Probabilistic Termination by Monadic Affine Sized Typing

Ugo Dal Lago, Charles Grellois

Probabilistic Termination by Monadic Affine Sized Typing[⋆]

Ugo Dal Lago[1] and Charles Grellois[2]

[1] University of Bologna & INRIA Sophia Antipolis
ugo.dallago@unibo.it
[2] INRIA Sophia Antipolis
charles.grellois@inria.fr

Abstract. We introduce a system of monadic affine sized types, which substantially generalise usual sized types, and allows this way to capture probabilistic higher-order programs which terminate almost surely. Going beyond plain, strong normalisation without losing soundness turns out to be a hard task, which cannot be accomplished without a richer, quantitative notion of types, but also without imposing some affinity constraints. The proposed type system is powerful enough to type classic examples of probabilistically terminating programs such as random walks. The way typable programs are proved to be almost surely terminating is based on reducibility, but requires a substantial adaptation of the technique.

1 Introduction

Probabilistic models are more and more pervasive in computer science [1–3]. Moreover, the concept of algorithm, originally assuming determinism, has been relaxed so as to allow probabilistic evolution since the very early days of theoretical computer science [4]. All this has given impetus to research on probabilistic programming languages, which however have been studied at a large scale only in the last twenty years, following advances in randomized computation [5], cryptographic protocol verification [6, 7], and machine learning [8]. Probabilistic programs can be seen as ordinary programs in which specific instructions are provided to make the program evolve probabilistically rather than deterministically. The typical example are instructions for sampling from a given distribution toolset, or for performing probabilistic choice.

One of the most crucial properties a program should satisfy is *termination*: the execution process should be guaranteed to end. In (non)deterministic computation, this is easy to formalize, since any possible computation path is only considered qualitatively, and termination is a boolean predicate on programs: any non-deterministic program either terminates – in must or may sense – or it

[⋆] This work is partially supported by the ANR projects 12IS02001 PACE and 14CE250005 ELICA.

does not. In probabilistic programs, on the other hand, any terminating computation path is attributed a probability, and thus termination becomes a *quantitative* property. It is therefore natural to consider a program terminating when its terminating paths form a set of measure one or, equivalently, when it terminates with maximal probability. This is dubbed "almost sure termination" (AST for short) in the literature [9], and many techniques for automatically and semi-automatically checking programs for AST have been introduced in the last years [10–13]. All of them, however, focus on imperative programs; while probabilistic functional programming languages are nowadays among the most successful ones in the realm of probabilistic programming [8]. It is not clear at all whether the existing techniques for imperative languages could be easily applied to functional ones, especially when higher-order functions are involved.

In this paper, we introduce a system of monadic affine sized types for a simple probabilistic λ-calculus with recursion and show that it guarantees the AST property for all typable programs. The type system, described in Section 4, can be seen as a non-trivial variation on Hughes et al.'s sized types [14], whose main novelties are the following:

- Types are generalised so as to be *monadic*, this way encapsulating the kind of information we need to type non-trivial examples. This information, in particular, is taken advantage of when typing recursive programs.
- Typing rules are *affine*: higher-order variables cannot be freely duplicated. This is quite similar to what happens when characterising polynomial time functions by restricting higher-order languages akin to the λ-calculus [15]. Without affinity, the type system is bound to be unsound for AST.

The necessity of both these variations is discussed in Section 2 below. The main result of this paper is that typability in monadic affine sized types entails AST, a property which is proved using an adaptation of the Girard-Tait reducibility technique [16]. This adaptation is technically involved, as it needs substantial modifications allowing to deal with possibly infinite and probabilistic computations. In particular, every reducibility set must be parametrized by a quantitative parameter p guaranteeing that terms belonging to this set terminate with probability at least p. The idea of parametrizing such sets already appears in work by the first author and Hofmann [17], in which a notion of realizability parametrized by resource monoids is considered. These realizability models are however studied in relation to linear logic and to the complexity of normalisation, and do not fit as such to our setting, even if they provided some crucial inspiration. In our approach, the fact that recursively-defined terms are AST comes from a continuity argument on this parameter: we can prove, by unfolding such terms, that they terminate with probability p for every $p < 1$, and continuity then allows to take the limit and deduce that they are AST. This soundness result is technically speaking the main contribution of this paper, and is described in Section 5.

An extended version with more details and proofs is available [18].

1.1 Related Works

Sized types have been originally introduced by Hughes, Pareto, and Sabry [14] in the context of reactive programming. A series of papers by Barthe and colleagues [19–21] presents sized types in a way similar to the one we will adopt here, although still for a deterministic functional language. Contrary to the other works on sized types, their type system is proved to admit a decidable type inference, see the unpublished tutorial [20]. Abel developed independently of Barthe and colleagues a similar type system featuring size informations [22]. These three lines of work allow polymorphism, arbitrary inductive data constructors, and ordinal sizes, so that data such as infinite trees can be manipulated. These three features will be absent of our system, in order to focus the challenge on the treatment of probabilistic recursive programs. Another interesting approach is the one of Xi's Dependent ML [23], in which a system of lightweight dependent types allows a more liberal treatment of the notion of size, over which arithmetic or conditional operations may in particular be applied. Termination is ensured by checking during typing that a given metrics decreases during recursive calls. This system is well-adapted for practical termination checking and can be extended with mutual recursion, inductive types and polymorphism, but does not feature ordinal sizes. See [22] for a detailed comparison of the previously cited systems. Some works along these lines are able to deal with coinductive data, as well as inductive ones [14, 19, 22]. They are related to Amadio and Coupet-Grimal's work on guarded types ensuring productivity of infinite structures such as streams [24]. None of these works deal with probabilistic computation, and in particular with almost sure termination.

There has been a lot of interest, recently, about probabilistic termination as a verification problem in the context of imperative programming [10–13]. All of them deal, invariably, with some form of while-style language without higher-order functions. A possible approach is to reduce AST for probabilistic programs to termination of non-deterministic programs [10]. Another one is to extend the concept of ranking function to the probabilistic case. Bournez and Garnier obtained in this way the notion of Lyapunov ranking function [25], but such functions capture a notion more restrictive than AST: *positive* almost sure termination, meaning that the program is AST and terminates in expected finite time. To capture AST, the notion of ranking supermartingale [26] has been used. Note that the use of ranking supermartingales allows to deal with programs which are both probabilistic and non-deterministic [11, 13] and even to reason about programs with real-valued variables [12].

