mathtt{then}\ x\ \mathtt{else}$$
$$\left((\varphi^3(x+1) \oplus_e \varphi^2(x+1)) \oplus_{sig(x)} \varphi^2(x+1)\right)$$

We sample the error value e (the higher e is, the more damaged the print) and accept the print whenever $e \leq p$. If the print is unacceptable, we replace the binary choice in Ex. 5.1 with one that depends on the sampled value of e. In the extended version [2] we show how Thm. 5.9 and Lem. 5.10 can be used to prove this program AST whenever $p \geq \sqrt{7} - 2 \approx 0.646$. As this example illustrates well, termination analysis of terms that use continuous random samples as first-class values can become very intricate. Such examples are not expressible in PHORS [34] or with binary probabilistic choice [20, 32, 50]. Our framework can analyse such examples efficiently, even automatically.

Special Case: Affine Recursion. Every affine-recursive program [20, §4.1] has recursive rank at most 1, so by Cor. 5.13, ϵ-RA for *any* $\epsilon > 0$ implies AST. This can be seen as the functional equivalent of the *zero-one-law for termination* (c.f. [42, Sec. 2.6]). But the real novelty of our result lies in the fact that sophisticated methods are necessary to deal with the case of non-affine recursion. Our proof rules (Thm. 5.9 and Cor. 5.13) give a powerful tool to verify AST

for non-affine programs where the standard zero-one law fails. Similarly to the language studied e.g. in [39], we can use this to design languages that are AST-by-construction. In particular, in any probabilistic programming system we can (safely) add a special fixpoint operator that comes with the guarantee of ϵ-RA for a sufficiently large ϵ, whose size can be determined statically via the recursive rank. This corresponds to a generalization of the stochastic while-loop in [39]. As demonstrated in [39], probabilistic programming languages with this seemingly severe restriction can still describe complex models with arbitrary precision and convergence guarantee, supporting correct inference of (AST) programs.

6 A Proof System For Non-affine Recursion

The framework of Sec. 5 (Thm. 5.9) relies on the *counting pattern*, $\{[\![\mu_x^\varphi.M \mid r]\!]\}_{r \in \mathbb{R}}$, of the program $\mu_x^\varphi.M$. This family can contain uncountably many different counting distributions, making it impractical for analysis. As we saw in Ex. 5.8, for the counting pattern $\{[\![\mu_x^\varphi.M \mid r]\!]\}_{r \in \mathbb{R}}$ of Ex. 5.1, we have $[\![\mu_x^\varphi.M \mid r]\!] \neq [\![\mu_x^\varphi.M \mid r']\!]$ for every $r \neq r'$. So how can we automate analysis so that Thm. 5.9 can be applied without explicitly computing the counting pattern? In this section we use a simple *game-playing* perspective to solve the problem. We show that we can replace probabilistic branching that depends on the actual argument, by nondeterministic branching and thus obtain a sound method to apply Thm. 5.9. As a rule of thumb, our system can verify all programs that exhibit an AST counting pattern which is independent of the (exact values of the) actual arguments (of the recursive function in question). In this section we give an overview of our approach, and direct readers to [2] for a full account, including more complex examples.

6.1 Stochastic symbolic Execution

The first idea we use for our system is *stochastic symbolic execution*. Instead of executing a program on a fixed trace (as done in $\xrightarrow{\star}$, Fig. 5) we evaluate on a trace of *sample variables* $(\alpha_0, \alpha_1, \cdots)$ whose values can be instantiated later. We organise execution in the form a binary tree where each branching represents a conditional which is annotated with the value on which control flow branches. We also record score-statements as well as recursive calls. We now replace the actual argument with an unknown value $\circledast$ (corresponding to the analysis of $\mathbf{body}_{\mu_x^\varphi.M}(\circledast)$ in Sec. 5.2). In Fig. 6a the execution tree that corresponds to the running example, Ex. 5.1, is depicted, we have coloured each branching that relies on the concrete argument $\circledast$ in red.

6.2 Strategies on Trees

As some of the branching (coloured red) depends on the actual argument, we cannot analyse it probabilistically. This can be overcome by an intuitive 2-player game reading of the execution tree: for every such node the Environment (player)

Figure 6. Symbolic execution trees for the running example and all possible strategies **(b)**.

can resolve its branching by an explicit strategy that indicates a left/right choice for each coloured node. For example, all possible strategies for the tree in Fig. 6a are depicted in Fig. 6b. As a strategy no longer relies on branching at nodes that contain $\circledast$, we can recover a probabilistic interpretation of paths, which is just the Lebesgue-measure of the possible assignment to the sample variables $\alpha_0, \alpha_1, \cdots$, such that a path is indeed followed. Given a strategy $\mathfrak{S}$, we denote with $\mathbb{P}(\mathfrak{S}, n)$ the probability of taking a path such that *at most n* recursive calls are made (i.e., a fixpoint node is traversed at most n times). Depending on the set of primitive functions, this value can be computed effectively. We now define the counting distribution $\mathbb{P}_{approx}$ by

$$\mathbb{P}_{approx}(0) \coloneqq \min_{\mathfrak{S} \in Strat(\mathfrak{T})} \mathbb{P}(\mathfrak{S}, 0)$$

$$\mathbb{P}_{approx}(n > 0) \coloneqq \left(\min_{\mathfrak{S} \in Strat(\mathfrak{T})} \mathbb{P}(\mathfrak{S}, n) \right) - \left(\min_{\mathfrak{S} \in Strat(\mathfrak{T})} \mathbb{P}(\mathfrak{S}, n-1) \right)$$

We can understand $\mathbb{P}_{approx}(n)$ as the least probability that n calls are made even if the Environment chooses in the worst (meaning maximal no. of recursive calls) possible way.

Example 6.1. Consider all strategies for the running example listed in Fig. 6b. We can compute $\mathbb{P}_{approx}(0) = p$; $\mathbb{P}_{approx}(2) = \mathbb{P}_{approx}(3) = (1 - p) \cdot \frac{1}{2}$; and $\mathbb{P}_{approx}(n) = 0$ for all other n.

We can show that replacing probabilistic branching with nondeterministic one gives a lower bound (w.r.t. to the order $\sqsubseteq$) on the counting pattern.

Theorem 6.2. *For every* $r \in \mathbb{R}$, $\mathbb{P}_{approx} \sqsubseteq [\![\mu_x^\varphi.M \mid r]\!]$

Thus, if $\overline{\mathbb{P}_{approx}}$ is AST (which is checkable via Thm. 5.4) we get that $\{\overline{[\![\mu_x^\varphi.M \mid r]\!]}\}_{r \in \mathbb{R}}$ is uniform AST; and via Thm. 5.9 $\mu_x^\varphi.M$ is AST on every actual argument. Thm. 6.2 and the values computed in Ex. 6.1 allow us to deduce (*automatically*) that Ex. 5.1 is AST on every argument if $p \geq \frac{3}{5}$. Similarly, our tool can verify AST of Ex. 5.15 if $p \geq \sqrt{7} - 2$. For a formal description of our algorithm, correctness proof and further examples see [2].

7 Implementation

We provide prototype implementations for computing lower bound of the termination probability, and of AST verification, building on Sec. 3 and Sec. 6 respectively. See [2] for a fuller account.

7.1 Lower Bounds of Termination Probability

We have implemented a prototype lower bound computation tool that exploits the completeness of the interval-traces semantics by iterative search for terminating interval traces. Even in its current, unoptimized, form our tool is able to compute meaningful lower bounds in a reasonable time. A full evaluation can be found in [2].

7.2 AST Verification

The challenge in implementing the ideas from Sec. 6 lies in the computation of branching probabilities. We adopt a geometric interpretation of probability and make use of various results and implementation techniques for volume computation. For simplicity we restrict primitive operations to addition, and multiplication by a constant (and thus subtraction), as the probability of branching can be seen as the volume of a convex polytope [21] (a subset of $\mathbb{R}^d$ of the from $\{\vec{x} \mid A\vec{x} \leq b\}$). We make use of the analytic formula for this volume in [37] and its subsequent implementation in [11], and appeal to Thm. 5.4. Our prototype implementation can verify many examples, including those from Sec. 1.1, Sec. 5, and Sec. 6, thereby illustrating that our approach is well-suited to implementation.

8 Related Work and Conclusion

Our interval-traces approach can be seen as a probabilistic interpretation of interval analysis, a standard approach to infer bounds on program variables [12, 46]. The attractive feature of intervals in our work is its completeness w.r.t. the Lebesgue measure for a broad range of primitive operations.

The only comparable lower bound computation we are aware of is presented in [34]. Kobayashi et al. show that the termination probability of CbN order-n probabilistic recursion schemes (n-PHORS) can be obtained as the least fixpoint of suitable order-$(n-1)$ fixpoint equations, which can be solved using standard Kleene fixpoint iteration. By contrast, our approach works on programs directly, and can handle continuous distributions. It is worth noting that (order-n) PHORS is readily encodable as (order-n) CbN SPCF, but the former is strictly less expressive (because the underlying recursion schemes are not Turing complete). Some interesting SPCF terms such as Ex. 5.15 cannot be expressed as PHORS. Since recursive Markov chains [24] (equivalently, probabilistic pushdown automata [8]) are essentially equivalent to 1-PHORS [34], it follows that order-1 SPCF (which contains the term in Ex. 5.15) is strictly more expressive than recursive Markov chains.

Our intersection type system is inspired by, and builds upon, the ideas of [9, 23]. However, unlike [9], we cannot prove correctness directly (because of continuous samples), rather we need to appeal to the completeness of our interval-based semantics. The step annotation of types enables us to reason about expected termination time. We conjecture these ideas to also be applicable to the system of [9].

Our AST verification method is closely related to [50], in that they also study recursive programs and allow for non-affine behaviour. Our work differs nonetheless in several key aspects: While they study an imperative language with discrete distributions, we work with a purely functional language with continuous distributions. Though their proposed rules can produce lower bounds on the probability of termination, they seem cumbersome to use. Their rule informally reads: if, for all n, we assume that each recursive call terminates with probability l_n after n fixpoint unfoldings, and we can prove that it terminates with probability at least l_{n+1} after $n+1$ unfoldings, then the program terminates with probability at least $\sup_n l_n$ (c.f. [50, Thm. 4.2]). In order to apply this rule, the user must manually find an explicit (and often non-trivial) sequence $(l_n)_{n \in \mathbb{N}}$. By contrast, our system provides a sound reduction to a random walk which can be analysed efficiently in linear time.

As already mentioned, our method can be seen as orthogonal to that in [20]. It is not at all obvious if their techniques can be extended to our setting with sampling from the uniform distribution. An interesting future direction is to develop a unified framework that analyses both the size-related information of the recursive function argument and the number of recursion call sites.

Conclusion

Recent advances in probabilistic programming systems and allied areas (such as [40]) provide strong impetus for the study of AST of programs with continuous distribution. We have presented a first comprehensive study of the lower bound problem, and ascertained the recursion-theoretic complexity of several termination problems. We have introduced a novel proof system for AST verification of non-affine programs which is easily implementable. While some of the existing AST proof methods support continuous distributions [16, 18, 25] the majority do not. It would be interesting to investigate if they [20, 31, 32, 34, 42, 50] can be so extended.

References

[1] Sheshansh Agrawal, Krishnendu Chatterjee, and Petr Novotný. 2018. Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. *Proc. ACM Program. Lang.* 2, POPL (2018), 34:1–34:32.

[2] Anonymous Author(s). 2020. On Probabilistic Termination of Functional Programs with Continuous Distributions. (2020). Full and extended version of this submission.

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

[4] Hendrik Pieter Barendregt. 1985. *The lambda calculus - its syntax and semantics*. Studies in logic and the foundations of mathematics, Vol. 103. North-Holland.

[5] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. 2019. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research* 20, 28 (2019), 1–6.

[6] Johannes Borgström, Ugo Dal Lago, Andrew D. Gordon, and Marcin Szymczak. 2016. A lambda-calculus foundation for universal probabilistic programming. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016*. 33–46.

[7] Tomás Brázdil, Václav Brozek, Kousha Etessami, Antonín Kucera, and Dominik Wojtczak. 2010. One-Counter Markov Decision Processes. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, January 17-19, 2010*. 863–874.

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

[9] Flavien Breuvart and Ugo Dal Lago. 2018. On Intersection Types and Probabilistic Lambda Calculi. In *Proceedings of the 20th International Symposium on Principles and Practice of Declarative Programming, PPDP 2018, Frankfurt am Main, Germany, September 03-05, 2018*. 8:1–8:13.

[10] Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. 2017. Non-idempotent intersection types for the Lambda-Calculus. *Log. J. IGPL* 25, 4 (2017), 431–464.

[11] Benno Büeler, Andreas Enge, and Komei Fukuda. 2000. *Exact Volume Computation for Polytopes: A Practical Study*. Birkhäuser Basel, 131–154.

[12] Michael G. Burke. 1990. An Interval-Based Approach to Exhaustive and Incremental Interprocedural Data-Flow Analysis. *ACM Trans. Program. Lang. Syst.* 12, 3 (1990), 341–395. https://doi.org/10.1145/78969.78963

[13] Bob Carpenter, Andrew Gelman, Matthew D Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. Stan: A probabilistic programming language. *Journal of statistical software* 76, 1 (2017).

[14] Aleksandar Chakarov and Sriram Sankaranarayanan. 2013. Probabilistic Program Analysis with Martingales. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*. 511–526.

[15] Krishnendu Chatterjee, Hongfei Fu, and Amir Kafshdar Goharshady. 2016. Termination Analysis of Probabilistic Programs Through Positivstellensatz's. In *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I*. 3–22.

[16] Krishnendu Chatterjee, Hongfei Fu, Petr Novotný, and Rouzbeh Hasheminezhad. 2018. Algorithmic Analysis of Qualitative and Quantitative Termination Problems for Affine Probabilistic Programs. *ACM Trans. Program. Lang. Syst.* 40, 2 (2018), 7:1–7:45.

[17] Krishnendu Chatterjee, Petr Novotný, and Dorde Zikelic. 2017. Stochastic invariants for probabilistic termination. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*. 145–160.

[18] Jianhui Chen and Fei He. 2020. Proving almost-sure termination by omega-regular decomposition. In *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*. 869–882.

[19] Marco F. Cusumano-Towner, Feras A. Saad, Alexander K. Lew, and Vikash K. Mansinghka. 2019. Gen: A General-Purpose Probabilistic Programming System with Programmable Inference. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2019)*. Association for Computing Machinery, New York, NY, USA, 221–236. https://doi.org/10.1145/3314221.3314642

[20] Ugo Dal Lago and Charles Grellois. 2019. Probabilistic Termination by Monadic Affine Sized Typing. *ACM Trans. Program. Lang. Syst.* 41, 2 (2019), 10:1–10:65.

[21] Martin E. Dyer and Alan M. Frieze. 1988. On the Complexity of Computing the Volume of a Polyhedron. *SIAM J. Comput.* 17, 5 (1988), 967–974. https://doi.org/10.1137/0217060

[22] Thomas Ehrhard, Michele Pagani, and Christine Tasson. 2018. Full Abstraction for Probabilistic PCF. *J. ACM* 65, 4 (2018), 23:1–23:44. https://doi.org/10.1145/3164540

[23] Thomas Ehrhard, Christine Tasson, and Michele Pagani. 2014. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*. 309–320.

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

[25] Luis María Ferrer Fioriti and Holger Hermanns. 2015. Probabilistic Termination: Soundness, Completeness, and Compositionality. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*. 489–501.

[26] Hong Ge, Kai Xu, and Zoubin Ghahramani. 2018. Turing: A Language for Flexible Probabilistic Inference. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1682–1690.

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

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

[29] Mingzhang Huang, Hongfei Fu, and Krishnendu Chatterjee. 2018. New Approaches for Almost-Sure Termination of Probabilistic Programs. In *Programming Languages and Systems - 16th Asian Symposium, APLAS 2018, Wellington, New Zealand, December 2-6, 2018, Proceedings*. 181–201.

[30] Mingzhang Huang, Hongfei Fu, Krishnendu Chatterjee, and Amir Kafshdar Goharshady. 2019. Modular verification for almost-sure termination of probabilistic programs. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 129:1–129:29.

[31] Benjamin Lucien Kaminski and Joost-Pieter Katoen. 2015. On the Hardness of Almost-Sure Termination. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part I*. 307–318.

[32] Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. 2018. Weakest Precondition Reasoning for Expected Runtimes of Randomized Algorithms. *J. ACM* 65, 5 (2018), 30:1–30:68.

[33] S. C. Kleene. 1955. Hierarchies of number-theoretic predicates. *Bull. Amer. Math. Soc.* 61 (05 1955), 193–213.

[34] Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. 2019. On the Termination Problem for Probabilistic Higher-Order Recursive Programs. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. 1–14.

[35] Dexter Kozen. 1981. Semantics of Probabilistic Programs. *J. Comput. Syst. Sci.* 22, 3 (1981), 328–350.

[36] Ugo Dal Lago and Margherita Zorzi. 2012. Probabilistic operational semantics for the lambda calculus. *RAIRO Theor. Informatics Appl.* 46, 3 (2012), 413–450. https://doi.org/10.1051/ita/2012012

[37] Jean B Lasserre. 1983. An analytical expression and an algorithm for the volume of a convex polyhedron in $\mathbb{R}^n$. *Journal of optimization theory and applications* 39, 3 (1983), 363–377.

[38] Wonyeol Lee, Hangyeol Yu, and Hongseok Yang. 2018. Reparameterization Gradient for Non-differentiable Models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.* 5558–5568.

[39] Alexander K. Lew, Marco F. Cusumano-Towner, Benjamin Sherman, Michael Carbin, and Vikash K. Mansinghka. 2020. Trace types and denotational semantics for sound programmable inference in probabilistic languages. *Proc. ACM Program. Lang.* 4, POPL (2020), 19:1–19:32.

[40] Carol Mak, C.-H. Luke Ong, Hugo Paquet, and Dominik Wagner. 2020. Densities of almost-surely terminating probabilistic programs are differentiable almost everywhere. *CoRR* abs/2004.03924 (2020). arXiv:2004.03924 https://arxiv.org/abs/2004.03924

[41] Vikash K. Mansinghka, Daniel Selsam, and Yura N. Perov. 2014. Venture: a higher-order probabilistic programming platform with programmable inference. *CoRR* abs/1404.0099 (2014). arXiv:1404.0099 http://arxiv.org/abs/1404.0099

[42] Annabelle McIver and Carroll Morgan. 2005. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer.

[43] Annabelle McIver, Carroll Morgan, Benjamin Lucien Kaminski, and Joost-Pieter Katoen. 2018. A new proof rule for almost-sure termination. *Proc. ACM Program. Lang.* 2, POPL (2018), 33:1–33:28.

[44] Sean Meyn and Richard L. Tweedie. 2009. *Markov Chains and Stochastic Stability* (2nd ed.). Cambridge University Press.

[45] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press. https://doi.org/10.1017/CBO9780511813603

[46] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. 2009. *Introduction to Interval Analysis*. SIAM. https://doi.org/10.1137/1.9780898717716

[47] R. Motwani and P. Raghavan. 1995. *Randomized algorithms*. Cambridge University Press.

[48] Akihiko Nishimura, David B Dunson, and Jianfeng Lu. 2020. Discontinuous Hamiltonian Monte Carlo for discrete parameters and discontinuous likelihoods. *Biometrika* 107, 2 (2020), 365–380.

[49] Jonathan Novak. 2014. Pólya's Random Walk Theorem. *Am. Math. Mon.* 121, 8 (2014), 711–716.

[50] Federico Olmedo, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. 2016. Reasoning about Recursive Probabilistic Programs. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016.* 672–681.

[51] Gordon D. Plotkin. 1977. LCF Considered as a Programming Language. *Theor. Comput. Sci.* 5, 3 (1977), 223–255.

[52] M. O. Rabin. 1976. Probabilistic algorithms. In *Algorithms and complexity: new directions and results*. Academic Press, New York, 21–39.

[53] Tom Rainforth. 2017. *Automating Inference, Learning, and Design Using Probabilistic Programming*. http://purl.org/dc/dcmitype/Text. University of Oxford.

[54] Tom Rainforth. 2017. *Automating Inference, Learning, and Design using Probabilistic Programming*. Ph.D. Dissertation.

[55] Reuven Y. Rubinstein and Dirk P. Kroese. 2017. *Simulation and the Monte Carlo Method* (3rd ed.). Wiley.

[56] Eugene S. Santos. 1969. Probabilistic Turing Machines and Computability. *Proc. Amer. Math. Soc.* 22, 3 (1969), 704–710. http://www.jstor.org/stable/2037463

[57] David Tolpin, Jan-Willem van de Meent, and Frank D. Wood. 2015. Probabilistic Programming in Anglican. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III.* 308–311.

[58] Dustin Tran, Alp Kucukelbir, Adji B Dieng, Maja Rudolph, Dawen Liang, and David M Blei. 2016. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787* (2016).

[59] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. 2018. An Introduction to Probabilistic Programming. *arXiv:1809.10756 [cs, stat]* (Sept. 2018). arXiv:1809.10756 [cs, stat]

[60] Yuan Zhou, Bradley J. Gram-Hansen, Tobias Kohn, Tom Rainforth, Hongseok Yang, and Frank Wood. 2019. LF-PPL: A Low-Level First Order Probabilistic Programming Language for Non-Differentiable Models. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan.* 148–157.

A Additional Material - Section 2

A.1 Typing Rules for SPCF

The full typing system of SPCF is given in Fig. 7.

A.2 Additional Proofs

Restatement of Lem. 2.3. *If M is AST then*

$$\mathbb{E}_{term}(M) = \sum_{n=0}^{\infty} \mu_{\mathbb{S}}\left(\mathbb{T}^n_{M,term}\right) \cdot n$$

Proof. We have $\biguplus_n \mathbb{T}^n_{M,\text{term}} = \mathbb{T}_{M,\text{term}}$. Now define

$$\mathbb{T}^{>n}_{M,\text{term}} \triangleq \biguplus_{i>n} \mathbb{T}^i_{M,\text{term}}$$

It is easy to see that $\mu_{\mathbb{S}}\left(\mathbb{T}^{>n}_{M,\text{term}}\right) + \mu_{\mathbb{S}}\left(\mathbb{T}^{\leq n}_{M,\text{term}}\right) = 1$ as by assumption $\mu_{\mathbb{S}}\left(\mathbb{T}_{M,\text{term}}\right) = 1$. Now

$$\mathbb{E}_{\text{term}}(M) = \sum_{n=0}^{\infty} \left(1 - \mu_{\mathbb{S}}\left(\mathbb{T}^{\leq n}_{M,\text{term}}\right)\right)$$

$$\overset{(1)}{=} \sum_{n=0}^{\infty} \mu_{\mathbb{S}}\left(\mathbb{T}^{>n}_{M,\text{term}}\right)$$

$$\overset{(2)}{=} \sum_{n=0}^{\infty} \sum_{j>n} \mu_{\mathbb{S}}\left(\mathbb{T}^i_{M,\text{term}}\right)$$

$$\overset{(3)}{=} \sum_{n=0}^{\infty} \mu_{\mathbb{S}}\left(\mathbb{T}^n_{M,\text{term}}\right) \cdot n$$

where **(1)** holds as $\mu_{\mathbb{S}}\left(\mathbb{T}^{>n}_{M,\text{term}}\right) + \mu_{\mathbb{S}}\left(\mathbb{T}^{\leq n}_{M,\text{term}}\right) = 1$, **(2)** follows as the union in the definition of $\mathbb{T}^{>n}_{M,\text{term}}$ is disjoint and **(3)** is an easy combinatorial argument. □

Lemma A.1 (PAST implies AST). *If M is PAST then M is AST*

Proof. Assume M is PAST so by definition

$$\sum_{n=0}^{\infty} \left(1 - \mu_{\mathbb{S}}\left(\mathbb{T}^{\leq n}_{M,\text{term}}\right)\right)$$

is a finite sum. As this sum converges to a finite value the sequence $\left(\mu_{\mathbb{S}}\left(\mathbb{T}^{\leq n}_{M,\text{term}}\right)\right)_{n\in\mathbb{N}}$ must converge to 1. And as $\mathbb{T}^{\leq n}_{M,\text{term}} \subseteq \mathbb{T}_{M,\text{term}}$ for every n and $\mu_{\mathbb{S}}$ is a measure (in particular monotone w.r.t. to $\subseteq$ and $\leq$) we get $\mu_{\mathbb{S}}\left(\mathbb{T}_{M,\text{term}}\right) = 1$, so M is AST. □

A.3 Call by Value

We now introduce a Call by Value evaluation strategy for SPCF. We define call by value redexes and evaluation contexts by:

$$R := (\lambda x.M)V \mid (\mu^{\varphi}_x.M)V \mid \text{if}(\underline{r}, N, P)$$
$$\mid f(\underline{r_1}, \cdots, \underline{r_{|f|}}) \mid \text{sample} \mid \text{score}(\underline{r})$$

$$E := [\cdot] \mid EM \mid (\lambda y.M)E \mid (\mu^{\varphi}_x.M)E \mid \text{if}(E, N, P)$$
$$\mid f(\underline{r_1}, \cdots, \underline{r_{k-1}}, E, M_{k+1}, \cdots, M_{|f|}) \mid \text{score}(E)$$

Note that for a β-redex to reduce, the argument must be a value and we conversely reduce the left hand side of applications. We define the CbV reduction relation by the rules in Fig. 8. The definitions in Sec. 2.4 regarding AST and PAST extend naturally to CbV[7]. Throughout this paper we always make clear what evaluation strategy we are using, so the notation never clashes.

B Additional Material - Section 3

B.1 Interval-Based Semantics

Restatement of Lem. 3.2. *If $f : \mathbb{R}^n \to \mathbb{R}$ is continuous then f is interval preserving*

Proof. Let $A \triangleq [a_1, b_1] \times \cdots \times [a_n, b_n]$ as in the definition of interval perseverance. We need a higher-dimensional version of the intermediate value theorem (IVT): if $\mathrm{x}, \mathrm{y} \in A$ and $c \in \mathbb{R}$ be such that $f(\mathrm{x}) \leq c \leq f(\mathrm{y})$ then there is a $\mathrm{z} \in A$ such that $f(\mathrm{z}) = c$ **(1)**. The IVT implies that $f(A)$ is a connected set. A standard property of continuous functions is that the images of compact sets are compact sets. Due to the Heine–Borel theorem, compact euclidean sets are exactly those that are bounded and closed. As A is obviously compact, we get that $f(A)$ is compact and thus bounded and closed. As $f(A)$ is also connected (by the IVT), it is a closed (bounded) interval as required.

It remains to show **(1)**: Let $\gamma : [0, 1] \to A$ be defined by $\gamma(t) \triangleq t\mathrm{x} + (1 - t)\mathrm{y}$ which is obviously continuous. In particular, note that since A is a box (and thus convex) $\gamma(t) \in A$ for every $t \in [0, 1]$. Now define $\phi : [0, 1] \to \mathbb{R}$ by $\phi \triangleq f \circ \gamma$ which is the composition of continuous functions and thus also continuous. Now $\phi(0) = f(\mathrm{x}) \leq c \leq f(\mathrm{y}) = \phi(1)$ so by the intermediate value theorem in the 1d case there exists a $t \in [0, 1]$ with $\phi(t) = c$. We can define $\mathrm{z} \triangleq \gamma(t)$ which satisfies the requirement by definition of ϕ. □

B.2 Interval-Based Reduction

The full (CbN) reduction system for interval terms is given in Fig. 9.

B.3 Soundness

This subsection is devoted to give a full proof of Thm. 3.4.

