Simple Types for Probabilistic Termination

Willem Heijltjes
Department of Computer Science, University of Bath, UK

Georgina Majury
Department of Computer Science, University of Bath, UK

Abstract

We present a new typing discipline to guarantee the probability of termination in probabilistic lambda-calculi. The main contribution is a particular naturality and simplicity: our probabilistic types are as simple types, but generated from probabilities as base types, representing a least probability of termination. Simple types are recovered by restricting probabilities to one.

Our vehicle is the Probabilistic Event Lambda-Calculus by Dal Lago, Guerrieri, and Heijltjes, which presents a solution to the issue of confluence in probabilistic lambda-calculi. Our probabilistic type system provides an alternative solution to that using counting quantifiers by Antonelli, Dal Lago, and Pistone, for the same calculus.

The problem that both type systems address is to give a lower bound on the probability that terms head-normalize. Following the recent Functional Machine Calculus by Heijltjes, our development takes the (simplified) Krivine machine as primary, and proceeds via an extension of the calculus with sequential composition and identity on the machine. Our type system then gives a natural account of termination probability on the Krivine machine, reflected back onto head-normalization for the original calculus. In this way we are able to avoid the use of counting quantifiers, while improving on the termination bounds given by Antonelli, Dal Lago, and Pistone.

2012 ACM Subject Classification Theory of computation → Lambda calculus; Theory of computation → Type theory; Theory of computation → Probabilistic computation

Keywords and phrases lambda-calculus, probabilistic termination, simple types

Digital Object Identifier 10.4230/LIPIcs.CSL.2025.31

Acknowledgements We would like to thank Melissa Antonelli, Ugo Dal Lago, and Paolo Pistone for the constructive conversation about both our approaches, and the anonymous referees for their helpful commentary.

1 Introduction

While the study of probabilistic computation can be traced to the 1950s [11], the first study of the probabilistic λ-calculus in particular is considered to be by Saheb-Djahromi in the late 70s [30]. In the near half century since, many variations on higher-order probabilistic computation have been considered [10, 12, 19, 20, 26, 29]. In recent years, perhaps due to the potential for applications in machine learning and modelling of probabilistic systems, the area has seen a return to popularity [5, 6, 16, 17, 24, 31]. An important computational phenomenon in its own right, the study of probabilistic choice can also provide a "foot in the door" for understanding how more general effects might manifest, leading for instance to the recent *Functional Machine Calculus* (FMC) as a confluent λ-calculus with effects [3, 18] that will play a central role in our development.

In this paper we consider the problem of *probabilistic termination*, the probability that a given reduction mechanism reaches the normal form of a term, a key consideration for probabilistic computation and the subject of significant recent attention [4, 7, 8, 15, 22]. [1]

[1] Note that this objective is distinct from *almost-sure* termination: it considers *exact* probabilities, not those converging in the limit. Almost-sure termination is more commonly studied for iterative constructs [21, 27]; extending the lambda-calculus with an almost-surely terminating iterator is the subject of future work.

Termination being one of the prime considerations of type systems in general, the question of type systems for probabilistic termination is pertinent [1, 7]. Our contribution in this paper is a new typing discipline for probabilistic termination, following the recent line of work on the *probabilistic event λ-calculus* [9] and the FMC. Their prime features are *confluence* for probabilistic computation and other effects, and the encoding of multiple reduction strategies within a single calculus.

One of the greatest challenges facing the study of probabilistic computation is confluence. In most variants the outcome of duplicating a probabilistic term is strategy dependent. Consider the instruction "write down the result of flipping a coin twice": it is ambiguous whether "twice" refers to "write" or "flipping", and the choice of interpretation changes the possible outcomes. In most of the literature results are therefore either restricted to a single strategy, such as call–by–value or call–by–name, or rely on a modified definition of reduction. This raises the question whether it is possible to have confluence for probabilistic computation, while expressing different strategies, and rewriting without restriction on contexts, properties which lend elegance to the λ-calculus. The *probabilistic event λ-calculus* [9] proposes a solution to this problem by decomposing a probabilistic sum $M \oplus N$ into a *choice* operation $M\, a\, N$, understood as a conditional "*if* a *then* M *else* N", and a probabilistic *generator* $\boxed{a}.\, P$, representing a coin toss whose outcome is bound to the boolean variable a in the term P, which then determines the choice for $M\, a\, N$. When in argument position $\boxed{a}.\, M$ acts as a wrapper, similar to *thunking* in call–by–push value [25] and the bang-calculus [14, 15], protecting the operator from evaluation. These factors combine to return confluence to the calculus, allowing for arbitrary evaluation strategies with an unrestricted β-reduction.

The probabilistic event λ-calculus (PEΛ) was introduced in [9] with a simple type system, corresponding to that on the lambda-calculus. This system ignored the probabilistic elements of the calculus, and thus gave little insight into the properties of this extension. In a deterministic calculus a type system provides various safety guarantees. These may be qualitative: termination, outputs, composability; or quantitative: run time, term size. In the probabilistic setting, however, it is natural to wonder what guarantees can be made, when the behaviour of a single term can vary between iterations. If, as in the type system mentioned, the probabilities are ignored, strong results can be obtained, albeit on a fairly uninteresting fragment of the calculus. Once probability is acknowledged by the type system the guarantees made become similarly qualified: probability of termination, almost-sure termination, expected run time. Here, we consider the first of these.

One proposed type system for the PEΛ [1], labelled C$\lambda_{\to}$, introduces *counting quantifiers* as a Curry–Howard correspondent to an intuitionistic counting propositional logic. These provide a mechanism to express "proportion of truth" by quantifying the number of satisfying assignments to a formula within a given model, in the way that existential and universal quantification may be understood via the existence of a satisfying assignment, respectively the satisfaction of all assignments. By taking the assignments to a formula to describe the branches of a probabilistic computation, counting quantifiers may be used to describe probability bounds. By this mechanism, C$\lambda_{\to}$ provides a lower bound on the probability of head normalisation. However, as illustrated in [1], the bounds provided by C$\lambda_{\to}$ are not tight, even in situations where this would be expected (see Example 8).

In this paper we present an alternative approach to the same problem, for the same calculus, using a different inspiration, to provide improved termination bounds. Deriving from the PEΛ, the FMC provides an additional feature that, to us, seemed crucial to a natural account of probabilistic termination via the type system. First, the "machine" in question is the (simplified) *Krivine machine* [23], whose evaluation is closely related to head

normalisation. The FMC here adds an intriguing aspect: it introduces *sequential composition* and *identity* on the machine, where the latter, the imperative *skip*, provides a notion of *successful termination*, notably absent from the machine for standard λ-calculus. Moreover, this becomes the primary interpretation of types: a type derivation in the FMC is a proof that the machine successfully terminates. Our main question for this work was how to adapt this to capture *probabilistic termination*. The answer, described in Section 6, is an extended *sequential probabilistic event* λ*-calculus* $\mathsf{SPE}\Lambda$, with a type system $\mathsf{SPE}\Lambda^{\mathbb{Q}\Rightarrow}$ that naturally describes probabilistic termination of the machine, as well as probabilistic termination of head reduction.

The type system $\mathsf{SPE}\Lambda^{\mathbb{Q}\Rightarrow}$ for the extended sequential calculus reflects back onto a natural type system for the original probabilistic event λ-calculus, $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$, which gives probabilistic head normalization in the following way: types generalize simple types by replacing the base type with a *probability*. Formally,

$$A, B \ ::= \ p \ \mid \ A \to B$$

where $p \in [0,1] \cap \mathbb{Q}$ is a rational number between zero and one inclusive. The intuition is that the base type for simple types, o, may be understood as signifying successful evaluation, even if it is uninhabited. After all, a constant base type such as for booleans or integers signifies the successful return of a corresponding value. This further matches the interpretation in the FMC, where the type o is inhabited by *skip*, and corresponds to termination of the machine without producing a result. Our approach, then, is to replace *certain* termination with a *probability* of termination, replacing the base type o with a probability p. We consider the striking simplicity and natural intuition of this approach one of our main contributions.

2 The probabilistic event lambda-calculus

We recall the *probabilistic event lambda-calculus* $\mathsf{PE}\Lambda$ from [9]. Assume countable sets of *variables*, ranged over by x, y, z, and of *events*, ranged over by a, b, c. The former are *term* variables, to be instantiated by terms of the calculus, and the latter are *boolean* variables, to be instantiated by $\top$ (*true*) or $\bot$ (*false*). The $\mathsf{PE}\Lambda$ extends the λ-calculus with a *generator* $\boxed{a}.M$, which flips a coin and binds the result ($\top$ or $\bot$) to a, and a *choice* or *conditional* $M\,a\,N$, which evaluates to M if a is *true*, and to N otherwise. A traditional probabilistic sum of terms, $M \oplus N$, may be encoded as $\boxed{a}.M\,a\,N$. Generators are normally *fair*, with equal probability for $\top$ or $\bot$, though on occasion we may need a *biased* generator $\boxed{a}_p$ which chooses $\top$ with probability p, and $\bot$ with $1-p$.

▶ **Definition 1.** Terms *are given by the following grammar,*

$$M, N \ ::= \ x \ \mid \ \lambda x.M \ \mid \ M\,N \ \mid \ \boxed{a}.M \ \mid \ M\,a\,N$$

with, from left to right: a variable*; an* abstraction*, which binds* x *in* M*; an* application*; a* generator*, which binds* a *in* M*; and a* choice*.*

The *free variables* and *free events* of a term M are written $\mathsf{fv}(M)$ and $\mathsf{fe}(M)$ respectively. Substitution is written prefix: $\{N/x\}M$ is the capture-avoiding substitution of N for x in M. For an event a we define two *projection* functions $\pi^a_\top$ and $\pi^a_\bot$, which apply the effect of instantiating a with *true* and *false* respectively, to a term M.

▶ **Definition 2.** *The* projection *functions* π_i^a *for an event* a *and* $i \in \{\bot, \top\}$ *are given as follows, where* $a \neq b$.

$$\begin{aligned} \pi_i^a x &= x \\ \pi_i^a(\lambda x.\, M) &= \lambda x.\, \pi_i^a M \\ \pi_i^a(M\, N) &= (\pi_i^a M)\,(\pi_i^a N) \end{aligned} \qquad \begin{aligned} \pi_\top^a(M\, a\, N) &= \pi_\top^a M \\ \pi_\bot^a(M\, a\, N) &= \pi_\bot^a N \end{aligned} \qquad \begin{aligned} \pi_i^a(M\, b\, N) &= (\pi_i^a M)\, b\, (\pi_i^a N) \\ \pi_i^a(\boxed{b}.\, M) &= \boxed{b}.\, \pi_i^a M \end{aligned}$$

We use the standard notions of *context*, *head context*, and *applicative context* to define the reduction relations. Note that a head context is of the form $\lambda x_1 \ldots \lambda x_n.\, \{\}\, M_1 \ldots M_m$ where m and n are potentially zero.