Some recent work by Cappai, the first author, and Parisen Toldin [27, 28] introduce type systems ensuring that all typable programs can be evaluated in probabilistic polynomial time. This is too restrictive for our purposes. On the one hand, we aim at termination, and restricting to polynomial time algorithms would be an overkill. On the other hand, the above-mentioned type systems guarantee that the length of *all* probabilistic branches are uniformly bounded (by the same polynomial). In our setting, this would restrict the focus to terms in which infinite computations are forbidden, while we simply want the set of such computations to have probability 0. In fact, the results we present in this paper can be seen as a first step towards a type system characterizing average polynomial time, in the style of implicit computational complexity [29].

2 Why is Monadic Affine Typing Necessary?

In this section, we justify the design choices that guided us in the development of our type system. As we will see, the nature of AST requires a significant and non-trivial extension of the system of sized types originally introduced to ensure termination in the deterministic case [14].

Sized Types for Deterministic Programs. The simply-typed λ-calculus endowed with a typed recursion operator letrec and appropriate constructs for the natural numbers, sometimes called PCF, is already Turing-complete, so that there is no hope to prove it strongly normalizing. Sized types [14] refine the simple type system by enriching base types with annotations, so as to ensure the termination of any recursive definition. Let us explain the idea of sizes in the simple, yet informative case in which the base type is Nat. Sizes are defined by the grammar

$$\mathfrak{s} \quad ::= \quad \mathfrak{i} \;\big|\; \infty \;\big|\; \widehat{\mathfrak{s}}$$

where $\mathfrak{i}$ is a size variable and $\widehat{\mathfrak{s}}$ is the successor of the size $\mathfrak{s}$ — with $\widehat{\infty} = \infty$. These sizes permit to consider decorations $\mathsf{Nat}^{\mathfrak{s}}$ of the base type Nat, whose elements are natural numbers of size at most $\mathfrak{s}$. The type system ensures that the only constant value of type $\mathsf{Nat}^{\widehat{\mathfrak{i}}}$ is $\mathsf{0}$, that the only constant values of type $\mathsf{Nat}^{\widehat{\widehat{\mathfrak{i}}}}$ are $\mathsf{0}$ or $\underline{1} = \mathsf{S}\ \mathsf{0}$, and so on. The type Nat^{∞} is the one of all natural numbers, and is therefore often denoted as Nat.

The crucial rule of the sized type system, which we present here following Barthe et al. [19], allows one to type recursive definitions as follows:

$$\frac{\Gamma, f : \mathsf{Nat}^{\mathfrak{i}} \to \sigma \vdash M : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \sigma[\widehat{\mathfrak{i}}/\mathfrak{i}] \qquad \mathfrak{i} \text{ pos } \sigma}{\Gamma \vdash \mathsf{letrec}\ f = M : \mathsf{Nat}^{\mathfrak{s}} \to \sigma[\mathfrak{s}/\mathfrak{i}]} \tag{1}$$

This typing rule ensures that, to recursively define the function $f = M$, the term M taking an input of size $\widehat{\mathfrak{i}}$ calls f on inputs of *strictly lesser* size $\mathfrak{i}$. This is for instance the case when typing the program

$$M_{DBL} \quad = \quad \mathsf{letrec}\ f = \lambda x.\mathsf{case}\ x\ \mathsf{of}\ \left\{ \mathsf{S} \to \lambda y.\mathsf{S}\ \mathsf{S}\ (f\ y) \;\middle|\; \mathsf{0} \to \mathsf{0} \right\}$$

computing recursively the double of an input integer, as the hypothesis of the fixpoint rule in a typing derivation of M_{DBL} is

$$f : \mathsf{Nat}^{\mathfrak{i}} \to \mathsf{Nat} \vdash \lambda x.\mathsf{case}\ x\ \mathsf{of}\ \left\{ \mathsf{S} \to \lambda y.\mathsf{S}\ \mathsf{S}\ (f\ y) \;\middle|\; \mathsf{0} \to \mathsf{0} \right\} : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \mathsf{Nat}$$

The fact that f is called on an input y of strictly lesser size $\mathfrak{i}$ is ensured by the rule typing the case construction:

$$\frac{\Gamma \vdash x : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \qquad \Gamma \vdash \lambda y.\mathsf{S}\ \mathsf{S}\ (f\ y) : \mathsf{Nat}^{\mathfrak{i}} \to \mathsf{Nat} \qquad \Gamma \vdash \mathsf{0} : \mathsf{Nat}}{\Gamma \vdash \mathsf{case}\ x\ \mathsf{of}\ \left\{ \mathsf{S} \to \lambda y.\mathsf{S}\ \mathsf{S}\ (f\ y) \;\middle|\; \mathsf{0} \to \mathsf{0} \right\} : \mathsf{Nat}}$$

where $\Gamma = f : \mathsf{Nat}^{\mathfrak{i}} \to \mathsf{Nat},\ x : \mathsf{Nat}^{\widehat{\mathfrak{i}}}$. The soundness of sized types for strong normalization allows to conclude that M_{DBL} is indeed SN.

A Naïve Generalization to Probabilistic Terms. The aim of this paper is to obtain a probabilistic, *quantitative* counterpart to this soundness result for sized types. Note that unlike the result for sized types, which was focusing on *all* reduction strategies of terms, we only consider a *call-by-value* calculus[3]. Terms can now contain a probabilistic choice operator $\oplus_p$, such that $M \oplus_p N$ reduces to the term M with probability $p \in \mathbb{R}_{[0,1]}$, and to N with probability $1 - p$. The language and its operational semantics will be defined more extensively in Section 3. Suppose for the moment that we type the choice operator in a naïve way:

$$\text{Choice} \quad \frac{\Gamma \vdash M : \sigma \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M \oplus_p N : \sigma}$$