Assume (Ω, Σ_Ω) is a measurable space and μ is a measure on (Ω, Σ_Ω). $A, B \in \Sigma_\Omega$ are called almost-disjoint if $\mu(A \cap B) =$

---

[7]While the concepts extend naturally, they obviously are not identical. E.g., the probability of a termination in CbV may very well differ from the one in CbN.

$$\frac{x : \alpha \in \Gamma}{\Gamma \vdash x : \alpha} \qquad \frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \to \beta} \qquad \frac{\Gamma, \varphi : \alpha \to \beta, x : \alpha \vdash M : \beta}{\Gamma \vdash \mu_x^\varphi.M : \alpha \to \beta} \qquad \frac{}{\Gamma \vdash \underline{r} : \mathbf{R}}$$

$$\frac{\Gamma \vdash M : \beta \to \alpha \qquad \Gamma \vdash N : \beta}{\Gamma \vdash MN : \alpha} \qquad \frac{\Gamma \vdash M : \mathbf{R} \qquad \Gamma \vdash N : \alpha \qquad \Gamma \vdash P : \alpha}{\Gamma \vdash \mathtt{if}(M, N, P) : \alpha}$$

$$\frac{}{\Gamma \vdash \mathtt{sample} : \mathbf{R}} \qquad \frac{\Gamma \vdash M_1 : \mathbf{R} \qquad \cdots \qquad \Gamma \vdash M_{|f|} : \mathbf{R}}{\Gamma \vdash f(M_1, \cdots, M_{|f|}) : \mathbf{R}} \qquad \frac{\Gamma \vdash M : \mathbf{R}}{\Gamma \vdash \mathtt{score}(M) : \mathbf{R}}$$

Figure 7. Full SPCF Typing rules

$$\frac{}{\langle (\lambda x.M)V, s \rangle \xrightarrow{\mathfrak{B}} \langle M[V/x], s \rangle} \qquad \frac{}{\langle (\mu_x^\varphi.M)V, s \rangle \xrightarrow{\mathfrak{B}} \langle M[V/x, (\mu_x^\varphi.M)/\varphi], s \rangle}$$

$$\frac{r \le 0}{\langle \mathtt{if}(\underline{r}, N, P), s \rangle \xrightarrow{\mathfrak{B}} \langle N, s \rangle} \qquad \frac{r > 0}{\langle \mathtt{if}(\underline{r}, N, P), s \rangle \xrightarrow{\mathfrak{B}} \langle P, s \rangle} \qquad \frac{}{\langle \mathtt{sample}, r :: s \rangle \xrightarrow{\mathfrak{B}} \langle \underline{r}, s \rangle}$$

$$\frac{}{\langle f(\underline{r_1}, \cdots, \underline{r_{|f|}}), s \rangle \xrightarrow{\mathfrak{B}} \langle \underline{f(r_1, \cdots, r_{|f|})}, s \rangle} \qquad \frac{r \ge 0}{\langle \mathtt{score}(\underline{r}), s \rangle \xrightarrow{\mathfrak{B}} \langle \underline{r}, s \rangle} \qquad \frac{\langle R, s \rangle \xrightarrow{\mathfrak{B}} \langle M, s' \rangle}{\langle E[R], s \rangle \xrightarrow{\mathfrak{B}} \langle E[M], s' \rangle}$$

Figure 8. Call by Value small-step reduction $\xrightarrow{\mathfrak{B}}$ for SPCF. If it is clear from the context that we work in a CbV strategy we drop the annotation and simply write $\to$.

$$\frac{}{\langle (\lambda x.\mathcal{M})\mathcal{N}, \wp \rangle \rightsquigarrow \langle \mathcal{M}[\mathcal{N}/x], \wp \rangle} \qquad \frac{}{\langle (\mu_x^\varphi.\mathcal{M})\mathcal{N}, \wp \rangle \rightsquigarrow \langle \mathcal{M}[\mathcal{N}/x, (\mu_x^\varphi.\mathcal{M})/\varphi], \wp \rangle}$$

$$\frac{b \le 0}{\langle \mathtt{if}(\underline{[a, b]}, \mathcal{N}, \mathcal{P}), \wp \rangle \rightsquigarrow \langle \mathcal{N}, \wp \rangle} \qquad \frac{a > 0}{\langle \mathtt{if}(\underline{[a, b]}, \mathcal{N}, \mathcal{P}), \wp \rangle \rightsquigarrow \langle \mathcal{P}, \wp \rangle}$$

$$\frac{}{\langle \mathtt{sample}, [a, b] :: \wp \rangle \rightsquigarrow \langle \underline{[a, b]}, \wp \rangle} \qquad \frac{0 \le a}{\langle \mathtt{score}(\underline{[a, b]}), \wp \rangle \rightsquigarrow \langle \underline{[a, b]}, \wp \rangle}$$

$$\frac{}{\langle f(\underline{[a_1, b_1]}, \cdots, \underline{[a_{|f|}, b_{|f|}]}), \wp \rangle \rightsquigarrow \langle \underline{\hat{f}(a_1, b_1, \cdots, a_{|f|}, b_{|f|})}, \wp \rangle} \qquad \frac{\langle \mathcal{R}, \wp \rangle \rightsquigarrow \langle \mathcal{M}, \wp' \rangle}{\langle \mathcal{E}[\mathcal{R}], \wp \rangle \rightsquigarrow \langle \mathcal{E}[\mathcal{M}], \wp' \rangle}$$

Figure 9. Internal-based (CbN) small-step reduction.

0. In the case of $\Omega = \mathbb{R}$ we get that intervals $[a, b]$ and $[c, d]$ are almost disjoint iff $b \le c$ or $d \le a$.

Embedding and Refinement. To state soundness it is fruitful to investigate the embedding of standard terms in interval terms $(\cdot^{2\Im})$. We define a relation $M \triangleleft \mathcal{M}$ in Fig. 10 which models the intuitive idea of viewing every interval numeral $\underline{[a, b]}$ as any value within $[a, b]$. Then $M \triangleleft \mathcal{M}$ is derivable if and only if M and $\mathcal{M}$ agree structurally and every standard numeral in M is contained in the repressive interval numeral in $\mathcal{M}$. We can see that the canonical embedding

is compatible with this refinement, i.e., for every standard term M, $M \triangleleft M^{2\Im}$. We can also define a refinement between standard traces and interval by

$$r_0 \cdots r_{n-1} \triangleleft [a_0, b_0] \cdots [a_{n-1}, b_{n-1}] \Leftrightarrow \forall i : r_i \in [a_i, b_i]$$

For an interval trace $\wp$ we define $(\!| \wp |\!) := \{s \mid s \triangleleft \wp\}$, i.e., the set of all traces refining $\wp$.

Lemma B.1. *If $\langle \mathcal{M}, \wp \rangle \rightsquigarrow^n \langle \mathcal{N}, \wp' \rangle$ and $M \triangleleft \mathcal{M}$ and $s \triangleleft \wp$ then there exists a $N \triangleleft \mathcal{N}$ and $s' \triangleleft \wp'$ such that $\langle M, s \rangle \to^n \langle N, s' \rangle$.*

$$\frac{}{x \triangleleft x} \qquad \frac{}{\texttt{sample} \triangleleft \texttt{sample}} \qquad \frac{r \in [a,b]}{r \triangleleft [a,b]} \qquad \frac{M \triangleleft \mathcal{M}}{\lambda x.M \triangleleft \lambda x.\mathcal{M}}$$

$$\frac{M \triangleleft \mathcal{M} \qquad N \triangleleft \mathcal{N}}{MN \triangleleft \mathcal{M}\mathcal{N}} \qquad \frac{M \triangleleft \mathcal{M}}{\mu_x^\varphi.M \triangleleft \mu_x^\varphi.\mathcal{M}} \qquad \frac{M \triangleleft \mathcal{M}}{\texttt{score}(M) \triangleleft \texttt{score}(\mathcal{M})}$$

$$\frac{M \triangleleft \mathcal{M} \qquad N \triangleleft \mathcal{N} \qquad P \triangleleft \mathcal{P}}{\texttt{if}(M,N,P) \triangleleft \texttt{if}(\mathcal{M},\mathcal{N},\mathcal{P})} \qquad \frac{M_1 \triangleleft \mathcal{M}_1 \qquad \cdots \qquad M_{|f|} \triangleleft \mathcal{M}_{|f|}}{f(M_1,\cdots,M_{|f|}) \triangleleft f(\mathcal{M}_1,\cdots,\mathcal{M}_{|f|})}$$

Figure 10. Inductive definition of the refinement relation $\triangleleft$ between the set of well-typed terms Λ and the set of well-typed interval terms $\Lambda_{\mathfrak{I}}$.

Proof. We first observe the following obvious result: If $M \triangleleft \mathcal{M}$ and $N_i \triangleleft \mathcal{N}_i$ for $i \in [n]$ then $M[N_i/x_i]_{i \in [n]} \triangleleft \mathcal{M}[\mathcal{N}_i/x_i]_{i \in [n]}$ which can be proved by induction on M (or $\mathcal{M}$). Call this observation **(1)**. We now show the statement for $n = 1$. The case for $n = 0$ is trivial and for $n > 1$ follows by a simple induction. We do structural induction on $\mathcal{M}$.

- If $\mathcal{M} = (\lambda x.\mathcal{P})\mathcal{Q}$: then $\mathcal{N} = \mathcal{P}[\mathcal{Q}/x]$ and as $M \triangleleft \mathcal{M}$, $M = (\lambda x.P)V$ for some $P \triangleleft \mathcal{P}$, $Q \triangleleft \mathcal{Q}$ and $\varphi' = \varphi$. Define $s' \triangleq s$ and $N \triangleq P[Q/x]$. Clearly $\langle M, s \rangle \rightarrow \langle N, s' \rangle$. Now $s' \triangleleft \varphi'$ is obvious and from **(1)** we also get $N \triangleleft \mathcal{N}$.
- If $\mathcal{M} = (\mu_x^\varphi.\mathcal{P})\mathcal{Q}$: similar to the previous case.
- $\mathcal{M} = f(\underline{[a_1,b_1]}, \cdots, \underline{[a_{|f|}, b_{|f|}]})$. Then

$$\mathcal{N} = \hat{f}(a_1, b_1, \cdots, a_{|f|}, b_{|f|})$$

  As $M \triangleleft \mathcal{M}$ we get $M = f(r_1, \cdots, r_{|f|})$ and $r_i \in [a_i, b_i]$. Define $s' \triangleq s$ and $N \triangleq \underline{f(r_1, \cdots, r_n)}$. Clearly $\langle M, s \rangle \rightarrow \langle N, s' \rangle$. As f is interval preserving we also get that $f(r_1, \cdots, r_{|f|}) \in \hat{f}(a_1, b_1, \cdots, a_{|f|}, b_{|f|})$ so $N \triangleleft \mathcal{N}$.
- $\mathcal{M} = \texttt{if}(\underline{[a,b]}, \mathcal{P}, \mathcal{Q})$. Assume $b \leq 0$, the case where $a > 0$ is analogous. So $\mathcal{N} = \mathcal{P}$. As $M \triangleleft \mathcal{M}$, $M = \texttt{if}(r, P, Q)$ for some $r \in [a,b]$ and $P \triangleleft \mathcal{P}$. So we can choose $N = P$.
- $\mathcal{M} = \texttt{sample}$ so $\varphi = [a,b] :: \varphi'$ and $\mathcal{N} = [a,b]$. Now $M = \texttt{sample}$ and as $s \triangleleft \varphi$, $s = r :: s'$ for $r \in [a,b]$. Now define $N = \underline{r}$.
- $\mathcal{M} = \texttt{score}(\underline{[a,b]})$: So $M = \texttt{score}(\underline{r})$ and $r \in [a,b]$. So $N = \underline{r}$.
- $\mathcal{M} = \mathcal{E}[\mathcal{R}]$ for $\mathcal{E} \neq [\cdot]$: Then $\langle \mathcal{R}, \varphi \rangle \rightarrow \langle \mathcal{M}, \varphi' \rangle$. As $M \triangleleft \mathcal{M}$ we have $M = E[R]$ for $R \triangleleft \mathcal{R}$. Now by induction on $\mathcal{R}$ we get a M with $M \triangleleft \mathcal{M}$ and $s' \triangleleft \varphi'$ such that $\langle R, s \rangle \rightarrow \langle M, s' \rangle$. Now $E[M] \triangleleft \mathcal{E}[\mathcal{M}]$ as $\triangleleft$ is obviously closed under evaluation contexts. And $\langle E[R], s \rangle \rightarrow \langle E[M], s' \rangle$ as required.

$\square$

Lemma B.2. *If* $\varphi \in \mathbb{T}^{\mathfrak{I}}_{\mathcal{M}, term}$ *and* $M \triangleleft \mathcal{M}$ *then* $(\!| \varphi |\!) \subseteq \mathbb{T}_{M, term}$ *and for each* $s \in (\!| \varphi |\!)$, $\#_\downarrow^\varphi(\mathcal{M}) = \#_\downarrow^s(M)$

Proof. Follow directly from Lem. B.1. $\square$

Proposition B.3. *For every countable set of* pairwise compatible *traces* $A \subseteq \mathbb{T}^{\mathfrak{I}}_{\mathcal{M}, term}$ *and every* $M \triangleleft \mathcal{M}$ *we have the following:*

- $\omega(A) \leq \mathbb{P}_{term}(M)$
- $\mathbb{E}^{\mathcal{M}}(A) \leq \mathbb{E}_{term}(M)$

Proof. We first note that for every interval trace φ, $(\!| \varphi |\!)$ is a measurable set of traces and furthermore $\mu_{\mathbb{S}}((\!| \varphi |\!)) = \omega(\varphi)$ by definition of the Lebesgue measure.

Now as A is by assumption pairwise compatible the family $((\!| \varphi |\!))_{\varphi \in A}$ is pairwise almost disjoint. Thus

$$\omega(A) = \sum_{\varphi \in A} \omega(\varphi) \overset{(1)}{=} \sum_{\varphi \in A} \mu_{\mathbb{S}}((\!| \varphi |\!))$$

$$\overset{(2)}{=} \mu_{\mathbb{S}}\left(\bigcup_{\varphi \in A} (\!| \varphi |\!) \right) \overset{(3)}{\leq} \mu_{\mathbb{S}}(\mathbb{T}_{M, term}) = \mathbb{P}_{term}(M)$$

where **(1)** follows from the definition of the Lebesgue measure on boxes, **(2)** from the fact that family is pairwise almost disjoint and thus differs by a countable union of null sets. **(3)** follows from Lem. B.2.

For the second part we can observe that if $\varphi \in \mathbb{T}^{\mathfrak{I}}_{\mathcal{M}, term}$, $M \triangleleft \mathcal{M}$ and $s \triangleleft \varphi$, then $\#_\downarrow^s(M) = \#_\downarrow^\varphi(\mathcal{M})$. Now

$$\mathbb{E}^{\mathcal{M}}(A) = \sum_{\varphi \in A} \omega(\varphi) \cdot \#_\downarrow^\varphi(\mathcal{M})$$

$$\overset{(1)}{=} \sum_{n=0}^{\infty} \omega(\{\varphi \in A \mid \#_\downarrow^\varphi(\mathcal{M}) = n\}) \cdot n$$

$$\overset{(2)}{\leq} \sum_{n=0}^{\infty} \mu_{\mathbb{S}}\left(\mathbb{T}^n_{M, term} \right) \cdot n \overset{(3)}{\leq} \mathbb{E}_{term}(M)$$

where **(1)** follows from simple reordering, **(2)** from the fact that every interval trace in $\{\varphi \in A \mid \#_\downarrow^\varphi(\mathcal{M}) = n\}$ we get $(\!| \varphi |\!) \subseteq \mathbb{T}^n_{M, term}$ and the same reasoning as above. **(3)** is standard and can e.g. be inferred from the proof of Lem. 2.3. $\square$

Restatement of Thm. 3.4. *For every countable set of* pairwise compatible *traces* $A \subseteq \mathbb{T}^{\mathfrak{I}}_{M^{2\mathfrak{I}}, term}$ *the following holds:*

- $\omega(A) \leq \mathbb{P}_{term}(M)$
- $\mathbb{E}(M^{2\mathfrak{I}}, A) \leq \mathbb{E}_{term}(M)$

Proof. Follows directly from Prop. B.3 as $M \triangleleft M^{2\mathfrak{I}}$. □

B.4 Completeness

Restatement of Lem. 3.7. *If $f : \mathbb{R}^n \to \mathbb{R}$ is continuous and for every $y \in \mathbb{R}$, $f^{-1}(\{y\})$ is a Lebesgue Null-set then f is $\mathbb{Q}$-interval separable.*

Proof. Let $I = [a, b]$ be an interval as in the definition of interval separable. We have $f^{-1}([a, b]) = f^{-1}((a, b)) \cup f^{-1}(\{a\}) \cup f^{-1}(\{b\})$. By assumption $f^{-1}(\{a\})$ and $f^{-1}(\{b\})$ are null sets. As f is continuous and (a, b) is an open set we get that $f^{-1}(a, b)$ is an open set. A well know result in $\mathbb{R}^n$ is that every open-set can be covered exactly by a countable number of boxes that have rational endpoints. So there are boxes $B_1, B_2, \cdots$ (with rational endpoints) such that $\cup_i B_i = f^{-1}(a, b)$. □

The remaining pats of this section are devoted to give a proof of Thm. 3.8. We assume that all primitive function $f \in \mathbb{F}$ are interval separable. As a simple example while this is not easy consider the following:

Example B.4. Consider the term $M \triangleq \text{if}(\text{sample}-\underline{0.5}, \underline{0}, \underline{1})$ which is clearly AST. In fact, we have $\mathbb{T}_{M,\text{term}} = \{s_1 \mid s_1 \in \mathbb{R}_{[0,1]}\}$, so the set of terminating traces is itself an interval. However, the interval trace $\wp = [0, 1]$ is not terminating for $M^{2\mathfrak{I}}$ (formally $[0, 1] \notin \mathbb{T}^{\mathfrak{I}}_{M^{2\mathfrak{I}},\text{term}}$).

The key step to constructing a countable set of interval traces is to focus on branching. We, therefore, annotate the reduction relation with explicit information which branch of a conditional was taken. We define the set of *directions* by $D = \{L, R\}$. A *conditional oracle* is then a sequence $\kappa \in D^*$. To define the meaning of a conditional oracle we use a modified reduction relation $\xrightarrow{co} \subseteq (\Lambda \times \mathbb{S} \times D^*)^2$ via the rules in Fig. 11. We can easily see:

Lemma B.5. *If $s \in \mathbb{T}_{M,\text{term}}$ then there exists a unique $\kappa \in D^*$ and value V with $\langle M, s, \kappa \rangle \xrightarrow{co}{}^* \langle V, \epsilon, \epsilon \rangle$.*

We now partition the set of terminating traces according to their branching behaviour. For $\kappa \in D^*$ we define

$$\mathbb{T}^{(\kappa)}_{M,\text{term}} \triangleq \{s \in \mathbb{S} \mid \exists V : \langle M, s, \kappa \rangle \xrightarrow{co}{}^* \langle V, \epsilon, \epsilon \rangle\}$$

I.e., all traces that branch according to κ. By Lem. B.5 it is easy to see that that the family $\{\mathbb{T}^{(\kappa)}_{M,\text{term}}\}_{\kappa \in D^*}$ forms a partition of the set of terminating traces. Note that by fixing the branching, we also fix the number of reduction steps and the number of samples: if $s_1, s_2 \in \mathbb{T}^{(\kappa)}_{M,\text{term}}$, $\#^{s_1}_{\downarrow}(M) = \#^{s_2}_{\downarrow}(M)$ and $|s_1| = |s_2|$.

In Ex. B.4, we have seen that there exist interval traces $\wp \in \mathbb{S}_{\mathfrak{I}}$ with $(\!|\wp|\!) \subseteq \mathbb{T}_{M,\text{term}}$ that are not terminating for the canonical embedding $M^{2\mathfrak{I}}$ (i.e., $\wp \notin \mathbb{T}^{\mathfrak{I}}_{M^{2\mathfrak{I}},\text{term}}$). We can, however, show that if all traces in $(\!|\wp|\!)$ follow the same branching $\wp \in \mathbb{T}^{\mathfrak{I}}_{M^{2\mathfrak{I}},\text{term}}$ does hold. We first need the following:

Lemma B.6. *Let $\mathcal{M}$ be any term not already a value, $\kappa, \kappa' \in D^*$ and $\wp \in \mathbb{S}_{\mathfrak{I}}$ and $n \in \mathbb{N}$. If for every pair (M, s) with $M \triangleleft \mathcal{M}$ and $s \triangleleft \wp$*

$$\langle M, s, \kappa \rangle \xrightarrow{co}{}^n \langle M_{(M,s)}, s_{(M,s)}, \kappa' \rangle$$

(Note that $M_{(M,s)}$ and $s_{(M,s)}$ are uniquely determined). Then $\langle \mathcal{M}, \wp \rangle \rightsquigarrow^n \langle \mathcal{M}', \wp' \rangle$ for some $\mathcal{M}'$ and $\wp'$ such that for every pair (M, s), $M_{(M,s)} \triangleleft \mathcal{M}'$ and $s_{(M,s)} \triangleleft \wp'$.

Proof. We show the result for $n = 1$. The case for $n = 0$ is trivial and for $n > 1$ follows by easy induction using the case for $n = 1$. The proof goes by induction on $\mathcal{M}$. We only consider the case were $\mathcal{M}$ is itself a redex. The case where $\mathcal{M} = \mathcal{E}[\mathcal{R}]$ follow by induction on $\mathcal{R}$. The only interesting case is where $\mathcal{M}$ is a conditional redex: So lets focus on the case where $\mathcal{M} = \text{if}([a, b], \mathcal{N}, \mathcal{P})$: Now any $M \triangleleft \mathcal{M}$ must have the from $M = \text{if}(r_M, N_M, P_M)$ and there exist at least on such (as we work with closed, non, empty intervals). As any such M can reduce via $\xrightarrow{co}$ by assumption we get that $\kappa = L :: \kappa'$ or $\kappa = R :: \kappa'$ as otherwise no reduction can take place. W.l.o.g. assume $\kappa = L\kappa'$. We now claim that $b \leq 0$. Assume for contradiction that $b > 0$. Then choose the term $\dot{M} \triangleleft \text{if}(b, N, P)$ where $N \triangleleft \mathcal{N}$ and $P \triangleleft \mathcal{P}$ are arbitrary (they always exist). Now $\langle \dot{M}, s, L :: \kappa' \rangle$ cannot make a reduction step via $\xrightarrow{co}$ which contradicts the assumption. So $b \leq 0$.

This means that $\mathcal{M} = \text{if}([a, b], \mathcal{N}, \mathcal{P}) \rightsquigarrow \mathcal{N}$. Now any $M \triangleleft \mathcal{M}$ of the form $M = \text{if}(r_M, N_M, P_M)$ and reduces to N_M. So $M_{(M,s)} = N_M \triangleleft \mathcal{N} = \mathcal{M}'$ as required and obviously $s_{(M,s)} = s \triangleleft \wp = \wp'$. □

Lemma B.7. *If $\wp \in \mathbb{S}_{\mathfrak{I}}$, $\kappa \in D^*$ and $(\!|\wp|\!) \subseteq \mathbb{T}^{(\kappa)}_{M,\text{term}}$ then $\wp \in \mathbb{T}^{\mathfrak{I}}_{M^{2\mathfrak{I}},\text{term}}$.*

Proof. We show the following stronger lemma which immediately implies the result as the only term refining $M^{2\mathfrak{I}}$ is M itself: If $\mathcal{M}$ is any interval term, $\wp \in \mathbb{S}_{\mathfrak{I}}$ and $\kappa \in D^*$ and for all $M \triangleleft \mathcal{M}$, $(\!|\wp|\!) \subseteq \mathbb{T}^{(\kappa)}_{M,\text{term}}$ then $\wp \in \mathbb{T}^{\mathfrak{I}}_{\mathcal{M},\text{term}}$.

We can proof this as follows: As for any $M \triangleleft \mathcal{M}$, $(\!|\wp|\!) \subseteq \mathbb{T}^{(\kappa)}_{M,\text{term}}$ we get that every $M \triangleleft \mathcal{M}$ and $s \triangleleft \wp$, $\langle M, s, \kappa \rangle \xrightarrow{co}{}^n \langle V_{(M,s)}, \epsilon, \epsilon \rangle$ for a fixed n (As soon as κ is fixed each reduction takes the same number of steps). We can thus apply Lem. B.6 and get that $\langle \mathcal{M}, \wp \rangle \rightsquigarrow^n \langle \mathcal{V}, \epsilon \rangle$, so $\wp \in \mathbb{T}^{\mathfrak{I}}_{\mathcal{M},\text{term}}$ as required. □

B.5 Symbolic Terms and Symbolic Execution

The second key ingredient is symbolic execution as this gives us a better understanding of the sets $\mathbb{T}^{(\kappa)}_{M,\text{term}}$. The idea of symbolic terms is to not evaluate a term on a fixed trace of real numbers but instead on a generic trace consisting of variables. Whenever we resolve a sample-statement we do not substitute in a real number but a variable. This does prohibit us from evaluating primitive functions or resolve conditionals. To circumvent the former we use symbolic values, which

$$\frac{}{\langle(\lambda x.M)N, \mathbf{s}, \kappa\rangle \xrightarrow{co} \langle M[N/x], \mathbf{s}, \kappa\rangle} \qquad \frac{r \leq 0}{\langle \mathrm{if}(\underline{r}, N, P), \mathbf{s}, \mathbf{L} :: \kappa\rangle \xrightarrow{co} \langle N, \mathbf{s}, \kappa\rangle} \qquad \frac{r > 0}{\langle \mathrm{if}(\underline{r}, N, P), \mathbf{s}, \mathbf{R} :: \kappa\rangle \xrightarrow{co} \langle P, \mathbf{s}, \kappa\rangle}$$

$$\frac{}{\langle \mathrm{sample}, r :: \mathbf{s}, \kappa\rangle \xrightarrow{co} \langle \underline{r}, \mathbf{s}, \kappa\rangle} \qquad \frac{}{\langle(\mu_x^\varphi.M)N, \mathbf{s}, \kappa\rangle \xrightarrow{co} \langle M[N/x, (\mu_x^\varphi.M)/\varphi], \mathbf{s}, \kappa\rangle} \qquad \frac{r \geq 0}{\langle \mathrm{score}(\underline{r}), \mathbf{s}, \kappa\rangle \xrightarrow{co} \langle \underline{r}, \mathbf{s}, \kappa\rangle}$$

$$\frac{}{\langle f(\underline{r_1}, \cdots, \underline{r_{|f|}}), \mathbf{s}, \kappa\rangle \xrightarrow{co} \langle \underline{f(r_1, \cdots, r_{|f|})}, \mathbf{s}, \kappa\rangle} \qquad \frac{\langle R, \mathbf{s}, \kappa\rangle \xrightarrow{co} \langle M, \mathbf{s}', \kappa'\rangle}{\langle E[R], \mathbf{s}, \kappa\rangle \xrightarrow{co} \langle E[M], \mathbf{s}', \kappa'\rangle}$$

Figure 11. Small-step reduction relation with conditional oracles.

can be seen as partially evaluate primitive functions. To resolve the latter we make use of the conditional oracles. For an overview of a similar system of symbolic execution we refer the reader to [40].

Let $\alpha_0, \alpha_1, \cdots$ be a denumerable set of *sample-variables* indexed by natural numbers. We use them to postpone every sample statement by instead substitution a fresh variable. Symbolic values and terms are defined by:

$$\mathfrak{B} \triangleq x \mid \underline{r} \mid \alpha_j \mid \lambda x.\mathfrak{M} \mid \mu_x^\varphi.\mathfrak{M} \mid \boxed{f}(\mathfrak{B}_1, \cdots, \mathfrak{B}_{|f|})$$

$$\mathfrak{M}, \mathfrak{N}, \mathfrak{P} \triangleq \mathfrak{B} \mid \mathfrak{M}\mathfrak{N} \mid \mathrm{if}(\mathfrak{M}, \mathfrak{N}, \mathfrak{P})$$
$$\mid f(\mathfrak{M}_1, \cdots, \mathfrak{M}_{|f|}) \mid \mathrm{sample} \mid \mathrm{score}(\mathfrak{M})$$

Note that the only new syntactic additions, compared with standard SPCF, are the sample variables α_j and the symbolic primitive functions $\boxed{f}(\mathfrak{M}_1, \cdots, \mathfrak{M}_{|f|})$. We again focus on typable terms. The simple type system for standard SPCF (given in Fig. 1) naturally extends to symbolic terms when we add the following two rules:

$$\frac{}{\Gamma \Vdash \alpha_j : \mathbf{R}}$$

$$\frac{\Gamma \Vdash \mathfrak{M}_1 : \mathbf{R} \qquad \cdots \qquad \Gamma \Vdash \mathfrak{M}_{|f|} : \mathbf{R}}{\Gamma \Vdash \boxed{f}(\mathfrak{M}_1, \cdots, \mathfrak{M}_{|f|}) : \mathbf{R}}$$

Let Λ_{sym} be the set of all typable symbolic terms. Note that any $M \in \Lambda$ directly corresponds to a symbolic term in the canonical way.