▶ **Definition 3.** Contexts C, head *contexts* H, *and* applicative *contexts* A *are defined as follows. A context* C *with the hole replaced by* M, *capturing variables, is written* $C\{M\}$.

$$\begin{aligned} C \;::=\; & \{\} \mid \lambda x.\, C \mid C\, M \mid M\, C \\ & \mid \boxed{a}.\, C \mid C\, a\, M \mid M\, a\, C \end{aligned} \qquad \begin{aligned} H \;::=\; & \lambda x.\, H \mid A \\ A \;::=\; & A\, M \mid \{\} \end{aligned}$$

A *probability* p, q is a rational number between 0 and 1 inclusive. Probabilistic reduction will return a *multi-(sub-)distribution* [2], a finite multiset of weighted terms $\mathcal{M}$ written as $[p_1 \cdot M_1, \ldots, p_n \cdot M_n]$ whose *weight* $|\mathcal{M}| = \sum_{i \leq n} p_i$ is (at most) one.[2] For simplicity, we will refer to these as *distributions*, and we convert implicitly between terms M and the singleton distribution $[1 \cdot M]$. Multiset union is written $\mathcal{S} + \mathcal{T}$, and the empty multiset as $\varnothing$. A distribution $\mathcal{M}$ is scaled to $p\mathcal{M}$ by multiplying each weight in $\mathcal{M}$ by p. The underlying probability (sub-)distribution of $\mathcal{M}$, the finite function from terms to probabilities obtained by collecting like terms $p \cdot M$ and $q \cdot M$ as $(p{+}q) \cdot M$, is written $\lfloor \mathcal{M} \rfloor$.

▶ **Definition 4.** Beta-reduction $\twoheadrightarrow_\beta$ *and* head β-reduction $\twoheadrightarrow_{\beta \mathsf{h}}$ *are given by closing the beta-rule below under all contexts* C *respectively under head contexts* H, *and implicitly return a singleton distribution.*

$$(\lambda x.\, M)N \;\twoheadrightarrow_\beta\; \{N/x\}M$$

Projective reduction $\twoheadrightarrow_\pi$ *is the following reduction relation from terms to distributions.*

$$H\{\boxed{a}.\, M\} \;\twoheadrightarrow_\pi\; [\, \tfrac{1}{2} \cdot H\{\pi_\top^a M\}\,,\, \tfrac{1}{2} \cdot H\{\pi_\top^a M\}\,]$$

Head reduction $(\twoheadrightarrow_{\mathsf{h}}) = (\twoheadrightarrow_{\beta \mathsf{h}}) \cup (\twoheadrightarrow_\pi)$ *is the union of head* β*-reduction and projective reduction. Reduction is lifted to distributions of terms in the expected way: if* $M \twoheadrightarrow \mathcal{N}$ *then* $[p \cdot M] + \mathcal{M} \twoheadrightarrow (p\mathcal{N}) + \mathcal{M}$. *We write* $\twoheadrightarrow\!\!\!\!\twoheadrightarrow$ *for the reflexive-transitive closure of a reduction relation* $\twoheadrightarrow$.

The PEΛ features a second notion of reduction, *permutative reduction* $\twoheadrightarrow_{\mathsf{p}}$ [9], which gives a more fine-grained evaluation of probabilistic sums. It is this reduction for which confluence is particularly significant. The effect of permutative reduction is to bridge the gap between the decomposed operators $\boxed{a}$ and $M\, a\, N$ and the standard probabilistic sum $M \oplus N$, encoded as $\boxed{a}.\, M\, a\, N$, by internalizing the reduction of $\boxed{a}.\, M$ to the sum of its two projections [9, Proposition 29]:

$$\boxed{a}.\, M \;\twoheadrightarrow\!\!\!\!\twoheadrightarrow_{\mathsf{p}}\; \boxed{a}.\, (\pi_\top^a M)\; a\; (\pi_\bot^a M) \quad (\text{if } a \in \mathsf{fe}(M)) \,.$$

[2] We use multi-distributions rather than distributions to accommodate the reduction measure for the proof of head normalization in Appendix A.2.

$$\frac{}{\Gamma,\, x\colon A \vdash x\colon A} \qquad \frac{\Gamma,\, x\colon A \vdash M\colon B}{\Gamma \vdash \lambda x.\, M\colon A \to B} \qquad \frac{\Gamma \vdash M\colon A \to B \qquad \Gamma \vdash N\colon A}{\Gamma \vdash M\, N\colon B}$$

$$\frac{\Gamma \vdash M\colon A \qquad \Gamma \vdash N\colon A}{\Gamma \vdash M\, a\, N\colon A} \qquad \frac{\Gamma \vdash M\colon A}{\Gamma \vdash \boxed{a}.\, M\colon A}$$

Figure 1 The simple type system $\mathsf{PE}\Lambda^{\to}$.

In this paper we will work with projective reduction, since we are interested in termination of head reduction. We will, on occasion, consider terms where probabilistic sums are of the traditional form $M \oplus N = \boxed{a}.\, M\, a\, N$ with $a \notin \mathsf{fe}(M), \mathsf{fe}(N)$. As per the above, these are effectively the normal forms of permutative reduction.

3 Probabilistic types

Before introducing our probabilistic type system, we recall *simple types* for the $\mathsf{PE}\Lambda$ [9].

▶ **Definition 5.** Simple types *are given by the following grammar.*

$$A, B \;::=\; o \;\mid\; A \to B$$

A typing context Γ *is a finite function from term variables to types, written as a sequence* $x_1\colon A_1, \ldots, x_n\colon A_n$. *A* typing judgment $\Gamma \vdash M\colon A$ *assigns the type* A *to the term* M *in the context* Γ. *The* simply-typed probabilistic event λ-calculus $\mathsf{PE}\Lambda^{\to}$ *is given by the typing rules in Figure 1.*

Simple types ignore the probabilistic constructs of the calculus, *generator* and *choice*, requiring only that branches of a choice have equal types. This gives the expected result: typed terms are strongly normalizing, since every possible branch of the computation is typed. For *probabilistic* termination, we wish to capture that a given fraction of all branches of a computation, terminates – where our notion of *termination* is given by *head normalization*.

Our approach derives from the following idea: if the base type o of simple types, even if not inhabited, denotes *certainty* of termination, then we may generalise this to *probability* of termination by replacing base types with arbitrary probabilities. This yields the following notion of types.

▶ **Definition 6.** Probabilistic types *are given by the following grammar, where* $p \in [0,1] \cap \mathbb{Q}$.

$$A, B \;::=\; p \;\mid\; A \to B$$

The intuitive meaning of, for example, assigning a term M a type $A \to B \to p$ is: given inputs of type A and B, M terminates with probability at least p. The identity term $I = \lambda x.\, x$ may be assigned any type $p \to p$: given an input N that terminates with probability p, the term $I\, N$ does so as well. The type system will include an axiom that any term, in particular a non-terminating one such as $\Omega = (\lambda x.\, xx)(\lambda x.\, xx)$, may be assigned a termination probability of zero. This is expressed by a type $A_1 \to \ldots \to A_n \to 0$: for any inputs A_1 through A_n, the term will terminate with probability at least zero.

Note that these types generalise simple types with a single, uninhabited base type o. A probabilistic system with multiple base types, say integers and booleans, would pair the return type with a probability, for instance an integer with $\frac{1}{2}$ probability, or a boolean with $\frac{1}{4}$ probability. The extension of the calculus with sequencing, in Sections 5 and 6, will give a more concrete account of how probabilities interact with return values and return types.

$$\frac{}{E \mid \Gamma, x\colon A \vdash x\colon A}\,\mathsf{var} \qquad \frac{}{E \mid \Gamma \vdash M\colon \overline{A} \to 0}\,\mathsf{zero} \qquad \frac{E \mid \Gamma \vdash M\colon \overline{A} \to q}{E \mid \Gamma \vdash M\colon \overline{A} \to p}\,\mathsf{low}\ (p < q)$$

$$\frac{E \mid \Gamma, x\colon A \vdash M\colon B}{E \mid \Gamma \vdash \lambda x.\, M\colon A \to B}\,\mathsf{abs} \qquad \frac{E \mid \Gamma \vdash M\colon A \to B \qquad E \mid \Gamma \vdash N\colon A}{E \mid \Gamma \vdash M\,N\colon B}\,\mathsf{app}$$

$$\frac{E, a \mapsto i \mid \Gamma \vdash M_i\colon A}{E, a \mapsto i \mid \Gamma \vdash M_\top\, a\, M_\bot\colon A}\,\mathsf{chc} \qquad \frac{E, a \mapsto \top \mid \Gamma \vdash M\colon \overline{A} \to p \qquad E, a \mapsto \bot \mid \Gamma \vdash M\colon \overline{A} \to q}{E \mid \Gamma \vdash \boxed{a}.\, M\colon \overline{A} \to \tfrac{1}{2}p + \tfrac{1}{2}q}\,\mathsf{gen}$$

Figure 2 The probabilistic type system $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$.

The term $\boxed{a}.\, M\, a\, \Omega$ gives a fair probabilistic choice between M and Ω. For the computation to be well-typed regardless of the choice, M and Ω should have the same input types, so that they can be applied to the same arguments. Hence, if M has type $A \to B \to p$, we may choose $A \to B \to 0$ for Ω. Then the type for $\boxed{a}.\, M\, a\, \Omega$ should be $A \to B \to \frac{1}{2}p$: given arguments of type A and B, the computation chooses M with probability $\frac{1}{2}$ and so terminates with probability at least $\frac{1}{2}p$.

These considerations motivate the shape of our probabilistic type system, with one further aspect to explain. An arbitrary generator term $\boxed{a}.\, M$ reduces with a fair probability to either $\pi^a_\top M$ or $\pi^a_\bot M$, which may then terminate with different probabilities. However, using projections would mean the type system loses the property of being inductive on terms. We will instead record the assignment of a truth value to a in an additional context E, that we call an *event valuation*. The type derivation then projects a term $M\, a\, N$ to M or to N according to the value of a in the event valuation E.

▶ **Definition 7.** *The* probabilistically typed probabilistic event λ-calculus $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ *is given by the typing rules in Figure 2, using the following definitions. An* event valuation E *is a finite function from events to* $\{\bot, \top\}$*, written as a sequence* $a_1 \mapsto i_1, \ldots, a_n \mapsto i_n$*. A* typing context Γ *is a finite function from variables to types, written* $x_1\colon A_1, \ldots, x_n\colon A_n$*. A* probabilistic typing judgement $E \mid \Gamma \vdash M\colon A$ *assigns a type* A *to the term* M *in the context of* E *and* Γ*. A sequence of antecedents* $A_1 \to \ldots \to A_n \to p$ *is abbreviated with vector notation as* $\overline{A} \to p$*.*

The typing rules are syntax-driven, except for the low rule to lower a given bound. The rule is not essential to the type system, since it is already implied in the meaning of types as giving a lower bound; but for the same reason, since it is implied, including it brings more clarity than omitting it. Note further that the typing rule gen assumes a fair coin toss for the generator $\boxed{a}$, which gives the resulting probability of $\frac{1}{2}p + \frac{1}{2}q$. An unfair toss $\boxed{a}_r$ with ratio r would give a probability $rp + (1-r)q$.