On the one hand, the original system of sized types features subtyping, which allows some flexibility to "unify" the types of M and N to σ. On the other hand, it is easy to realise that *all* probabilistic branches would have to be terminating, without any hope of capturing interesting AST programs: nothing has been done to capture the *quantitative* nature of probabilistic termination. An instance of a term which is not strongly normalizing but which is almost-surely terminating — meaning that it normalizes with probability 1 — is

$$M_{BIAS} = \Big(\mathsf{letrec}\ f = \lambda x.\mathsf{case}\ x\ \mathsf{of}\ \Big\{ \mathsf{S} \to \lambda y.f(y) \oplus_{\frac{2}{3}} (f(\mathsf{S}\,\mathsf{S}\,y))) \ \mid\ 0 \to 0 \Big\}\Big)\ \underline{n} \qquad (2)$$

simulating a *biased random walk* which, on $x = m+1$, goes to m with probability $\frac{2}{3}$ and to $m+2$ with probability $\frac{1}{3}$. The naïve generalization of the sized type system only allows us to type the body of the recursive definition as follows:

$$f : \mathsf{Nat}^{\widehat{\widehat{\mathfrak{i}}}} \to \mathsf{Nat}^{\infty} \vdash \lambda y.f(y) \oplus_{\frac{2}{3}} (f(\mathsf{S}\,\mathsf{S}\,y))) : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \mathsf{Nat}^{\infty} \qquad (3)$$

and thus does not allow us to deduce any relevant information on the *quantitative* termination of this term: nothing tells us that the recursive call $f(\mathsf{S}\,\mathsf{S}\,y)$ is performed with a relatively low probability.

A Monadic Type System. Along the evaluation of M_{BIAS}, there is *indeed* a quantity which decreases during each recursive call to the function f: the *average* size of the input on which the call is performed. Indeed, on an input of size $\widehat{\mathfrak{i}}$, f calls itself on an input of smaller size $\mathfrak{i}$ with probability $\frac{2}{3}$, and on an input of greater size $\widehat{\widehat{\mathfrak{i}}}$ with probability only $\frac{1}{3}$. To capture such a relevant *quantitative* information on the recursive calls of f, and with the aim to capture almost sure termination, we introduce a *monadic* type system, in which *distributions of types* can be used to type in a finer way the functions to be used recursively. Contexts $\Gamma \mid \Theta$ will be generated by a context Γ attributing sized types to any number

[3] Please notice that choosing a reduction strategy is crucial in a probabilistic setting, otherwise one risks getting nasty forms of non-confluence [30].

of variables, while Θ will attribute a *distribution* of sized types to at most *one* variable — typically the one we want to use to recursively define a function. In such a context, terms will be typed by a distribution type, formed by combining the Dirac distributions of types introduced in the Axiom rules using the following rule for probabilistic choice:

$$\text{Choice}\quad \frac{\Gamma \mid \Theta \vdash M : \mu \qquad \Gamma \mid \Psi \vdash N : \nu \qquad \langle \mu \rangle = \langle \nu \rangle}{\Gamma \mid \Theta \oplus_p \Psi \vdash M \oplus_p N : \mu \oplus_p \nu}$$

The guard condition $\langle \mu \rangle = \langle \nu \rangle$ ensures that μ and ν are distributions of types decorating of the *same* simple type. Without this condition, there is no hope to aim for a decidable type inference algorithm.

The Fixpoint Rule. Using these monadic types, instead of the insufficiently informative typing (3), we can derive the sequent

$$f : \left\{ \left(\mathsf{Nat}^{\mathfrak{i}} \to \mathsf{Nat}^{\infty}\right)^{\frac{2}{3}}, \left(\mathsf{Nat}^{\widehat{\widehat{\mathfrak{i}}}} \to \mathsf{Nat}^{\infty}\right)^{\frac{1}{3}} \right\} \vdash \lambda y. f(y) \oplus_{\frac{2}{3}} (f(\mathsf{S}\,\mathsf{S}\,y))) \;:\; \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \mathsf{Nat}^{\infty} \tag{4}$$

in which the type of f contains finer information on the sizes of arguments over which it is called recursively, and with which probability. This information enables us to perform a first switch from a qualitative to a quantitative notion of termination: we will adapt the hypothesis

$$\Gamma, f : \mathsf{Nat}^{\mathfrak{i}} \to \sigma \vdash M : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \sigma[\widehat{\mathfrak{i}}/\mathfrak{i}] \tag{5}$$

of the original fix rule (1) of sized types, expressing that f is called on an argument of size one less than the one on which M is called, to a condition meaning that there is probability 1 to call f on arguments of a lesser size *after enough iterations of recursive calls.* We therefore define a random walk associated to the distribution type μ of f, the *sized walk* associated to μ, and which is as follows for the typing (4):

- the random walk starts on 1, corresponding to the size $\widehat{\mathfrak{i}}$,
- on an integer $n+1$, the random walk jumps to n with probability $\frac{2}{3}$ and to $n+2$ with probability $\frac{1}{3}$,
- 0 is stationary: on it, the random walk loops.

This random walk – as all sized walks will be – is an instance of *one-counter Markov decision problem* [31], so that it is decidable in polynomial time whether the walk reaches 0 with probability 1. We will therefore replace the hypothesis (5) of the letrec rule by the quantitative counterpart we just sketched, obtaining

$$\mathsf{letrec}\quad \frac{\begin{array}{c}\left\{ (\mathsf{Nat}^{\mathfrak{s}_j} \to \nu[\mathfrak{s}_j/\mathfrak{i}])^{p_j} \;\middle|\; j \in \mathcal{J} \right\} \text{induces an AST sized walk} \\ \Gamma \mid f : \left\{ (\mathsf{Nat}^{\mathfrak{s}_j} \to \nu[\mathfrak{s}_j/\mathfrak{i}])^{p_j} \;\middle|\; j \in \mathcal{J} \right\} \vdash V : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \nu[\widehat{\mathfrak{i}}/\mathfrak{i}]\end{array}}{\Gamma, \Delta \mid \Theta \vdash \mathsf{letrec}\ f = V : \mathsf{Nat}^{\mathfrak{r}} \to \nu[\mathfrak{r}/\mathfrak{i}]}$$

where we omit two additional technical conditions to be found in Section 4 and which justify the weakening on contexts incorporated to this rule. The resulting type system allows to type a varieties of examples, among which the following program computing the geometric distribution over the natural numbers:

$$M_{EXP} \;=\; \Big(\mathsf{letrec}\ f \;=\; \lambda x.x \oplus_{\frac{1}{2}} \mathsf{S}\ \ (f\ x)\Big)\ \mathsf{0} \tag{6}$$

and for which the decreasing quantity is the size of the set of probabilistic branches of the term making recursive calls to f. Another example is the unbiased random walk

$$M_{UNB} \;=\; \Big(\mathsf{letrec}\ f \;=\; \lambda x.\mathsf{case}\ x\ \mathsf{of}\ \Big\{\mathsf{S} \to \lambda y.f(y) \oplus_{\frac{1}{2}} (f(\mathsf{S}\,\mathsf{S}\,y))) \ \mid\ \mathsf{0} \to \mathsf{0}\Big\}\Big)\ \underline{n} \tag{7}$$

for which there is no clear notion of decreasing measure during recursive calls, but which yet terminates almost surely, as witnessed by the sized walk associated to an appropriate derivation in the sized type system. We therefore claim that the use of this external guard condition on associated sized walks, allowing us to give a general condition of termination, is satisfying as it both captures an interesting class of examples, and is computable in polynomial time.