B.5.1 Symbolic Execution. We now give an operational small-step semantics to symbolic terms. This symbolic execution closely corresponds to reduction in the standard (CbN) semantics with the exception that every sample-statement is resolved by a sample variable. Symbolic redexes and evaluation contexts are defined as expected:

$$\mathfrak{R} \triangleq (\lambda x.\mathfrak{M})\mathfrak{N} \mid (\mu_x^\varphi.\mathfrak{M})\mathfrak{N} \mid \mathrm{if}(\mathfrak{B}, \mathfrak{N}, \mathfrak{P})$$
$$\mid f(\mathfrak{B}_1, \cdots, \mathfrak{B}_{|f|}) \mid \mathrm{sample} \mid \mathrm{score}(\mathfrak{B})$$

$$\mathfrak{E} \triangleq [\cdot] \mid \mathfrak{E}\mathfrak{M} \mid \mathrm{if}(\mathfrak{E}, \mathfrak{N}, \mathfrak{P}) \mid \mathrm{score}(\mathfrak{E})$$
$$\mid f(\mathfrak{B}_1, \cdots, \mathfrak{B}_{k-1}, \mathfrak{E}, \mathfrak{M}_{k+1}, \cdots, \mathfrak{M}_{|f|})$$

Symbolic Values. As sample variables are taken in for real-valued numerals, whenever we resolve a sample statement, we can no longer evaluate primitive functions as some of the arguments may be variables. A function symbol f applied to arguments, therefore, does not evaluate to the function value but instead we postpone the evaluation and use the symbolic construct $\boxed{f}$. In particular, a (closed) symbolic value of type $\mathbf{R}$ is no longer always a numeral. We can view $\boxed{f}$ as a function evaluation that is postponed. If we fix the value of the sample variables, a symbolic value, therefore, does again denotes a real number: Let $\mathfrak{B}$ be a symbolic value of type $\mathbf{R}$ (no λ or μ-abstraction) with sample-variables within $\{\alpha_0, \cdots, \alpha_{m-1}\}$. We can view a vector $\sigma \in \mathbb{R}_{[0,1]}^m$ as a substitution and define $\mathfrak{B}[\sigma] \in \mathbb{R}$ in the obvious way by substituting in values and evaluating primitive functions. Given $A \subseteq \mathbb{R}$ we define $\mathfrak{B}^{-1}(A) \triangleq \{\sigma \in \mathbb{R}_{[0,1]}^m \mid \mathfrak{B}[\sigma] \in A\}$.

Symbolic Inequality. We define a symbolic inequality as pairs of the from $(\mathfrak{B} \bowtie r)$ where $\mathfrak{B}$ is a symbolic value, $\bowtie \in \{\leq, <, \geq, >\}$ and $r \in \mathbb{R}$. A symbolic constraint Δ is a set of symbolic inequalities. Given a symbolic constraint $\Delta = \{(\mathfrak{B}_i \bowtie_i r_i)\}_{i \in [n]}$ with sample variables contained within $\alpha_0, \cdots, \alpha_{m-1}$ we can define

$$\mathrm{Sat}_m(\Delta) \triangleq \{\sigma \in \mathbb{R}_{[0,1]}^m \mid \forall i \in [n] : \mathfrak{B}_i[\sigma] \bowtie_i r_i\}$$

We can see every $\sigma \in \mathrm{Sat}_m(\Delta)$ also as an element in $\mathbb{S}^m$, i.e., a standard trace of length m.

Symbolic Configuration and Symbolic Execution. A symbolic configuration has the from $\begin{vmatrix} \mathfrak{M}, \kappa, n \\ \Delta \end{vmatrix}$ where $\mathfrak{M}$ is a symbolic term, $\kappa \in D^*$ a sequence of directions $n \in \mathbb{N}$ a natural number and Δ a symbolic constraint. The conditional oracle κ is used to resolve branching. During execution the constraints that a trace needs to satisfy to actually follow κ are recorded in the constraint Δ. The natural number in each configuration references the number of sample variables that have already been substituted. We define the symbolic small-step reduction relation $\xrightarrow{\mathrm{sym}}$ via the rules in Fig. 12.
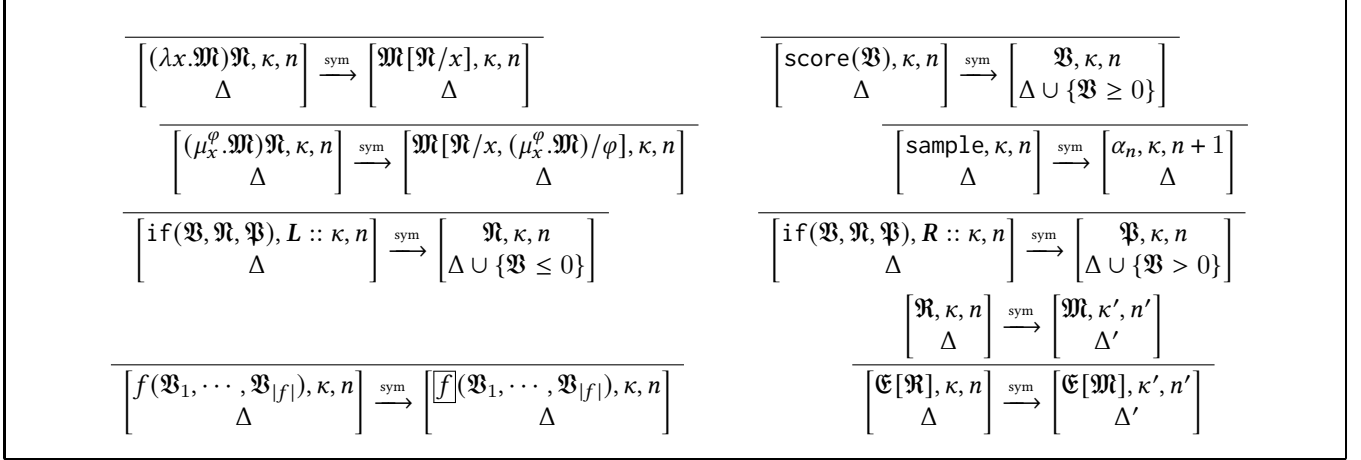
$$\dfrac{}{\left[\begin{matrix}(\lambda x.\mathfrak{M})\mathfrak{N},\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\mathfrak{M}[\mathfrak{N}/x],\kappa,n\\\Delta\end{matrix}\right]}$$

$$\dfrac{}{\left[\begin{matrix}\text{score}(\mathfrak{B}),\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\mathfrak{B},\kappa,n\\\Delta\cup\{\mathfrak{B}\geq 0\}\end{matrix}\right]}$$

$$\dfrac{}{\left[\begin{matrix}(\mu_x^\varphi.\mathfrak{M})\mathfrak{N},\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\mathfrak{M}[\mathfrak{N}/x,(\mu_x^\varphi.\mathfrak{M})/\varphi],\kappa,n\\\Delta\end{matrix}\right]}$$

$$\dfrac{}{\left[\begin{matrix}\text{sample},\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\alpha_n,\kappa,n+1\\\Delta\end{matrix}\right]}$$

$$\dfrac{}{\left[\begin{matrix}\text{if}(\mathfrak{B},\mathfrak{N},\mathfrak{P}),L::\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\mathfrak{N},\kappa,n\\\Delta\cup\{\mathfrak{B}\leq 0\}\end{matrix}\right]}$$

$$\dfrac{}{\left[\begin{matrix}\text{if}(\mathfrak{B},\mathfrak{N},\mathfrak{P}),R::\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\mathfrak{P},\kappa,n\\\Delta\cup\{\mathfrak{B}>0\}\end{matrix}\right]}$$

$$\dfrac{}{\left[\begin{matrix}f(\mathfrak{B}_1,\cdots,\mathfrak{B}_{|f|}),\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\boxed{f}(\mathfrak{B}_1,\cdots,\mathfrak{B}_{|f|}),\kappa,n\\\Delta\end{matrix}\right]}$$

$$\dfrac{\left[\begin{matrix}\mathfrak{R},\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\mathfrak{M},\kappa',n'\\\Delta'\end{matrix}\right]}{\left[\begin{matrix}\mathfrak{E}[\mathfrak{R}],\kappa,n\\\Delta\end{matrix}\right]\xrightarrow{\text{sym}}\left[\begin{matrix}\mathfrak{E}[\mathfrak{M}],\kappa',n'\\\Delta'\end{matrix}\right]}$$

Figure 12. Small-step reduction for symbolic terms (symbolic execution).

The symbolic execution we present here differs (especially on first glance) from the one used in [40]. In their semantics, a symbolic configuration at all times contains a set of traces that can take this path. In contrast, we annotate a symbolic configuration with an explicit set of symbolic inequalities. As we fixed the outcomes of conditionals beforehand our reductions is deterministic.

Correspondence. We can show the following correspondence theorem:

Proposition B.8. *For any term M. If $\kappa \in D^*$ and there exist $\mathfrak{B}, n, \Delta$ (If they exists, they are unique) such that*

$$\left[\begin{matrix}M,\kappa,0\\\emptyset\end{matrix}\right]\xrightarrow{\text{sym}*}\left[\begin{matrix}\mathfrak{B},\epsilon,n\\\Delta\end{matrix}\right]$$

then $\text{Sat}_n(\Delta) = \mathbb{T}_{M,term}^{(\kappa)}$ *otherwise* $\mathbb{T}_{M,term}^{(\kappa)} = \emptyset$.

Proof. The proof is analogous to the proof in [40, Thm. 13]. Every symbolic configuration $\left[\begin{smallmatrix}\mathfrak{M},\kappa,n\\\Delta\end{smallmatrix}\right]$ can be seen as the pair $\langle\!\langle \mathfrak{M}, \_, \text{Sat}_n(\Delta)\rangle\!\rangle$ in the setting of [40] when we omit the weight parameter (denoted by $\_$). □

Example B.9. Consider the term

$$\mathfrak{M} \triangleq \text{if sample} + \text{sample} - \underline{1} \text{ then } x \text{ else}$$
$$(\text{if } \underline{0} \text{ then } \underline{3} \text{ else } \underline{4})$$

For the conditional oracle **RL** we get

$$\left[\begin{matrix}\mathfrak{M},RL,0\\\emptyset\end{matrix}\right]\xrightarrow{\text{sym}*}\left[\begin{matrix}\underline{3},\epsilon,2\\\{\alpha_0\boxplus\alpha_1\boxminus\underline{1}>0,\underline{0}\leq 0\}\end{matrix}\right]$$

And the solution Sat_2 of the symbolic constraint

$$\Delta \triangleq \{\alpha_0\boxplus\alpha_1\boxminus\underline{1}>0,\underline{0}\leq 0\}$$

is the set $\{s_0 s_1 \in \mathbb{S}^2 \mid s_0 + s_1 > 1\}$ which is exactly the set $\mathbb{T}_{M,\text{term}}^{(RL)}$ as stated in Prop. B.8.

B.5.2 Completeness Proof.

Lemma B.10. *If $\mathfrak{B}$ is a symbolic value of type $\boldsymbol{R}$ with sample variables among $\{\alpha_0,\cdots,\alpha_{m-1}\}$ where each variable occurs at most once and $[a,b] \in \mathfrak{I}$ an interval then there exists a countable family of boxes $\{B_i\}_{i\in\mathcal{I}}$ ($B_i \subseteq \mathbb{R}_{[0,1]}^m$) such that $\bigcup_i B_i \Subset \mathfrak{B}^{-1}([a,b])$.*

Proof. We do induction on the structure of $\mathfrak{B}$. The case of $\mathfrak{B} = \underline{r}$ and $\mathfrak{B} = \alpha_j$ is trivial. So let $\mathfrak{B} = \boxed{f}(\mathfrak{B}_1,\cdots,\mathfrak{B}_{|f|})$.

As f is by assumption interval-separable there exist countable boxes $(B_i)_{i\in\mathcal{I}}$ s.t., $\cup_i B_i \Subset f^{-1}([a,b])$ for some countable set $\mathcal{I}$. Each B_i is a box and can thus be written as $B_i = [a_i^1,b_i^1]\times\cdots\times[a_i^{|f|},b_i^{|f|}]$. Now define $C_i \triangleq \bigcap_{1\leq j\leq|f|}\mathfrak{B}_j^{-1}([a_i^j,b_i^j]) \subseteq \mathbb{R}_{[0,1]}^m$. These are all assignments such that each $\mathfrak{B}_j$ takes on a value in $[a_i^j,b_i^j]$. As the countable union of Lebesgue null sets is a null we get $\bigcup_{i\in\mathcal{I}} C_i \Subset \mathfrak{B}^{-1}([a,b])$. Call this fact **(1)**.

Now by induction for each $1\leq j\leq|f|$ there exists a family of boxes $(B_k^{i,j})_{k\in\mathcal{I}_i^j}$ for some countable index set $\mathcal{I}_i^j$, such that $\bigcup_{k\in\mathcal{I}_i^j} B_k^{i,j} \Subset \mathfrak{B}_j^{-1}([a_i^j,b_i^j])$. Now $\bigcap_{1\leq j\leq|f|}\bigcup_{k\in\mathcal{I}_i^j} B_k^{i,j} = \bigcup_{(k_1,\cdots,k_{|f|})\in\mathcal{I}_i^1\times\cdots\times\mathcal{I}_i^{|f|}} B_{k_1}^{i,1}\cap\cdots B_{k_{|f|}}^{i,|f|}$ by distributing the intersection over the union. We can put this together and get the following by again using the fact that the countable union of null sets is a null set:

$$\bigcup_{(k_1,\cdots,k_{|f|})\in\mathcal{I}_i^1\times\cdots\times\mathcal{I}_i^{|f|}} B_{k_1}^{i,1}\cap\cdots B_{k_{|f|}}^{i,|f|}$$
$$=\bigcap_{1\leq j\leq|f|}\bigcup_{k\in\mathcal{I}_i^j} B_k^{i,j} \Subset \bigcap_{1\leq j\leq|f|}\mathfrak{B}_j^{-1}([a_i^j,b_i^j])=C_i$$

Note that the index set $\mathcal{I}_i^1\times\cdots\times\mathcal{I}_i^{|f|}$ is countable. Combined with **(1)** we get

$$\bigcup_{i\in\mathcal{I}}\bigcup_{(k_1,\cdots,k_{|f|})\in\mathcal{I}_i^1\times\cdots\times\mathcal{I}_i^{|f|}} B_{k_1}^{i,1}\cap\cdots B_{k_{|f|}}^{i,|f|} \Subset \mathfrak{B}^{-1}([a,b])$$

Note that the set $\{(i, k_1, \cdots, k_{|f|}) \mid i \in \mathcal{I}, (k_1, \cdots, k_{|f|}) \in \mathcal{I}_i^1 \times \cdots \times \mathcal{I}_i^{|f|}\}$ is also countable as the countable product of countable sets. Also note that the finite intersection of boxes in the equation above is again a box. We are thus done. □

Lemma B.11. *If A, B are two boxes in $\mathbb{R}^m$ then there exist finite boxes $\{C_i\}_{i \in [n]}$ that are pairwise almost disjoint and satisfy $A \cup B = \bigcup_i C_i$.*

Restatement of Thm. 3.8. *If every $f \in \mathbb{F}$ is interval separable, then for every $M \in \Lambda_0$ there exists a countable set of pairwise-compatible interval traces $A \subseteq \mathbb{T}_{M^{2\mathfrak{I}}, term}^{\mathfrak{I}}$ such that $\omega(A) = \mathbb{P}_{term}(M)$; and if M is AST then $\mathbb{E}(M^{2\mathfrak{I}}, A) = \mathbb{E}_{term}(M)$.*

Proof. We can naturally identify traces in $\mathbb{S}^m$ with elements in $\mathbb{R}_{[0,1]}^m$. The definition of almost-surely fully contained, $\Subset$, naturally extends to traces. Fix any $\kappa \in D^*$. In the first step, we show that there exists a countable family of boxes $\{B_l\}_{l \in \mathcal{I}}$ ($B_l \subseteq \mathbb{R}_{[0,1]}^m$) such that $\bigcup_{l \in \mathcal{I}} B_l \Subset \mathbb{T}_{M, term}^{(\kappa)}$.

First: There either exists a value $\mathfrak{B}$ a natural number m and constraint Δ (all of them unique) such that $\begin{bmatrix} \mathfrak{M}, \kappa, 0 \\ \emptyset \end{bmatrix} \xrightarrow{\text{sym}*} \begin{bmatrix} \mathfrak{B}, \epsilon, m \\ \Delta \end{bmatrix}$ or there exists none. In either case, we apply Prop. B.8. In the latter case, we are done as $\mathbb{T}_{M, term}^{(\kappa)} = \emptyset$. In the former case, we get $\mathsf{Sat}_m(\Delta) = \mathbb{T}_{M, term}^{(\kappa)}$. Note that $\mathbb{T}_{M, term}^{(\kappa)} \subseteq \mathbb{S}^m$. Let $\Delta = \{(\mathfrak{B}_i \bowtie_i r_i)\}_{i \in [n]}$. Now by definition of Sat_m, $\mathsf{Sat}_m(\Delta) = \{\sigma \in \mathbb{R}_{[0,1]}^m \mid \forall i \in [n] : \mathfrak{B}_i[\sigma] \bowtie_i r_i\}$. We can write this as $\bigcap_{i \in [n]} \mathfrak{B}_i^{-1}(I_i)$ where I_i is one of (r_i, ∞), $[r_i, \infty)$, $(-\infty, r_i)$ or $(-\infty, r_i]$ depending on $\bowtie_i$. Due to the CbN evaluation, each symbolic value contains each sample variable at at most one position. As all of these sets can be given as a countable union of closed bounded intervals, we can apply Lem. B.10 and get a family $(B_k^i)_{k \in \mathcal{I}_i}$ such that $\bigcup_{k \in \mathcal{I}_i} B_k^i \Subset \mathfrak{B}_i^{-1}(I_i)$. Now the finite intersection of countable unions of boxes is itself a countable union of boxes (refer to the proof of Lem. B.10). There thus exists a family $(B_l)_{l \in \mathcal{I}}$ with $\bigcup_{l \in \mathcal{I}} B_l \Subset \bigcap_{i \in [n]} \mathfrak{B}_i^{-1}(I_i) = \mathbb{T}_{M, term}^{(\kappa)}$.

Second: So $\bigcup_{l \in \mathcal{I}} B_l \Subset \mathbb{T}_{M, term}^{(\kappa)}$. By Lem. B.11 we can assume that this family is pairwise almost disjoint. Now each box $B_l \subseteq \mathbb{R}_{[0,1]}^m$ can naturally be seen as an interval trace within $\mathbb{S}_{\mathfrak{I}}^m$. Let $A^{(\kappa)}$ be this set of interval traces. As the boxes are pairwise almost disjoint the traces are pairwise compatible. For each $\wp \in A^{(\kappa)}$ we have $(\!|\wp|\!) \in \mathbb{T}_{M, term}^{(\kappa)}$ so by Lem. B.7 we get that $\wp \in \mathbb{T}_{M^{2\mathfrak{I}}, term}^{\mathfrak{I}}$. So $A^{(\kappa)} \subseteq \mathbb{T}_{M^{2\mathfrak{I}}, term}^{\mathfrak{I}}$. As the set of conditional oracles D^* is countable we can take the union of all interval traces $A^{(\kappa)}$ for all $\kappa \in D^*$. There thus exists a countable set of interval traces $A \subseteq \mathbb{T}_{M^{2\mathfrak{I}}, term}^{\mathfrak{I}}$ such that $\bigcup_{\wp \in A}(\!|\wp|\!) \Subset \mathbb{T}_{M, term}$. This already implies that $\omega(A) = \mu_{\mathbb{S}}(\mathbb{T}_{M, term})$. For the expected time to termination recall that for all $s \in (\!|\wp|\!)$, $\#_{\downarrow}^{\wp}(M^{2\mathfrak{I}}) = \#_{\downarrow}^s(M)$. □

C   Additional Material - Section 4

To state properties of the type system it is actually easiest to decompose this reduction relation. A relation $\rightarrow_{det}$, handling deterministic steps, and a relation $\rightarrow_{[a,b]}$ for $[a, b] \in \mathfrak{I}_{0,1}$ performing probabilistic steps. Those relations are defined in Fig. 13.

While our type system is designed such that the least upper bound over all derivation equals the probability of termination and thus looks very similar to the monadic system in Breuvart and Dal Lago [9], we have to approach on an entirely different way. The system by Breuvart and Dal Lago relies on the countable nature of the execution tree and can state subject reduction by taking the weighted (finite) sum over the reduction relation. Due to the uncountable nature of SPCF, we cannot follow this approach. Instead, in our system we allow for enumeration of terminating interval traces and make use of the soundness and completeness of the interval-based semantics shown in Sec. 3. We write $[n]$ for the set $\{0, \cdots, n-1\}$, i.e., the first n integers.

Lemma C.1. *If $\vdash M : \mathcal{A}$ and $\mathcal{B} \subseteq \mathcal{A}$ then $\vdash M : \mathcal{B}$*

Proof. Easy induction on $\vdash M : \mathcal{A}$. □

C.1   Subject Reduction and Soundness

C.1.1   Subject Reduction. We begin by showing that our system does enjoy subject reduction. In our setting, the τ component gives the number of steps to termination. Matching this intuition, the τ decrease by 1 in each step. Furthermore as each $\wp$ is a terminating trace, each probabilistic reduction consumes the first element (c.f. [9]).

Lemma C.2 (Substitution). *If $\Gamma; \{x_i : \sigma_i\}_{i \in [n]} \vdash M : \mathcal{A}$ for distinct x_i and for all $i \in [n]$ and $\mathcal{B} \in \sigma_i$, $\Gamma \vdash N_i : \mathcal{B}$ then $\Gamma \vdash M[N_i/x_i]_{i \in [n]} : \mathcal{A}$*

Proof. An easy induction on M. □

Lemma C.3 (Deterministic Subject Reduction). *If $\vdash M : \mathcal{A}$, $\mathcal{A} \neq \{\}$ and M has a deterministic redex and then $M \rightarrow_{det} M'$ and $\vdash M' : \mathcal{A}^{(\uparrow\epsilon, -1)}$.*

Proof. Induction on $M \rightarrow_{det} M'$. Case analysis on M.

- $M = (\lambda x. N)\mathcal{P} \rightarrow_{det} N[\mathcal{P}/x]$: Then the last step must have been:

$$\frac{\dfrac{\{x : \sigma\} \vdash N : \mathcal{B}}{\vdash \lambda x. N : \{(\sigma \rightarrow \mathcal{B}, \epsilon, 0)\}} \text{ (abs)} \qquad \{\vdash \mathcal{P} : C \mid \forall C \in \sigma\}}{\vdash (\lambda x. N)\mathcal{P} : \mathcal{B}^{(\uparrow\epsilon, 1)} = \mathcal{A}} \text{ (app)}$$

  By substitution (Lem. C.2) we can type $\vdash N[\mathcal{P}/x] : \mathcal{B} = \mathcal{A}^{(\uparrow\epsilon, -1)}$ as required.
- $M = (\mu_x^{\varphi}. N)\mathcal{P} \rightarrow_{det} N[\mathcal{P}/x, (\mu_x^{\varphi}. N)/\varphi]$: Then the last step must have been via (app) and (fix), similar to above. We conclude via (Lem. C.2).
- $M = \text{if}([a, b], N, \mathcal{P}) \rightarrow_{det} N$ and $b \leq 0$: Then the last step must have been:

$$\frac{}{(\lambda x.\mathcal{M})\mathcal{N} \to_{\text{det}} \mathcal{M}[\mathcal{N}/x]} \quad \frac{}{(\mu_x^\varphi.\mathcal{M})\mathcal{N} \to_{\text{det}} \mathcal{M}[\mathcal{N}/x, (\mu_x^\varphi.\mathcal{M})/\varphi]} \quad \frac{}{f([\underline{a_1,b_1}], \cdots, [\underline{a_{|f|},b_{|f|}}]) \to_{\text{det}} \hat{f}(a_1,b_1,\cdots,a_{|f|},b_{|f|})}$$

$$\frac{a \geq 0}{\text{score}([\underline{a,b}]) \to_{\text{det}} [\underline{a,b}]} \quad \frac{b \leq 0}{\text{if}([\underline{a,b}],\mathcal{N},\mathcal{P}) \to_{\text{det}} \mathcal{N}} \quad \frac{a > 0}{\text{if}([\underline{a,b}],\mathcal{N},\mathcal{P}) \to_{\text{det}} \mathcal{P}} \quad \frac{\mathcal{R} \to_{\text{det}} \mathcal{M}}{\mathcal{E}[\mathcal{R}] \to_{\text{det}} \mathcal{E}[\mathcal{M}]}$$

$$\frac{}{\text{sample} \to_{[a,b]} [\underline{a,b}]} \qquad\qquad \frac{\mathcal{R} \to_{[a,b]} \mathcal{M}}{\mathcal{E}[\mathcal{R}] \to_{[a,b]} \mathcal{E}[\mathcal{M}]}$$

Figure 13. Decomposed reduction into deterministic steps $\to_{\text{det}}$ and probabilistic steps $\to_{[a,b]}$

$$\frac{\vdash [\underline{a,b}] : \{([\underline{a,b}],\epsilon,0)\} \qquad \vdash \mathcal{N} : \mathcal{B}_{([a,b],\epsilon,0)}}{\vdash \text{if}([\underline{a,b}],\mathcal{N},\mathcal{P}) : \mathcal{B}_{([a,b],\epsilon,0)}^{(\uparrow\epsilon,1)}} \text{ (if)}$$

So $\vdash \mathcal{N} : \mathcal{B}_{([a,b],\epsilon,0)} = \mathcal{A}^{(\uparrow\epsilon,-1)}$.

- $\mathcal{M} = \text{if}([\underline{a,b}],\mathcal{N},\mathcal{P}) \to_{\text{det}} \mathcal{P}$ and $a > 0$. Similar to the previous case.
- $\mathcal{M} = \text{if}([\underline{a,b}],\mathcal{N},\mathcal{P})$ and $a < 0$ and $b \geq 0$. Note possible as by assumption $\mathcal{A} \neq \{\}$.
- $\mathcal{M} = f([\underline{a,b}],[\underline{c,d}]) \to_{\text{det}} \hat{f}(a,b,c,d)$. Then the last step must have been via (f) and (num) and $\mathcal{A} = \{(\hat{f}(a,b,c,d),\epsilon,1)\}$ we can type

$$\vdash \hat{f}(a,b,c,d) : \{(\hat{f}(a,b,c,d),\epsilon,0)\}$$

via (num) .

- $\mathcal{M} = \text{score}([\underline{a,b}]) \to_{\text{det}} [\underline{a,b}]$ and $a \geq 0$. Then the last step must have been via (score) and (num) to $\mathcal{A} = \{([\underline{a,b}],\epsilon,1)\}$ and we can type $\vdash [\underline{a,b}] : \{([\underline{a,b}],\epsilon,0)\}$ via (num) as required.
- $\mathcal{M} = \text{score}([\underline{a,b}])$ and $a < 0$. Not possible as by assumption $\mathcal{A} \neq \{\}$.
- $\mathcal{M} = \mathcal{N}\mathcal{P} \to_{\text{det}} \mathcal{N}'\mathcal{P}$ and $\mathcal{N} \to_{\text{det}} \mathcal{N}'$. Then the last step must have been:

$$\frac{\vdash \mathcal{N} : \mathcal{B} \qquad \{\vdash \mathcal{P} : \mathcal{D} \mid \forall(\sigma \to C, \wp, \tau) \in \mathcal{B}, \mathcal{D} \in \sigma\}}{\vdash \mathcal{N}\mathcal{P} : \bigcup_{(\sigma\to C,\wp,\tau)\in\mathcal{B}} C^{(\uparrow\wp,\tau+1)}} \text{ (app)}$$

By induction we get $\vdash \mathcal{N}' : \mathcal{B}^{(\uparrow\epsilon,-1)}$. We can conclude using (app) , by choosing the same type derivations for each element in $\mathcal{B}^{(\uparrow\epsilon,-1)}$ as in the original derivation..