▶ **Example 8.** To demonstrate our type system we reprise the example $t[a, b]$ from [1, Section 6.2], written here as the term $\boxed{a}.\, \boxed{b}.\, \boxed{c}.\, T$ below. In Figure 3 we derive the following type.

$$\boxed{a}.\, \boxed{b}.\, \boxed{c}.\, T\colon 1 \to \tfrac{3}{8} \quad \text{where} \quad T = ((I\, c\, \Omega)\, b\, \Omega)\, a\, (\Omega\, b\, I)$$

We will discuss a number of aspects of our type system. First, as a particularly natural feature, observe that the rules var, abs and app make it conservative over standard simply-typed λ-calculus – which, interestingly, is agnostic to the choice of base types.

$$D_{\bot,\bot,\bot} = \dfrac{\dfrac{\dfrac{\dfrac{}{a\mapsto\bot,b\mapsto\bot,c\mapsto\bot \mid x\colon 1\vdash x\colon 1}\,\mathsf{var}}{a\mapsto\bot,b\mapsto\bot,c\mapsto\bot \mid \vdash \lambda x.\,x\colon 1\to 1}\,\mathsf{abs}}{a\mapsto\bot,b\mapsto\bot,c\mapsto\bot \mid \vdash I\,c\,\Omega\colon 1\to 1}\,\mathsf{chc}}{a\mapsto\bot,b\mapsto\bot,c\mapsto\bot \mid \vdash (I\,c\,\Omega)\,b\,\Omega\colon 1\to 1}\,\mathsf{chc}}{a\mapsto\bot,b\mapsto\bot,c\mapsto\bot \mid \vdash ((I\,c\,\Omega)\,b\,\Omega)\,a\,(\Omega\,b\,I)\colon 1\to 1}\,\mathsf{chc}$$

$$D_{\bot,\bot} = \dfrac{D_{\bot,\bot,\bot}\quad a\mapsto\bot,b\mapsto\bot,c\mapsto\bot \mid \vdash T\colon 1\to 1 \qquad \dfrac{}{a\mapsto\bot,b\mapsto\bot,c\mapsto\top \mid \vdash T\colon 1\to 0}\,\mathsf{zero}}{a\mapsto\bot,b\mapsto\bot \mid \vdash \boxed{c}.\,T\colon 1\to\tfrac{1}{2}}\,\mathsf{gen}$$

$$D_{\bot} = \dfrac{D_{\bot,\bot}\quad a\mapsto\bot,b\mapsto\bot \mid \vdash \boxed{c}.\,T\colon 1\to\tfrac{1}{2} \qquad \dfrac{}{a\mapsto\bot,b\mapsto\top \mid \vdash \boxed{c}.\,T\colon 1\to 0}\,\mathsf{zero}}{a\mapsto\bot \mid \vdash \boxed{b}.\,\boxed{c}.\,T\colon 1\to\tfrac{1}{4}}\,\mathsf{chc}$$

$$D_{\top,\top,i} = \dfrac{\dfrac{\dfrac{\dfrac{}{a\mapsto\top,b\mapsto\top,c\mapsto i \mid x\colon 1\vdash x\colon 1}\,\mathsf{var}}{a\mapsto\top,b\mapsto\top,c\mapsto i \mid \vdash \lambda x.\,x\colon 1\to 1}\,\mathsf{abs}}{a\mapsto\top,b\mapsto\top,c\mapsto i \mid \vdash \Omega\,b\,I\colon 1\to 1}\,\mathsf{chc}}{a\mapsto\top,b\mapsto\top,c\mapsto i \mid \vdash ((I\,c\,\Omega)\,b\,\Omega)\,a\,(\Omega\,b\,I)\colon 1\to 1}\,\mathsf{chc}$$

$$D_{\top,\top} = \dfrac{D_{\top,\top,\bot}\quad a\mapsto\top,b\mapsto\top,c\mapsto\bot \mid \vdash T\colon 1\to 1 \qquad D_{\top,\top,\top}\quad a\mapsto\top,b\mapsto\top,c\mapsto\top \mid \vdash T\colon 1\to 1}{a\mapsto\top,b\mapsto\top \mid \vdash \boxed{c}.\,T\colon 1\to 1}\,\mathsf{gen}$$

$$D_{\top} = \dfrac{\dfrac{}{a\mapsto\top,b\mapsto\bot \mid \vdash \boxed{c}.\,T\colon 1\to 0}\,\mathsf{zero} \qquad D_{\top,\top}\quad a\mapsto\top,b\mapsto\top \mid \vdash \boxed{c}.\,T\colon 1\to 1}{a\mapsto\top \mid \vdash \boxed{b}.\,\boxed{c}.\,T\colon 1\to\tfrac{1}{2}}\,\mathsf{gen}$$

$$D = \dfrac{D_{\bot}\quad a\mapsto\bot \mid \vdash \boxed{b}.\,\boxed{c}.\,T\colon 1\to\tfrac{1}{4} \qquad D_{\top}\quad a\mapsto\top \mid \vdash \boxed{b}.\,\boxed{c}.\,T\colon 1\to\tfrac{1}{2}}{\mid \vdash \boxed{a}.\,\boxed{b}.\,\boxed{c}.\,T\colon 1\to\tfrac{3}{8}}\,\mathsf{gen}$$

Figure 3 Typing derivations for Example 8, where $T = ((I\,c\,\Omega)\,b\,\Omega)\,a\,(\Omega\,b\,I)$.

Second, a key difference with the simple type system of Figure 1 is that probabilistic branching occurs in the *generator* rule, whereas for simple types it is the *choice* rule. This is due to aiming for *head normalization* instead of *strong normalization*. Consider the example term $\boxed{a}.\, M\, a\, (\Omega\, a\, N)$. Head reduction projects it to M and N, removing Ω, which will not be reduced. This is reflected in the probabilistic type system, where the branches of the generator typing rule project to M and N via the event valuation. The simple type system, reflecting strong normalization, would require also Ω to be typed – which of course it cannot.

For terms of the form $M \oplus N = \boxed{a}.\, M\, a\, N$ this difference is moot: both type systems branch similarly for this construct, to M and N. This leaves as only distinction the generalisation of types themselves, from a single base type o to probabilities p. In accordance with the meaning of a base type as the probability of termination, simple types are then recovered by restricting to base type $p = 1$. This rules out the rules zero and low, assigning a zero-weighted type $\overline{A} \to 0$ and reducing the weight of a type. It is easy to observe that the remaining rules preserve the restriction $p = 1$, to give the following proposition.

▶ **Proposition 9.** *For the fragment below left, the simply-typed* PEΛ *coincides with the probabilistically-typed* PEΛ *restricted to the types below right.*

$$M,\, N \;::=\; x \mid \lambda x.\, M \mid M\, N \mid \boxed{a}.\, M\, a\, N \qquad\qquad A,\, B \;::=\; 1 \mid A \to B$$

Our main result for the probabilistically-typed PEΛ is that a type $\overline{A} \to p$ guarantees head normalization with probability at least p. Formally, this is stated by a head reduction to a distribution of which a proportion of at least p is in head-normal form.

▶ **Theorem 10.** *For closed M, if $M : \overline{A} \to p$ then $M \twoheadrightarrow_{\mathsf{h}} \mathcal{N}_0 + \mathcal{N}_1$ where all terms in $\mathcal{N}_0$ are head normal and $|\mathcal{N}_0| \geq p$.*

The result will follow directly from the corresponding Theorem 22 for the expanded probabilistic calculus with sequential composition, introduced in Section 5.

4 Comparison with counting quantifiers

In this section we will give a close comparison with the type system C$\lambda_{\to}$ of Antonelli, Dal Lago, and Pistone [1], and demonstrate that our approach gives tighter bounds on the probability of termination.

The first distinction between C$\lambda_{\to}$ and PEΛ$^{\mathbb{Q}\to}$ is that the former uses indexed event variables x_a^i for $i \in \mathbb{N}$ instead of events a. However, there is no formal need for this; since the indices i are static, we may replace x_a^i and x_a^j for distinct i and j simply with distinct events a and b. This simplification extends to the semantics. Probabilities in C$\lambda_{\to}$ are given by boolean formulas $\mathscr{b}$ over the variables x_a^i, indicating a subset of the space $(2^{\mathbb{N}})^X$ where X is a finite set of events. However, since the indices i are fixed and bounded, it is sufficient to consider finite spaces 2^X. The elements of this set are the event valuations E with domain X, and a boolean formula $\mathscr{b}$ over X indicates the set of event valuations $[\![\mathscr{b}]\!]_X = \{E \in 2^X \mid E \models \mathscr{b}\}$, where $E \models \mathscr{b}$ is characterized syntactically as expected:

$$\frac{}{E, a \mapsto \top \;\models\; a} \qquad \frac{}{E, a \mapsto \bot \;\models\; \neg a} \qquad \frac{}{E \;\models\; \top} \qquad \frac{E \;\models\; \mathscr{b} \qquad E \;\models\; \mathscr{c}}{E \;\models\; \mathscr{b} \wedge \mathscr{c}} \qquad \frac{E \;\models\; \mathscr{b}}{E \;\models\; \mathscr{b} \vee \mathscr{c}}$$

For a set $\mathcal{E} \subseteq 2^X$ the measure $\mu_X(\mathcal{E})$ is given by

$$\mu_X(\mathcal{E}) = \frac{|\mathcal{E}|}{2^{|X|}}$$

where $|S|$ denotes the size of a set S. Then $\mu(\mathscr{b})$ is $\mu_X([\![\mathscr{b}]\!]_X)$ where X is the domain of $\mathscr{b}$. This coincides with the measure $\mu(\mathscr{b})$ over $(2^{\mathbb{N}})^X$ in [1], as the latter makes no essential use of the infinity offered by $\mathbb{N}$.

The second difference is the syntax of types: $\mathrm{C}\lambda_{\to}$ introduces probabilities through counting quantifiers C^p, where $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ has probabilities as base types. Types are nevertheless isomorphic: $\mathrm{C}\lambda_{\to}$ types $\mathfrak{s}$ are of the form $\mathrm{C}^p(\mathfrak{s}_1 \to \cdots \to \mathfrak{s}_n \to o)$, with a fixed outer counting quantifier, and map 1-to-1 onto $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ types A of the form $A_1 \to \ldots \to A_n \to p$ by associating the probability p instead with the consequent o. Formally, we encode types $\mathfrak{s}$ and typing contexts Γ of $\mathrm{C}\lambda_{\to}$ into our setting as follows.

$$\begin{aligned}[\![\mathrm{C}^q(o)]\!] &= q \\ [\![\mathrm{C}^q(\mathfrak{s} \Rightarrow \tau)]\!] &= [\![\mathfrak{s}]\!] \to [\![\mathrm{C}^q(\tau)]\!]\end{aligned} \qquad [\![x_1 : \mathfrak{s}_1, \ldots, x_n : \mathfrak{s}_n]\!] = x_1 : [\![\mathfrak{s}_1]\!], \ldots, x_n : [\![\mathfrak{s}_n]\!]$$

Having connected boolean formulas $\mathscr{b}$ to event valuations E, and $\mathrm{C}\lambda_{\to}$ types to $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ types, we may state the following conservativity result of $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ over $\mathrm{C}\lambda_{\to}$.

▶ **Proposition 11.** *If* $E \in [\![\mathscr{b}]\!]_X$ *then*

$$\Gamma \vdash^X M : \mathscr{b} \rightarrowtail \mathfrak{s} \quad \textit{implies} \quad E \mid [\![\Gamma]\!] \vdash M : [\![\mathfrak{s}]\!] \ .$$

Proof. By induction on the typing derivation for $\Gamma \vdash^X M : \mathscr{b} \rightarrowtail \mathfrak{s}$. ◀

▶ **Corollary 12.** *For a given closed term* M *the type system* $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ *gives the same or higher termination bounds than* $\mathrm{C}\lambda_{\to}$.