In Section 5, we prove that this shift from a qualitative to a quantitative hypothesis in the type system results in a shift from the soundness for strong normalization of the original sized type system to a soundness for its quantitative counterpart: *almost-sure termination.*

Why Affinity? To ensure the soundness of the letrec rule, we need one more structural restriction on the type system. For the sized walk argument to be adequate, we must ensure that the recursive calls of f are indeed precisely modelled by the sized walk, and this is not the case when considering for instance the following term:

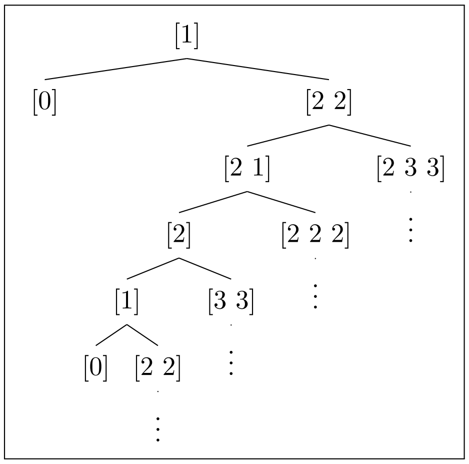

Fig. 1. A Tree of Recursive Calls.

$$M_{NAFF} \;=\; \Big(\mathsf{letrec}\ f \;=\; \lambda x.\mathsf{case}\ x\ \mathsf{of}\ \Big\{\mathsf{S} \to \lambda y.f(y) \oplus_{\frac{2}{3}} (f(\mathsf{S}\,\mathsf{S}\,y)\,;\, f(\mathsf{S}\,\mathsf{S}\,y)) \ \mid\ \mathsf{0} \to \mathsf{0}\Big\}\Big)\ \underline{n} \tag{8}$$

where the sequential composition ; is defined in this call-by-value calculus as $M\,;N \;=\; (\lambda x.\lambda y.\mathsf{0})\ M\ N$. Note that M_{NAFF} calls recursively f *twice* in the right branch of its probabilistic choice, and is not therefore modelled appropriately by the sized walk associated to its type. In fact, we would need a generalized notion of random walk to model the recursive calls of this process; it would be a random walk on *stacks* of integers. In the case where $n = 1$, the recursive calls to f can indeed be represented by a tree of stacks as depicted in Figure 1, where leftmost edges have probability $\frac{2}{3}$ and rightmost ones $\frac{1}{3}$. The root indicates that the first call on f was on the integer 1. From it, there is either a call of f on 0 which terminates, or *two* calls on 2 which are put into a stack of calls, and so on. We could prove that, without the *affine* restriction we are about to formulate, the term M_{NAFF} is typable with monadic sized types and the fixpoint rule we just designed. However, this term is not almost-surely terminating. Notice, indeed, that the *sum* of the integers appearing in a stack labelling a node of the tree in Figure 1 decreases by 1 when the left edge of probability $\frac{2}{3}$ is taken, and increases by *at least* 3 when the right edge of probability $\frac{1}{3}$ is taken. It follows that the expected increase of the sum of the elements of the stack during one step is at least $-1 \times \frac{2}{3} + 3 \times \frac{1}{3} = \frac{1}{3} > 0$. This implies that the probability that f is called on an input of size 0 after enough iterations is strictly less than 1, so that the term M_{NAFF} cannot be almost surely terminating.

Such general random processes have stacks as states and are rather complex to analyse. To the best of the authors' knowledge, they do not seem to have been considered in the literature. We also believe that the complexity of determining whether 0 can be reached almost surely in such a process, if decidable, would be very high. This leads us to the design of an *affine* type system, in which the management of contexts ensures that a given probabilistic branch of a term may only use at most once a given higher-order symbol. We do not however formulate restrictions on variables of simple type Nat, as affinity is only used on the letrec rule and thus on higher-order symbols. Remark that this is in the spirit of certain systems from implicit computational complexity [15, 32].

Another restriction imposed by this reduction of almost-sure termination checking for higher-order programs to almost-sure termination checking for one-counter Markov decision processes is the fact that we do not allow a general form of nested recursion. This restriction is encoded in the system by allowing *at most* one variable to have a distribution of types in the context. It follows that programs making use of mutual recursion can not be typed in this system.

3 A Simple Probabilistic Functional Programming Language

We consider the language $\lambda_\oplus$, which is an extension of the λ-calculus with recursion, constructors for the natural numbers, and a choice operator. In this section, we introduce this language and its operational semantics, and use them to define the crucial notion of *almost-sure termination.*

Terms and Values. Given a set of variables $\mathcal{X}$, terms and values of the language $\lambda_\oplus$ are defined by mutual induction as follows:

$$\begin{array}{llll}\text{Terms:} & M, N, \ldots & ::= & V \;\big|\; V\,W \;\big|\; \mathsf{let}\ x = M\ \mathsf{in}\ N \;\big|\; M \oplus_p N \\ & & & \big|\; \mathsf{case}\ V\ \mathsf{of}\ \{\mathsf{S} \to W \mid \mathsf{0} \to Z\}\end{array}$$

$$\begin{array}{llll}\text{Values:} & V, W, Z, \ldots & ::= & x \;\big|\; \mathsf{0} \;\big|\; \mathsf{S}\,V \;\big|\; \lambda x.M \;\big|\; \mathsf{letrec}\ f = V\end{array}$$

where $x, f \in \mathcal{X}$, $p \in]0,1[$. When $p = \frac{1}{2}$, we often write $\oplus$ as a shorthand for $\oplus_{\frac{1}{2}}$. The set of terms is denoted $\Lambda_\oplus$ and the set of values is denoted $\Lambda_\oplus^V$.