- $\mathcal{M} = \text{if}(\mathcal{N},\mathcal{P},\mathcal{Q}) \to_{\text{det}} \text{if}(\mathcal{N}',\mathcal{P},\mathcal{Q})$ and $\mathcal{N} \to_{\text{det}} \mathcal{N}'$: Then the last step must have been via (if) . We can use the IH on $\mathcal{N}$ and conclude via (if) by choosing the same derivations.
- $\mathcal{M} = \text{score}(\mathcal{N}), \mathcal{M} = f(\mathcal{N},\mathcal{P}), \mathcal{M} = f([\underline{a,b}],\mathcal{N})$ are trivial.

□

Lemma C.4 (Probabilistic Subject Reduction). *If $\vdash \mathcal{M} : \{(\alpha,\wp,\tau)\}$ and $\mathcal{M}$ has a probabilistic redex then $\wp = [a,b]\wp'$ and we have $\mathcal{M} \to_{[a,b]} \mathcal{M}'$ and $\vdash \mathcal{M}' : \{(\alpha,\wp',\tau-1)\}$*

Proof. Case analysis on $\mathcal{M}$.

- $\mathcal{M} = \text{sample}$. Then the last step is:

$$\frac{}{\vdash \text{sample} : \{([\underline{a,b}],[\underline{a,b}],1)\}} \text{ (sample)}$$

for some a,b. We get $\text{sample} \to_{[a,b]} \mathcal{M}' \triangleq [\underline{a,b}]$ and can obviously type: $\vdash \mathcal{M}' : \{([\underline{a,b}],\epsilon,0)\}$ using (num) .

- $\mathcal{M} = \mathcal{N}\mathcal{P}$ and $\mathcal{N}$ has a probabilistic redex. The last step must have been:

$$\frac{\vdash \mathcal{N} : \{(\sigma \to \{(\alpha,\wp_3,\tau_3)\},\wp_1,\tau_1)\} \qquad \{\vdash \mathcal{P} : C \mid \forall C \in \sigma\}}{\vdash \mathcal{N}\mathcal{P} : \{(\alpha,\wp_1\wp_3,\tau_1+\tau_3+1)\}} \text{ (app)}$$

By induction we get that $\wp_1 = [a,b]\wp_2$, $\mathcal{N} \to_{[a,b]} \mathcal{N}'$ and $\vdash \mathcal{N} : \{(\sigma \to \{(\alpha,\wp_3,\tau_3)\},\wp_2,\tau_1-1)\}$. So $\wp_1\wp_3 = [a,b]\wp_2\wp_3$. Now $\mathcal{N}\mathcal{P} \to_{[a,b]} \mathcal{N}'\mathcal{P}$ and we can conclude $\vdash \mathcal{N}'\mathcal{P} : \{(\alpha,\wp_2\wp_3,\tau_1+\tau_3)\}$ via (app) .

- All the other closre cases, i.e., $\mathcal{M} = \text{if}(\mathcal{N},\mathcal{P},\mathcal{Q})$, $\mathcal{M} = f(\mathcal{N},\mathcal{P})$, $\mathcal{M} = f([\underline{a,b}],\mathcal{N})$ and $\mathcal{M} = \text{score}(\mathcal{N})$ where $\mathcal{N}$ has a probabilistic redex follow in the same fashion as above.

□

Lemma C.5 (Subject Reduction). *If $\vdash \mathcal{M} : \{(\alpha,\wp,\tau)\}$ and $\mathcal{M}$ is not a value, then either*

- *$\mathcal{M}$ has a deterministic redex and $\mathcal{M} \to_{\text{det}} \mathcal{M}'$ and $\vdash \mathcal{M}' : \{(\alpha,\wp,\tau-1)\}$, or*
- *$\mathcal{M}$ has a probabilistic redex then $\wp = [a,b]\wp'$ and we have $\mathcal{M} \to_{[a,b]} \mathcal{M}'$ and $\vdash \mathcal{M}' : \{(\alpha,\wp',\tau-1)\}$*

Proof. Follows from Lem. C.4 and Lem. C.3. □

Lemma C.6. *If $\vdash \mathcal{M} : \{(\alpha_i,\wp_i,\tau_i) \mid i \in [n]\}$ then $\wp_i \in \mathbb{T}_{\mathcal{M},term}^{\mathfrak{I}}$ and $\#_\downarrow^{\wp_i}(\mathcal{M}) = \tau_i$ for all $i \in [n]$*

Proof. We show the easier observation that if $\vdash \mathcal{M} : \{\alpha,\wp,\tau\}$, then $\wp \in \mathbb{T}_{\wp,\text{term}}^{\mathfrak{I}}$ and $\#_\downarrow^\wp(\mathcal{M}) = \tau$. The result then follows by Lem. C.1 as we get $\vdash \mathcal{M} : \{\alpha_i,\wp_i,\tau_i\}$ for every $i \in [n]$.

As an easy corollary from Subject reduction (Lem. C.5) combined with the obvious properties of the decomposed semantics, we get that if $\vdash \mathcal{M} : \{(\alpha,\wp,\tau)\}$ and $\langle \mathcal{M},\wp \rangle \rightsquigarrow$

$\langle \mathcal{M}', \wp' \rangle$ we have $\vdash \mathcal{M}' : \{(\alpha, \wp', \tau - 1)\}$. Call this observation **(1)**.

We first show $\wp \in \mathbb{T}_{\mathcal{M},\text{term}}^{\Im}$. Let $\langle M, \wp \rangle \triangleq \langle \mathcal{M}_0, \wp_0 \rangle \rightsquigarrow \langle \mathcal{M}_1, \wp_1 \rangle \rightsquigarrow \langle \mathcal{M}_2, \wp_2 \rangle \rightsquigarrow \cdots$ be the possibly infinite reduction sequence. From **(1)** we get $\vdash \mathcal{M}_i : \{(\alpha, \wp_i, \tau - i)\}$. The sequence can thus make *at most* τ-steps and is hence finite. Let $\langle M, \wp \rangle = \langle \mathcal{M}_0, \wp_0 \rangle \rightsquigarrow \langle \mathcal{M}_1, \wp_1 \rangle \rightsquigarrow \langle \mathcal{M}_2, \wp_2 \rangle \rightsquigarrow \cdots \rightsquigarrow \langle \mathcal{M}_n, \wp_n \rangle$ be this finite, maximal sequence. We assume for contraction that $\mathcal{M}_n$ is not a value. As $\vdash \mathcal{M}_n : \{(\alpha, \wp_n, \tau - n)\}$ we can use subject reduction (Lem. C.5) and get that $\langle \mathcal{M}_n, \wp_n \rangle$ can make a further step which contradicts the maximality. Now as M_n is a value, we can inspect the typing rules and get that $\wp_n = \epsilon$ as values can only be typed with an empty interval trace. This already shows that $\wp \in \mathbb{T}_{\mathcal{M},\text{term}}^{\Im}$.

Now by definition of the number of steps $\#_{\downarrow}^{\wp}(\mathcal{M}) = n$. As $\mathcal{M}_n$ is a value and $\vdash \mathcal{M}_n : \{(\alpha, \wp_n, \tau - n)\}$ we get by inspection that $\tau - n = 0$, so $\tau = n = \#_{\downarrow}^{\wp}(\mathcal{M})$. $\qquad\square$

C.1.2 Pairwise Compatibility. We can easily see:

Lemma C.7 (Pairwise Compatibility). *If $\vdash \mathcal{M} : \{(\alpha_i, \wp_i, \tau_i) \mid i \in [n]\}$ then $\{\wp_i\}_i$ are pairwise compatible.*

C.1.3 Soundness.

Proposition C.8 (Soundness). *For every interval term M and $M \triangleleft \mathcal{M}$*

- $\displaystyle\bigvee_{\vdash \mathcal{M} : \mathcal{A}} \omega(\mathcal{A}) \le \mathbb{P}_{term}(M)$ • $\displaystyle\bigvee_{\vdash \mathcal{M} : \mathcal{A}} \mathbb{E}(\mathcal{A}) \le \mathbb{E}_{term}(M)$

Proof. Assume $\vdash \mathcal{M} : \mathcal{A}$ and $\mathcal{A} = \{(\alpha_i, \wp_i, \tau_i) \mid i \in [n]\}$. By Lem. C.6 each $\wp_i \in \mathbb{T}_{\mathcal{M},\text{term}}^{\Im}$. Furthermore, by Lem. C.7 the interval traces are pairwise compatible. By the Soundness of the interval-based semantics (Thm. 3.4) we, therefore, conclude that

$$\omega(\mathcal{A}) \le \mathbb{P}_{\text{term}}(M)$$

For the second claim we can again use Thm. 3.4 and the fact that $\tau_i = \#_{\downarrow}^{\wp_i}(\mathcal{M})$ (shown in Lem. C.6) and get

$$\mathbb{E}(\mathcal{A}) \le \mathbb{E}_{\text{term}}(M)$$

As this holds for all $\vdash \mathcal{M} : \mathcal{A}$ it also holds for the least upper bound. $\qquad\square$

C.2 Subject Expansion and Completeness

C.2.1 Subject Expansion.

Lemma C.9 (Reverse Substitution). *If $\vdash \mathcal{M}[\mathcal{N}_i/x_i]_{i\in[n]} : \mathcal{A}$ for distinct x_i then there exist a $\{a_i\}_{i\in[n]}$, s.t., $\{x_i : a_i\}_{i\in[n]} \vdash \mathcal{M} : \mathcal{A}$ and for all $i \in [n]$ and $\mathcal{B} \in a_i, \vdash \mathcal{N}_i : \mathcal{B}$*

Proof. Standard. By induction on $\mathcal{M}$. $\qquad\square$

Lemma C.10 (Deterministic Subject Expansion). *If $\vdash \mathcal{M} : \mathcal{A}$ and $\mathcal{M}' \rightarrow_{det} \mathcal{M}$ then $\vdash \mathcal{M}' : \mathcal{A}^{(\uparrow\epsilon,1)}$*

Proof. We assume w.l.o.g. that $\mathcal{A} \ne \emptyset$. Induction on $\vdash \mathcal{M} : \mathcal{A}$. Case analysis on $\mathcal{M}'$.

- $\mathcal{M}' = (\lambda x.\mathcal{N})\mathcal{P} \rightarrow_{det} \mathcal{N}[\mathcal{P}/x]$: So $\vdash \mathcal{N}[\mathcal{P}/x] : \mathcal{A}$. By Lem. C.9 we get an σ, s.t., $\{x : \sigma\} \vdash \mathcal{N} : \mathcal{A}$ and for all $\mathcal{B} \in \sigma, \vdash \mathcal{P} : \mathcal{B}$. We can conclude $\vdash \lambda x.\mathcal{N} : \{(\sigma \rightarrow \mathcal{A}, \epsilon, 1)\}$ using **(abs)** and can the derive $\vdash (\lambda x.\mathcal{N})\mathcal{P} : \mathcal{A}^{(\uparrow\epsilon,1)}$ via **(app)** .

- $\mathcal{M}' = (\mu_x^{\varphi}.\mathcal{N})\mathcal{P} \rightarrow_{det} \mathcal{N}[\mathcal{P}/x, (\mu_x^{\varphi}.\mathcal{N})/\varphi]$. So $\vdash \mathcal{N}[\mathcal{P}/x, (\mu_x^{\varphi}.\mathcal{N})/\varphi] : \mathcal{A}$. By Lem. C.9 we get a_x, a_{φ} such that $\{x : a_x, \varphi : a_{\varphi}\} \vdash \mathcal{N} : \mathcal{A}$ and for all $\mathcal{B} \in a_x$, $\vdash \mathcal{P} : \mathcal{B}$ and all $\mathcal{B} \in a_{\varphi}, \vdash \mu_x^{\varphi}.\mathcal{N} : \mathcal{B}$. We can thus type $\vdash \mu_x^{\varphi}.\mathcal{N} : \{(a_x \rightarrow \mathcal{A}, \epsilon, 0)\}$ using **(fix)** and conclude $\vdash (\mu_x^{\varphi}.\mathcal{N})\mathcal{P} : \mathcal{A}^{(\uparrow\epsilon,1)}$ via **(app)** .

- $\mathcal{M}' = \text{if}(\underline{[a,b]}, \mathcal{N}, \mathcal{P}) \rightarrow_{det} \mathcal{N}$ and $b \le 0$ and $\vdash \mathcal{N} : \mathcal{A}$. We can type $\vdash \underline{[a,b]} : \{([a,b], \epsilon, 0)\}$ via **(num)** and as $b \le 0$ we can derive $\vdash \text{if}(\underline{[a,b]}, \mathcal{N}, \mathcal{P}) : \mathcal{A}^{(\uparrow\epsilon,1)}$ using **(if)** .

- $\mathcal{M}' = \text{if}(\underline{a,b}, \mathcal{N}, \mathcal{P}) \rightarrow_{det} \mathcal{P}$ and $a > 0$. Similar as the case above.

- $\mathcal{M}' = f(\underline{[a,b]}, \underline{[c,d]}) \rightarrow_{det} \hat{f}(a,b,c,d)$: So $\vdash \hat{f}(a,b,c,d) : \mathcal{A}$, so we get that $\mathcal{A} = \{(\hat{f}(a,b,c,d), \epsilon, 0)\}$ as only **(num)** is applicable. We can type $\vdash \underline{[a,b]} : \{([a,b], \epsilon, 0)\}$ via **(num)** and similar for $\underline{[c,d]}$ and can conclude using (f_2) .

- $\mathcal{M}' = \mathcal{N}'\mathcal{P} \rightarrow_{det} \mathcal{N}\mathcal{P}$ and $\mathcal{N}' \rightarrow_{det} \mathcal{N}$. As $\vdash \mathcal{N}\mathcal{P} : \mathcal{A}$ we get that the last step must have been:

$$\frac{\vdash \mathcal{N} : \mathcal{B} \qquad \{\vdash \mathcal{P} : \mathcal{D} \mid \forall (\sigma \rightarrow C, \wp, \tau) \in \mathcal{B}, \mathcal{D} \in \sigma\}}{\vdash \mathcal{N}\mathcal{P} : \bigcup_{(\sigma \rightarrow C, \wp, \tau) \in \mathcal{B}} C^{(\uparrow\wp, \tau+1)} = \mathcal{A}} \text{ (app)}$$

By IH we get $\vdash \mathcal{N}' : \mathcal{B}^{(\uparrow\epsilon,1)}$ and conclude using **(app)** by choosing the same derivations as in the original derivation.

- $\mathcal{M}' = \text{if}(\mathcal{N}', \mathcal{P}, \mathcal{Q}) \rightarrow_{det} \text{if}(\mathcal{N}, \mathcal{P}, \mathcal{Q})$ and $\mathcal{N}' \rightarrow_{det} \mathcal{N}$. As $\vdash \text{if}(\mathcal{N}, \mathcal{P}, \mathcal{Q})$ we get that the last step must have been via **(if)** . As in the previous case we can apply induction choose the same derivations for $\mathcal{P}$ and $\mathcal{Q}$ and conclude back via **(if)** .

- $\mathcal{M}' = \text{score}(\mathcal{N}'), \mathcal{M}' = f(\mathcal{N}', \mathcal{P}), \mathcal{M}' = f(\underline{[a,b]}, \mathcal{N}')$. Trivial.

$\square$

Lemma C.11 (Probabilistic Subject Expansion). *If $\vdash \mathcal{M}_i : \mathcal{A}_i$ and $\mathcal{M} \rightarrow_{[a_i,b_i]} \mathcal{M}_i$ where $\{[a_i, b_i]\}_i$ are almost disjoint then*

$$\vdash \mathcal{M} : \bigcup_i \mathcal{A}_i^{(\uparrow[a_i,b_i],1)}$$

Proof. We can assume that $\mathcal{A}_i \ne \{\}$ as this case is trivial. By induction on $\mathcal{M}$.

- $\mathcal{M} = \text{sample}$: Then $\mathcal{M}_i = \underline{[a_i, b_i]}$ and as $\vdash \mathcal{M}_i : \mathcal{A}_i$ we get that $\mathcal{A}_i = \{([a_i, b_i], \epsilon, 0)\}$ as the last step must be via **(num)** . As $[a_i, b_i]$ are almost disjoint we can use the **(sample)** -rule to type sampleas required.

- $\mathcal{M} = \mathcal{N}\mathcal{P}$ and $\mathcal{N}$ does a reduction step, i.e., $\mathcal{N} \rightarrow_{[a_i,b_i]} \mathcal{N}_i$ and $\mathcal{M}_i = \mathcal{N}_i\mathcal{P}$. As $\vdash \mathcal{N}_i\mathcal{P} : \mathcal{A}_i$ we get that the last step must have been:

$$\frac{\vdash \mathcal{N}_i : \mathcal{B}_i \qquad \{\vdash \mathcal{P} : \mathcal{D} \mid \forall(\sigma \to C, \wp, \tau) \in \mathcal{B}_i, \mathcal{D} \in \sigma\}}{\vdash \mathcal{N}_i\mathcal{P} : \bigcup_{(\sigma \to C, \wp, \tau) \in \mathcal{B}_i} C^{(\uparrow\wp, \tau+1)} = \mathcal{A}_i} \text{ (app)}$$

By induction we get $\vdash \mathcal{N} : \bigcup_i \mathcal{B}_i^{(\uparrow[a_i, b_i], 1)}$. Define $\mathcal{B} \triangleq \bigcup_i (\mathcal{B}_i)^{(\uparrow[a_i, b_i], 1)}$. We get $\{\vdash \mathcal{P} : \mathcal{D} \mid \forall(\sigma \to C, \wp, \tau) \in \mathcal{B}, \mathcal{D} \in c\}$ as $\mathcal{B}$ is just the concatenation of all $\mathcal{B}_i$, i.e., every type in $\mathcal{B}$ is in at least on $\mathcal{B}_i$. By using (app) we can thus type $\vdash \mathcal{N}\mathcal{P} : \bigcup_{(\sigma \to C, \wp, \tau) \in \mathcal{B}} C^{(\uparrow\wp, \tau+1)}$.

$$\bigcup_{(\sigma \to C, \wp, \tau) \in \mathcal{B}} C^{(\uparrow\wp, \tau+1)} = \bigcup_i \bigcup_{(\sigma \to C, \wp, \tau) \in \mathcal{B}_i} C^{\uparrow([a_i, b_i]\wp, \tau+1+1)}$$
$$= \bigcup_i \mathcal{A}_i^{(\uparrow[a_i, b_i], 1)}$$

as required.

- $\mathcal{M} = \text{if}(\mathcal{N}, \mathcal{P}, Q)$ and $\mathcal{N}$ does a reduction step, i.e., $\mathcal{N} \to_{[a_i, b_i]} \mathcal{N}_i$ and $\mathcal{M}_i = \text{if}(\mathcal{N}_i, \mathcal{P}, Q)$. The last step in each derivation must have been via (if) so $\vdash \mathcal{N}_i : \mathcal{B}_i$,

$$\{\vdash \mathcal{P} : C_{([a, b], \wp, \tau), i} \mid ([a, b], \wp, \tau) \in \mathcal{B}_i, b \le 0\}$$

and

$$\{\vdash Q : \mathcal{D}_{([a, b], \wp, \tau), i} \mid ([a, b], \wp, \tau) \in \mathcal{B}_i, a > 0\}$$

and

$$\mathcal{A}_i = \bigcup_{([a, b], \wp, \tau) \in \mathcal{B}_i | b \le 0} C_{([a, b], \wp, \tau), i}^{(\uparrow\wp, \tau)} \cup \bigcup_{([a, b], \wp, \tau) \in \mathcal{B}_i | a > 0} \mathcal{D}_{([a, b], \wp, \tau)}^{(\uparrow\wp, \tau), i}$$

By induction we get $\vdash \mathcal{N} : \bigcup_i \mathcal{B}_i^{(\uparrow[a_i, b_i], 1)} \triangleq \mathcal{B}$. Now each element $([a, b], \wp, \tau) \in \mathcal{B}$ stems from exactly one category i (from one $\mathcal{B}_i$). So elements in $\mathcal{B}$ can be seen as having the from $([a, b], \wp, \tau), i \in \mathcal{B}$. (This is just needed to take care of the indices). Using (if) we can type

$$\vdash \text{if}(\mathcal{N}, \mathcal{P}, Q) : \bigcup_{([a, b], \wp, \tau), i \in \mathcal{B} | b \le 0} C_{([a, b], \wp, \tau), i}^{(\uparrow\wp, \tau)} \cup \bigcup_{([a, b], \wp, \tau), i \in \mathcal{B} | a > 0} \mathcal{D}_{([a, b], \wp, \tau), i}^{(\uparrow\wp, \tau)}$$
$$= \bigcup_i \left(\bigcup_{([a, b], \wp, \tau) \in \mathcal{B}_i | b \ge 0} C_{([a, b], \wp, \tau), i}^{(\uparrow\wp, \tau)} \cup \bigcup_{([a, b], \wp, \tau) \in \mathcal{B}_i | a < 0} \mathcal{D}_{([a, b], \wp, \tau), i}^{(\uparrow\wp, \tau)} \right)$$
$$= \bigcup_i \mathcal{A}_i^{(\uparrow[a_i, b_i], 1)}$$

as required.

- $\mathcal{M} = f(\mathcal{N}, \mathcal{P})$ and $\mathcal{N}$ does a reduction step, i.e., $\mathcal{N} \to_{[a_i, b_i]} \mathcal{N}_i$ and $\mathcal{M}_i = f(\mathcal{N}_i, \mathcal{P})$. As $\vdash f(\mathcal{N}_i, \mathcal{P}) : \mathcal{A}_i$ we get that the last step must have been via (f_2), i.e., $\vdash \mathcal{N}_i : \mathcal{B}_i$, $\{\vdash \mathcal{P} : C_{([a, b], \wp, \tau), i} \mid ([a, b], \wp, \tau) \in \mathcal{B}_i\}$ and

$$\mathcal{A}_i = \bigcup_{([a, b], \wp, \tau) \in \mathcal{B}_i} \bigcup_{([c, d], \wp', \tau') \in C_{([a, b], \wp, \tau), i}} \{(\hat{f}(a, b, c, d), \wp\wp', \tau + \tau' + 1)\}$$

By induction we get $\vdash \mathcal{N} : \bigcup_i \mathcal{B}_i^{(\uparrow[a_i, b_i], 1)} \triangleq \mathcal{B}$. We can now conclude using (f_2) as in the previous cases.

- $\mathcal{M} = f([a, b], \mathcal{N})$ and $\mathcal{N}$ does a reduction step, i.e., $\mathcal{N} \to_{[a_i, b_i]} \mathcal{N}_i$ and $\mathcal{M}_i = f([a, b], \mathcal{N}_i)$. The last steps must thus have been:

$$\frac{\dfrac{}{\vdash [a, b] : \{([a, b], \epsilon, 0)\}} \text{ (num)} \qquad \vdash \mathcal{N}_i : \mathcal{B}_{([a, b], \epsilon, 0), i}}{\vdash f([a, b], \mathcal{N}_i) : \bigcup_{([c, d], \wp, \tau) \in \mathcal{B}_{([a, b], \epsilon, 0), i}} \{(\hat{f}(a, b, c, d), \wp, \tau + 1)\}} (f_2)$$

By induction we get $\vdash \mathcal{N} : \bigcup_i \mathcal{B}_i^{(\uparrow[a_i, b_i], 1)} \triangleq \mathcal{B}$. We can trivially conclude via (f_2) and (num).

- $\mathcal{M} = \text{score}(\mathcal{N})$ and $\mathcal{N}$ does a reduction step, i.e., $\mathcal{N} \to_{[a_i, b_i]} \mathcal{N}_i$ and $\mathcal{M}_i = \text{score}(\mathcal{N}_i)$. The last step must have been via (score). We can apply induction and trivially conclude via (score).

□

Lemma C.12 (Subject Expansion)**.** *It holds that:*

- *If* $\vdash \mathcal{M} : \mathcal{A}$ *and* $\mathcal{N} \to_{det} \mathcal{M}$ *then* $\vdash \mathcal{N} : \mathcal{A}^{(\uparrow\epsilon, 1)}$
- *If* $\vdash \mathcal{M}_i : \mathcal{A}_i$ *and* $\mathcal{N} \to_{[a_i, b_i]} \mathcal{M}_i$ *where* $\{[a_i, b_i]\}_i$ *are almost disjoint then*

$$\vdash \mathcal{N} : \bigcup_i \mathcal{A}_i^{(\uparrow[a_i, b_i], 1)}$$

Proof. Follows from Lem. C.10 and Lem. C.11. □

C.2.2 Completeness.

Naïve Attempt on Completeness: We have seen in the soundness proof that if $\vdash \mathcal{M} : \{(\alpha_i, \wp_i, \tau_i) \mid i \in [n]\}$ each $\wp_i$ is a terminating trace. For completeness we would like to reverse that process and show that any pairwise compatible set of traces can be achieved via a type derivations: That is, if $\{\wp_i \mid i \in [n]\} \subseteq \mathbb{T}_{\mathcal{M}, \text{term}}^{\mathfrak{I}}$ are pairwise compatible then

$$\vdash \mathcal{M} : \{(\alpha_i, \wp_i, \#_\downarrow^{\wp_i}(\mathcal{M})) \mid i \in [n]\}$$

for some types $\{\alpha_i\}_{i \in [n]}$. This would immediately give us a completeness theorem as the interval-based semantics is itself complete. However, the above does **not** hold.

Example C.13. As an example consider the following simple term: $\mathcal{M} \triangleq \text{if}\left(\text{sample} - \frac{1}{2}, \text{sample}, \underline{0}\right)^{2\mathfrak{I}}$ Then the two interval traces $\wp_1 \triangleq [0, \frac{1}{2}][0, \frac{1}{2}]$, $\wp_2 \triangleq [0, \frac{1}{3}][\frac{1}{2}, 1]$ are clearly compatible but cannot be typed with the above system. The interested reader is advised to try find a typing derivation.

Strong Pairwise Compatibility. To show completeness, we need to introduce the new concept of *strong compatibility*. We call $\wp_1, \wp_2$ *strongly compatible* if $\wp_1 \leftrightarrow \wp_2$ is derivable by the following rules

$$\frac{}{\epsilon \leftrightarrow [a, b]\wp} \qquad \frac{[a, b], [c, d] \text{ are almost disjoint}}{[a, b]\wp_1 \leftrightarrow [c, d]\wp_2}$$

$$\frac{}{[a, b]\wp \leftrightarrow \epsilon} \qquad \frac{\wp_1 \leftrightarrow \wp_2}{[a, b]\wp_1 \leftrightarrow [a, b]\wp_2}$$

Strongly compatible traces are either pairwise almost disjoint in the first position or agree on the first position and the remainder is also strongly compatible. If two traces are strongly compatible they can thus share a common, identical prefix but must be pairwise almost disjoint at the first position where they differ. Clearly every strongly compatible pair of interval traces is also compatible but not the other way around. As an example the two traces in Ex. C.13 are compatible but not strongly compatible. We can show the following:
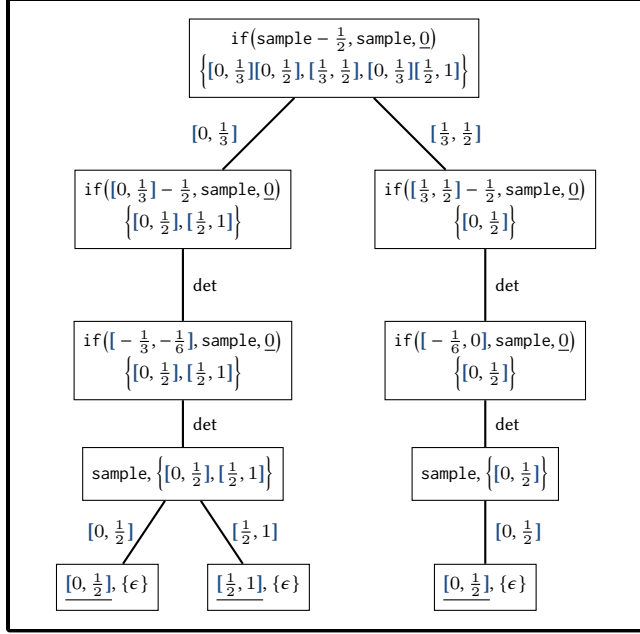
Figure 14. Example reduction for the term from Ex. C.13 on a set of pairwise strongly compatible traces. Probabilistic and deterministic reduction steps are arranged as a tree.