In the reverse direction, Example 8 and its counterpart in [1, Section 6.2] show that the two type systems do not give the exact same bounds, and in some cases $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ gives a strictly higher bound. The reason is that $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ locates branching between alternatives at the gen-rule for $\boxed{a}.M$, where $\mathrm{C}\lambda_{\to}$ branches for *choice* terms $M\,a\,N$ or in a contraction rule. Crucially, the gen-rule allows branches with different termination bounds. We illustrate this further by attempting to simulate the rule for $\boxed{a}.M$ in $\mathrm{C}\lambda_{\to}$.

$$\frac{E, a \mapsto \top \mid [\![\Gamma]\!] \vdash M : \overline{A} \to p \qquad E, a \mapsto \bot \mid [\![\Gamma]\!] \vdash M : \overline{A} \to q}{E \mid [\![\Gamma]\!] \vdash \boxed{a}.M : \overline{A} \to \tfrac{1}{2}p + \tfrac{1}{2}q}\,\mathsf{gen}$$

The branching at this rule in $\mathrm{C}\lambda_{\to}$ is captured with a contraction on the two premisses, which requires the probabilities to be equal, i.e. $p = q$. The derivation is as follows, where $\mathscr{d} = \top$ and $\mu(\mathscr{d}) = 1$ for the counting rule.

$$\frac{\dfrac{\Gamma \vdash^{X\cup\{a\}} M : \mathscr{b} \wedge a \rightarrowtail \mathrm{C}^p\sigma \qquad \Gamma \vdash^{X\cup\{a\}} M : \mathscr{b} \wedge \neg a \rightarrowtail \mathrm{C}^p\sigma \qquad \mathscr{b} \models (\mathscr{b} \wedge a) \vee (\mathscr{b} \wedge \neg a)}{\Gamma \vdash^{X\cup\{a\}} M : \mathscr{b} \rightarrowtail \mathrm{C}^p\sigma}}{\Gamma \vdash^X \boxed{a}.M : \mathscr{b} \rightarrowtail \mathrm{C}^p\sigma}$$

This is the issue illustrated by Example 8 and its counterpart in [1, Section 6.2], for which $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ gives the actual termination probability of $\frac{3}{8}$, while $\mathrm{C}\lambda_{\to}$ gives a best approximation of $\frac{1}{4}$. Reprising the example in $\mathrm{C}\lambda_{\to}$, the two sub-derivations for $\boxed{b}.\boxed{c}.T$, given in condensed form below, assign probabilities of $\frac{1}{4}$ and $\frac{1}{2}$. These may only be combined in a contraction by lowering the first probability to match the $\frac{1}{4}$ of the first.

$$\frac{\dfrac{\vdash^{\{a,b,c\}} T : a \wedge b \wedge c \rightarrowtail \mathrm{C}^1\sigma}{\vdash^{\{a,b\}} \boxed{c}.T : a \wedge b \rightarrowtail \mathrm{C}^{\frac{1}{2}}\sigma}}{\vdash^{\{a\}} \boxed{b}.\boxed{c}.T : a \rightarrowtail \mathrm{C}^{\frac{1}{4}}\sigma} \qquad \frac{\dfrac{\vdash^{\{a,b,c\}} T : \neg a \wedge \neg b \rightarrowtail \mathrm{C}^1\sigma}{\vdash^{\{a,b\}} \boxed{c}.T : \neg a \wedge \neg b \rightarrowtail \mathrm{C}^1\sigma}}{\vdash^{\{a\}} \boxed{b}.\boxed{c}.T : \neg a \rightarrowtail \mathrm{C}^{\frac{1}{2}}\sigma}$$

The above analysis suggests that this issue with $\mathrm{C}\lambda_{\to}$ may be fixed by adopting the generator typing rule of $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$, adjusted appropriately as follows. The key here is that different branches feature the dual atoms a and $\neg a$, not seen in either the simple type system $\mathsf{PE}\Lambda^{\mathbb{Q}\to}$ nor the intersection type system in [1].

$$\frac{\Gamma \vdash^{X\cup\{a\}} M : \mathscr{c} \wedge a \rightarrowtail \mathrm{C}^{p}\sigma \qquad \Gamma \vdash^{X\cup\{a\}} M : \mathscr{d} \wedge \neg a \rightarrowtail \mathrm{C}^{q}\sigma \qquad \mathscr{b} \models \mathscr{c} \vee \mathscr{d}}{\Gamma \vdash^{X} \boxed{a}.\, M : \mathscr{b} \rightarrowtail \mathrm{C}^{\frac{1}{2}p+\frac{1}{2}q}\sigma}$$

5 Sequencing

Two observations about probabilistic λ-calculi motivate the developments in the remainder of this paper. The first is the primary role of *head reduction*. It is well known that head normalization corresponds closely to evaluation on the Krivine Machine [23], and sometimes the machine gives a more natural model of what is being studied. The second is that probabilistic evaluation in λ-calculi needs to account for the difference between call–by–value (cbv) and call–by–name (cbn), to which end additional constructions are introduced, sometimes ad-hoc. The $\mathsf{PE}\Lambda$ is an example, as is the separate consideration of a cbv- and a cbn-probabilistic sum by Faggian and Ronchi Della Rocca [16], while Antonelli, Dal Lago, and Pistone [1] add to the standard cbn-application a second cbv-application.

These observations prompted us to consider probabilistic termination from the perspective of the *Functional Machine Calculus* (FMC) [18], a λ-calculus with computational effects. Firstly, the Krivine Machine plays a central role in the FMC (indeed it is the "M" in "FMC"), while the FMC provides the machine with a new notion of *successful termination*, absent from the standard λ-calculus. It is then a natural question if and how this may be used to capture the *probability* of successful termination. Secondly, the FMC may express both cbn and cbv behaviour, with the cbn λ-calculus a fragment and the cbv λ-calculus encoded in the syntax. The need for ad-hoc constructs to control reduction behaviour is thus avoided.

The Krivine Machine, simplified by replacing *environments* with *substitution*, evaluates a λ-term in the presence of a stack of input terms. An abstraction $\lambda x.\, M$ pops the top off the stack, say N, and continues as $\{N/x\}M$, while an application $M\, N$ pushes its argument N and continues as M. A λ-term may thus be viewed as a language of instruction sequences for this machine: application–*push*, abstraction–*pop*, variable–*execute*.

The FMC then extends the λ-calculus with *sequential composition* $M\,;N$ and its unit, the imperative *skip* $\star$, with the expected semantics: *concatenation* of machine instructions and the *empty* instruction. As in models of imperative languages, *skip* indicates successful termination of the machine. This gives a fragment called the *sequential λ-calculus*.

We adopt these modifications in the $\mathsf{PE}\Lambda$ to give the *sequential probabilistic event λ-calculus* ($\mathsf{SPE}\Lambda$), defined below. Following [18] we render abstraction as $\langle x\rangle.\, M = \lambda x.\, M$ and application as $[N].\, M = M\, N$ to emphasise the machine behaviour of *pop* and *push*, retaining the standard syntax as a shorthand. In particular, the new notation clarifies the interaction between *push* and *sequencing*: the following three terms are equivalent, rendered first in standard notation and second with prefix application.

$$(\star\, N)\,;M \;\sim\; (\star\,;M)\, N \;\sim\; M\, N \qquad\qquad ([N].\,\star)\,;M \;\sim\; [N].\,(\star\,;M) \;\sim\; [N].\,M$$

The full FMC further generalises to a machine with multiple independent stacks, addressed by a set of *locations*, in which *pop* and *push* are then parameterised to operate on the corresponding stack. This allows us to encode the effects of mutable higher-order store, input/output, and indeed probabilistic computation: the generator $\boxed{a}.\, M$ of the $\mathsf{PE}\Lambda$ is, in the FMC, an abstraction $\mathsf{rnd}\langle\, a\,\rangle.\, M$ parameterised to draw from a stream of random values labelled rnd. The $\mathsf{SPE}\Lambda$ is thus a fragment of the FMC with two locations.

▶ **Definition 13.** *The* sequential probabilistic event λ-calculus $\mathsf{SPE}\Lambda$ *is given as follows.*

$$M, N, P ::= x \mid \langle x \rangle . M \mid [N] . M \mid \boxed{a} . M \mid M\, a\, N \mid \star \mid N \,;\, M$$

Prefixing binds tighter than sequencing, $[N] . M \,;\, P = ([N] . M) \,;\, P$, and sequencing associates right, $M \,;\, N \,;\, P = M \,;\, (N \,;\, P)$. *Projections* and *contexts* extend to the $\mathsf{SPE}\Lambda$ as below; *head contexts* and *applicative contexts* are as for the $\mathsf{PE}\Lambda$.

$$\pi_i^a \star = \star \qquad \pi_i^a (N \,;\, M) = \pi_i^a N \,;\, \pi_i^a M \qquad C ::= \ldots \mid C \,;\, M \mid M \,;\, C$$

The interaction between sequentiality and the λ-calculus is governed by the following *sequencing* reduction rules. These make sequential composition right-associative, and let the *prefixing* of *push*, *pop*, and the generator propagate past it (as in a standard list concatenation algorithm). The result is to make the first such action on the abstract machine the leading construct.

$$\star \,;\, P \to_\sigma P \qquad \begin{array}{ll} (N \,;\, M) \,;\, P \to_\sigma N \,;\, (M \,;\, P) & \langle x \rangle . M \,;\, P \to_\sigma \langle x \rangle . (M \,;\, P) \quad (x \notin \mathsf{fv}(P)) \\ [N] . M \,;\, P \to_\sigma [N] . (M \,;\, P) & \boxed{a} . M \,;\, P \to_\sigma \boxed{a} . (M \,;\, P) \quad (a \notin \mathsf{fe}(P)) \end{array}$$

The *sequencing* relation $\to_\sigma$ is given by closing these rules under all contexts C, and *head sequencing* $\to_{\sigma\mathsf{h}}$ by closing under head contexts H only. The β-reduction rule in $\mathsf{SPE}\Lambda$ notation is as below left, with $\to_\beta$ given by closing under all contexts and $\to_{\beta\mathsf{h}}$ by closing under head contexts. Projective reduction, below right, is as previously.

$$[N] . \langle x \rangle . M \to_\beta \{N/x\} M \qquad H\{\boxed{a} . M\} \to_\pi [\, \tfrac{1}{2} \cdot H\{\pi_\top^a M\} \,,\, \tfrac{1}{2} \cdot H\{\pi_\top^a M\} \,]$$

Head reduction $\to_\mathsf{h}$ is the union of all three head relations:

$$\to_\mathsf{h} \;=\; \to_{\beta\mathsf{h}} \cup \to_{\sigma\mathsf{h}} \cup \to_\pi \;.$$

We round off by observing the shape of head-normal forms, assuming no free event variables.

▶ **Proposition 14.** *The head-normal forms of event-closed* $\mathsf{SPE}\Lambda$*-terms are of one of the three forms* $H\{\star\}$*,* $H\{x\}$*, and* $H\{x \,;\, M\}$*.*

5.1 Encoding call–by–value

Sequential composition provides an essential element that the λ-calculus lacks, and which is at the heart of the cbv/cbn dichotomy: *control over execution*. The cbv behaviour of an application $M\, N$ is encoded almost as $N \,;\, M$: *first* evaluate the argument N, *then* evaluate the function M (the full encoding, below, includes an extra part to also *execute* M).