Terms of the calculus are assumed to be in A-normal form [33]. This allows to formulate crucial definitions in a simpler way, concentrating in the Let construct the study of the probabilistic behaviour of terms. We claim that all traditional constructions can be encoded in this formalism. For instance, the usual application $M\ N$ of two terms can be harmlessly recovered via the encoding $\mathsf{let}\ x = M\ \mathsf{in}\ (\mathsf{let}\ y = N\ \mathsf{in}\ x\ y)$. In the sequel, we write $c\ \overrightarrow{V}$ when a value may be either $\mathsf{0}$ or of the shape $\mathsf{S}\ V$.

Term Distributions. The introduction of a probabilistic choice operator in the syntax leads to a *probabilistic* reduction relation. It is therefore meaningful to consider the (operational) semantics of a term as a *distribution* of values modelling the outcome of *all* the finite probabilistic reduction paths of the term. For instance, the term M_{EXP} defined in (6) evaluates to the term distribution assigning probability $\frac{1}{2^{n+1}}$ to the value $\underline{n}$. Let us define this notion more formally:

Definition 1 (Distribution). *A* distribution *on X is a function $\mathscr{D} : X \to [0,1]$ satisfying the constraint $\sum \mathscr{D} = \sum_{x \in X} \mathscr{D}(x) \leq 1$, where $\sum \mathscr{D}$ is called the* sum *of the distribution $\mathscr{D}$. We say that $\mathscr{D}$ is* proper *precisely when $\sum \mathscr{D} = 1$. We denote by $\mathcal{P}$ the set of all distributions, would they be proper or not. We define the* support $\mathsf{S}(\mathscr{D})$ *of a distribution $\mathscr{D}$ as: $\mathsf{S}(\mathscr{D}) = \left\{ x \in X \;\middle|\; \mathscr{D}(x) > 0 \right\}$. When $\mathsf{S}(\mathscr{D})$ consists only of closed terms, we say that $\mathscr{D}$ is a* closed *distribution. When it is finite, we say that $\mathscr{D}$ is a* finite *distribution. We call* Dirac *a proper distribution $\mathscr{D}$ such that $\mathsf{S}(\mathscr{D})$ is a singleton. We denote by 0 the null distribution, mapping every term to the probability 0.*

When $X = \Lambda_{\oplus}$, we say that $\mathscr{D}$ is a *term distribution.* In the sequel, we will use a more practical notion of *representation* of distributions, which enumerates the terms with their probabilities as a family of assignments. For technical reasons, notably related to the subject reduction property, we will also need *pseudo-representations*, which are essentially multiset-like decompositions of the representation of a distribution.

Definition 2 (Representations and Pseudo-Representations). *Let $\mathscr{D} \in \mathcal{P}$ be of support $\left\{ x_i \;\middle|\; i \in \mathcal{I} \right\}$, where $x_i = x_j$ implies $i = j$ for every $i, j \in \mathcal{I}$. The* representation *of $\mathscr{D}$ is the set $\mathscr{D} = \left\{ x_i^{\mathscr{D}(x_i)} \;\middle|\; i \in \mathcal{I} \right\}$ where $x_i^{\mathscr{D}(x_i)}$ is just an intuitive way to write the pair $(x_i, \mathscr{D}(x_i))$. A* pseudo-representation *of $\mathscr{D}$ is any multiset $\left[y_j^{p_j} \;\middle|\; j \in \mathcal{J} \right]$ such that*

$$\forall j \in \mathcal{J}, \quad y_j \in \mathsf{S}(\mathscr{D}) \qquad\qquad \forall i \in \mathcal{I}, \quad \mathscr{D}(x_i) = \sum_{y_j = x_i} p_j.$$

By abuse of notation, we will simply write $\mathscr{D} = \left[y_j^{p_j} \;\middle|\; j \in \mathcal{J} \right]$ to mean that $\mathscr{D}$ admits $\left[y_j^{p_j} \;\middle|\; j \in \mathcal{J} \right]$ as pseudo-representation. Any distribution has a canonical pseudo-representation obtained by simply replacing the set-theoretic notation with the multiset-theoretic one.

Distributions support operations like affine combinations and sums – the latter being only a partial operation. We extend these operations to (pseudo)-representations, in a natural way. Distributions, endowed with the pointwise partial-order $\preccurlyeq$, form an ω-CPO, but not a lattice, since the join of two distributions is not guaranteed to exist.

Definition 3 (Value Decomposition of a Term Distribution). *Let $\mathscr{D}$ be a term distribution. We write its* value decomposition *as $\mathscr{D} \overset{VD}{=} \mathscr{D}_{|V} + \mathscr{D}_{|T}$, where $\mathscr{D}_{|V}$ is the maximal subdistribution of $\mathscr{D}$ whose support consists of values, and $\mathscr{D}_{|T} = \mathscr{D} - \mathscr{D}_{|V}$ is the subdistribution of "non-values" contained in $\mathscr{D}$.*

Operational Semantics. The semantics of a term will be the value distribution to which it reduces via the probabilistic reduction relation, iterated up to the limit. As a first step, we define the call-by-value reduction relation $\to_v \subseteq \mathcal{P} \times \mathcal{P}$ on Figure 2. Note that we write Dirac distributions simply as terms on the left side of $\to_v$, to improve readability. As usual, we denote by $\to_v^n$ the n-th iterate of the relation $\to_v$, with $\to_v^0$ being the identity relation. We then define the relation $\Rrightarrow_v^n$ as follows. Let $\mathscr{D} \to_v^n \mathscr{E} \overset{VD}{=} \mathscr{E}_{|V} + \mathscr{E}_{|T}$. Then $\mathscr{D} \Rrightarrow_v^n \mathscr{E}_{|V}$. Note that, for every $n \in \mathbb{N}$ and $\mathscr{D} \in \mathcal{P}$, there is a unique distribution $\mathscr{E}$ such that $\mathscr{D} \to_v^n \mathscr{E}$. Moreover, $\mathscr{E}_{|V}$ is the only distribution such that $\mathscr{D} \Rrightarrow_v^n \mathscr{E}_{|V}$.