Lemma C.14. *If* $\{\wp_i \mid i \in [n]\} \subseteq \mathbb{S}_{\mathfrak{I}}$ *then there exists interval traces* $\{\wp'_j \mid j \in [m]\} \subseteq \mathbb{S}_{\mathfrak{I}}$ *that are pairwise* strongly *compatible with* $\bigcup_{i\in[n]} (\!|\wp_i|\!) = \bigcup_{j\in[m]} (\!|\wp'_j|\!)$ *and for each* $j \in [m]$, $(\!|\wp'_j|\!) \subseteq (\!|\wp_i|\!)$ *for some* $i \in [n]$.

Proof. We can give a constructive proof: We first analyse $\{\wp_i(0)\}_{i\in[n]}$, i.e., the interval at the first position. Clearly there exists intervals $\{[a_k, b_k]\}_{k\in\mathcal{K}}$ for a finite $\mathcal{K}$ that are all pairwise almost disjoint such that for each i there is a set $\mathcal{K}_i \subseteq \mathcal{K}$ with $\wp_i(0) = \bigcup_{k\in\mathcal{K}_i} [a_k, b_k]$. This holds as we can always partition at overlapping position. We can thus replace every $\wp_i$ with $|\mathcal{K}_i|$ many interval traces by replacing the first interval with the intervals in $[a_k, b_k]$ from $k \in \mathcal{K}_i$. The resulting set of standard traces agrees with the one we started from. The first position of those traces are either pairwise identical or almost disjoint as required by the definition of strong compatibility. For all traces that are identical on the first position we can proceed inductively. □

The two traces in Ex. C.13 are not strongly compatible but can be replaced by the 3 traces $\wp'_1 \triangleq [0, \frac{1}{3}][0, \frac{1}{2}]$, $\wp'_2 \triangleq [\frac{1}{3}, \frac{1}{2}][0, \frac{1}{2}]$ and $\wp'_3 \triangleq [0, \frac{1}{3}][\frac{1}{2}, 1]$ that denote the same set of standard traces but are pairwise strongly compatible.

Subject Expansion For Strongly Compatible Traces. The crucial observation is that in the the statement of subject expansion (Lem. C.12), the intervals in a probabilistic step should be pairwise almost disjoint. As we have seen in

the example above pairwise compatible traces must not necessarily be almost disjoint in the first position. But pairwise strongly compatible traces are: the first position is either almost disjoint or identical. To make use of this idea we represent the reduction of a term given a set of interval traces as a tree. Nodes in the tree are of the from $(\mathcal{M}, A)$ we $\mathcal{M}$ is an interval term and $\emptyset \neq A \subseteq \mathbb{T}^{\mathfrak{I}}_{\mathcal{M},\text{term}}$ a set of strongly compatible interval traces. The successors of a node are given by a relation $\leadsto$ where each transition is either labelled by det, to represents a deterministic reduction or by an interval $[a, b] \in \mathfrak{I}_{0,1}$:

$$\frac{\mathcal{M} \to_{\text{det}} \mathcal{N}}{(\mathcal{M}, A) \leadsto_{\text{det}} (\mathcal{N}, A)}$$

$$\frac{\mathcal{M} \to_{[a,b]} \mathcal{N} \qquad B = \{\wp \mid [a, b]\wp \in A\} \neq \emptyset}{(\mathcal{M}, A) \leadsto_{[a,b]} (\mathcal{N}, B)}$$

If we again consider the example term fro Ex. C.13 and the pairwise strongly compatible traces $\wp'_1, \wp'_2, \wp'_3$ from before we get the tree depicted in Fig. 14. Every det step corresponds to a deterministic reduction. For every probabilistic reduction the set of interval traces is stripped by its first position. As the set of traces is strongly compatible, the outgoing edges of every node are labelled by almost disjoint intervals.

Proposition C.15 (Completeness)**.** *If* $\{\wp_i \mid i \in [n]\} \subseteq \mathbb{T}^{\mathfrak{I}}_{\mathcal{M},\text{term}}$ *are pairwise* **strongly** *compatible then*

$$\vdash \mathcal{M} : \left\{(\alpha_i, \wp_i, \#^{\wp_i}_{\downarrow}(\mathcal{M})) \mid i \in [n]\right\}$$

for some types $\{\alpha_i\}_{i\in[n]}$

Proof. We first make the following easy observation that follows immediately by the definition of strong compatibility: If $A \subseteq \mathbb{T}^{\mathfrak{I}}_{\mathcal{M},\text{term}}$ is pairwise strongly compatible and $(\mathcal{M}, A) \leadsto^* (\mathcal{N}, B)$ and $(\mathcal{N}, B) \leadsto_{[a_i,b_i]} (\mathcal{N}_i, B_i)$ for $i \in [n]$ then $[a_i, b_i]$ are almost disjoint. Call this observation **(1)**. For our proof we consider the tree that is generated by $(\mathcal{M}, \{\wp_i \mid i \in [n]\})$. Note that this tree is finite. We claim that for every node $(\mathcal{N}, A)$ where $A = \{\dot\wp_i \mid i \in [k]\}$ in this tree we can type $\vdash \mathcal{N} : \left\{(\alpha_i, \dot\wp_i, \#^{\dot\wp_i}_{\downarrow}(\mathcal{N})) \mid i \in [k]\right\}$. We show this inductively by traversing the tree from the leafs up. Formally, we do induction on the shortest path to a leaf. In the base case, the node in question is a leaf: as by assumption each $\wp_i \in \mathbb{T}^{\mathfrak{I}}_{\mathcal{M},\text{term}}$ we get that each leaf of this tree has the from $(\mathcal{V}, \{\epsilon\})$ for some closed value $\mathcal{V}$. It is easy to check that for every value we can type $\vdash \mathcal{V} : \left\{(\alpha, \epsilon, 0)\right\}$ for some α, by either using **(num)** (in case of a numeral) or **(abs)** or **(fix)** followed by ⟨{}⟩ (in case of λ-or μ-abstraction). Now consider the case where $(\mathcal{N}, A)$ is a inner node. There are again two cases:

- $(\mathcal{N}, A) \leadsto_{\text{det}} (\mathcal{P}, A)$, so $\mathcal{N} \to_{\text{det}} \mathcal{P}$. Write $A = \{\dot\wp_i \mid i \in [k]\}$. By induction we can type $\vdash \mathcal{P} : \left\{(\alpha_i, \dot\wp_i, \#^{\dot\wp_i}_{\downarrow}(\mathcal{P})) \mid i \in [k]\right\}$. Now by Subject Expansion (Lem. C.12) we can type $\vdash \mathcal{N} : \left\{(\alpha_i, \dot\wp_i, \#^{\dot\wp_i}_{\downarrow}(\mathcal{P}) + 1) \mid i \in [k]\right\}$ as required,

- In the other case, $(\mathcal{N}, A) \leadsto_{[a_i, b_i]} (\mathcal{P}_i, B_i)$ for $i \in [m]$. Lets write $B_i = \{\wp_i^j \mid j \in [k_i]\}$. We have $A = \bigcup_{i \in [m]} \{[a_i, b_i] \wp_i^j \mid j \in [k_i]\}$. By induction we $\vdash \mathcal{P}_i : \{(\alpha_j^i, \dot{\wp}_j^i, \#_{\downarrow}^{\wp_j^i}(\mathcal{P})) \mid j \in [k_i]\} \triangleq \mathcal{A}_i$. By **(1)** we get that the $[a_i, b_i]$ are pairwise almost disjoint. By Lem. C.12 we can thus type $\vdash \mathcal{N} : \bigcup_i \mathcal{A}_i^{(\uparrow [a_i, b_i], 1)}$ as required.

$\square$

C.3 Soundness and Completeness

We can finally combine everything for a proof of Thm. 4.1.

Restatement of Thm. 4.1. *For every term $M \in \Lambda_0$,*

1. $\bigvee_{\vdash M^{2\Im} : \mathcal{A}} \omega(\mathcal{A}) = \mathbb{P}_{term}(M)$, *and*
2. *If M is AST,* $\bigvee_{\vdash M^{2\Im} : \mathcal{A}} \mathbb{E}(\mathcal{A}) = \mathbb{E}_{term}(M)$

Proof. We first show the first part:
We already showed $\bigvee_{\vdash M^{2\Im} : \mathcal{A}} \omega(\mathcal{A}) \leq \mathbb{P}_{term}(M)$ in Prop. C.8 as $M \triangleleft M^{2\Im}$. It remains to show that they are actually equal. Let $\epsilon > 0$. We show that there exist a $\vdash M^{2\Im} : \mathcal{A}$ such that $\omega(\mathcal{A}) \geq \mathbb{P}_{term}(M) - \epsilon$. Using the completeness of the interval-based semantics (Thm. 3.8), we get a *finite* set of pairwise compatible interval traces $\{\wp_j' \mid j \in [n]\} \subseteq \mathbb{T}_{M^{2\Im}, term}^{\Im}$ such that $\sum_{j \in [n]} \omega(\wp_j') \geq \mathbb{P}_{term}(M) - \epsilon$. Now from Lem. C.14 there exists a finite set of interval traces with the same weight that is furthermore pairwise strongly compatible. Let $\{\wp_i \mid i \in [m]\}$ be this set. Note that $\{\wp_i \mid i \in [m]\} \subseteq \mathbb{T}_{M^{2\Im}, term}^{\Im}$ as byLem. C.14 for every $i \in [m]$, $(\!(\wp_i)\!) \subseteq (\!(\wp_j')\!)$ for some $j \in [n]$. By Prop. C.15 we get that $\vdash M^{2\Im} : \{(\alpha_i, \wp_i, \#_{\downarrow}^{\wp_i}(M)) \mid i \in [m]\} \triangleq \mathcal{A}$ for some types α_i. Now obviously $\omega(\mathcal{A}) = \sum_i \omega(\wp_i') \geq \mathbb{P}_{term}(M) - \epsilon$, so we are done as we can let ϵ tend to 0.

For the second part we can proceed as before by using the second part of Thm. 3.8. $\square$

D Additional Material - Section 5

Restatement of Thm. 5.4. *A finite step distribution s is AST if and only if all of the following hold*

a) $\sum_{i \in \mathbb{Z}} s(i) = 1$ b) $s \neq \delta_0$ c) $\sum_{i \in \mathbb{Z}} i \cdot s(i) \leq 0$

Proof. $\Longleftarrow$: We begin with the (arguably more interesting direction) that the three conditions together imply AST. We first note that due to condition a) the error state, $\bot$, is never reachable. We can thus concentrate on paths consisting of natural numbers and can neglect the possibly of moving to the error state. Instead of considering the random walk on the half line we, we consider the more general wok on the integers, i.e., we remove the truncation at 0. That is the Markov chain $\mathfrak{M} = (\mathbb{Z}, \mathfrak{P})$ where the transition matrix $\mathfrak{P}$ is defined by $\mathfrak{P}(x, y) = s(y - x)$. It is easy to see that s is AST

if and only if $\mathfrak{M}$ eventually visits the non-positive numbers a.s.

We then begin by checking the third condition (c)) for equality of strict inequality:

- In the case of strict inequality, we have $\sum_{i \in \mathbb{Z}} i \cdot s(i) < 0$: Fix any starting state m as in the definition of AST. We define integer valued random variables $X_0, X_1, \cdots$ by $X_i = m + \sum_{k=1}^{i} Y_i$ where Y_i are independent random variables that are distributed according to s. It is easy to see that by the construction of the Markov chain $\mathfrak{M}$ we have $\mathfrak{P}^n(x, y) = \mathbb{P}(X_n = y)$, i.e., for the random variable X_i the probability of $X_i = y$ is the probability of being in state y after n steps. Here $\mathbb{P}$ is the probability distribution on the underlying (not specified) measurable space on which the X_is are defined.
  With $\mathbb{E}(X_i)$ we denote the expectation of X_i and with $Var(X_i)$ the variance defined in the standard way. With $\mathbb{E}(s)$ we denote the expectation of s. We obviously have $\mathbb{E}(X_i) = m + i \cdot \mathbb{E}(s)$ and as each of the Y_i are independent also $Var(X_i) = i \cdot Var(s)$. By assumption $\mathbb{E}(s) = \sum_{i \in \mathbb{Z}} i \cdot s(i) < 0$. Let $\epsilon = -\sum_{i \in \mathbb{Z}} i \cdot s(i) > 0$. So $\mathbb{E}(X_i) = m - i \cdot \epsilon$. All that remains now is to apply an appropriate concentration bound. Let N be such that for every $i > N$ we have $\mathbb{E}(X_i) < 0$, which exists as $\mathbb{E}(X_i) = m + i \cdot \epsilon$. For each $i > N$ we have:

$$\mathbb{P}(X_i > 0) \leq \mathbb{P}\left(|X_i - \mathbb{E}(X_i)| > -\mathbb{E}(X_i)\right)$$
$$\overset{(1)}{\leq} \frac{Var(X_i)}{(-\mathbb{E}(X_i))^2} = \frac{i \cdot Var(s)}{(m - \epsilon \cdot i)^2}$$

  where **(1)** follows from Chebyshev's inequality. If we let $i \to \infty$ we thus get that $\mathbb{P}(X_i > 0)$ converges to 0.
- In the case of equality, we have $\sum_{i \in \mathbb{Z}} i \cdot s(i) = 0$: First note that in this case the above reposing does not work. It does not even hold that $\mathbb{P}(X_i > 0)$ tends to 0 as it has in the previous case. We again use the same construction of the RV X_i as before. We will show that X_i does eventually become negative at least once.
  From condition c) together with b) we get that there exists an $i^* < 0$ with $s(i^*) > 0$. From any state k we can now reach a non-positive number in $\lceil k/i^* \rceil$ steps with probability at least $s(i^*)^{\lceil k/i^* \rceil} > 0$ (just take the relative change i^* so many times). Our proof now hinges on a famous theorem proved by George Pólya that states that any random walk on $\mathbb{Z}^1$ or $\mathbb{Z}^2$ with zero mean is recurrent (For a modern proof see e.g. [49]). As our random walk starts in m, i.e., $X_0 = m$ we thus get that the process $(X_i)_i$ does return to m with probability 1. We can then use the strong Markov property that states that if we have any stopping time τ (in our case the first time we revisit m) the process after τ is identical to the original one. We thus get that as $(X_i)_i$ is recurrent, i.e., visits m again almost-surely, it also visits m infinity many times a.s. As we have just shown, every time we visit m there is a (lower bounded) positive probability

(of $s(i^*)^{\lceil k/i^* \rceil} > 0$) of visiting a negative numbers. So we eventually visit a negative number with certainty (see the zero-one law in [42]).

$\boxed{\Rightarrow}$: We now show the other direction. We prove this by contraposition. It is easy to see that if $\sum_i s(i) < 1$ the walk is not AST as we have a positive probability of moving to the error state from any state. Similar if $s = \delta_0$ we are obviously not terminating. Lastly if $\sum_{i \in \mathbb{Z}} i \cdot s(i) > 0$ we can follow similar reasoning as in the case of strictly negative expectation and show that the expectation *increases* in each step. □

Restatement of Lem. 5.6. *If* $\{s_i\}_{i \in \mathcal{I}}$ *is a* finite *family of step distributions and each* s_i *is AST then* $\{s_i\}_{i \in \mathcal{I}}$ *is uniform AST.*

Proof. Fix any m. Fix any $\epsilon > 0$. By assumption

$$\lim_{n \to \infty} \mathfrak{P}_{s_i}^n(m, 0) = 1$$

for every i. So for any i there exist a $N_i \in \mathbb{N}$ such that for every $n \geq N_i$, $\mathfrak{P}_{s_i}^n(m, 0) \geq 1 - \epsilon$ (By definition of the limit). Now define $N = \sum_i N_i$ which is finite as $\mathcal{I}$ is finite.

Now choose any $n \geq N$. We claim

$$\inf_{i_1, \cdots, i_n} \mathfrak{P}_{s_{i_1}} \cdots \mathfrak{P}_{s_{i_n}}(m, 0) \geq 1 - \epsilon$$

which would immediately give us the result. Choose arbitrary indices $i_1, \cdots, i_n$. We show $\mathfrak{P}_{s_{i_1}} \cdots \mathfrak{P}_{s_{i_n}}(m, 0) \geq 1 - \epsilon$. As $n \geq N$ there must exists a $i_* \in \mathcal{I}$ that occurs at least N_{i_*}-many times among $i_1, \cdots, i_n$ by the pigeon hole principle.

As 0 is an absorbing state we can see that $\mathfrak{P}_{s_i} \mathfrak{P}(m, 0) \geq \mathfrak{P}(m, 0)$ for any stochastic matrix $\mathfrak{P}$ **(1)**. Note that multiplication is commutative, i.e., $\mathfrak{P}_{s_i} \mathfrak{P}_{s_j}(m, 0) = \mathfrak{P}_{s_j} \mathfrak{P}_{s_i}(m, 0)$ (This does *not* hold for general matrix multiplication but holds for $\mathfrak{P}_{s_i}$ as a random walk is invariant of the current state). We can thus reorder the indices $i_1, \cdots, i_n$ such that i_* fills the last N_{i_*} positions. Together with **(1)** we thus get

$$\mathfrak{P}_{s_{i_1}} \cdots \mathfrak{P}_{s_{i_n}}(m, 0) \geq \mathfrak{P}_{s_{i_*}}^{N_{i_*}}(m, 0) \geq 1 - \epsilon$$

as required. □

D.1 Proof of Thm. 5.9

This section is devoted to a proof of Thm. 5.9. We begin by giving a rough outline of the proof for orientation. The fundamental idea is to decompose the set of terminating traces according to the number of recursive calls made (not only on the first level). We formalize this decomposition by a special kind of tree structure called number tree. We then show a direct correspondence between number trees and and terminating runs of the random walk generated by $\{[\![\mu_x^\varphi.M \mid r]\!]\}_{r \in \mathbb{R}}$. Henceforth fix a term $\mu_x^\varphi.M$.

Number Trees. We define a *number tree* by the following:

$$\mathcal{S} \triangleq n \triangleright [\mathcal{S}_1, \cdots, \mathcal{S}_n]$$

where $n \in \mathbb{N}$. We can depict each number tree by viewing n as the label of the node and $\mathcal{S}_1, \cdots, \mathcal{S}_n$ as the children as
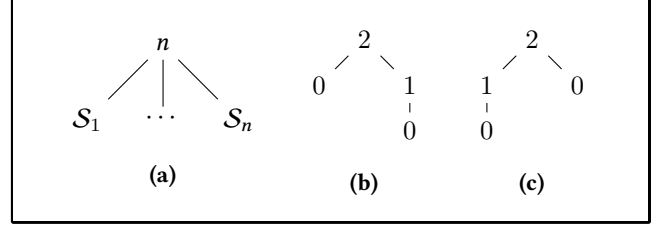

Figure 15. Example number trees.

depicted in Fig. 15a. Note that the simplest tree is given by $0 \triangleright []$. Two (distinct) example trees are given in Fig. 15b and Fig. 15c.

Summary Semantics. We define a *summary* as an element $\square_{r'}^r$, for $r, r' \in \mathbb{R}$. With $\mathfrak{D}$ we denote the set of all summaries and with $\mathbb{S}^\square \triangleq (\mathfrak{D} \cup \mathbb{R}_{[0,1]})^*$ the set of summary traces. We can define a summary semantics working on summary traces in Fig. 16. For every recursive call, we substitute in a summary, i.e., an abbreviation for a trace, on the traces. Note that this semantics closely corresponds to the semantics in Fig. 5 used to count the recursive calls. The only difference is that we do not blindly substitute in a dummy value $\star$ but predefine the outcome via a summary[8]. We set

$$\mathbb{T}_{r \mapsto r'}^\square = \{s \in \mathbb{S}^\square \mid \langle \mathbf{body}_{\mu_x^\varphi.M}(r), s \rangle \xrightarrow{\square}^* \langle \underline{r'}, \epsilon \rangle\}$$

as all summary traces on which the term on argument r evaluates to argument r'.

Number Trees as Traces. The summary semantics explicitly lists recursive calls, as we can view the summary $\square_{r'}^r$ as a placeholder for a trace such that $(\mu_x^\varphi.M)\underline{r}$ evaluates to $\underline{r'}$. The summaries allow us to partition the set of terminating traces according to the number of calls made on each level. We can specify the number of calls by a number tree. For a tree $n \triangleright [\mathcal{S}_1, \cdots, \mathcal{S}_n]$ there should be n direct recursive calls and inside those calls the number of calls is inductively specified by $\mathcal{S}_i$. We consider the following example for some intuition:

Example D.1. Consider the term $\mu_x^\varphi.\varphi(\varphi x) \oplus (\underline{0} \oplus \varphi x)$ and the number tree in Fig. 15b. All traces that correspond to this tree should make 2 recursive calls in the first level. In the first of those calls, no further call is made and on the second a single one is made and afterwards none. This corresponds to the following set of terminating traces:

$$\{s \in \mathbb{S}^7 \mid s_1 \in [0, \tfrac{1}{2}], s_2 \in (\tfrac{1}{2}, 1], s_3 \in [0, \tfrac{1}{2}],$$
$$s_4 \in (\tfrac{1}{2}, 1], s_5 \in (\tfrac{1}{2}, 1], s_6 \in (\tfrac{1}{2}, 1], s_7 \in [0, \tfrac{1}{2}]\}$$

---

[8]Note that, unlike in $\xrightarrow{\star}$ we do not need to count the number of calls, as we can simply count the number of summaries in a trace which equals the number of calls made.

$$\frac{}{\langle(\lambda x.M)V, s\rangle \xrightarrow{\square} \langle M[V/x], s\rangle}$$

$$\frac{}{\langle(\underline{\mu})\underline{r}, \square_{r'}^r :: s\rangle \xrightarrow{\square} \langle\underline{r'}, s\rangle}$$

$$\frac{}{\langle \texttt{sample}, r :: s\rangle \xrightarrow{\square} \langle\underline{r}, s\rangle}$$

$$\frac{r \leq 0}{\langle\texttt{if}(\underline{r}, N, P), s\rangle \xrightarrow{\square} \langle N, s\rangle}$$

$$\frac{r > 0}{\langle\texttt{if}(\underline{r}, N, P), s\rangle \xrightarrow{\square} \langle P, s\rangle}$$

$$\frac{r \geq 0}{\langle\texttt{score}(\underline{r}), s\rangle \xrightarrow{\square} \langle\underline{r}, s\rangle}$$

$$\frac{\langle R, s\rangle \xrightarrow{\square} \langle M, s'\rangle}{\langle E[R], s\rangle \xrightarrow{\square} \langle E[M], s'\rangle}$$

$$\frac{}{\langle f(\underline{r_1}, \cdots, \underline{r_{|f|}}), s\rangle \xrightarrow{\square} \langle\underline{f(r_1, \cdots, r_{|f|})}, s\rangle}$$

Figure 16. Small-step reduction rules for $\xrightarrow{\square}$.

Definition D.2. For each number tree $\mathcal{S}$ we can define a family of sets of traces $\{\mathbb{A}_{r \mapsto r'}^{\mathcal{S}}\}_{r, r' \in \mathbb{R}}$ by induction on $\mathcal{S}$ as follows:

$$\frac{s_1 \square_{r_1'}^{r_1} s_2 \cdots \square_{r_n'}^{r_n} s_{n+1} \in \mathbb{T}_{r \mapsto r'}^{\square} \qquad \left\{\dot{s}_{r_i'}^{r_i} \in \mathbb{A}_{r_i \mapsto r_i'}^{\mathcal{S}_i}\right\}_{i=1}^n}{s_1 \dot{s}_{r_1'}^{r_1} s_2 \cdots \dot{s}_{r_n'}^{r_n} s_{n+1} \in \mathbb{A}_{r \mapsto r'}^{n \triangleright [\mathcal{S}_1, \cdots, \mathcal{S}_n]}}$$

Elements in $\mathbb{A}_{r \mapsto r'}^{n \triangleright [\mathcal{S}_1, \cdots, \mathcal{S}_n]}$ are thus obtained by taking every summary trace with exactly n summaries that takes r to r'. For the ith summary (the ith recursive call) we then substitute in a trace from $\mathbb{A}_{r_i \mapsto r_i'}^{\mathcal{S}_i}$ that is recursively obtained from the ith child ($\mathcal{S}_i$). Every number tree thus defines a specific set of traces. By induction it is easy to see that for distinct trees the obtained sets of traces are disjoint. The interested reader is advised to match this definition with Ex. D.1. We define $\mathbb{A}_{r \mapsto \mathbb{R}}^{\mathcal{S}} \triangleq \bigcup_{r' \in \mathbb{R}} \mathbb{A}_{r \mapsto r'}^{\mathcal{S}}$ as all terminating traces with recursion according to $\mathcal{S}$. It is easy to see that $\mathbb{A}_{r \mapsto \mathbb{R}}^{\mathcal{S}}$ is measurable.

Probability Distributions on Number Trees. We can view a number tree as being sampled from a counting distribution $t : \mathbb{N} \to \mathbb{R}_{[0,1]}$. For every node we sample a number n according to p, add n nodes and continue by sampling the child nodes. Every counting distribution $t : \mathbb{N} \to \mathbb{R}_{[0,1]}$ thus gives a natural probability to a number tree $\mathcal{S}$ as just the product over all nodes in $\mathcal{S}$. More generally we define:

Definition D.3. For a family of counting distributions $\{t_k\}_k : \mathbb{N} \to \mathbb{R}_{[0,1]}$ and a number tree $\mathcal{S}$ we define $\mathbb{P}_{\inf}^{\{t_k\}_k}(\mathcal{S})$ by induction as

$$\mathbb{P}_{\inf}^{\{t_k\}_k}(n \triangleright [\mathcal{S}_1, \cdots, \mathcal{S}_n]) \triangleq \left(\inf_k t_k(n)\right) \cdot \prod_{i=1}^n \mathbb{P}_{\inf}^{\{t_k\}_k}(\mathcal{S}_i)$$

where we follow the usual convention that $\prod_{i=1}^0 = 1$.

Note that if $\{t_k\}_k$ consist of a single element $\mathbb{P}_{\inf}^{\{t_k\}_k}(\mathcal{S})$ for a tree $\mathcal{S}$ is the product of the probability of every node in that tree. Taking the infimum follows the general scheme as e.g. in the definition uniform AST (Def. 5.5). We show that if

we take the family $(\llbracket \mu_x^\varphi.M \mid r \rrbracket)_{r \in \mathbb{R}}$ the probability of tree is a lower bound on the measure of $\mathbb{A}_{r \mapsto \mathbb{R}}^{\mathcal{S}}$.

Example D.4. For Ex. D.1 we get that the family $(\llbracket \mu_x^\varphi.M \mid r \rrbracket)_{r \in \mathbb{R}}$ comprises a single element, namely the function t defined by $t(2) = \frac{1}{2}$, $t(1) = \frac{1}{4}$ and $t(0) = \frac{1}{4}$. The probability of the tree $\mathcal{S}$ in Fig. 15b, as defined above then equals $\frac{1}{2}\frac{1}{4}\frac{1}{4}\frac{1}{4} = \frac{1}{128}$ which is less than or equal (in this case equal) to the measure of $\mathbb{A}_{r \mapsto \mathbb{R}}^{\mathcal{S}}$.