We demonstrate the encoding of the cbv-probabilistic λ-calculus $\Lambda_\oplus^{\mathsf{cbv}}$ of Faggian and Ronchi Della Rocca [16]. The encoding of cbv λ-terms is standard: see [13, 18, 28]. *Values* V, W and *terms* M, N encode by the translations $-_\mathsf{v}$ and $-_\mathsf{t}$ respectively, below.

$$\begin{array}{lrlrl} \text{Values } V, W: & x_\mathsf{v} &= x & \text{Terms } M, N: & V_\mathsf{t} = [V_\mathsf{v}] . \star \\ & (\lambda x.M)_\mathsf{v} &= \langle x \rangle . M_\mathsf{t} & & (M\, N)_\mathsf{t} = N_\mathsf{t} \,;\, M_\mathsf{t} \,;\, \langle x \rangle . x \\ & & & & (M \oplus N)_\mathsf{t} = \boxed{a} . M_\mathsf{t}\, a\, N_\mathsf{t} \end{array}$$

The operational intuition is that a *push* represents a *return value*: a term M_t evaluates until it is of the form $V_\mathsf{t} = [V_\mathsf{v}] . \star$, at which point V_v is pushed to the stack and the machine terminates. Then β-reduction is simulated as follows.

$$\begin{array}{rcl}((\lambda x.M)\,V)_{\mathsf{t}} & = & [V_{\mathsf{v}}].\ \star\ ;\ [\langle x\rangle.\,M_{\mathsf{t}}].\ \star\ ;\ \langle y\rangle.\,y \\ & \twoheadrightarrow_{\sigma} & [V_{\mathsf{v}}].\,[\langle x\rangle.\,M_{\mathsf{t}}].\,\langle y\rangle.\,y \\ & \to_{\beta} & [V_{\mathsf{v}}].\,\langle x\rangle.\,M_{\mathsf{t}} \\ & \to_{\beta} & \{V_{\mathsf{v}}/x\}M_{\mathsf{t}} \\ & = & (\{V/x\}M)_{\mathsf{t}}\end{array}$$

The probabilistic reduction rule of $\Lambda^{\mathsf{cbv}}_{\oplus}$ is that of projective reduction, under the given encoding, but it applies in *surface contexts*:

$$S \ ::= \ \{\} \ \mid \ M\,S \ \mid \ S\,M$$

Then the translation of $S\{M \oplus N\}$ indeed does not place the probabilistic redex inside a *push*, the requirement for correct behaviour.

5.2 The abstract machine

The small-step operational semantics of the SPEΛ is given by the following abstract machine. A *state* is a triple (S, M, K), where M is a term and S and K are stacks of terms. S is the *operand stack*, with the head to the right as $S\,N$, and K is the *continuation stack*, with the head to the left as $N\,K$. In both cases the empty stack is written ε, and concatenation by juxtaposition, $S\,T$. *Transitions* or *steps* are probabilistic: a transition rule is written as below left, expressing that the machine transitions from a state (S, M, K) to (T, N, L) with probability p. We may omit p when $p = 1$. A *run* is a sequence of steps, written as below centre, where probabilities are multiplied, i.e. p below is the product of the probabilities of all steps. A run is *successful* if it terminates with *skip* and an empty continuation stack, as below right; the stack T then holds the *return values* of the computation.

$$\text{step: } p\frac{(\,S\ ,\ M\ ,\ K\,)}{(\,T\ ,\ N\ ,\ L\,)} \qquad \text{run: } p\frac{(\,S\ ,\ M\ ,\ K\,)}{(\,T\ ,\ N\ ,\ L\,)} \qquad \text{successful run: } p\frac{(\,S\ ,\ M\ ,\ K\,)}{(\,T\ ,\ \star\ ,\ \varepsilon\,)}$$

▶ **Definition 15.** *The* sequential probabilistic machine *(SPM) is given by the following probabilistic* transitions*.*

$$\frac{(\,S\ ,\ [N].\,M\ ,\ K\,)}{(\,S\,N\ ,\ M\ ,\ K\,)} \qquad \frac{(\,S\ ,\ N\,;M\ ,\ K\,)}{(\,S\ ,\ N\ ,\ M\,K\,)} \qquad \frac{1}{2}\frac{(\,S\ ,\ \boxed{a}.\,M\ ,\ K\,)}{(\,S\ ,\ \pi^a_{\top}M\ ,\ K\,)}$$

$$\frac{(\,S\,N\ ,\ \langle x\rangle.\,M\ ,\ K\,)}{(\,S\ ,\ \{N/x\}M\ ,\ K\,)} \qquad \frac{(\,S\ ,\ \star\ ,\ M\,K\,)}{(\,S\ ,\ M\ ,\ K\,)} \qquad \frac{1}{2}\frac{(\,S\ ,\ \boxed{a}.\,M\ ,\ K\,)}{(\,S\ ,\ \pi^a_{\bot}M\ ,\ K\,)}$$

5.3 Big-step semantics

Running the machine for a given term and input stack gives a distribution of return stacks. We use the following notation, extending from that for distributions over terms.

$$\mathcal{T} \ = \ [p_1\cdot T_1, \ldots, p_n\cdot T_n] \ = \ [p_i\cdot T_i]_{i\le n}$$

▶ **Definition 16.** *The* evaluation *relation $S, M \Downarrow \mathcal{T}$ is defined inductively by the following rules.*

$$\frac{}{S, \star \Downarrow [1\cdot S]} \qquad \frac{S, \{N/x\}M \Downarrow \mathcal{T}}{S\,N, \langle x\rangle.\,M \Downarrow \mathcal{T}} \qquad \frac{R, M \Downarrow [p_i\cdot S_i]_{i\le n} \qquad (\,S_i, N \Downarrow \mathcal{T}_i\,)_{i\le n}}{R, M\,;N \Downarrow \sum_{i\le n} p_i\mathcal{T}_i}$$

$$\frac{}{S, M \Downarrow \varnothing} \qquad \frac{S\,N, M \Downarrow \mathcal{T}}{S, [N].\,M \Downarrow \mathcal{T}} \qquad \frac{S, \pi^a_{\top}M \Downarrow \mathcal{T}_{\top} \qquad S, \pi^a_{\bot}M \Downarrow \mathcal{T}_{\bot}}{S, \boxed{a}.\,M \Downarrow \frac{1}{2}\mathcal{T}_{\top} + \frac{1}{2}\mathcal{T}_{\bot}}$$

We demonstrate that small-step and big-step semantics agree.

▶ **Proposition 17.** *$S, M \Downarrow \mathcal{T}$ if and only if there is a finite collection of n distinct runs*

$$p_i \frac{(\,S\ ,\ M\ ,\ \varepsilon\,)}{(\,T_i\ ,\ \star\ ,\ \varepsilon\,)} \quad (i \leq n) \quad \textit{such that} \quad \mathcal{T} = [p_i \cdot T_i]_{i \leq n}\ .$$

Proof. ($\Longrightarrow$) By induction on $S, M \Downarrow \mathcal{T}$. ($\Longleftarrow$) By induction each run in the collection of n runs. ◀

6 Sequential probabilistic types

Types for the sequential λ-calculus are of the form below, with the meaning: given an input stack of terms typed by A_1 through A_n, the machine will terminate successfully and return a stack with types B_1 through B_m.

$$A_n \dots A_1 \Rightarrow B_1 \dots B_m$$

For the SPEΛ, we parameterize this with the *probability* of successful termination.

▶ **Definition 18.** Sequential probabilistic types *are given by the following grammars, where p is a probability.*

$$\begin{array}{rcll} A, B, C & ::= & \overline{A} \overset{p}{\Rightarrow} \overline{C} & \textit{(types)} \\ \overline{A} & ::= & A_1 \dots A_n & \textit{(type vectors)} \end{array}$$

A typing judgement $E \mid \Gamma \vdash M \colon A$ *assigns a term M the type A in the context of E and Γ, and $E \mid \Gamma \vdash S \colon \overline{A}$ assigns a stack of terms S a type vector $\overline{A}$. The sequential probabilistic type system* $\mathsf{SPE}\Lambda^{\mathbb{Q}\Rightarrow}$ *is given by the typing rules in Figure 4. We may omit p when $p = 1$.*

There are no base types: their rôle is subsumed by types with empty vectors ($\overset{p}{\Rightarrow}$). Observe that because stacks are last-in-first-out, the identity term on two elements is $\langle x \rangle . \langle y \rangle . [y] . [x] . \star$, i.e. with the order of x and y reversed between popping and pushing. We match this reversal in types, and assign this term the type $AB \Rightarrow BA$. Since we want identity types to be of the form $\overline{A} \Rightarrow \overline{A}$, in a type $\overline{A} \overset{p}{\Rightarrow} \overline{C}$ we consider the antecedent type vector $\overline{A}$ to be reversed, i.e.

$$\overline{A} \Rightarrow \overline{A} \quad = \quad A_n \dots A_1 \Rightarrow A_1 \dots A_n\ .$$

Probabilistic PEΛ-types embed into sequential types by $\overline{A} \to p \quad = \quad \overline{A} \overset{p}{\Rightarrow} \varepsilon$. With this identification, for PEΛ-terms and -types the two type systems coincide.

▶ **Proposition 19.** *For M a* PEΛ*-term, $E \mid \Gamma \vdash M \colon \overline{A} \to p$ if and only if $E \mid \Gamma \vdash M \colon \overline{A} \overset{p}{\Rightarrow} \varepsilon$.*

Every type is inhabited by a closed term. For a type A, define the *zero term* 0_A as follows: for $A = B_1 \dots B_m \overset{p}{\Rightarrow} C_1 \dots C_n$,

$$0_A \ = \ \langle x_1 \rangle \dots \langle x_m \rangle . [0_{C_1}] \dots [0_{C_n}] . \star\ .$$

▶ **Proposition 20** (Type inhabitation)**.** *Every type A is inhabited by its zero term, $\vdash 0_A \colon A$.*

Proof. By induction on the type A, using the low rule to lower the termination probability from 1 to an arbitrary p. ◀

$$\frac{}{E \mid \Gamma,\, x\colon \overline{A} \overset{p}{\Rightarrow} \overline{C} \vdash x\colon \overline{A}\,\overline{B} \overset{p}{\Rightarrow} \overline{B}\,\overline{C}}\ \mathsf{var} \qquad \frac{}{E \mid \Gamma \vdash M\colon \overline{A} \overset{0}{\Rightarrow} \overline{C}}\ \mathsf{zero}$$

$$\frac{E \mid \Gamma,\, x\colon A \vdash M\colon \overline{B} \overset{p}{\Rightarrow} \overline{C}}{E \mid \Gamma \vdash \langle x\rangle.\, M\colon A\,\overline{B} \overset{p}{\Rightarrow} \overline{C}}\ \mathsf{abs} \qquad \frac{E \mid \Gamma \vdash N\colon A \qquad E \mid \Gamma \vdash M\colon A\,\overline{B} \overset{p}{\Rightarrow} \overline{C}}{E \mid \Gamma \vdash [N].\, M\colon \overline{B} \overset{p}{\Rightarrow} \overline{C}}\ \mathsf{app}$$

$$\frac{E, a{\mapsto} i \mid \Gamma \vdash M_i\colon A}{E, a{\mapsto} i \mid \Gamma \vdash M_\top\, a\, M_\bot\colon A}\ \mathsf{chc} \qquad \frac{E, a{\mapsto}\top \mid \Gamma \vdash M\colon \overline{A} \overset{p}{\Rightarrow} \overline{C} \qquad E, a{\mapsto}\bot \mid \Gamma \vdash M\colon \overline{A} \overset{q}{\Rightarrow} \overline{C}}{E \mid \Gamma \vdash \boxed{a}.\, M\colon \overline{A} \overset{\frac{1}{2}p+\frac{1}{2}q}{\Rightarrow} \overline{C}}\ \mathsf{gen}$$

$$\frac{}{E \mid \Gamma \vdash \star\colon \overline{A} \overset{1}{\Rightarrow} \overline{A}}\ \mathsf{skip} \qquad \frac{E \mid \Gamma \vdash M\colon \overline{A} \overset{p}{\Rightarrow} \overline{B} \qquad E \mid \Gamma \vdash N\colon \overline{B} \overset{q}{\Rightarrow} \overline{C}}{E \mid \Gamma \vdash M\,;N\colon \overline{A} \overset{pq}{\Rightarrow} \overline{C}}\ \mathsf{seq}$$

$$\frac{E \mid \Gamma \vdash M\colon \overline{A} \overset{q}{\Rightarrow} \overline{C}}{E \mid \Gamma \vdash M\colon \overline{A} \overset{p}{\Rightarrow} \overline{C}}\ \mathsf{low}\ (p{<}q) \qquad \frac{(E \mid \Gamma \vdash M_i\colon A_i)_{\,i\le n}}{E \mid \Gamma \vdash \varepsilon\, M_1 \dots M_n\colon A_1 \dots A_n}\ \mathsf{stk}$$

Figure 4 The sequential probabilistic type system $\mathsf{SPE}\Lambda^{\mathbb{Q}\Rightarrow}$.