Lemma 1. *Let $n, m \in \mathbb{N}$ with $n < m$. Let $\mathscr{D}_n$ (resp $\mathscr{D}_m$) be the distribution such that $M \Rrightarrow_v^n \mathscr{D}_n$ (resp $M \Rrightarrow_v^m \mathscr{D}_m$). Then $\mathscr{D}_n \preccurlyeq \mathscr{D}_m$.*

Definition 4 (Semantics of a Term, of a Distribution). *The semantics of a distribution $\mathscr{D}$ is the distribution $[\![\, \mathscr{D} \,]\!] = \sup_{n\in\mathbb{N}} \left(\left\{ \mathscr{D}_n \;\middle|\; \mathscr{D} \Rrightarrow_v^n \mathscr{D}_n \right\}\right)$. This supremum exists thanks to Lemma 1, combined with the fact that $(\mathcal{P}, \preccurlyeq)$ is an ω-CPO. We define the semantics of a term M as $[\![\, M \,]\!] = [\![\, \{ M^1 \} \,]\!]$.*

We now have all the ingredients required to define the central concept of this paper, the one of almost-surely terminating term:

Definition 5 (Almost-Sure Termination). *We say that a term M is* almost-surely terminating *precisely when $\sum [\![\, M \,]\!] = 1$.*

4 Monadic Affine Sized Typing

Following the discussion from Section 2, we introduce in this section a non-trivial lifting of sized types to our probabilistic setting. As a first step, we design an *affine* simple type system for $\lambda_\oplus$. This means that no higher-order variable may be used more than once in the same probabilistic branch. However, variables of base type Nat may be used freely. In spite of this restriction, the resulting system allows to type terms corresponding to any probabilistic Turing machine. In Section 4.2, we introduce a more sophisticated type system, which will be *monadic* and affine, and which will be sound for almost-sure termination as we prove in Section 5.

$$\frac{}{\mathsf{let}\ x\ =\ V\ \mathsf{in}\ M \quad \to_v \quad \left\{ (M[V/x])^1 \right\}} \qquad \frac{}{(\lambda x.M)\ V \quad \to_v \quad \left\{ (M[V/x])^1 \right\}}$$

$$\frac{}{M\ \oplus_p\ N \quad \to_v \quad \left\{ M^p,\, N^{1-p} \right\}}$$

$$\frac{M \quad \to_v \quad \left\{ L_i^{p_i} \ \middle|\ i \in \mathcal{I} \right\}}{\mathsf{let}\ x\ =\ M\ \mathsf{in}\ N \quad \to_v \quad \left\{ (\mathsf{let}\ x\ =\ L_i\ \mathsf{in}\ N)^{p_i} \ \middle|\ i \in \mathcal{I} \right\}}$$

$$\frac{}{\mathsf{case}\ \mathsf{S}\ V\ \mathsf{of}\ \left\{ \mathsf{S} \to W \ \middle|\ \mathsf{0} \to Z \right\} \quad \to_v \quad \left\{ (W\ V)^1 \right\}}$$

$$\frac{}{\mathsf{case}\ \mathsf{0}\ \mathsf{of}\ \left\{ \mathsf{S} \to W \ \middle|\ \mathsf{0} \to Z \right\} \quad \to_v \quad \left\{ (Z)^1 \right\}}$$

$$\frac{}{(\mathsf{letrec}\ f\ =\ V)\ \left(c\ \overrightarrow{W}\right) \quad \to_v \quad \left\{ \left(V[(\mathsf{letrec}\ f\ =\ V)/f]\ \left(c\ \overrightarrow{W}\right)\right)^1 \right\}}$$

$$\frac{\mathscr{D} \ \overset{VD}{=}\ \left\{ M_j^{p_j} \ \middle|\ j \in \mathcal{J} \right\} + \mathscr{D}_{|V} \qquad \forall j \in \mathcal{J},\ \ M_j \quad \to_v \quad \mathscr{E}_j}{\mathscr{D} \quad \to_v \quad \left(\sum_{j \in \mathcal{J}} p_j \cdot \mathscr{E}_j \right) + \mathscr{D}_{|V}}$$

Fig. 2. Call-by-value reduction relation $\to_v$ on distributions.

4.1 Affine Simple Types for $\boldsymbol{\lambda_\oplus}$

The terms of the language $\lambda_\oplus$ can be typed using a variant of the simple types of the λ-calculus, extended to type letrec and $\oplus_p$, but also restricted to an *affine* management of contexts. Recall that the constraint of affinity ensures that a given higher-order symbol is used at most *once* in a probabilistic branch. We define simple types over the base type Nat in the usual way: $\kappa, \kappa', \ldots \ ::= \mathsf{Nat} \ \big|\ \kappa \to \kappa'$ where, by convention, the arrow associates to the right. Contexts Γ, Δ, ... are sequences of simply-typed variables $x\ ::\ \kappa$. We write sequents as $\Gamma \vdash M\ ::\ \kappa$ to distinguish these sequents from the ones using distribution types appearing later in this section. Before giving the rules of the type system, we need to define two policies for contracting contexts: an affine and a general one.

Context Contraction. Contexts can be combined in two different ways. On the one hand, one can form the *non-affine contraction* $\Gamma \cup \Delta$ of two contexts, for which Γ and Δ are allowed to share some variables, but these variables must be attributed the *same type* in both contexts. On the other hand, one can form the *affine contraction* $\Gamma \uplus \Delta$, in which variables in common between Γ and Δ must be attributed the type Nat.

$$\text{Var}\quad \frac{}{\Gamma, x :: \kappa \vdash x :: \kappa} \qquad \frac{\Gamma \vdash V :: \mathsf{Nat}}{\Gamma \vdash \mathsf{S}\, V :: \mathsf{Nat}} \qquad \frac{}{\Gamma \vdash \mathsf{0} :: \mathsf{Nat}}$$

$$\lambda\quad \frac{\Gamma, x :: \kappa \vdash M :: \kappa'}{\Gamma \vdash \lambda x.M :: \kappa \to \kappa'} \qquad \frac{\Gamma \vdash V :: \kappa \to \kappa' \qquad \Delta \vdash W :: \kappa}{\Gamma \uplus \Delta \vdash V\, W :: \kappa'}\quad \text{App}$$

$$\text{Choice}\quad \frac{\Gamma \vdash M :: \kappa \qquad \Delta \vdash N :: \kappa}{\Gamma \cup \Delta \vdash M \oplus_p N :: \kappa}$$

$$\text{Let}\quad \frac{\Gamma \vdash M :: \kappa \qquad \Delta, x :: \kappa \vdash N :: \kappa'}{\Gamma \uplus \Delta \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N :: \kappa'}$$

$$\text{Case}\quad \frac{\Gamma \vdash V :: \mathsf{Nat} \qquad \Delta \vdash W :: \mathsf{Nat} \to \kappa \qquad \Delta \vdash Z :: \kappa}{\Gamma \uplus \Delta \vdash \mathsf{case}\ V\ \mathsf{of}\ \{\mathsf{S} \to W \mid \mathsf{0} \to Z\} :: \kappa}$$

$$\mathsf{letrec}\quad \frac{\Gamma, f :: \mathsf{Nat} \to \kappa \vdash V :: \mathsf{Nat} \to \kappa \qquad \forall x \in \Gamma,\ x :: \mathsf{Nat}}{\Gamma \vdash \mathsf{letrec}\ f = V :: \mathsf{Nat} \to \kappa}$$

Fig. 3. Affine simple types for $\lambda_\oplus$.