Proposition D.5. *For every number tree $\mathcal{S}$ and any r we have*

$$\mathbb{P}_{\inf}^{\{\llbracket \mu_x^\varphi.M \mid r' \rrbracket\}_{r'}}(\mathcal{S}) \leq \mu_{\mathbb{S}}\left(\mathbb{A}_{r \mapsto \mathbb{R}}^{\mathcal{S}}\right)$$

Proof. By induction on $\mathcal{S}$ with r universally quantified. Let $\mathcal{S} = n \triangleright [\mathcal{S}_1, \cdots, \mathcal{S}_n]$. We first analyse $\mathbb{A}_{r \mapsto \mathbb{R}}^{\mathcal{S}}$: by definition every $s \in \mathbb{A}_{r \mapsto r'}^{\mathcal{S}}$ has the from $s = s_1 \dot{s}_{r_1'}^{r_1} s_2 \cdots \dot{s}_{r_n'}^{r_n} s_{n+1}$ for some $s_1 \square_{r_1'}^{r_1} s_2 \cdots \square_{r_n'}^{r_n} s_{n+1} \in \mathbb{T}_{r \mapsto r'}^{\square}$ and $\dot{s}_{r_i'}^{r_i} \in \mathbb{A}_{r_i \mapsto r_i'}^{\mathcal{S}_i}$. We can observe that $\{s_1 \cdots s_{n+1} \mid s_1 \square_{r_1'}^{r_1} s_2 \cdots \square_{r_n'}^{r_n} s_{n+1} \in \mathbb{T}_{r \mapsto r'}^{\square}, r' \in \mathbb{R}\} = \mathbb{T}_{\mathbf{body}_{\mu_x^\varphi.M}(r); n}^{\star}$ by comparing the relation $\xrightarrow{\star}$ and $\xrightarrow{\square}$. If we concatenate two sets of traces the measure of the new set is the product of the individual measures. As we know that n traces are substituted we can take the infimum over all possible arguments which gives us a trivial lower bound on $\mu_{\mathbb{S}}\left(\mathbb{A}_{r \mapsto \mathbb{R}}^{\mathcal{S}}\right)$:

$$\mu_{\mathbb{S}}\left(\mathbb{T}_{\mathbf{body}_{\mu_x^\varphi.M}(r); n}^{\star}\right) \cdot \inf_{r_1, \cdots, r_n} \prod_{i=1}^n \mu_{\mathbb{S}}\left(\mathbb{A}_{r_i \mapsto \mathbb{R}}^{\mathcal{S}_i}\right) \leq \mu_{\mathbb{S}}\left(\mathbb{A}_{r \mapsto \mathbb{R}}^{\mathcal{S}}\right) \quad \textbf{(i)}$$

By induction we get that $\mathbb{P}_{\inf}^{\{\llbracket \mu_x^\varphi.M \mid r' \rrbracket\}_{r'}}(\mathcal{S}_i) \leq \mu_{\mathbb{S}}\left(\mathbb{A}_{\dot{r} \mapsto \mathbb{R}}^{\mathcal{S}_i}\right)$ for every i and every $\dot{r}$. Note that the left hand side does not depend on $\dot{r}$ so in particular

$$\prod_{i=1}^n \mathbb{P}_{\inf}^{\{\llbracket \mu_x^\varphi.M \mid r' \rrbracket\}_{r'}}(\mathcal{S}_i) \leq \inf_{r_1, \cdots, r_n} \prod_{i=1}^n \mu_{\mathbb{S}}\left(\mathbb{A}_{r_i \mapsto \mathbb{R}}^{\mathcal{S}_i}\right) \quad \textbf{(ii)}$$

We can now put this all together and get:

$$\mathbb{P}^{\{\|\mu_x^\varphi.M|r'\|\}_{r'}}_{\inf}(\mathcal{S}) \overset{(1)}{=} \inf_{r'} \|\mu_x^\varphi.M \mid r' \|(n) \cdot \prod_{i=1}^{n} \mathbb{P}^{\{\|\mu_x^\varphi.M|r'\|\}_{r'}}_{\inf}(\mathcal{S}_i)$$

$$\overset{(2)}{\le} \|\mu_x^\varphi.M \mid r \|(n) \cdot \inf_{r_1,\cdots,r_n} \prod_{i=1}^{n} \mu_{\mathbb{S}}(\mathbb{A}^{\mathcal{S}_i}_{r_i \mapsto \mathbb{R}})$$

$$\overset{(3)}{\le} \mu_{\mathbb{S}}(\mathbb{T}^\star_{\mathbf{body}_{\mu_x^\varphi.M}(r);n}) \cdot \inf_{r_1,\cdots,r_n} \prod_{i=1}^{n} \mu_{\mathbb{S}}(\mathbb{A}^{\mathcal{S}_i}_{r_i \mapsto \mathbb{R}})$$

$$\overset{(4)}{\le} \mu_{\mathbb{S}}(\mathbb{A}^{\mathcal{S}}_{r \mapsto \mathbb{R}})$$

where **(1)** follows from the definition of $\mathbb{P}^{\{\|\mu_x^\varphi.M|r'\|\}_{r'}}_{\inf}(\mathcal{S})$, **(2)** from the fact that $\inf_{r'} \|\mu_x^\varphi.M \mid r' \|(n) \le \|\mu_x^\varphi.M \mid r \|(n)$ together with fact **(ii)**, **(3)** from the definition of $\|\mu_x^\varphi.M \mid r'\|$ and **(4)** from **(i)**. □

Number Trees as Terminating Runs. It is easy to see that for every family of subprobability mass functions on the natural numbers $\{t_k\}_k$, $\sum_\mathcal{S} \mathbb{P}^{\{t_k\}_k}_{\inf}(\mathcal{S}) \le 1$. Here the sum is taken over the countable set of (finite) number trees. What we can show is the following:

Lemma D.6. *If $\{t_k\}_k$ is a family of counting distributions and $\{\overline{t_k}\}_k$ is uniform AST then*

$$\sum_\mathcal{S} \mathbb{P}^{\{t_k\}_k}_{inf}(\mathcal{S}) = 1$$

Proof. We define the following set of *absolute* runs, i.e., sequences of states:

$$Runs_A \triangleq \{U \in \mathbb{N}^* \mid U(i+1) - U(i) \ge -1, U(0) = 1,$$
$$U(|u|-1) = 0, \forall 1 \le i < |U|-1 : U(i) \neq 0\}$$

Think of elements in $Runs_A$ as terminating runs of the Markov chain that start in state 1 and eventually reach state 0. The condition $U(i+1) - U(i) \ge -1$ is there to ensure that in each step the value never decrease by more than 1 (Note that $\overline{t_k}$ never assign positive probability to values less than -1). We associate a probability, $\mathrm{P}(U)$ to elements $U \in Runs_A$ by:

$$\mathrm{P}(U) = \inf_{k_0,\cdots,k_{|U|-2}} \prod_{i=0}^{|U|-2} \mathfrak{P}_{\overline{t_{k_i}}}(U(i), U(i+1))$$

which is just the probability of that run when taking the infimum over all possible choices of transition distribution. What we observe now is that

$$\lim_{n \to \infty} \left(\inf_{k_1,\cdots,k_n} \mathfrak{P}_{\overline{t_{k_1}}} \cdots \mathfrak{P}_{\overline{t_{k_n}}}(1,0) \right) = \sum_{U \in Runs_A} \mathrm{P}(U)$$

, i.e., the probability of eventually reaching 0 from 1 is the same as the sum over the probability of each path that terminates starting in 1. By assumption $(\overline{t_k})_k$ is uniformly AST so the left hand side equals 1. Call this **(1)**. Instead of analysis the absolute path we can also consider the relative change
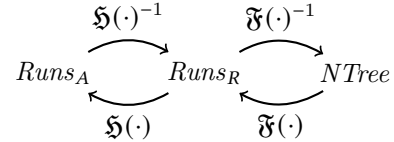
in each step. We define

$$Runs_R \triangleq \{u \in (\mathbb{N} \cup \{-1\})^* \mid \sum_{i=0}^{|u|-1} u(i) = -1,$$
$$\forall m < |u|-1 \sum_{i=0}^{m} u(i) > -1\}$$

Each element $u \in Runs_R$ gives the relative change in each step such that starting from state 1 we eventual terminate. The sum of the relative change should thus be -1 but the sum of every strict prefix is at least 0 (so that termination only occurs in the last step). There exists a bijective correspondence between elements in $Runs_A$ and $Runs_R$: For each $u \in Runs_R$, define $\mathfrak{H}(u) \in Runs_A$ as the sequence of length $|u|+1$ defined by $\mathfrak{H}(u)(i) \triangleq 1 + \sum_{j=0}^{i-1} u(j)$. It is easy to verify that $\mathfrak{H}(\cdot)$ is a bijection.

Now lastly we observe that there is a bijection between the set of number trees and $Runs_R$. For each number tree $\mathcal{S}$ we inductively define a sequence of integers $\mathfrak{F}(\mathcal{S}) \in Runs_R$ by

$$\mathfrak{F}(n \triangleright [\mathcal{S}_1, \cdots, \mathcal{S}_n]) = (n-1) :: \mathfrak{F}(\mathcal{S}_1) \cdots \mathfrak{F}(\mathcal{S}_n)$$

It is an easy proof to show that $\mathfrak{F}(\cdot)$ forms a bijection. We thus have the bijective situation depicted below.

$$Runs_A \underset{\mathfrak{H}(\cdot)}{\overset{\mathfrak{H}(\cdot)^{-1}}{\rightleftarrows}} Runs_R \underset{\mathfrak{F}(\cdot)}{\overset{\mathfrak{F}(\cdot)^{-1}}{\rightleftarrows}} NTree$$

It is now easy to see that for every number tree $\mathcal{S}$, $\mathbb{P}^{\{t_k\}_k}_{\inf}(\mathcal{S}) = \mathrm{P}(\mathfrak{H}(\mathfrak{F}(\mathcal{S})))$ as $\mathbb{P}^{\{t_k\}_k}_{\inf}$ gives probability of the relative change $\{t_k\}_k$ which is exactly the same as weighting the transition directly as in the definition of P.

As $\mathfrak{H} \circ \mathfrak{F}$ is a bijection and by **(1)** we thus get $\sum_\mathcal{S} \mathbb{P}^{\{t_k\}_k}_{\inf}(\mathcal{S}) = 1$ as required. □

Restatement of Thm. 5.9. *If $\{\overline{\|\mu_x^\varphi.M \mid r\|}\}_{r \in \mathbb{R}}$ is uniform AST then $\mu_x^\varphi.M$ terminates a.s. on every argument.*

Proof. We obviously have $\mathbb{T}_{(\mu_x^\varphi.M)\underline{r},\text{term}} \supseteq \biguplus_\mathcal{S} \mathbb{A}^{\mathcal{S}}_{r \mapsto \mathbb{R}}$ (in fact they are equal but we do not require this for the proof). We can thus deduce:

$$\mu_{\mathbb{S}}(\mathbb{T}_{(\mu_x^\varphi.M)\underline{r},\text{term}}) \ge \mu_{\mathbb{S}}\left(\biguplus_\mathcal{S} \mathbb{A}^{\mathcal{S}}_{r \mapsto \mathbb{R}} \right)$$

$$\overset{(1)}{=} \sum_\mathcal{S} \mu_{\mathbb{S}}(\mathbb{A}^{\mathcal{S}}_{r \mapsto \mathbb{R}})$$

$$\overset{(2)}{\ge} \sum_\mathcal{S} \mathbb{P}^{\{\|\mu_x^\varphi.M|r'\|\}_{r'}}_{\inf}(\mathcal{S})$$

$$\overset{(3)}{=} 1$$

where **(1)** follows from the fact that $\mathbb{A}^{\mathcal{S}}_{r\mapsto\mathbb{R}}$ is disjoint for distinct number trees, **(2)** from Prop. D.5 and **(3)** from Lem. D.6.

□

D.2 Partial Order on Counting Distributions

We first show:

Lemma D.7. *Let $p, q : \mathbb{N} \to \mathbb{R}_{[0,1]}$ be finite counting distributions with $p \sqsubseteq q$. If $\overline{p}$ is an AST step distribution then $\overline{q}$ is an AST step distribution.*

Proof. For convenience lets write $\hat{p}(i) \triangleq \sum_{j \leq i} p(j)$ similarly for $\hat{q}$. As $p \sqsubseteq q$, we have $\hat{p}(i) \leq \hat{q}(i)$ for every i. We now use Thm. 5.4: As $\overline{p}$ is an AST step distribution we have $\sum_i i \cdot p(i) = 1$, $\sum_i i \cdot p(i) \leq 1$ and $p \neq \delta_1$. We now show that q satisfies all those conditions as well and can then use the other direction from Thm. 5.4.

As p is finite we can view $p : \{0, \cdots, k\} \to \mathbb{R}_{[0,1]}$ for some k. As $\sum_i i \cdot p(i) = 1$ we get $\hat{p}(k) = 1$ and thus by assumption $\sum_i i \cdot q(i) = \hat{q}(k) = 1$ **(1)**. We can hence also view q as a function $q : \{0, \cdots, k\} \to \mathbb{R}_{[0,1]}$. As $\sum_i i \cdot p(i) \leq 1$ and $p \neq \delta_1$ we get that $\hat{p}(0) = p(0) > 0$, so $q(0)$ is also positive and thus $q \neq \delta_1$ **(2)**.

In the following, it remains to show that $\sum_i i \cdot q(i) \leq 1$. We use the combinatorial fact that $\sum_{i\in\mathbb{N}} i \cdot p(i) = \sum_{i\in\mathbb{N}} \sum_{j>i} p(j)$ and show:

$$\sum_{i\in\mathbb{N}} i \cdot p(i) = \sum_{i\in\mathbb{N}} \sum_{j>i} p(j) = \sum_i \left(1 - \sum_{j\leq i} p(j)\right)$$
$$= \sum_{i=0}^{k} \left(1 - \hat{p}(i)\right)$$
$$= (k+1) - \sum_{i=0}^{k} \hat{p}(i)$$

Analogously $\sum_{i\in\mathbb{N}} i \cdot q(i) = (k+1) - \sum_{i=0}^{k} \hat{q}(i)$. As $\hat{p}(i) \leq \hat{q}(i)$ for all i we get:

$$\sum_i i \cdot q(i) = (k+1) - \sum_{i=0}^{k} \hat{q}(i)$$
$$\leq (k+1) - \sum_{i=0}^{k} \hat{p}(i)$$
$$= \sum_i i \cdot p(i)$$
$$\leq 1 \qquad\qquad\qquad \textbf{(3)}$$

We are done as **(1)**, **(2)** and **(3)** imply that $\overline{q}$ is AST by Thm. 5.4.

□

Similarly we can show the following as we can easily extend $\sqsubseteq$ to distributions on $\mathbb{N}$ that result from runs of the Markov chain $\mathfrak{P}_{t_i}$.

Restatement of Lem. 5.10. *If $s, \{t_i\}_{i\in\mathcal{I}}$ are counting distributions and for all $i \in \mathcal{I}$, $s \sqsubseteq t_i$ and $\overline{s}$ is AST then $\{\overline{t_i}\}_{i\in\mathcal{I}}$ is uniform AST.*

D.3 Ensure Progress

The problem with formally counting the number of recursive calls is that the returned value of a prior call can influence not only what the next calls are but also how many calls are made. As an example consider $\mu_x^\varphi.\text{if } fx \text{ then } fx \text{ else } fx + fx$ where the number of recursive calls is either 2 or 3 depending on the outcome of the first. We present a type system that guarantees evaluation via $\to^\star$ to succeed, i.e., whenever $\textbf{body}_{\mu_x^\varphi.M}(r)$ is typable, $\sum_n [\![\mu_x^\varphi.M \mid r]\!] (n) = 1$ for all $r \in \mathbb{R}$. There are two conceptually different reasons why $\sum_n [\![\mu_x^\varphi.M \mid r]\!] (n) = 1$. Either we get stuck (on a non-null set of traces) on terms of the from $\text{if}(\star, N, P)$ or $\text{score}(\star)$. The other case is to get stuck on terms of the from $\text{score}(r)$ for $r < 0$. We focus on the first cause, which informally occurs whenever a recursive outcome is subsequently used in guards or scores and thereby influences the control flow. The second cause, on the other hand depends on the concrete denotation of a program, and at such can not be analysed statically.

The crux of our approach is thus to disallow the outcome of recursive calls to influence branching in the programs, i.e., recursive outcomes may not be used inside guards of conditionals or score-constructs. Obviously, this cannot be characterised purely syntactically as the property we seek is semantic in nature. We enforce this by a more involved simple type system where we add a dedicated type $\mathbf{R}^\top$ for recursive outcomes that cannot be used within guards. We define simple types in Fig. 17a. The idea of $\mathbf{R}^\top$ being more restrictive than $\mathbf{R}$ can be formalized via a subtyping relation given in Fig. 17b. Typing judgments are of the form $\Gamma \vdash M : \alpha$ and given in Fig. 17c. The crucial step is the rule for conditionals combined with the fixpoint rule. For conditionals we require that the term in the guard position has $\mathbf{R}$ and at the same time the recursive abstraction $\overline{\mu}$ has the more restrictive return type $\mathbf{R}^\top$. Combined with subtyping this gives a semantic guarantee that the recursive abstraction cannot be used inside conditionals. Note that the type system works with the simplified fixpoint constructs, $\overline{\mu}$. For now on we assume that the fixed $\mu_x^\varphi.M$ satisfies $\vdash \textbf{body}_{\mu_x^\varphi.M}(r) : \mathbf{R}^\top$ for some r[9].

Whenever $\textbf{body}_{\mu_x^\varphi.M}(r)$ is typable in the system in Fig. 17 recursive outcomes cannot be used inside the conditionals or score constructs. If, in addition, no score-constructs get stuck, this already ensures that $\xrightarrow{\star}$ enjoys progress[10].

---

[9]We obviously have that $\textbf{body}_{\mu_x^\varphi.M}(r)$ is typable for some r iff it is typable for all r.

[10]While we can statically ensure that a recursive outcome, $\star$, never occurs insider a guard or a score-construct, we can not ensure that we only score

$$\alpha, \beta \triangleq \mathbf{R} \mid \mathbf{R}^\top \mid \alpha \to \beta \qquad\qquad \overline{\alpha \sqsubseteq \alpha} \qquad\qquad \overline{\mathbf{R} \sqsubseteq \mathbf{R}^\top} \qquad\qquad \frac{\alpha' \sqsubseteq \alpha \qquad \beta \sqsubseteq \beta'}{\alpha \to \beta \sqsubseteq \alpha' \to \beta'}$$

(a)   **(b)**

$$\frac{x : \alpha \in \Gamma}{\Gamma \vdash x : \alpha} \qquad \frac{\Gamma; x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \to \beta} \qquad \overline{\Gamma \vdash \boxed{\mu} : \mathbf{R}^\top \to \mathbf{R}^\top} \qquad \overline{\Gamma \vdash \underline{r} : \mathbf{R}} \qquad \overline{\Gamma \vdash \mathtt{sample} : \mathbf{R}}$$

$$\frac{\Gamma \vdash M : \alpha \qquad \alpha \sqsubseteq \beta}{\Gamma \vdash M : \beta} \qquad \frac{\Gamma \vdash M : \alpha \to \beta \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta} \qquad \frac{\Gamma \vdash M : \mathbf{R} \qquad \Gamma \vdash N : \alpha \qquad \Gamma \vdash P : \alpha}{\Gamma \vdash \mathtt{if}(M, N, P) : \alpha}$$

$$\frac{\Gamma \vdash M_1 : \mathbf{R} \quad \cdots \quad \Gamma \vdash M_{|f|} : \mathbf{R}}{\Gamma \vdash f(M_1, \cdots, M_{|f|}) : \mathbf{R}} \qquad \frac{\Gamma \vdash M : \mathbf{R}}{\Gamma \vdash \mathtt{score}(M) : \mathbf{R}} \qquad \frac{\Gamma \vdash M_1 : \mathbf{R}^\top \quad \cdots \quad \Gamma \vdash M_{|f|} : \mathbf{R}^\top}{\Gamma \vdash f(M_1, \cdots, M_{|f|}) : \mathbf{R}^\top}$$

(c)

Figure 17. Simple typing judgments that guarantee that recursive outcomes are never used inside inside conditionals.

Lemma D.8. *If $\mathbf{body}_{\mu_x^\varphi.M}(r)$ is typable in the system from Fig. 17 and no subterm of the from $\mathtt{score}(r)$ for $r < 0$ is reachable, then $\sum_n [\![\mu_x^\varphi.M \mid r]\!](n) = 1$ for all r.*

Proof. We extend the system in Fig. 17 by the axiom $\Gamma \vdash \star : \mathbf{R}^\top$ and cn show subject reduction w.r.t. $\overset{\star}{\to}$. As terms of the from $\mathtt{if}(\star, N, P)$ or $\mathtt{score}(\star)$ are not typable in Fig. 17 execution via $\overset{\star}{\to}$ can never reach terms that contain such subterms. As by assumption no $\mathtt{score}$-construct can fail, our reduction does enjoy progress. This directly implies that $\sum_n [\![\mu_x^\varphi.M \mid r]\!](n) = 1$ for all r by the same argument as in [40, Lem. 7]. ∎

D.4 Intersection Counting System

To count the number of occurrences we employ a non-idempotent intersection (NII) type system. Intersection types are defined by the mutually recursive grammar $\alpha, \beta \triangleq \mathbf{R} \mid a \to \alpha$ and $a \triangleq [\alpha_1, \cdots, \alpha_n]$ where $[\alpha_1, \cdots, \alpha_n]$ denotes a *multiset*. A typing context Γ is a partial map from variables to intersections. The disjoint union of two contexts Γ, Δ denoted by $\Gamma \uplus \Delta$ is the elementwise disjoint union of multiset. For an overview on non-idempotent intersection types see [10]. Typing judgments are of the from $\Gamma \vdash M : \alpha$ and given by the rules in Fig. 18. Due to the non-idempotent nature, for each type derivation we can read of the number of semantic occurrences as the cardinality of the intersection type. E.g. $\mu_x^\varphi.M$ is a first-order fixpoint and $\{\varphi : a, x : b\} \vdash M : \mathbf{R}$ we get a path in which φ is used exactly $|a|$-many times. Here $|a|$ denote the cardinality of an intersection type.

---

on non-negative values. Checking if the argument of every score is non-negative requires the inspection of the denotion of a subprogram and is thus very involved. For most interesting program it is, however, easy to verify the concrete score value as it is e.g. a constant.

We can easily see that the type system gives a upper bound on the recursive rank, as each type derivation outlines a possible execution and the cardinality of a intersection type represents the semantic use cases of a variable.

Lemma D.9. *Let $\mu_x^\varphi.M$ be a first-order fixpoint term with recuive rank m (as defined in Sec. 5.4). Then $m \le \max\limits_{\{\varphi:a,x:b\} \vdash M : \mathbf{R}} |a|$*

This lemma justifies the use of the NII-type system to upper bound the recursive rank and thus make use of Cor. 5.13 without computing the recursive rank directly. A direct computation would involve probabilistic reasoning, whereas the type system gives a "easy to compute" upper bound. Note that for a term $\mu_x^\varphi.M$, the quantity $\max\limits_{\{\varphi:a,x:b\} \vdash M : \mathbf{R}} |a|$ used in Lem. D.9 is effectively computable.

D.5 Further example

We consider the Ex. 5.15, i.e., the addition of Ex. 5.1 where we use probabilistic outcomes as first class citizens. Recall that p is the acceptance probability of a print. For each print we first sample a value e uniform on $[0, 1]$ which represents how broken the product is, i.e., $e = 0$ is a completely fine product and $e = 1$ would correspond to a total failure. Whenever the quality is less than p we accept the print. In the other case, as before in Ex. 5.1, there is a chance of $sig(x)$ of the staff being tired and making mistakes. In case a mistake is made, we do however not have a fair binary choice between printing 2 or 3 copies, but instead this depends on the quality of the most recent print q. With probability e we reprint 3 copies and otherwise only 2. With increasing e, i.e., the more damaged the last print was, the more likely it is to reprint 3 instead of 2. The term we analyse in Ex. 5.15 is the following

$$\frac{}{\{x : [\alpha]\} \vdash x : \alpha} \qquad \frac{\Gamma; x : a \vdash M : \alpha}{\Gamma \vdash \lambda x.M : a \to \alpha} \qquad \frac{\Gamma \vdash M : [\alpha_i] \to \beta \quad \{\Gamma_i \vdash N : \alpha_i\}}{\uplus_i \Gamma_i \uplus \Gamma \vdash MN : \beta}$$

$$\frac{\Gamma \vdash M : \mathbf{R} \quad \Delta \vdash N : \alpha}{\Gamma \uplus \Delta \vdash \mathtt{if}(M, N, P) : \alpha} \qquad \frac{\Gamma \vdash M : \mathbf{R} \quad \Delta \vdash P : \alpha}{\Gamma \uplus \Delta \vdash \mathtt{if}(M, N, P) : \alpha} \qquad \frac{}{\emptyset \vdash \mathtt{sample} : \mathbf{R}}$$

$$\frac{\Gamma_1 \vdash M_1 : \mathbf{R} \quad \cdots \quad \Gamma_{|f|} \vdash M_{|f|} : \mathbf{R}}{\Gamma_1 \uplus \cdots \uplus \Gamma_{|f|} \vdash f(M_1, \cdots, M_{|f|}) : \mathbf{R}} \qquad \frac{}{\emptyset \vdash \underline{r} : \mathbf{R}} \qquad \frac{\Gamma \vdash M : \mathbf{R}}{\Gamma \vdash \mathtt{score}(M) : \mathbf{R}}$$

Figure 18. Non-idempotent Intersection Type System that counts the number of semantic uses of a variable. Note that this system is not syntax-guided due to the two rules for conditionals.
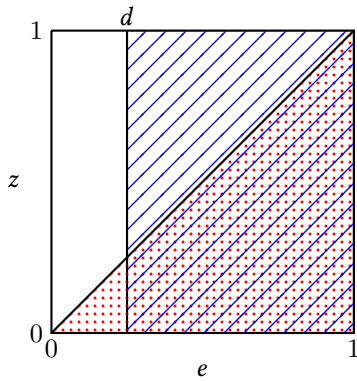
Figure 19. A geometric interpretation of the probability in Sec. D.5. Each point in the square correspond to a value of e (the error value sampled in the let) and z the sampled value in the binary choice. The blue (striped) area are all value pairs such that $e > d$, i.e., all sampled values for e such that the first conditional takes the right branch. The red (dotted) area contains all value pairs such that $z \le e$, i.e., the left branching in the binary sample is taken.

(parameterised by p):

$$\mu_x^\varphi.\mathtt{let}\ e = \mathtt{sample}\ \mathtt{in}\ \mathtt{if}\ e \le p\ \mathtt{then}\ x\ \mathtt{else}$$
$$\left(\left(\varphi^3(x+1) \oplus_e \varphi^2(x+1)\right) \oplus_{sig(x)} \varphi^2(x+1)\right)$$

In particular, note the use of a probabilistic sample (e) as a first class value and the subsequent use as a probability $\oplus_e$. Such behavior cannot be modelled via discrete distributions, as the quality e is an intrinsic continuous value.

We want to check for which instantiation of p this term is AST on every input. To make use of Thm. 5.9 we extract the counting pattern $\llbracket \mu_x^\varphi.M \mid r \rrbracket$ for the program above. If we fix p this would become easier. However, for our demonstration, we treat p as a variable. As we want the p to stay flexible, this can be done via some basic geometric reasoning. It is easy to see that $\llbracket \mu_x^\varphi.M \mid r \rrbracket (0) = p$, $\llbracket \mu_x^\varphi.M \mid r \rrbracket (1) = 0$ and $\llbracket \mu_x^\varphi.M \mid r \rrbracket (n) = 0$ for all $n > 3$, so it remains to compute $\llbracket \mu_x^\varphi.M \mid r \rrbracket (2)$ and $\llbracket \mu_x^\varphi.M \mid r \rrbracket (3)$. Lets start

with $\llbracket \mu_x^\varphi.M \mid r \rrbracket (3)$. In order to make 3 recursive calls we must have that the sampled value e satisfies $e > p$ and in the later binary choice ($\oplus_e$) the sampled value (lets cal it z) must satisfy $z \le e$. If we let e and z be sampled iid from a uniform distribution on $[0, 1]$ we can interpret the desired probability as volume of the intersection of the red (dotted) and blue (striped) area in Fig. 19. We can compute this volume which is $\frac{1-p^2}{2}$. We thus get $\llbracket \mu_x^\varphi.M \mid r \rrbracket (3) = sig(r) * \frac{1-p^2}{2}$. Similarly we can compute $\llbracket \mu_x^\varphi.M \mid r \rrbracket (2) = (1 - p)(1 - \frac{1+p}{2} sig(r))$.