Type inhabitation is an important justification for the interpretation of types as a guarantee for successful machine termination: it means that for a typed term $M\colon A \overset{p}{\Rightarrow} \overline{C}$, a suitable input stack $S : \overline{A}$ always exists. Our main theorem, below, formalizes this interpretation: for a term $M\colon A \overset{p}{\Rightarrow} \overline{C}$ and stack $S : \overline{A}$, the machine successfully returns a stack $T : \overline{C}$ with probability at least p.

▶ **Theorem 21.** *For a typed, closed term $M\colon \overline{A} \overset{p}{\Rightarrow} \overline{C}$ and stack $S : \overline{A}$ there is a finite set of distinct successful runs*

$$p_i \frac{(S\ ,\ M\ ,\ \varepsilon)}{(T_i\ ,\ \star\ ,\ \varepsilon)} \qquad (i \le n)$$

with sum probability $\sum_{i\le n} p_i \ge p$.

Proof. See Appendix A.1. ◀

The lower bound given by the theorem is an exact bound when two conditions are met: the typing derivation does not use the low rule to lower the termination bound, and every use of the zero rule to assign a zero probability applies to a term that is in fact non-terminating. In such a case, such as Example 8, types can be seen to give an exact bound.

Since head reduction as a strategy closely follows machine evaluation, it is no surprise that the termination bound for the machine is also a lower bound for probabilistic head reduction. This is formalised in the following theorem.

▶ **Theorem 22.** *For a closed M, if $M\colon \overline{A} \overset{p}{\Rightarrow} \overline{C}$ then $M \twoheadrightarrow_{\mathsf{h}} \mathcal{N}_0 + \mathcal{N}_1$ where all terms in $\mathcal{N}_0$ are head normal and $|\mathcal{N}_0| \ge p$.*

Proof. See Appendix A.2. ◀

Observe that our main theorem on probabilistic termination for the PEΛ, Theorem 10, follows immediately from the corresponding Theorem 22 above by conservativity of the SPEΛ over the PEΛ. This is despite the fact that the proof of Theorem 22 makes essential use of the notion of successful termination made available by sequencing, absent from the PEΛ.

7 Conclusions

To us, what stands out about our approach is the simplicity and the natural intuition of our type system. This manifests in the transparent reasoning in our proofs, which despite using deep techniques such as abstract reducibility, give a direct and clear connection between types, machine behaviour, and reduction.

Compared with the approach in [1], the simplicity of our approach manifests in several advantages. First is to avoid the need for *counting quantifiers*, associating probabilities instead with base types, and the use of simple event valuations over boolean formulas. Second, it is clear that the expression of both call–by–name and call–by–value behaviour is an essential ingredient for a probabilistic calculus. We eschew the introduction of ad-hoc constructs, relying instead on a principled interpretation of `cbv` via sequential composition. Finally, where the main example [1, Example 6.2] requires intersection types to produce the exact bound, our approach does so directly in Example 8, while morally remaining within the realm of simple types, and strictly improving on the bounds provided by $C\lambda_{\to}$.

References

1 Melissa Antonelli, Ugo Dal Lago, and Paolo Pistone. Curry and howard meet borel. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'22)*, pages 45:1–45:13. ACM, 2022. doi:10.1145/3531130.3533361.

2 Martin Avanzini, Ugo Dal Lago, and Akihisa Yamada. On probabilistic term rewriting. In John P. Gallagher and Martin Sulzmann, editors, *Proc. Functional and Logic Programming - 14th International Symposium, FLOPS 2018*, volume 10818 of *Lecture Notes in Computer Science*, pages 132–148. Springer, 2018. doi:10.1007/978-3-319-90686-7_9.

3 Chris Barrett, Willem Heijltjes, and Guy McCusker. The Functional Machine Calculus II: Semantics. In *31st EACSL Annual Conference on Computer Science Logic (CSL 2023)*, volume 252 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:18, 2023. doi:10.4230/LIPIcs.CSL.2023.10.

4 Raven Beutner and Luke Ong. On probabilistic termination of functional programs with continuous distributions. In Stephen N. Freund and Eran Yahav, editors, *Proceedings 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI)*, pages 1312–1326. ACM, 2021. doi:10.1145/3453483.3454111.

5 Flavien Breuvart and Ugo Dal Lago. On intersection types and probabilistic lambda calculi. In *roceedings of the 20th International Symposium on Principles and Practice of Declarative Programming, PPDP 2018*, pages 8:1–8:13. ACM, 2018. doi:10.1145/3236950.3236968.

6 Fredrik Dahlqvist and Dexter Kozen. Semantics of higher-order probabilistic programs with conditioning. *Proceedings of the ACM on Programming Languages*, 4(POPL):57:1–57:29, 2020. doi:10.1145/3371125.

7 Ugo Dal Lago, Claudia Faggian, and Simona Ronchi Della Rocca. Intersection types and (positive) almost-sure termination. *Proceedings of the ACM on Programming Languages*, 5(POPL):1–32, 2021. doi:10.1145/3434313.

8 Ugo Dal Lago and Charles Grellois. Probabilistic termination by monadic affine sized typing. *ACM Transactions on Programming Languages and Systems*, 41(2):10:1–10:65, 2019. doi:10.1145/3293605.

9 Ugo Dal Lago, Giulio Guerrieri, and Willem Heijltjes. Decomposing probabilistic lambda-calculi. In *Proceedings of Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 12077 of *LNCS*, pages 136–156, 2020. doi:10.1007/978-3-030-45231-5_8.

10 Ugo Dal Lago and Margherita Zorzi. Probabilistic operational semantics for the lambda calculus. *RAIRO - Theoretical Informatics and Applications*, 46(3):413–450, 2012. doi:10.1051/ita/2012012.

11 Karel de Leeuw, Edward F. Moore, Claude E. Shannon, and Norman Shapiro. Computability by probabilistic machines. *Automata studies*, 34:183–198, 1956.

12 Ugo De'Liguoro and Adolfo Piperno. Non deterministic extensions of untyped lambda-calculus. *Information and Computation*, 122(2):149–177, 1995. doi:10.1006/inco.1995.1145.

13 Rémi Douence and Pascal Fradet. A systematic study of functional language implementations. *ACM Transactions on Programming Languages and Systems*, 20(2):344–387, 1998. doi:10.1145/276393.276397.

14 Thomas Ehrhard and Giulio Guerrieri. The bang calculus: An untyped lambda-calculus generalizing call-by-name and call-by-value. In *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming (PPDP'16)*, pages 174–187, 2016. doi:10.1145/2967973.2968608.

15 Thomas Ehrhard and Christine Tasson. Probabilistic call by push value. *Logical Methods in Computer Science*, 15(1), 2019. doi:10.23638/LMCS-15(1:3)2019.

16 Claudia Faggian and Simona Ronchi Della Rocca. Lambda calculus and probabilistic computation. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019*, pages 1–13, 2019. doi:10.1109/LICS.2019.8785699.

17 Jean Goubault-Larrecq. A probabilistic and non-deterministic call-by-push-value language. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019*, pages 1–13. IEEE Computer Society, 2019. doi:10.1109/LICS.2019.8785809.

18 Willem Heijltjes. The functional machine calculus. In *Proceedings of the 38th Conference on the Mathematical Foundations of Programming Semantics, MFPS XXXVIII*, volume 1 of *ENTICS*, 2022. doi:10.46298/ENTICS.10513.

19 C. Jones and G. D. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89)*, pages 186–195. IEEE Computer Society, 1989. doi:10.1109/LICS.1989.39173.

20 Achim Jung and Regina Tix. The troublesome probabilistic powerdomain. *Electronic Notes in Theoretical Computer Science*, 13:70–91, 1998. doi:10.1016/S1571-0661(05)80216-6.

21 Benjamin Kaminski. *Advanced Weakest Precondition Calculi for Probabilistic Programs*. PhD thesis, Fakultät für Mathematik, Informatik und Naturwissenschaften, RWTH Aachen University, 2019. doi:10.18154/RWTH-2019-01829.

22 Naoki Kobayashi, Ugo Dal Lago, and Charles Grellois. On the termination problem for probabilistic higher-order recursive programs. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019*, pages 1–14. IEEE, 2019. doi:10.1109/LICS.2019.8785679.

23 Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007. doi:10.1007/s10990-007-9018-9.

24 Thomas Leventis. A deterministic rewrite system for the probabilistic λ-calculus. *Mathematical Structures in Computer Science*, 29(10):1479–1512, 2019. doi:10.1017/S0960129519000045.

25 Paul Blain Levy. *Call-by-push-value: A functional/imperative synthesis*, volume 2 of *Semantic Structures in Computation*. Springer Netherlands, 2003. doi:10.1007/978-94-007-0954-6.

26 Udi Manber and Martin Tompa. Probabilistic, nondeterministic, and alternating decision trees. In *14th Annual ACM Symposium on Theory of Computing*, pages 234–244, 1982. doi:10.1145/800070.802197.

27 Annabelle McIver and Carroll Morgan. Abstraction and refinement in probabilistic systems. *SIGMETRICS Perform. Eval. Rev.*, 32(4):41–47, March 2005. doi:10.1145/1059816.1059824.

28 A.J. Power and Hayo Thielecke. Closed Freyd- and κ-categories. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 1644 of *Lecture Notes in Computer Science*, pages 625–634. Springer, 1999. doi:10.1007/3-540-48523-6_59.

29 Norman Ramsey and Avi Pfeffer. Stochastic lambda calculus and monads of probability distributions. In *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '02, pages 154–165, 2002. doi:10.1145/503272.503288.

30 Nasser Saheb-Djahromi. Probabilistic LCF. In *Mathematical Foundations of Computer Science 1978, Proceedings, 7th Symposium*, volume 64 of *Lecture Notes in Computer Science*, pages 442–451. Springer, 1978. doi:10.1007/3-540-08921-7_92.

31 Davide Sangiorgi and Valeria Vignudelli. Environmental bisimulations for probabilistic higher-order languages. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016*, pages 595–607, 2016. doi:10.1145/2837614.2837651.

A Proofs for Section 6

A.1 Probabilistic machine termination

Our main theorem for the typed SPEΛ will be that a type $\overline{A} \overset{p}{\Rightarrow} \overline{B}$ guarantees a probability of termination of the machine of at least p, given an input stack of type $\overline{A}$, and returning a stack of type $\overline{B}$. The proof is a direct application of abstract reducibility. For each type we define a set $\mathrm{RUN}(A)$ that holds the terms with the above property, and we proceed to prove that every term of type A belongs to this set.

We write $\mathcal{D}_p(X)$ for the set of finite distributions $\mathcal{X}$ of weight $|\mathcal{X}| \geq p$ over a set X.