The Affine Type System. The affine simple type system is then defined in Figure 3. All the rules are quite standard. Higher-order variables can occur at most once in any probabilistic branch because all binary typing rules – except probabilistic choice – treat contexts affinely. We set $\Lambda^V_\oplus(\Gamma, \kappa) = \{V \in \Lambda^V_\oplus \mid \Gamma \vdash V :: \kappa\}$ and $\Lambda_\oplus(\Gamma, \kappa) = \{M \in \Lambda_\oplus \mid \Gamma \vdash M :: \kappa\}$. We simply write $\Lambda^V_\oplus(\kappa) = \Lambda^V_\oplus(\emptyset, \kappa)$ and $\Lambda_\oplus(\kappa) = \Lambda_\oplus(\emptyset, \kappa)$ when the terms or values are closed. These closed, typable terms enjoy subject reduction and the progress property.

4.2 Monadic Affine Sized Types

This section is devoted to giving the basic definitions and results about monadic affine sized types (MASTs, for short), which can be seen as decorations of the affine simple types with some *size information.*

Sized Types. We consider a set $\mathcal{S}$ of *size variables*, denoted $\mathfrak{i}$, $\mathfrak{j}$, ... and define *sizes* (called *stages* in [19]) as:

$$\mathfrak{s}, \mathfrak{r} \quad ::= \quad \mathfrak{i} \mid \infty \mid \widehat{\mathfrak{s}}$$

where $\widehat{\cdot}$ denotes the *successor* operation. We denote the iterations of $\widehat{\cdot}$ as follows: $\widehat{\widehat{\mathfrak{s}}}$ is denoted $\widehat{\mathfrak{s}}^2$, $\widehat{\widehat{\widehat{\mathfrak{s}}}}$ is denoted $\widehat{\mathfrak{s}}^3$,and so on. By definition, at most one variable $\mathfrak{i} \in \mathcal{S}$ appears in a given size $\mathfrak{s}$. We call it its *spine variable*, denoted as $\mathrm{spine}(\mathfrak{s})$. We write $\mathrm{spine}(\mathfrak{s}) = \emptyset$ when there is no variable in $\mathfrak{s}$. An order $\preccurlyeq$ on sizes can be defined as follows:

$$\frac{}{\mathfrak{s} \preccurlyeq \mathfrak{s}} \qquad \frac{\mathfrak{s} \preccurlyeq \mathfrak{r} \qquad \mathfrak{r} \preccurlyeq \mathfrak{t}}{\mathfrak{s} \preccurlyeq \mathfrak{t}} \qquad \frac{}{\mathfrak{s} \preccurlyeq \widehat{\mathfrak{s}}} \qquad \frac{}{\mathfrak{s} \preccurlyeq \infty}$$

Notice that these rules imply notably that $\widehat{\infty}$ is equivalent to ∞, i.e., $\widehat{\infty} \preccurlyeq \infty$ and $\infty \preccurlyeq \widehat{\infty}$. We consider sizes modulo this equivalence. We can now define sized types and distribution types by mutual induction, calling distributions of (sized) types the distributions over the set of sized types:

Definition 6 (Sized Types, Distribution Types). *Sized types and distribution types are defined by mutual induction, contextually with the function $\langle \cdot \rangle$ which maps any sized or distribution type to its* underlying *affine type.*

$$\begin{array}{lrcl}
\textit{Sized types:} & \sigma,\, \tau & ::= & \sigma \to \mu \;\;\big|\;\; \mathsf{Nat}^{\mathfrak{s}} \\
\textit{Distribution types:} & \mu,\, \nu & ::= & \left\{ \sigma_i^{p_i} \;\;\big|\;\; i \in \mathcal{I} \right\}, \\
\textit{Underlying map:} & \langle \sigma \to \mu \rangle & = & \langle \sigma \rangle \to \langle \mu \rangle \\
 & \langle \mathsf{Nat}^{\mathfrak{s}} \rangle & = & \mathsf{Nat} \\
 & \langle \left\{ \sigma_i^{p_i} \;\big|\; i \in \mathcal{I} \right\} \rangle & = & \langle \sigma_j \rangle
\end{array}$$

For distribution types we require additionally that $\sum_{i \in \mathcal{I}} p_i \leq 1$, that $\mathcal{I}$ is a finite non-empty set, and that $\langle \sigma_i \rangle = \langle \sigma_j \rangle$ for every $i, j \in \mathcal{I}$. In the last equation, j is any element of $\mathcal{I}$.

The definition of sized types is *monadic* in that a higher-order sized type is of the shape $\sigma \to \mu$ where σ is again a sized type, and μ is a *distribution* of sized types.