We now want to use Thm. 5.9 to find values of p such that the term is AST on every input. We again make use of Lem. 5.10. We define

$$s := p\delta_0 + \frac{(1-p)^2}{2}\delta_2 + \frac{1-p^2}{2}\delta_3$$

It is routine to check that $s \sqsubseteq \llbracket \mu_x^\varphi.M \mid r \rrbracket$ for every r. To analyse for which p $\bar{s}$ is AST we use Thm. 5.4; so we need to find values for p such that the expectation of p is less than or equal 1 [11]. We can compute:

$$p * 0 + \frac{(1-p)^2}{2} * 2 + \frac{1-p^2}{2} * 3 \le 1$$
$$\Leftrightarrow (1 - p)^2 + \frac{3(1-p^2)}{2} \le 1$$
$$\Leftrightarrow -\frac{1}{2}p^2 - 2p + \frac{3}{2} \le 0$$
$$\Leftrightarrow (p \le -2 - \sqrt{7}) \vee (p \ge \sqrt{7} - 2)$$

As we assumed $p \in [0, 1]$, $\bar{s}$ is AST iff $p \ge \sqrt{7} - 2$. We have $s \sqsubseteq \llbracket \mu_x^\varphi.M \mid r \rrbracket$ for every r so we can appeal to Lem. 5.10 and get that $\{\llbracket \mu_x^\varphi.M \mid r \rrbracket_r\}_{r \in \mathbb{R}}$ is uniform AST whenever $p \ge \sqrt{7} - 2$. By Thm. 5.9 we can thus conclude that when $p \ge \sqrt{7} - 2$ the program is AST on every input.

This example demonstrated well, that when we use (continuous) random outcomes as first class values, the analysis becomes very intricate. Such examples can not be expressed in PHORS [34] or with binary probabilistic choice [20, 32, 50]. Our framework can analyse such example efficiently.

---

[11] Note that this is equivalent to the fact that the exception of $\bar{s}$ (which is shifted by -1) is less than or equal 0

As we will see in the next section, we can automate this process entirely. I.e., the probably computation and the derivation of s can be done fully automatically (for a fixed $p \geq \sqrt{7} - 2$).

E Additional Material - Section 6

E.1 Detailed Algorithm Description

In this section we give a detailed (and formal) description of our algorithm.

Symbolic Execution Trees. The key step is to evaluate a term symbolically and use sample variables to postpone sample decision (c.f. symbolic terms in Sec. B.5). We extend the syntax of symbolic terms by a new symbol, $\circledast$, that will be used as an unknown argument. We now present symbolic execution as a big-step semantics, where branching on the term level is represented as branching of a tree. As we are, in particular, interested in recursive calls we annotate each call made in the semantics. We define (symbolic) executions trees by:

$$ETree \ni \mathfrak{T} \triangleq \circ \, \mathfrak{B} \;\; | \; \boxed{\mu} \, (\mathfrak{T}) \;\; | \; \boxed{s} \, (\mathfrak{B})(\mathfrak{T})$$
$$| \; \circ(\mathfrak{B})(\mathfrak{T}_1, \mathfrak{T}_2) \; | \; \bullet(\mathfrak{B})(\mathfrak{T}_1, \mathfrak{T}_2)$$

where $\mathfrak{B} \in Val$ is a symbolic value[12]. We choose a more space-economical way to present trees. We occasionally depict execution trees as tree of degree 2 where $\circ(\mathfrak{B})(\mathfrak{T}_1, \mathfrak{T}_2)$ represents a binary branch. Note that an execution tree condenses all of the information we are interested in. For branching, it records the symbolic value as the condition; for score, it records the symbolic value that is scored, and finally every recursive call is recorded. To construct an execution tree from a program it remains to fold[13] execution trees:

Definition E.1. Given a function $H : Val \to ETree$ we can define the lifted tree fold $H^\dagger : ETree \to ETree$ by induction as follows:

$$H^\dagger\big(\circ \, \mathfrak{B} \,\big) = H(\mathfrak{B})$$
$$H^\dagger\big(\boxed{\mu} \, (\mathfrak{T}) \,\big) = \boxed{\mu} \, (H^\dagger \mathfrak{T})$$
$$H^\dagger\big(\boxed{s} \, (\mathfrak{B})(\mathfrak{T}) \,\big) = \boxed{s} \, (\mathfrak{B})(H^\dagger \mathfrak{T})$$
$$H^\dagger \, (\circ(\mathfrak{B})(\mathfrak{T}_1, \mathfrak{T}_2)) = \circ(\mathfrak{B})(H^\dagger \mathfrak{T}_1, H^\dagger \mathfrak{T}_2)$$
$$H^\dagger \, (\bullet(\mathfrak{B})(\mathfrak{T}_1, \mathfrak{T}_2)) = \bullet(\mathfrak{B})(H^\dagger \mathfrak{T}_1, H^\dagger \mathfrak{T}_2)$$

We can now define a big-step semantics by giving a symbolic execution tree for each program, denoted $M \Downarrow \mathfrak{T}$. The big-step rules are given in Fig. 20 where $\beta(x)(y)$ performs a β-step, i.e., $\beta(\lambda x.M)(V) \triangleq M[V/x]$ and $\beta(\boxed{\mu})(V) \triangleq \star$. $\lambda\!\!\!\lambda$ binds the argument of an anonymous function. To avoid confusion, we use the special symbol to distinguish it from abstractions within our language. Note that this system inherits the structure of a standard big-step semantic (see e.g. [6]). As we execute symbolically and cannot resolve branching, we operate on trees and fold each reduction step. For each

---

[12]As we have done in Sec. 5.2, we extend symbolic values by a special symbol $\star$.

[13]Tree folding is standard in functional programming. In our case, fold traverses the tree and replaces every leaf with a tree given the folded function.
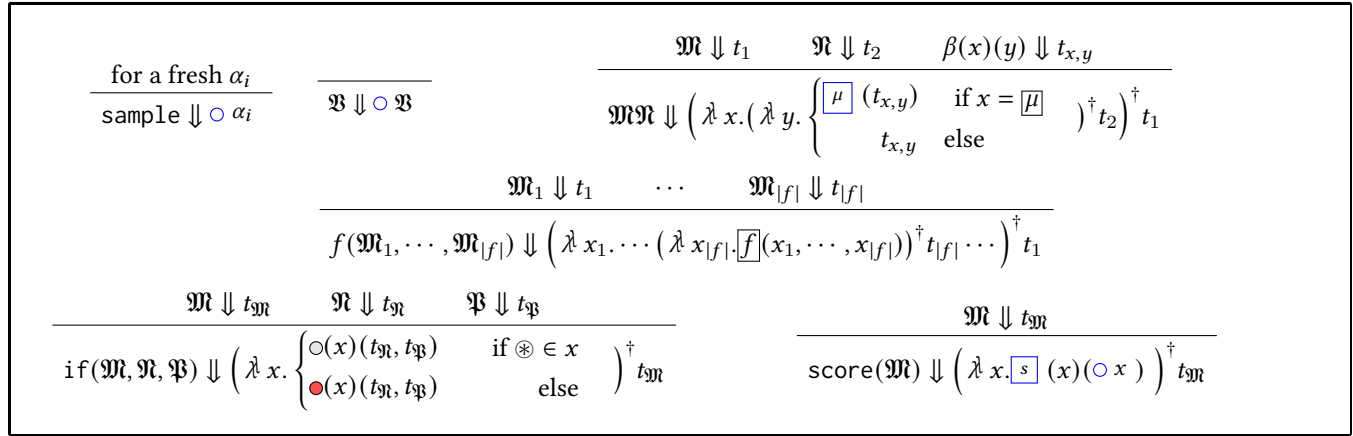
$$\frac{\text{for a fresh } \alpha_i}{\texttt{sample} \Downarrow \circ \alpha_i} \qquad \overline{\mathfrak{B} \Downarrow \circ \mathfrak{B}} \qquad \frac{\mathfrak{M} \Downarrow t_1 \qquad \mathfrak{N} \Downarrow t_2 \qquad \beta(x)(y) \Downarrow t_{x,y}}{\mathfrak{M}\mathfrak{N} \Downarrow \left(\lambda x. \left(\lambda y. \begin{cases} \boxed{\mu}\,(t_{x,y}) & \text{if } x = \boxed{\mu} \\ t_{x,y} & \text{else} \end{cases} \right)^{\dagger} t_2 \right)^{\dagger} t_1}$$

$$\frac{\mathfrak{M}_1 \Downarrow t_1 \qquad \cdots \qquad \mathfrak{M}_{|f|} \Downarrow t_{|f|}}{f(\mathfrak{M}_1, \cdots, \mathfrak{M}_{|f|}) \Downarrow \left(\lambda x_1. \cdots \left(\lambda x_{|f|}.\boxed{f}(x_1, \cdots, x_{|f|}) \right)^{\dagger} t_{|f|} \cdots \right)^{\dagger} t_1}$$

$$\frac{\mathfrak{M} \Downarrow t_{\mathfrak{M}} \qquad \mathfrak{N} \Downarrow t_{\mathfrak{N}} \qquad \mathfrak{P} \Downarrow t_{\mathfrak{P}}}{\texttt{if}(\mathfrak{M}, \mathfrak{N}, \mathfrak{P}) \Downarrow \left(\lambda x. \begin{cases} \circ(x)(t_{\mathfrak{N}}, t_{\mathfrak{P}}) & \text{if } \circledast \in x \\ \bullet(x)(t_{\mathfrak{N}}, t_{\mathfrak{P}}) & \text{else} \end{cases} \right)^{\dagger} t_{\mathfrak{M}}} \qquad \frac{\mathfrak{M} \Downarrow t_{\mathfrak{M}}}{\texttt{score}(\mathfrak{M}) \Downarrow \left(\lambda x.\boxed{s}\,(x)(\circ x\,) \right)^{\dagger} t_{\mathfrak{M}}}$$

Figure 20. Big-step symbolic execution where symbolic terms denote execution trees.
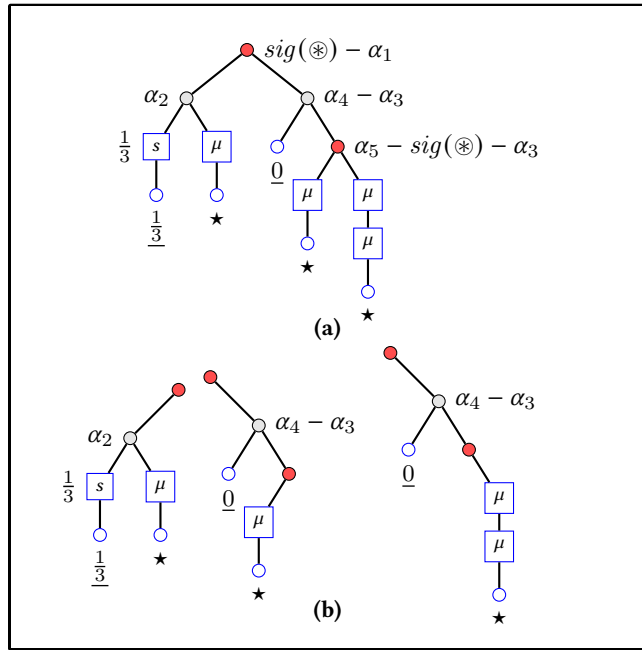
Figure 21. Symbolic execution trees for the running example and all possible strategies **(b)**.

resolved conditional we introduce a binary branch at every conditional, a $\boxed{s}\,(V)(\mathfrak{T})$ for every score construct, and a $\boxed{\mu}\,(\mathfrak{T})$ for every recursive call. For every term M there exist a $M \Downarrow \mathfrak{T}$ and $\mathfrak{T}$ is unique up to reordering of the sample variables. The term we analyse is $\mathbf{body}_{\mu_x^\varphi.M}(\circledast)$, i.e., the body with the argument replaced by the distinguished symbol $\circledast$.

Example E.2. As an running example to demonstrate our tool consider the following non-trivial term

$$\mu_x^\varphi.\big(\texttt{score}(\tfrac{1}{3}) \oplus \varphi\,x\big)\oplus_{sig(x)}\Big(\texttt{let } p = \texttt{sample in}$$
$$\underline{0} \oplus_p \big(\varphi\,x \oplus_{x+p} \varphi(\varphi\,x)\big)\Big)$$

where $sig(x)$ is the sigmoid function that squashes the real line into $[0, 1]$. Checking this program for AST is challenging as the analysis depends on a complex interplay between the actual argument x and the probabilistic outcomes. Note that for M as above: $[\![\mu_x^\varphi.M \mid r]\!] \neq [\![\mu_x^\varphi.M \mid r']\!]$, if $r \neq r'$. The term we analyse in our big-step system is $\mathbf{body}_{\mu_x^\varphi.M}(r)$ which in our case is:

$$\big(\texttt{score}(\tfrac{1}{3}) \oplus \boxed{\mu}\circledast\big)\oplus_{sig(\circledast)}$$
$$\Big(\texttt{let } p = \texttt{sample in } \underline{0} \oplus_p \big(\boxed{\mu}\circledast \oplus_{sig(\circledast)+p} \boxed{\mu}(\boxed{\mu}\circledast)\big)\Big)$$

The tree $\mathfrak{T}$ with $\mathbf{body}_{\mu_x^\varphi.M}(\circledast) \Downarrow \mathfrak{T}$ is depicted in Fig. 21a. The interested reader is advised to check the construction herself.

Strategies. Informally, each red inner node does contain the unknown argument $\circledast$ so we cannot determine its probabilistic behaviour without knowing its concrete value. The route we pursue here is to simply ignore every branching at red nodes and not treating it as a quantitative but non-deterministic branching. Loosely speaking, we let the environment decide which branch to take. We define strategies by

$$\mathfrak{S} \triangleq \circ\mathfrak{B} \mid \boxed{\mu}\,(\mathfrak{S}) \mid \boxed{s}\,(\mathfrak{B})(\mathfrak{S}) \mid \circ(\mathfrak{B})(\mathfrak{S}_1, \mathfrak{S}_2)$$
$$\mid \bullet(\mathfrak{B})(\mathfrak{S}, \times) \mid \bullet(\mathfrak{B})(\times, \mathfrak{S})$$

So strategies almost agree with execution trees but can choose which path to follow for each red node. A strategy $\mathfrak{S}$ is compatible with an execution tree $\mathfrak{T}$ (written $\mathfrak{S} \prec \mathfrak{T}$) if it matches the structure.

$$\mathsf{Const}^\star(\bigcirc\,\mathfrak{B}\,,\epsilon) \triangleq \mathbb{R}^m_{[0,1]}$$

$$\mathsf{Const}^\star(\boxed{\mu}\,(\mathfrak{T})\,,\kappa) \triangleq \mathsf{Const}^\star(\mathfrak{S},\kappa)$$

$$\mathsf{Const}^\star(\bullet(\mathfrak{S}_2)(\mathfrak{S}_1,\times)\mathfrak{B},L\kappa) \triangleq \mathsf{Const}^\star(\mathfrak{S}_1,\kappa)$$

$$\mathsf{Const}^\star(\bullet(\mathfrak{S}_2)(\times,\mathfrak{S}_1)\mathfrak{B},R\kappa) \triangleq \mathsf{Const}^\star(\mathfrak{S}_2,\kappa)$$

$$\mathsf{Const}^\star(\boxed{s}\,(\mathfrak{B})(\mathfrak{S})\,,\kappa) \triangleq \mathsf{Const}^\star(\mathfrak{S},\kappa) \cap \mathfrak{B}^{-1}[0,\infty)$$

$$\mathsf{Const}^\star(\bigcirc(\mathfrak{B})(\mathfrak{S}_1,\mathfrak{S}_2),L\kappa) \triangleq \mathsf{Const}^\star(\mathfrak{S}_1,\kappa) \cap \mathfrak{B}^{-1}(-\infty,0]$$

$$\mathsf{Const}^\star(\bigcirc(\mathfrak{B})(\mathfrak{S}_1,\mathfrak{S}_2),R\kappa) \triangleq \mathsf{Const}^\star(\mathfrak{S}_2,\kappa) \cap \mathfrak{B}^{-1}(0,\infty)$$

Figure 22. Inductive definition of $\mathsf{Const}^\star(\mathfrak{S},\kappa)$ for $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S})$.

Paths and Probability. As we arranged execution in a tree, we effectively postponed branching decision. Each branch in a strategy (or execution) corresponds to a branching path of the problem. A path is a sequence in $\kappa \in \{L,R\}^*$ that resolves binary branching decision. For a strategy $\mathfrak{S}$ we denote with $\mathsf{Paths}^{\blacktriangledown}(\mathfrak{T})$ the set of terminating paths, i.e., paths that lead to a leaf. In the first example strategy in Fig. 21b paths include LL and LR.

For any strategy $\mathfrak{S}$ and terminating path $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S})$ we count the numbers of recursive calls on that path, i.e., the number of times that a fixpoint node, $\boxed{\mu}\,(\cdot)$, is traversed. We denote this number with $|\boxed{\mu}|(\mathfrak{S},\kappa) \in \mathbb{N}$. For a set C of natural numbers we abbreviate $\mathsf{Paths}^{\blacktriangledown}(\mathfrak{S},C) \triangleq \{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}) \mid |\boxed{\mu}|(\mathfrak{S},\kappa) \in C\}$.

Assume that all sample variables occurring in a execution tree $\mathfrak{T}$ are within $\{\alpha_0,\cdots,\alpha_{m-1}\}$ and $\mathfrak{S} \prec \mathfrak{T}$ (so sample variables within $\mathfrak{S}$ are also within $\{\alpha_0,\cdots,\alpha_{m-1}\}$). Then each path $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S})$ denotes a measurable subset of $\mathbb{R}^m_{[0,1]}$ in the natural way as all assignment such that this path is followed. We denote this set with $\mathsf{Const}^\star(\mathfrak{S},\kappa) \subseteq \mathbb{R}^m_{[0,1]}$ and it is defined by induction in Fig. 22. $\mathsf{Const}^\star(\mathfrak{S},\kappa) \subseteq \mathbb{R}^m_{[0,1]}$ denotes the set of assignments for $\alpha_0,\cdots,\alpha_{m-1}$ such that the branching and score-constructs are evaluated according to κ. Red nodes are ignored as we do not interpret them probabilistically. It is easy to see that $\mathsf{Const}^\star(\mathfrak{S},\kappa)$ is measurable. We abbreviate $\mathbb{P}^\star(\mathfrak{S},\kappa) \triangleq \lambda_m(\mathsf{Const}^\star(\mathfrak{S},\kappa))$, i.e., the Lebesgue measure of all those assignments.

The Algorithm. We are now in a position to present our algorithm. Given a term $\mu_x^\varphi.M$ we begin by computing $\mathbf{body}_{\mu_x^\varphi.M}(\circledast) \Downarrow \mathfrak{T}_\circledast$. Note that such a tree always exist and is, up to sample variables, unique. For a strategy $\mathfrak{S}$, we abbreviate $\mathbb{P}^\star(\mathfrak{S},n) \coloneqq \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S},\{0,\cdots,n\})} \mathbb{P}^\star(\mathfrak{S},\kappa)$, i.e., the probability that in $\mathfrak{S}$ at most n call are made.

We can now define:

$$\mathbb{P}_{approx}(0) \coloneqq \min_{\mathfrak{S} \in Strat(\mathfrak{T}_\circledast)} \mathbb{P}^\star(\mathfrak{S},0)$$

$$\mathbb{P}_{approx}(n > 0) \coloneqq$$
$$\left(\min_{\mathfrak{S} \in Strat(\mathfrak{T}_\circledast)} \mathbb{P}^\star((\mathfrak{S},n))\right) - \left(\min_{\mathfrak{S} \in Strat(\mathfrak{T}_\circledast)} \mathbb{P}^\star((\mathfrak{S},n-1)\right)$$

We can understand $\mathbb{P}_{approx}(n)$ as the least probability that n calls are made even if the environment chooses in the worst (worst here meaning more recursive calls) possible way.

Example E.3. Consider all strategies for the running example listed in Fig. 21b. We can compute $\mathbb{P}_{\mathrm{approx}}(0) = \mathbb{P}_{\mathrm{approx}}(2) = \frac{1}{2}$ and $\mathbb{P}_{\mathrm{approx}}(n) = 0$ for all other n.

As the same sampling outcome can be used within multiple branching, we must make sure that the non-deterministic interpretation of branching does not lose any information. We call a execution tree $\mathfrak{T}$ *sufficiently independent* if every sample variable that is used in a red node is not used in the subtree rooted at that node. Informally speaking, this means that probabilistic outcomes that we over-approximated by switching to a non-deterministic view may not be used afterwards. They can, of course, be used prior to the non-deterministic node. The correctness of our approach is then stated as follows:

Restatement of Thm. 6.2. *If $\mathfrak{T}_\circledast$ is sufficiently independent, then for every $r \in \mathbb{R}$, $\mathbb{P}_{\mathrm{approx}} \sqsubseteq [\![\mu_x^\varphi.M \mid r]\!]$.*

Note that our approach still does not provide a straightforward way to implement it. While $\mathbf{body}_{\mu_x^\varphi.M}(\circledast) \Downarrow \mathfrak{T}_\circledast$ can be computed effectively and the (finitely many) strategies with $\mathfrak{S} \prec \mathfrak{T}$ can be enumerated we still to compute $\mathbb{P}(\mathfrak{S},n)$ and therefore the Lebesgue measure of a certain set. However, our approach does a big leap towards automation as we no longer need to consider individual arguments. As we argue later (in the implementation section) the Lebesgue measure of a set can be computed or approximated efficiently for certain primitive functions.

$$\mathsf{Const}(\circ\,\mathfrak{B}\,,\epsilon) \triangleq \mathbb{R}^m_{[0,1]}$$

$$\mathsf{Const}(\boxed{\mu}\,(\mathfrak{T})\,,\kappa) \triangleq \mathsf{Const}(\mathfrak{T},\kappa)$$

$$\mathsf{Const}(\boxed{s}\,(\mathfrak{B})(\mathfrak{T})\,,\kappa) \triangleq \mathsf{Const}(\mathfrak{T},\kappa) \cap \mathfrak{B}^{-1}[0,\infty)$$

$$\mathsf{Const}(\bigcirc(\mathfrak{B})(\mathfrak{T}_1,\mathfrak{T}_2), L\kappa) \triangleq \mathsf{Const}(\mathfrak{T}_1,\kappa) \cap \mathfrak{B}^{-1}(-\infty,0]$$

$$\mathsf{Const}(\bigcirc(\mathfrak{B})(\mathfrak{T}_1,\mathfrak{T}_2), R\kappa) \triangleq \mathsf{Const}(\mathfrak{T}_2,\kappa) \cap \mathfrak{B}^{-1}(0,\infty)$$

$$\mathsf{Const}(\bullet(\mathfrak{B})(\mathfrak{T}_1,\mathfrak{T}_2), L\kappa) \triangleq \mathsf{Const}(\mathfrak{T}_1,\kappa) \cap \mathfrak{B}^{-1}(-\infty,0]$$

$$\mathsf{Const}(\bullet(\mathfrak{B})(\mathfrak{T}_1,\mathfrak{T}_2), R\kappa) \triangleq \mathsf{Const}(\mathfrak{T}_2,\kappa) \cap \mathfrak{B}^{-1}(0,\infty)$$

Figure 23. Inductive definition of $\mathsf{Const}(\mathfrak{T},\kappa)$ for $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T})$.

E.2 Correctness Proof

It remains to show the correctness of our approach, by proving Thm. 6.2. For the proof it is actually easiest to ignore some of the previous work. Instead of analysing $\mathbf{body}_{\mu^\varphi_x.M}(\circledast)$ we fix a actual argumengt r and investigate $\mathbf{body}_{\mu^\varphi_x.M}(r)$. Most notably, we get that $\mathbf{body}_{\mu^\varphi_x.M}(r) \Downarrow \mathfrak{T}_r$ for a (up to sample variables unique) $\mathfrak{T}_r$ and we know that $\mathfrak{T}_r$ does not contain a single red node (as it does not contain $\boxed{\circledast}$).

Paths in Trees. Similar to the way we defined paths in strategies, we can also define paths in execution trees. For a execution tree $\mathfrak{T}$ we denote with $\mathsf{Paths}^{\blacktriangledown}(\mathfrak{T})$ all terminating paths in $\mathfrak{T}$ and for a $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T})$ with $|\boxed{\mu}|(\mathfrak{S},\kappa) \in \mathbb{N}$ the number of times a fixpoint node is traversed. The set of terminating traces for the execution tree in Fig. 21a includes e.g. $\mathbf{RRR\ LR}$. As before, for a set C of natural numbers we abbreviate $\mathsf{Paths}^{\blacktriangledown}(\mathfrak{T},C) \triangleq \{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}) \mid |\boxed{\mu}|(\mathfrak{T},\kappa) \in C\}$.

Correspondence. For every execution tree $\mathfrak{T}$ that does not contain $\circledast$ and $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T})$ we define a measurable set $\mathsf{Const}(\mathfrak{T},\kappa)$ by induction in Fig. 23. Note that $\mathfrak{T}$ must not be obtained via $\Downarrow$. This is similar to the definition in Fig. 22 with the exception that, as $\circledast$ is not contained, every branch (both red and white) restricts the set of assignments. Informally, $\mathsf{Const}(\mathfrak{T},\kappa)$ includes all assignments to the sample variables, such that the branching according to κ is taken and all scoreconstructs do not fail. As before, we define $\mathbb{P}(\mathfrak{T},\kappa) \triangleq \lambda_m(\mathsf{Const}(\mathfrak{T},\kappa))$. We can now show a intuitive correspondence between the paths in $\mathbf{body}_{\mu^\varphi_x.M}(r) \Downarrow \mathfrak{T}_r$ and the small step semantics $\xrightarrow{\star}$ from Fig. 5 (which is similar to Prop. B.8).

Proposition E.4. *If* $r \in \mathbb{R}$ *and* $\mathbf{body}_{\mu^\varphi_x.M}(r) \Downarrow \mathfrak{T}_r$ *and* $n \in \mathbb{N}$ *then,*

$$\sum_{\kappa\in\mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_r,\{n\})} \mathbb{P}(\mathfrak{T}_r,\kappa) = \mu_{\mathbb{S}}\left(\mathbb{T}^{\star}_{\mathbf{body}_{\mu^\varphi_x.M}(r);n}\right)$$

This proposition states, that if we are interested in the number of recursive calls, say n. Then the set of paths $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_r,\{n\})$ are all paths on which n calls are made and the constraints along those paths characterize exactly the traces on which n calls are made in the $\xrightarrow{\star}$ semantics (Fig. 5). Note that not all traces in $\mathbb{T}^{\star}_{\mathbf{body}_{\mu^\varphi_x.M}(r);n}$ are of length at most m.

Replacing Probabilistic by Nondeterministic Choice. We can show the following (which does not depend on the fact that $\mathfrak{T}$ must be obtained via our big-step semantics $\Downarrow$). In particular note, that all trees obtained via $\Downarrow$ and do not contain $\circledast$ also do not contain a red node. For general $\mathfrak{T}$ this does not hold, i.e., there can be trees containing red nodes but no $\circledast$. We need the following simple fact:

Lemma E.5. *If* $\dot{a} \leq a$ *and* $\dot{b} \leq b$ *and* $p \in \mathbb{R}_{[0,1]}$ *then* $\dot{a} \leq pa + (1-p)b$ *or* $\dot{b} \leq pa + (1-p)b$.