▶ **Definition 23.** *The set* $\mathrm{RUN}(-)$ *is defined by mutual induction on types* A *and type vectors* $\overline{A}$ *as a set of closed terms respectively of closed stacks, as follows.*

$$\mathrm{RUN}(\overline{A} \overset{p}{\Rightarrow} \overline{B}) = \{\ M \mid \forall S \in \mathrm{RUN}(\overline{A}).\, \exists \mathcal{T} \in \mathcal{D}_p(\mathrm{RUN}(\overline{B}))\,.\ S, M \Downarrow \mathcal{T}\ \}$$
$$\mathrm{RUN}(A_1 \dots A_n) = \{\ \varepsilon\, M_1 \dots M_n \mid M_i \in \mathrm{RUN}(A_i)\ \}$$

For the proof of Lemma 25, the main reducibility lemma, we need the following notation, as well as an additional lemma. We write σ for a substitution map $\{M_i/x_i\}_{i\leq n}$, where σM is the application of σ to M. The set $\mathrm{RUN}(-)$ then extends to contexts Γ as follows. Note that the definition implies that σ is closed (i.e. each substituting term is closed).

$$\mathrm{RUN}(x_1 \colon A_1, \dots, x_n \colon A_n) = \{\ \sigma \mid \sigma x_i \in \mathrm{RUN}(A_i)\ ,\ 1 \leq i \leq n\ \}$$

For an event valuation $E = (a_k \mapsto i_k)_{k\leq n}$ we write $\pi_E M$ for the projection on each a_i in E.

$$\pi_E M = \pi^{a_1}_{i_1}(\dots(\pi^{a_n}_{i_n} M)\dots)$$

We *expand* the stacks in a distribution $\mathcal{T}$ by prepending a stack S as $S\,\mathcal{T}$, where $S\,[p_i \cdot T_i]_{i\leq n} = [p_i \cdot S\,T_i]_{i\leq n}$.

▶ **Lemma 24.** *If* $S, M \Downarrow \mathcal{T}$ *then* $R\,S, M \Downarrow R\,\mathcal{T}$ *for any stack* R.

Proof. By induction on the definition of $\Downarrow$. ◀

The main reducibility lemma is then the following.

▶ **Lemma 25.** *If* $E \mid \Gamma \vdash M \colon A$ *and* $\sigma \in \mathrm{RUN}(\Gamma)$ *then* $\pi_E(\sigma M) \in \mathrm{RUN}(A)$.

Proof. By induction on the typing derivation for M. Note that since σ is closed, $\pi_E(\sigma M) = \sigma(\pi_E(M))$. We cover three key cases (sequencing, abstraction, and generator); the remaning are similar.

Sequencing case:

$$\frac{E \mid \Gamma \vdash N \colon \overline{A} \overset{p}{\Rightarrow} \overline{B} \quad E \mid \Gamma \vdash M \colon \overline{B} \overset{q}{\Rightarrow} \overline{C}}{E \mid \Gamma \vdash N\,; M \colon \overline{A} \overset{pq}{\Rightarrow} \overline{C}}\ \mathsf{seq}$$

Let $N' = \pi_E(\sigma N)$ and $M' = \pi_E(\sigma M)$. Given $R \in \textsc{run}(\overline{A})$, by induction we have $R, N' \Downarrow [p_i \cdot S_i]_{i \le n}$ with each $S_i \in \textsc{run}(\overline{B})$ and $\sum_{i \le n} p_i \ge p$. Again by induction, for each $i \le n$ we have $S_i, M' \Downarrow \mathcal{T}_i$ with $\mathcal{T}_i \in \mathcal{D}_q(\textsc{run}(\overline{C}))$. The definition of $\Downarrow$ gives $R, (N' ; M') \Downarrow \mathcal{T}$ for $\mathcal{T} = \sum_{i \le n} p_i \mathcal{T}_i$. Finally, observe that $\mathcal{T} \in \mathcal{D}_{pq}(\textsc{run}(\overline{C}))$ since $|\mathcal{T}| = \sum_{i \le n} p_i |\mathcal{T}_i| \ge pq$ and since each $\mathcal{T}_i$ is a distribution over $\textsc{run}(\overline{C})$.

Abstraction case:

$$\frac{E \mid \Gamma, x \colon A \vdash M \colon \overline{B} \overset{p}{\Rightarrow} \overline{C}}{E \mid \Gamma \vdash \langle x \rangle . M \colon A\,\overline{B} \overset{p}{\Rightarrow} \overline{C}}\,\textsf{abs}$$

Let $\sigma \in \textsc{run}(\Gamma)$ and $S\,N \in \textsc{run}(\overline{B}\,A)$. We need to demonstrate a distribution $\mathcal{T} \in \mathcal{D}_p(\textsc{run}(\overline{C}))$ such that $S\,N, \pi_E(\sigma(\langle x \rangle . M)) \Downarrow \mathcal{T}$. Let $M' = \pi_E(\sigma M)$ and note that since σ is not defined on x and σ and N are closed, $\pi_E(\sigma\{N/x\}M) = \{N/x\}M'$ and $\pi_E(\sigma(\langle x \rangle . M) = \langle x \rangle . M'$. The inductive hypothesis for the premise $E \mid \Gamma, x \colon A \vdash M \colon \overline{B} \overset{p}{\Rightarrow} \overline{C}$, with $\sigma\{N/x\} \in \textsc{run}(\Gamma, x \colon A)$, gives the desired $\mathcal{T} \in \mathcal{D}_p(\textsc{run}(\overline{C}))$ with evaluation $S, \{N/x\}M' \Downarrow \mathcal{T}$. Then by the definition of evaluation, $S\,N, M' \Downarrow \mathcal{T}$, and hence $\langle x \rangle . M' \in \textsc{run}(A\,\overline{B} \overset{p}{\Rightarrow} \overline{C})$.

Generator case:

$$\frac{E, a \mapsto \top \mid \Gamma \vdash M \colon \overline{A} \overset{p}{\Rightarrow} \overline{C} \qquad E, a \mapsto \bot \mid \Gamma \vdash M \colon \overline{A} \overset{q}{\Rightarrow} \overline{C}}{E \mid \Gamma \vdash \boxed{a} . M \colon \overline{A} \overset{\frac{1}{2}p + \frac{1}{2}q}{\Rightarrow} \overline{C}}\,\textsf{gen}$$

Let $\sigma \in \textsc{run}(\Gamma)$ and $S \in \textsc{run}(\overline{A})$. Let $M_i = \pi_{E, a \mapsto i}(\sigma M)$ for $i \in \top, \bot$. By the inductive hypothesis, $M_\top \in \textsc{run}(\overline{A} \overset{p}{\Rightarrow} \overline{C})$ and $M_\bot \in \textsc{run}(\overline{A} \overset{q}{\Rightarrow} \overline{C})$, which gives $S, M_\top \Downarrow \mathcal{T}_\top$ and $S, M_\bot \Downarrow \mathcal{T}_\bot$ for some $\mathcal{T}_\top \in \mathcal{D}_p(\textsc{run}(\overline{C}))$ and $\mathcal{T}_\bot \in \mathcal{D}_q(\textsc{run}(\overline{C}))$. Let $r = \frac{1}{2}(p + q)$; then $\mathcal{T} = \frac{1}{2}(\mathcal{T}_\top + \mathcal{T}_\bot) \in \mathcal{D}_r(\textsc{run}(\overline{C}))$ and by definition of evaluation, $S, \pi_E(\sigma(\boxed{a} . M)) \Downarrow \mathcal{T}$. We conclude that $\pi_E(\sigma(\boxed{a} . M)) \in \textsc{run}(\overline{A} \overset{r}{\Rightarrow} \overline{C})$. ◀

Our main theorem, the probability of machine termination, is a direct consequence.

▶ **Theorem 21 (restatement).** *For a typed, closed term $M \colon \overline{A} \overset{p}{\Rightarrow} \overline{C}$ and stack $S : \overline{A}$ there is a finite set of distinct successful runs*

$$p_i \frac{(S\ ,\ M\ ,\ \varepsilon)}{(T_i\ ,\ \star\ ,\ \varepsilon)} \qquad (i \le n)$$

with sum probability $\sum_{i \le n} p_i \ge p$.

Proof. By Lemma 25 we have $M \in \textsc{run}(\overline{A} \overset{p}{\Rightarrow} \overline{C})$ and $S \in \textsc{run}(\overline{A})$. By the definition of $\textsc{run}(-)$ we then have $S, M \Downarrow \mathcal{T}$ with $|\mathcal{T}| \ge p$. Proposition 17 then gives the desired set of machine runs. ◀

A.2 Probabilistic head normalization

Finally, we will relate machine termination to head reduction. For a redex in a head context $\lambda x_1 \ldots \lambda x_n . \{\}\, M_1 \ldots M_m$ the machine runs as follows: after the abstractions consume the top n elements off the stack, and the applications push the terms M_i onto it, then the redex itself is the first part of the term to be evaluated on the machine. Machine evaluation thus corresponds tightly to head reduction, with the same order of evaluation of redexes.

However, in this correspondence, the machine halts with a *variable*, while successful termination in our setting requires a *skip* with an empty continuation stack. We will thus use a different approach.

$$\frac{}{S, \star \Downarrow_0 [1 \cdot S]} \qquad \frac{S, \{N/x\}M \Downarrow_w \mathcal{T}}{S\,N, \langle x \rangle .\, M \Downarrow_{w+1} \mathcal{T}} \qquad \frac{R, M \Downarrow_w [p_i \cdot S_i]_{i \leq n} \qquad (\, S_i, N \Downarrow_{v_i} \mathcal{T}_i \,)_{i \leq n}}{R, M\,;N \Downarrow_{(w + \sum_{i \leq n} v_i)} \sum_{i \leq n} p_i \mathcal{T}_i}$$

$$\frac{}{S, M \Downarrow_0 \varnothing} \qquad \frac{S\,N, M \Downarrow_w \mathcal{T}}{S, [N] .\, M \Downarrow_{w+1} \mathcal{T}} \qquad \frac{S, \pi^a_\top(M) \Downarrow_v \mathcal{T}_\top \qquad S, \pi^a_\bot(M) \Downarrow_w \mathcal{T}_\bot}{S, \boxed{a} .\, M \Downarrow_{v+w+1} \frac{1}{2}\mathcal{T}_\top + \frac{1}{2}\mathcal{T}_\bot}$$

Figure 5 The weighted probabilistic evaluation relation.

The main idea is that reduction *shortens* the runs of the machine, by removing consecutive *push* and *pop* operations, or in the case of projective reduction, removing a generator. By annotating the evaluation relation to count abstractions, applications, and generators we may then observe that this measure reduces under head reduction.

▶ **Definition 26.** *The* weighted *evaluation relation* $S, M \Downarrow_w \mathcal{T}$ *is given in Figure 5, where the* weight w *is a natural number. We extend it to a distribution of terms* $\mathcal{M}$ *by the following rule, carrying a multiset of weights* $\mathcal{W}$*.*

$$\frac{(\, S, M_i \Downarrow_{w_i} \mathcal{T}_i \,)_{i \leq n}}{S, [p_i \cdot M_i]_{i \leq n} \Downarrow_{[w_i]_{i \leq n}} \sum_{i \leq n} p_i \mathcal{T}_i}$$

The core lemma establishes that the weight in the evaluation relation decreases for $\to_{\beta \mathsf{h}}$ and $\to_\pi$ reduction steps, and is stable under $\to_{\sigma \mathsf{h}}$. However, this does not apply to terms introduced by the zero-rule, as $S, M \Downarrow_0 \varnothing$, which are the potentially non-terminating terms. Crucially for our purpose, when the evaluation returns a non-empty distribution, head-reduction on the term progresses towards a head-normal form.