Proof. Assume for contradiction $pa + (1-p)b < \dot{a}$ and $pa + (1-p)b < \dot{b}$ then $pa+(1-p)b < p\dot{a}+(1-p)\dot{b}$. But obviously also $p\dot{a} + (1-p)\dot{b} \leq pa + (1-p)b$, a contradiction.   □

Proposition E.6. *If* $\mathfrak{T}$ *is sufficiently independent and does not contain* $\circledast$ *and* $C \subseteq \mathbb{N}$ *then there exists a strategy* $\mathfrak{S} \prec \mathfrak{T}$, *s.t.,*

$$\sum_{\kappa\in\mathsf{Paths}^{\blacktriangledown}(\mathfrak{S},C)} \mathbb{P}^{\star}(\mathfrak{S},\kappa) \leq \sum_{\kappa\in\mathsf{Paths}^{\blacktriangledown}(\mathfrak{T},C)} \mathbb{P}(\mathfrak{T},\kappa)$$

Proof. We generalize the statement. For a measurable set $A \subseteq \mathbb{R}^m_{[0,1]}$ we define $\mathbb{P}_A(\mathfrak{T},\kappa) \triangleq \lambda_m(\mathsf{Const}(\mathfrak{T},\kappa) \cap A)$ and $\mathbb{P}^{\star}_A(\mathfrak{S},p) \triangleq \lambda_m(\mathsf{Const}^{\star}(\mathfrak{S},p) \cap A)$. Note that $\mathbb{P}_{\mathbb{R}^m_{[0,1]}}(\mathfrak{T},p) = \mathbb{P}(\mathfrak{T},p)$ and $\mathbb{P}^{\star}_{\mathbb{R}^m_{[0,1]}}(\mathfrak{S},p) = \mathbb{P}^{\star}(\mathfrak{S},p)$.

We now show that the statement holds with $\mathbb{P}_A$ instead of $\mathbb{P}$ and $\mathbb{P}^{\star}_A$ instead of $\mathbb{P}^{\star}$ for any measurable $A \subseteq \mathbb{R}^m_{[0,1]}$ which obviously subsumes our initial obligation. The proof goes by induction on $\mathfrak{T}$ with $A \subseteq \mathbb{R}^m_{[0,1]}$ universally quantified.

- If $\mathfrak{T} = \circ\,\mathfrak{B}$ then define $\mathfrak{S} \triangleq \circ\,\mathfrak{B}$. It is easy to check that this strategy does satisfy the condition.
- If $\mathfrak{T} = \boxed{\mu}\,(\mathfrak{T}')$. By induction there is a $\mathfrak{S}' \prec \mathfrak{T}'$ that satisfies the conditions. Define $\mathfrak{S} \triangleq \boxed{\mu}\,(\mathfrak{S}')$ which trivial satisfies the condition.
- If $\mathfrak{T} = \boxed{s}\,(\mathfrak{B})(\mathfrak{T}')$. Define $A' \triangleq \mathfrak{B}^{-1}[0,\infty) \cap A$ which is obviously measurable. Now by induction there is a $\mathfrak{S}' \prec \mathfrak{T}'$ such that

$$\sum_{\kappa\in\mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}',C)} \mathbb{P}^{\star}_{A'}(\mathfrak{S}',\kappa) \leq \sum_{\kappa\in\mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}',C)} \mathbb{P}_{A'}(\mathfrak{T}',\kappa)$$

Define $\mathfrak{S} \triangleq \boxed{s}\,(\mathfrak{B})(\mathfrak{S}')$. Now for every $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S})$ we have

$$
\begin{aligned}
\mathbb{P}_A^{\star}(\mathfrak{S}, \kappa) &= \lambda_m\big(\mathsf{Const}^{\star}(\mathfrak{S}, \kappa) \cap A\big) \\
&= \lambda_m\big(\mathsf{Const}^{\star}(\mathfrak{S}', \kappa) \cap \mathfrak{B}^{-1}[0, \infty) \cap A\big) \\
&= \mathbb{P}_{\mathfrak{B}^{-1}[0,\infty) \cap A}^{\star}(\mathfrak{S}', \kappa) \\
&= \mathbb{P}_{A'}^{\star}(\mathfrak{S}', \kappa)
\end{aligned}
$$

Analogously $\mathbb{P}_A(\mathfrak{T}, \kappa) = \mathbb{P}_{A'}(\mathfrak{T}', \kappa)$. So using the IH we get

$$
\begin{aligned}
\sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S},C)} \mathbb{P}_A^{\star}(\mathfrak{S}, \kappa) &= \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}',C)} \mathbb{P}_{A'}^{\star}(\mathfrak{S}', \kappa) \\
&\leq \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}',C)} \mathbb{P}_{A'}(\mathfrak{T}', \kappa) \\
&= \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T},C)} \mathbb{P}_A(\mathfrak{T}, \kappa)
\end{aligned}
$$

- If $\mathfrak{T} = \bigcirc(\mathfrak{B})(\mathfrak{T}_1, \mathfrak{T}_2)$: We define the set $A_1 \triangleq \mathfrak{B}^{-1}(-\infty, 0] \cap A$ and $A_2 \triangleq \mathfrak{B}^{-1}(0, \infty) \cap A$. Both are measurable. By induction there are strategies $\mathfrak{S}_1, \mathfrak{S}_2$ such that

$$
\sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_i,C)} \mathbb{P}_{A_i}^{\star}(\mathfrak{S}_i, \kappa) \leq \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_i,C)} \mathbb{P}_{A_i}(\mathfrak{T}_i, \kappa) \quad \textbf{(1)}
$$

for $i \in \{1, 2\}$. We define $\mathfrak{S} \triangleq \bigcirc(\mathfrak{B})(\mathfrak{S}_1, \mathfrak{S}_2)$ and claim that this fulfils the criterion. We observe the following, for any $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_1)$ we have:

$$
\begin{aligned}
\mathbb{P}_A^{\star}(\mathfrak{S}, L\kappa) &= \lambda_m\big(\mathsf{Const}^{\star}(\mathfrak{S}, L\kappa) \cap A\big) \\
&= \lambda_m\big(\mathsf{Const}^{\star}(\mathfrak{S}_1, \kappa) \cap \mathfrak{B}^{-1}(-\infty, 0] \cap A\big) \\
&= \mathbb{P}_{\mathfrak{B}^{-1}(-\infty,0] \cap A}^{\star}(\mathfrak{S}_1, \kappa) = \mathbb{P}_{A_1}^{\star}(\mathfrak{S}_1, \kappa)
\end{aligned}
$$

and analogously for every $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_2)$, $\mathbb{P}_A^{\star}(\mathfrak{S}, R\kappa) = \mathbb{P}_{A_2}^{\star}(\mathfrak{S}_2, \kappa)$. The same also holds for $\mathbb{P}$ instead of $\mathbb{P}^{\star}$. We can now check:

$$
\begin{aligned}
&\sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S},C)} \mathbb{P}_A^{\star}(\mathfrak{S}, \kappa) \\
&= \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_1,C)} \mathbb{P}_A^{\star}(\mathfrak{S}, L\kappa) + \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_2,C)} \mathbb{P}_A^{\star}(\mathfrak{S}, R\kappa) \\
&= \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_1,C)} \mathbb{P}_{A_1}^{\star}(\mathfrak{S}_1, \kappa) + \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_2,C)} \mathbb{P}_{A_2}^{\star}(\mathfrak{S}_2, \kappa)
\end{aligned}
$$

And using the same reasoning we have

$$
\begin{aligned}
&\sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T},C)} \mathbb{P}_A(\mathfrak{T}, \kappa) \\
&= \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_1,C)} \mathbb{P}_{A_1}(\mathfrak{T}_1, \kappa) + \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_2,C)} \mathbb{P}_{A_2}(\mathfrak{T}_2, \kappa)
\end{aligned}
$$

We can now conclude using the inequalities we obtained via induction **(1)**.

- If $\mathfrak{T} = \bullet(f)(\mathfrak{T}_1, \mathfrak{T}_2)$: We can assume that $\lambda_m(A) > 0$ as otherwise the statement is obvious as any strategy would work since both sides are equal to zero.

Let $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_1)$: We make use of the assumption of sufficient independence. As by assumption $\mathfrak{B}$ does not contain sample variables occurring in $\mathfrak{T}_1$, we get that

$\mathfrak{B}^{-1}(-\infty, 0]$ and $\mathsf{Const}(\mathfrak{T}_1, \kappa)$ are conditionally independent w.r.t. to λ_m. In particular,

$$
\begin{aligned}
&\lambda_m\big(\mathfrak{B}^{-1}(-\infty, 0] \cap \mathsf{Const}(\mathfrak{T}_1, \kappa) \mid A\big) \\
&= \lambda_m\big(\mathfrak{B}^{-1}(-\infty, 0] \mid A\big) \cdot \lambda_m\big(\mathsf{Const}(\mathfrak{T}_1, \kappa) \mid A\big)
\end{aligned}
$$

We can multiply both sides by $\lambda_m(A)$ and derive

$$
\begin{aligned}
&\lambda_m\big(\mathfrak{B}^{-1}(-\infty, 0] \cap \mathsf{Const}(\mathfrak{T}_1, \kappa) \cap A\big) \\
&= \lambda_m\big(\mathsf{Const}(\mathfrak{T}_1, \kappa) \cap A\big) \cdot \lambda_m\big(\mathfrak{B}^{-1}(-\infty, 0] \mid A\big)
\end{aligned}
$$

We can now derive:

$$
\begin{aligned}
\mathbb{P}_A(\mathfrak{T}, L\kappa) &= \lambda_m\big(\mathsf{Const}(\mathfrak{T}, L\kappa) \cap A\big) \\
&= \lambda_m\big(\mathsf{Const}(\mathfrak{T}_1, \kappa) \cap \mathfrak{B}^{-1}(-\infty, 0] \cap A\big) \\
&= \lambda_m\big(\mathsf{Const}(\mathfrak{T}_1, \kappa) \cap A\big) \cdot \lambda_m\big(\mathfrak{B}^{-1}(-\infty, 0] \mid A\big) \\
&= \mathbb{P}_A(\mathfrak{T}_1, \kappa) \cdot \lambda_m\big(\mathfrak{B}^{-1}(-\infty, 0] \mid A\big)
\end{aligned}
$$

Analogously $\mathbb{P}_A(\mathfrak{T}, R\kappa) = \mathbb{P}_A(\mathfrak{T}_2, \kappa) \cdot \mu(\mathfrak{B}^{-1}(0, \infty) \mid A)$ for $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_2)$. Now:

$$
\begin{aligned}
&\sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T},C)} \mathbb{P}_A(\mathfrak{T}, \kappa) \\
&= \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_1,C)} \mathbb{P}_A^{\star}(\mathfrak{T}, L\kappa) + \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_2,C)} \mathbb{P}_A^{\star}(\mathfrak{T}, R\kappa) \\
&= \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_1,C)} \mathbb{P}_A(\mathfrak{T}_1, \kappa)\lambda_m\big(\mathfrak{B}^{-1}(-\infty, 0] \mid A\big) \\
&\quad + \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_2,C)} \mathbb{P}_A(\mathfrak{T}_2, \kappa)\lambda_m\big(\mathfrak{B}^{-1}(0, \infty) \mid A\big)
\end{aligned}
$$

By the IH there are strategies $\mathfrak{S}_1, \mathfrak{S}_2$ such that

$$
\sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_i,C)} \mathbb{P}_A^{\star}(\mathfrak{S}_i, \kappa) \leq \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_i,C)} \mathbb{P}_A(\mathfrak{T}_i, \kappa)
$$

for $i \in \{1, 2\}$. Now as $\lambda_m(\mathfrak{B}^{-1}(-\infty, 0] \mid A) + \lambda_m(\mathfrak{B}^{-1}(0, \infty) \mid A) = 1$ we can apply Lem. E.5. So there exists $i^* \in \{1, 2\}$ such that

$$
\sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_{i^*},C)} \mathbb{P}_A^{\star}(\mathfrak{S}_{i^*}, \kappa) \leq \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T},C)} \mathbb{P}_A(\mathfrak{T}, \kappa)
$$

In case where $i^* = 1$, we define $\mathfrak{S} = \bullet(\mathfrak{B})(\mathfrak{S}_1, \times)$. We can observe that for all $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_1)$ we have $\mathbb{P}_A^{\star}(\mathfrak{S}_1, \kappa) = \mathbb{P}_A^{\star}(\mathfrak{S}, L\kappa)$ as $\mathsf{Const}^{\star}$ does not add any constraint. So $\mathfrak{S}$ does satisfy the desired property. In the case of $i^* = 2$, define $\mathfrak{S} = \bullet(\mathfrak{B})(\times, \mathfrak{S}_2)$.

$\square$

Changing the Node Colour. As $\mathbf{body}_{\mu_x^\varphi.M}(r)$ does not contain $\circledast$ we get that, when $\mathbf{body}_{\mu_x^\varphi.M}(r) \Downarrow \mathfrak{T}_r$, $\mathfrak{T}_r$ does not contain any red nodes. We do however want to colour $\mathfrak{T}_r$ similarly to what we did with $\mathfrak{T}_\circledast$ (recall $\mathbf{body}_{\mu_x^\varphi.M}(\circledast) \Downarrow \mathfrak{T}_\circledast$). The first step is to observe that $\mathfrak{T}_\circledast$ and $\mathfrak{T}_r$ do agree structurally if we ignore node colours and the values at nodes. In fact if we replace every occurrence of $\circledast$ in $\mathfrak{T}_\circledast$ with r, we get, up to the colouring (and reordering of sample variables), exactly $\mathfrak{T}_r$. To fix the colouring we do the following: Denote with $\mathfrak{T}_r^{\bullet}$ the tree $\mathfrak{T}_r$ but with all nodes that depend on r coloured in red. Formally that is $\mathfrak{T}_r^{\bullet} \triangleq \mathfrak{T}_\circledast[r/\circledast]$ where

$\mathfrak{T}_{\circledast}[r/\circledast]$ denotes $\mathfrak{T}_{\circledast}$ with all occurrence of $\circledast$ replaced by r. In particular $\mathfrak{T}_r^{\bullet}$ and $\mathfrak{T}_r$ agree up to reordering of sample variables and colouring of nodes. Now $\mathfrak{T}_r^{\bullet}$ does contain red nodes, but does not contains $\circledast$, in particular every symbolic value at branching nodes (both red and white) denotes a function and we can use Prop. E.6. We can then finally show:

Restatement of Thm. 6.2. *If* $\mathfrak{T}_{\circledast}$ *is sufficiently independent, then for every* $r \in \mathbb{R}$, $\mathbb{P}_{\mathrm{approx}} \sqsubseteq [\![\mu_x^{\varphi}.M \mid r]\!]$

Proof. We have $\mathbf{body}_{\mu_x^{\varphi}.M}(\circledast) \Downarrow \mathfrak{T}_{\circledast}$. Choose any $r \in \mathbb{R}$ and any $n \in \mathbb{N}$. Let $\mathbf{body}_{\mu_x^{\varphi}.M}(r) \Downarrow \mathfrak{T}_r$. And $\mathfrak{T}_r^{\bullet} \triangleq \mathfrak{T}_{\circledast}[\underline{r}/\circledast]$. As we argued before $\mathfrak{T}_r^{\bullet}$ and $\mathfrak{T}_r$ are identical up to the colouring of nodes. Furthermore the strategies for $\mathfrak{T}_r^{\bullet}$ and $\mathfrak{T}_{\circledast}$ are identical (up to different labels of red nodes). By Prop. E.6 there exists a strategy $\mathfrak{S}_r \prec \mathfrak{T}_r^{\bullet}$ such that

$$\sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_r, \{0, \cdots, n\})} \mathbb{P}^{\star}(\mathfrak{S}_r, \kappa)$$
$$\leq \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_r^{\bullet}, \{0, \cdots, n\})} \mathbb{P}(\mathfrak{T}_r^{\bullet}, \kappa) \tag{i}$$

As the strategies for $\mathfrak{T}_r^{\bullet}$ and $\mathfrak{T}_{\circledast}$ are identical (up to red values at red nodes) we get that $\mathfrak{S}_r$ is also a strategy for $\mathfrak{T}_{\circledast}$ (after changing the values at red nodes). Thus

$$\sum_{m \leq n} \mathbb{P}_{\mathrm{approx}}(m) \stackrel{(1)}{=} \min_{\mathfrak{S}' \prec \mathfrak{T}_{\circledast}} \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}', \{0, \cdots, n\})} \mathbb{P}^{\star}(\mathfrak{S}', \kappa)$$
$$\stackrel{(2)}{\leq} \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S}_r, \{0, \cdots, n\})} \mathbb{P}^{\star}(\mathfrak{S}_r, \kappa)$$
$$\stackrel{(3)}{\leq} \sum_{\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{T}_r, \{0, \cdots, n\})} \mathbb{P}(\mathfrak{T}_r, \kappa)$$
$$\stackrel{(4)}{=} \sum_{m \leq n} \mu_{\mathbb{S}}\big(\mathbb{T}^{\star}_{\mathbf{body}_{\mu_x^{\varphi}.M}(r); m}\big)$$
$$\stackrel{(5)}{=} \sum_{m \leq n} [\![\mu_x^{\varphi}.M \mid r]\!](m)$$

where **(1)** is a simple telescoping sum (see the definition of $\mathbb{P}_{approx}$), **(2)** follows as $\mathfrak{S}_r \prec \mathfrak{T}_{\circledast}$ (as the strategies for $\mathfrak{T}_{\circledast}$ $\mathfrak{T}_r^{\bullet}$ are almost identical as we argued before), **(3)** is by the choice of $\mathfrak{S}_r$ (c.f. **(i)**), **(4)** follows from Prop. E.4 and **(5)** by definition of $[\![\mu_x^{\varphi}.M \mid r]\!]$. Thus $\mathbb{P}_{\mathrm{approx}} \sqsubseteq [\![\mu_x^{\varphi}.M \mid r]\!]$ as required. □

F Additional Material - Section 7: Implementation

F.1 Lower Bound Computation

We can turn our interval-based semantics into an effective lower bound computation algorithm by iteratively searching for terminating interval traces.

To do so effectively, our algorithm evaluates a given term symbolically (see Sec. B.5) in a breath-first manor. Once we identified a conditional oracle leading to termination, i.e., a probabilistic execution leading to a value, we collect the symbolic constraints along this path. Let $\{\mathfrak{B}_i \bowtie_i r_i\}_{i \in [m]}$ be those constraints.

To approximate the probability of this path, i.e., the Lebesgue measure of sample-variable assignments that satisfy all constraints along this path, we use our interval approach. Let $\alpha_1, \cdots, \alpha_n$ be the sample variables occurring in $\mathfrak{B}_1, \cdots, \mathfrak{B}_m$. We use a standard sweep algorithm to split $[0, 1]^n$ into smaller boxes. In each step we choose a variable among $\{\alpha_1, \cdots, \alpha_n\}$ and split the current box in half along the chosen dimension. For the resulting smaller boxes we check if the guards are satisfied (using the interval-based reasoning) and in case they are not, split the boxes again; If the box does satisfies all constraints we add the respective volume to the total count. We stop the computation once the analysed parts of the box exceed a user specified probability, i.e., the current branch is analysed such that discovering new terminating interval traces would only contribute very little to the lower bound.

Optimization. Our prototype implementation should be considered a proof of concept and as such is not optimized. The only optimization we use is a dependency analysis that identifies symbolic contains that do not share sample variables and computes the probability individually.

We conjecture, that our implementation can be optimized significantly, by optimizing the split routine. At the moment we split a box along its longest dimension to keep boxes as "square" as possible. Ideally one would have a heuristic, that identifies which dimension should be split and at which value to split to minimize the overall number on overall splits. This would decrease the number of computation steps significantly.

F.1.1 Experimental Results: As lower bound computation is a iterative, possibly non-terminating, process we set a termination condition. This can either be given as a time constraints, leading the termination to be stopped after a given time or as a depth constraints where terms are evaluated up to a given depth. We use the following example programs. Wherever possible we try to use examples used in the implementation of [34]. Our results are, however, only partially comparable to [34]. On the one hand, they only consider discrete distributions, which is obviously easier to analyse than the interval-based reasoning we use for continuous distributions. On the other hand, the main contribution of

Table 1. Experimental Results for Lower Bound Computations. We give the actual probability of termination (if known), the Lower Bound computed, the depth at which we stopped the exploration, the number of identified values and total nodes as well as the time in milliseconds.

Term M	$\mathbb{P}_{\text{term}}(M)$	LB	Depth	#V	#Nodes	t
$geo_{\frac{1}{2}}$	1 (see Thm. 5.9)	0.9999990463	100	20	356	78
$geo_{\frac{1}{5}}$	1 (see Thm. 5.9)	0.9995620416	200	40	1211	192
$1dRW_{\frac{1}{2},1}$	1	0.8036193847	200	65535	1376252	28223
$1dRW_{\frac{7}{10},1}$	1	0.9720964250	150	8191	204796	10224
gr	$\frac{\sqrt{5}-1}{2}$	0.6112594604	80	1773	2046981	4389
$print_{\frac{1}{2}}$	1 (see Thm. 5.9)	0.8318119049	90	23714	5056590	15749
$print_{\frac{1}{4}}$	?(< 1)	0.3328795089	90	23714	5056590	15749
$3print_{\frac{3}{4}}$	1 (see Thm. 5.9)	0.9606655982	80	1773	2046981	4622
$bin_{\frac{1}{2},2}$	1	0.9998493194	100	9445	118907	2265
$pedestrian$	1	0.6002376673	40	7	197	4493

[34] is the insight that the termination probability can be defined as the least fixpoint of higher-order fixpoint equations. Their tool therefore works on manually extracted fixpoint equations. As they already noted in their paper, not every fixpoint equation corresponds to a program; so we can only apply very few of their examples to our framework.

Examples.

- $geo_p := \left(\mu_x^{\varphi}.x \oplus_p \varphi(x+1) \right) \underline{0}$ The simple example Ex. 1.1 from Sec. 1 . This term "computes" the geometric distribution, i.e., the output follows the mass function $n \mapsto (1-p)^n p$. It is AST for every $p > 0$.

- $1dRW_{p,m} := \left(\mu_x^{\varphi}.\text{if}\, x\, \text{then}\, 0\, \text{else}\, \varphi(x-1) \oplus_p \varphi(x+1) \right) \underline{m}$
  The p-biased 1-dimensional random walk with a probability of p moving towards 0. The walk is known to be AST if and only if $p \geq \frac{1}{2}$. In case of $p = \frac{1}{2}$ this program is not PAST. Due to the non PAST nature this program (for $p = \frac{1}{2}$) is intrinsically hard to analyse, as the termination probability decreases significantly with increasing evaluation depth, requiring to consider very long executions. For a $p > \frac{1}{2}$ this program is PAST and, as a results, allow for better (faster) lower bound computation.

- $gr := \left(\mu_x^{\varphi}.0 \oplus \varphi(\varphi(\varphi x)) \right) \underline{0}$
  Program inspired by [50]. As we can infer from our counting based framework, this program is actually not AST and terminates with probability $\frac{\sqrt{5}-1}{2}$, the reciprocal of the golden ratio. (see [50])

- $print_p := \left(\mu_x^{\varphi}.x \oplus_p \varphi(\varphi(x+1)) \right) \underline{0}$ The example Ex. 1.1 from Sec. 1. This program is AST iff $p \geq \frac{1}{2}$ and is case of $p = \frac{1}{2}$ it is not PAST. For $p = \frac{1}{4}$ this is comparable to the term "Ex2.3-1" from the full version of [34].

- $3print_p := \left(\mu_x^{\varphi}.0 \oplus_p \varphi(\varphi(\varphi(x))) \right) \underline{0}$
  Similar to the previous case with three instead of 2 recursive calls. For $p = \frac{1}{4}$ this is comparable to the term "Ex2.3-v2" from the full version of [34].

- $bin_{p,m} := \left(\mu_x^{\varphi}.\text{if}\, x\, \text{then}\, 0\, \text{else}\, (f(x-1) \oplus_p f(x)) \right) \underline{m}$
  Inspired by [43].

-
$$pedestrian :=$$
$$\left(\mu_x^{\varphi}.\text{if}\, x\, \text{then}\, 0\, \text{else} \right.$$
$$\text{let}\, s = \text{sample}\, \text{in}$$
$$\left. s + \varphi\big((x-s) \oplus (x+s)\big) \right) \text{sample}$$

  The term describes a random walk on $\mathbb{R}_+$ that models the situation of a forgetful pedestrian. The example is taken from [40].

Experimental Setup. Our experiential results are listed in Table 1. Where $\mathbb{P}_{\text{term}}(M)$ gives the actual probability of termination, LB the lower bound computed by our tool[14], Depth gives the evaluation depth at which we abort the search[15], #V gives the number of values up to that depth and #Nodes the total number of terms explored. Finally t gives the time in milliseconds.

---

[14]We emphasis again that our tool works with rational numbers and thus perfect precision. For readability we give the first 10 decimal digits of the rational output.

[15]As mentioned previously the computation is a ongoing, possibly infinite computation that must be ended at some point. This can be done by either specifying a target depth of time. To keep the results as independent from the concrete machine as possible, we specify a target depth to increase reproducibility.

Table 2. Experimental Results for AST Verification. For each term (all of which our tool can verified to be AST) we give the counting distribution $\mathbb{P}_{approx}$ computed by our tool (which is analysed via Thm. 5.4). We also give the time used by our internal computation t_{int}, by our volume computation ([11]) t_{vol} and the total time $t = t_{int} + t_{vol}$ in milliseconds.

M	$\mathbb{P}_{approx}$	t_{int}	t_{vol}	t
$geo_{\frac{1}{2}}$	$\frac{1}{2}\delta_0 + \frac{1}{2}\delta_1$	140	99	239
Ex. 1.1, $p = \frac{1}{2}$ ($print_{\frac{1}{2}}$)	$\frac{1}{2}\delta_0 + \frac{1}{2}\delta_2$	138	99	237
$3print_{\frac{2}{3}}$	$\frac{2}{3}\delta_0 + \frac{1}{3}\delta_3$	274	123	297
Ex. 5.1, $p = 0.6$	$0.6\delta_0 + 0.2\delta_2 + 0.2\delta_3$	154	242	396
Ex. E.2	$0.5\delta_0 + 0.5\delta_2$	150	255	405
Ex. 5.15, $p = 0.65$	$0.65\delta_0 + 061250\delta_2 + 0.288750\delta_3$	158	215	373

F.2 AST Verification

Our proof method from Sec. 6 gives us a straightforward implementation as all operations are on a finite tree. Our tool first computes the execution tree and its strategies. The key difficultly is to compute $\mathbb{P}(\mathfrak{S}, \kappa)$ for strategy $\mathfrak{S}$ and a path $\kappa \in \mathsf{Paths}^{\blacktriangledown}(\mathfrak{S})$, i.e., compute the weight associated with a path. We restrict the primitive operations to addition and multiplication by a constant (and thus subtraction). Under this restriction, each symbolic value $\mathfrak{B}$ denotes a *linear* function in the sample variables. The weight of a path is thus the Lebesgue measure of an intersection of half planes or equivalently the volume of a polyhedron (a subset of $\mathbb{R}^d$ of the from $\{\vec{x} \mid A\vec{x} \leq b\}$) [21]. As shown in [37] the volume of such a polyhedron although $\#P$ hard, can be computed via a simple recursive scheme. We use the optimized implementation of this scheme in [11] to effective compute the volume. Our tool thus perform all basic operations on trees and refers to the tool from [11] for the probabilistic computations.

Experimental Results. Our tool can verify AST for all examples in this paper (with the identified bounds on free variables like p in Ex. 1.1 or Ex. 5.15). Our results are given in Table 2. The distribution $\mathbb{P}_{approx}$ is the one *automatically* inferred by our tool.

Extensions. Our approach from Sec. 6 is applicable for all measurable functions as primitive operations. However, to use the approach effectively and automatically, the probability of path must be computable. This is so in our case as the volume of a convex polyhedron is computable. Our framework and implementation can easily be adjusted to different primitive operations by merely changing the tool responsible for volume computation.

Index