▶ **Lemma 27.** *Let* $S, M \Downarrow_w \mathcal{T}$ *where* $\mathcal{T} \neq \varnothing$*.*

- *If* $M \to_{\sigma \mathsf{h}} N$ *then* $S, N \Downarrow_w \mathcal{T}$*.*
- *If* $M \to_{\beta \mathsf{h}} N$ *then* $S, N \Downarrow_v \mathcal{T}$ *with* $v < w$*.*
- *If* $M \to_\pi \mathcal{N}$ *then* $S, \mathcal{N} \Downarrow_{\mathcal{W}} \mathcal{T}$ *with* $v < w$ *for every* v *in* $\mathcal{W}$*.*

Proof. We first consider the reduction steps themselves, and then their closure under head contexts. The proof is by induction on $\Downarrow$. We consider three cases; the remaining are similar.

Beta $[N] .\, \langle x \rangle .\, M \to_\beta \{N/x\}M$ Since $\mathcal{T}$ is non-empty, the derivation must be as below left, as none of the inferences may be replaced by a zero-rule. The case is then as follows.

$$\frac{\dfrac{S, \{N/x\}M \Downarrow_w \mathcal{T}}{S\,N, \langle x \rangle .\, M \Downarrow_{w+1} \mathcal{T}}}{S, [N] .\, \langle x \rangle .\, M \Downarrow_{w+2} \mathcal{T}} \quad \to_\beta \quad S, \{N/x\}M \Downarrow_w \mathcal{T}$$

Projection $\boxed{a} .\, M \to_\pi [\, \frac{1}{2} \cdot \pi^a_\top M \,,\, \frac{1}{2} \cdot \pi^a_\bot M \,]$ Since $\mathcal{T}$ is non-empty, the derivation for the redex is the first below. The derivation for the reduct, second below, uses the evaluation rule for distributions.

$$\frac{S, \pi^a_\top M \Downarrow_v \mathcal{T}_\top \qquad S, \pi^a_\bot M \Downarrow_w \mathcal{T}_\bot}{S, \boxed{a} .\, M \Downarrow_{v+w+1} \frac{1}{2}\mathcal{T}_\top + \frac{1}{2}\mathcal{T}_\bot}$$

$$\frac{S, \pi^a_\top M \Downarrow_v \mathcal{T}_\top \qquad S, \pi^a_\bot M \Downarrow_w \mathcal{T}_\bot}{S, [\frac{1}{2} \cdot \pi^a_\top M, \frac{1}{2} \cdot \pi^a_\bot M] \Downarrow_{[v,w]} \frac{1}{2}\mathcal{T}_\top + \frac{1}{2}\mathcal{T}_\bot}$$

Sequence–generator $\boxed{a}.\, M\,;P \;\to_\sigma\; \boxed{a}.\,(M\,;P)$ $(a \notin \mathsf{fe}(P))$ The derivations are as follows, with premises stacked for space. In the second derivation $w' = w + \sum_{i\le n} u_i$ and $v' = v + \sum_{n<i\le m} u_i$.

$$\dfrac{\dfrac{R, \pi^a_\top M \Downarrow_w [p_i \cdot S_i]_{i\le n} \quad R, \pi^a_\bot M \Downarrow_v [p_i \cdot S_i]_{n<i\le m}}{R, \boxed{a}.\, M \Downarrow_{w+v+1} \frac{1}{2}[p_i \cdot S_i]_{i\le m}} \qquad (S_i, P \Downarrow_{u_i} \mathcal{T}_i)_{i\le m}}{R, \boxed{a}.\, M\,;P \Downarrow_{(w+v+1+\sum_{i\le m} u_i)} \sum_{i\le m} \frac{1}{2} p_i \mathcal{T}_i}$$

$$\dfrac{\dfrac{R, \pi^a_\top M \Downarrow_w [p_i \cdot S_i]_{i\le n} \quad (S_i, P \Downarrow_{u_i} \mathcal{T}_i)_{i\le n}}{R, \pi^a_\top M\,;P \Downarrow_{w'} \sum_{i\le n} p_i \mathcal{T}_i} \qquad \dfrac{R, \pi^a_\bot M \Downarrow_v [p_i \cdot S_i]_{n<i\le m} \quad (S_i, P \Downarrow_{u_i} \mathcal{T}_i)_{n<i\le m}}{R, \pi^a_\bot M\,;P \Downarrow_{v'} \sum_{n<i\le m} p_i \mathcal{T}_i}}{R, \boxed{a}.\,(M\,;P) \Downarrow_{(w+v+1+\sum_{i\le m} u_i)} \sum_{i\le m} \frac{1}{2} p_i \mathcal{T}_i}$$

This concludes the reduction steps. The remaining cases consider the closure under head contexts. For sequencing reduction, which leaves the weight unchanged, this is immediate, and the cases for beta-steps are straightforward and omitted. For projective reduction, let $M \to_\pi [\, \frac{1}{2} \cdot N_\top \; \frac{1}{2} \cdot N_\bot \,]$, and consider the remaining cases.

Application $[P].\, M \to_\pi [\, \frac{1}{2} \cdot [P].\, N_\top \; \frac{1}{2} \cdot [P].\, N_\bot \,]$ The derivation for $[P].\, M$ is as follows.

$$\dfrac{S\, P, M \Downarrow_w \mathcal{T}}{S, [P].\, M \Downarrow_{w+1} \mathcal{T}}$$

By induction, for the premise of this derivation we get one for the distribution $\mathcal{N} = [\frac{1}{2} \cdot N_\top, \frac{1}{2} \cdot N_\bot]$, below, where it follows that $\mathcal{T} = \frac{1}{2}\mathcal{T}_\top + \frac{1}{2}\mathcal{T}_\bot$ and $u, v < w$.

$$\dfrac{S\, P, N_\top \Downarrow_u \mathcal{T}_\top \qquad S\, P, N_\bot \Downarrow_v \mathcal{T}_\bot}{S\, P, \mathcal{N} \Downarrow_{[u,v]} \mathcal{T}}$$

Then for $\mathcal{P} = [\, \frac{1}{2} \cdot [P].\, N_\top \; \frac{1}{2} \cdot [P].\, N_\bot \,]$ we get the following derivation.

$$\dfrac{\dfrac{S\, P, N_\top \Downarrow_u \mathcal{T}_\top}{S, [P].\, N_\top \Downarrow_{u+1} \mathcal{T}_\top} \qquad \dfrac{S\, P, N_\bot \Downarrow_v \mathcal{T}_\bot}{S, [P].\, N_\bot \Downarrow_{v+1} \mathcal{T}_\bot}}{S, \mathcal{P} \Downarrow_{[u+1,v+1]} \mathcal{T}}$$

Abstraction $\langle x \rangle.\, M \to_\pi [\, \frac{1}{2} \cdot \langle x \rangle.\, N_\top \; \frac{1}{2} \cdot \langle x \rangle.\, N_\bot \,]$ The derivation for $\langle x \rangle.\, M$ is as follows.

$$\dfrac{S, \{P/x\}M \Downarrow_w \mathcal{T}}{S\, P, \langle x \rangle.\, M \Downarrow_{w+1} \mathcal{T}}$$

Given the reduction for M, we also have the following, where we abbreviate the reduct as $\mathcal{P}$.

$$\{P/x\}M \to_\pi [\, \tfrac{1}{2} \cdot \{P/x\}N_\top \; \tfrac{1}{2} \cdot \{P/x\}N_\bot \,] = \mathcal{P}$$

By induction this gives us the following derivation for $\mathcal{P}$, where again $\mathcal{T} = \frac{1}{2}\mathcal{T}_\top + \frac{1}{2}\mathcal{T}_\bot$ and $u, v < w$.

$$\dfrac{S, \{P/x\}N_\top \Downarrow_u \mathcal{T}_\top \qquad S, \{P/x\}N_\bot \Downarrow_v \mathcal{T}_\bot}{S, \mathcal{P} \Downarrow_{[u,v]} \mathcal{T}}$$

Then for $\mathcal{N} = [\,\frac{1}{2}\cdot\langle x\rangle.\,N_\top\ \frac{1}{2}\cdot\langle x\rangle.\,N_\bot\,]$ we get the required derivation.

$$\dfrac{\dfrac{S, \{P/x\}N_\top \Downarrow_u \mathcal{T}_\top}{S\,P, \langle x\rangle.\,N_\top \Downarrow_{u+1} \mathcal{T}_\top} \qquad \dfrac{S, \{P/x\}N_\bot \Downarrow_v \mathcal{T}_\bot}{S\,P, \langle x\rangle.\,N_\bot \Downarrow_{v+1} \mathcal{T}_\bot}}{S, \mathcal{N} \Downarrow_{[u+1,v+1]} \mathcal{T}}$$ ◀

We apply the above lemma to relate machine evaluation to head reduction in the following way: if evaluation returns a distribution of weight p, then head-reduction terminates with probability at least p.

▶ **Lemma 28.** *If $S, \mathcal{M} \Downarrow_{\mathcal{W}} \mathcal{T}$ then $\mathcal{M} \twoheadrightarrow_{\mathsf{h}} \mathcal{N}_0 + \mathcal{N}_1$ where all terms in $\mathcal{N}_0$ are head normal and $|\mathcal{N}_0| \geq |\mathcal{T}|$.*

Proof. Let $\mathcal{M} = [p_i \cdot M_i]_{i\leq n}$ evaluate by the following derivation.

$$\dfrac{(\,S, M_i \Downarrow_{w_i} \mathcal{T}_i\,)_{i\leq n}}{S, [p_i\cdot M_i]_{i\leq n} \Downarrow_{[w_i]_{i\leq n}} \sum_{i\leq n} p_i\mathcal{T}_i}$$

First, we head-reduce any M_i where $\mathcal{T}_i \neq \varnothing$. This process terminates by induction on $\mathcal{W}$, using Lemma 27: a β-step or projective step reduces $\mathcal{W}$, while sequencing reduction alone is terminating and does not increase it.

Then the distribution $\mathcal{N}_0 = [\,p_i\cdot M_i \mid i\leq n,\ \mathcal{T}_i \neq \varnothing\,]$ is head-normal. The weights of $\mathcal{N}_0$ and $\mathcal{T}$ are as follows.

$$|\mathcal{N}_0| = \sum[\,p_i \mid i\leq n,\ \mathcal{T}_i \neq \varnothing\,] \qquad |\mathcal{T}| = \sum_{i\leq n} p_i|\mathcal{T}_i|$$

It follows that $|\mathcal{N}_0| \geq |\mathcal{T}|$. ◀

Our final theorem then ties everything together: typing gives successful evaluation on the machine, which in turn gives termination of head reduction.

▶ **Theorem 22 (restatement).** *For a closed M, if $M : \overline{A} \overset{p}{\Rightarrow} \overline{C}$ then $M \twoheadrightarrow_{\mathsf{h}} \mathcal{N}_0 + \mathcal{N}_1$ where all terms in $\mathcal{N}_0$ are head normal and $|\mathcal{N}_0| \geq p$.*

Proof. We provide M with an input stack consisting only of zero terms 0_A. Let S be the stack of zero terms for $\overline{A}$. By Proposition 20 zero-terms inhabit their types, so that $S : \overline{A}$.

By Lemma 25 we have $S \in \textsc{run}(\overline{A})$ and $M \in \textsc{run}(\overline{A} \overset{p}{\Rightarrow} \overline{C})$. By the definition of $\textsc{run}(-)$ this gives an evaluation $S, M \Downarrow \mathcal{T}$ where $|\mathcal{T}| \geq p$.

For the corresponding weighted derivation $S, M \Downarrow_w \mathcal{T}$, Lemma 28 gives a head reduction $M \twoheadrightarrow_{\mathsf{h}} \mathcal{N}_0 + \mathcal{N}_1$ where all terms in $\mathcal{N}_0$ are head normal and $|\mathcal{N}_0| \geq |\mathcal{T}| \geq p$. ◀