Contexts and Operations on Them. Contexts are sequences of variables together with a sized type, and at most one distinguished variable with a distribution type:

Definition 7 (Contexts). *Contexts are of the shape $\Gamma \,|\, \Theta$, with*

$$\begin{array}{lrcl|ll}
\text{Sized contexts:} & \Gamma,\, \Delta,\, \ldots & ::= & \emptyset & x : \sigma,\, \Gamma & (x \notin dom(\Gamma)) \\
\text{Distribution contexts:} & \Theta,\, \Psi,\, \ldots & ::= & \emptyset & x : \mu &
\end{array}$$

As usual, we define the domain *$dom(\Gamma)$ of a sized context Γ by induction: $dom(\emptyset) = \emptyset$ and $dom(x : \sigma, \Gamma) = \{x\} \uplus dom(\Gamma)$. We proceed similarly for the domain $dom(\Theta)$ of a distribution context Θ. When a sized context $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$ ($n \geq 1$) is such that there is a simple type κ with $\forall i \in \{1, \ldots, n\},\ \langle \sigma_i \rangle = \kappa$, we say that Γ is* uniform *of simple type κ. We write this as $\langle \Gamma \rangle = \kappa$.*

We write Γ, Δ for the disjoint union *of these sized contexts: it is defined whenever $dom(\Gamma) \cap dom(\Delta) = \emptyset$. We proceed similarly for Θ, Ψ, but note that due to the restriction on the cardinality of such contexts, there is the additional requirement that $\Theta = \emptyset$ or $\Psi = \emptyset$.*

We finally define contexts *as pairs $\Gamma \,|\, \Theta$ of a sized context and of a distribution context, with the constraint that $dom(\Gamma) \cap dom(\Theta) = \emptyset$.*

Definition 8 (Probabilistic Sum). *Let μ and ν be two distribution types. We define their probabilistic sum $\mu \oplus_p \nu$ as the distribution type $p \cdot \mu + (1-p) \cdot \nu$. We extend this operation to a* partial *operation on distribution contexts:*

- *For two distribution types μ and ν such that $\langle \mu \rangle = \langle \nu \rangle$, we define $(x : \mu) \oplus_p (x : \nu) \;=\; x : \mu \oplus_p \nu$,*
- $(x : \mu) \oplus_p \emptyset \;=\; x : p \cdot \mu$,
- $\emptyset \oplus_p (x : \mu) \;=\; x : (1-p) \cdot \mu$,
- *In any other case, the operation is undefined.*

Definition 9 (Weighted Sum of Distribution Contexts). *Let $(\Theta_i)_{i \in \mathcal{I}}$ be a non-empty family of distribution contexts and $(p_i)_{i \in \mathcal{I}}$ be a family of reals of $[0,1]$. We define the weighted sum $\sum_{i \in \mathcal{I}} p_i \cdot \Theta_i$ as the distribution context $x : \sum_{i \in \mathcal{I}} p_i \cdot \mu_i$ when the following conditions are met:*

1. $\exists x, \;\; \forall i \in \mathcal{I}, \; \Theta_i = x : \mu_i$,
2. $\forall (i,j) \in \mathcal{I}^2, \;\; \langle \Theta_i \rangle = \langle \Theta_j \rangle$,
3. *and* $\sum_{i \in \mathcal{I}} p_i \leq 1$,

In any other case, the operation is undefined.

We define the substitution $[\mathfrak{r}/\mathfrak{i}]$ of a size variable in a size or in a sized or distribution type in the expected way; see the long version [18] for details. A subtyping relation allows to lift the order $\preccurlyeq$ on sizes to monadic sized types:

Definition 10 (Subtyping). *We define the subtyping relation $\sqsubseteq$ on sized types and distribution types as follows:*

$$\frac{}{\sigma \sqsubseteq \sigma} \qquad \frac{\mathfrak{s} \preccurlyeq \mathfrak{r}}{\mathsf{Nat}^{\mathfrak{s}} \sqsubseteq \mathsf{Nat}^{\mathfrak{r}}} \qquad \frac{\tau \sqsubseteq \sigma \qquad \mu \sqsubseteq \nu}{\sigma \to \mu \;\sqsubseteq\; \tau \to \nu}$$

$$\frac{\exists f : \mathcal{I} \to \mathcal{J}, \;\; \left(\forall i \in \mathcal{I}, \;\; \sigma_i \sqsubseteq \tau_{f(i)}\right) \text{ and } \left(\forall j \in \mathcal{J}, \;\; \sum_{i \in f^{-1}(j)} p_i \leq p'_j\right)}{\left\{ \sigma_i^{p_i} \;\mid\; i \in \mathcal{I} \right\} \sqsubseteq \left\{ \tau_j^{p'_j} \;\mid\; j \in \mathcal{J} \right\}}$$

Sized Walks and Distribution Types. As we explained in Section 2, the rule typing letrec in the monadic, affine type system relies on an external decision procedure, computable in polynomial time. This procedure ensures that the *sized walk* — a particular instance of *one-counter Markov decision process* (OC-MDP, see [31]), but which does not make use of non-determinism — associated to the type of the recursive function of interest indeed ensures almost sure termination. Let us now define the sized walk associated to a distribution type μ. For the precise connection with OC-MDPs, see the long version [18].

Definition 11 (Sized Walk). *Let $\mathcal{I} \subseteq_{fin} \mathbb{N}$ be a finite set of integers. Let $\{p_i\}_{i \in \mathcal{I}}$ be such that $\sum_{i \in \mathcal{I}} p_i \leq 1$. These parameters define a Markov chain whose set of states is $\mathbb{N}$ and whose transition relation is defined as follows:*

- *the state $0 \in \mathbb{N}$ is stationary (i.e. one goes from 0 to 0 with probability 1),*
- *from the state $s+1 \in \mathbb{N}$ one moves:*

- *to the state $s+i$ with probability p_i, for every $i \in \mathcal{I}$;*
- *to 0 with probability $1 - \left(\sum_{i \in \mathcal{I}} p_i\right)$.*

We call this Markov chain the sized walk *on $\mathbb{N}$ associated to $\left(\mathcal{I}, (p_i)_{i \in \mathcal{I}}\right)$. A sized walk is* almost surely terminating *when it reaches 0 with probability 1 from any initial state.*

Notably, checking whether a sized walk is terminating is relatively easy:

Proposition 1 (Decidability of AST for Sized Walks). *It is decidable in polynomial time whether a sized walk is AST.*

Proof. By encoding sized walks into OC-MDPs, which enjoy this property [31]. See the long version [18].

Definition 12 (From Types to Sized Walks). *Consider a distribution type $\mu = \left\{ (\mathsf{Nat}^{\mathfrak{s}_j} \to \nu_j)^{p_j} \;\middle|\; j \in \mathcal{J} \right\}$ such that $\forall j \in \mathcal{J},$ $\mathrm{spine}\,(\mathfrak{s}_j) = \mathfrak{i}$. Then μ induces a sized walk, defined as follows. First, by definition, $\mathfrak{s}_j$ must be of the shape $\widehat{\mathfrak{i}}^{k_j}$ with $k_j \geq 0$ for every $j \in \mathcal{J}$. We set $\mathcal{I} = \left\{ k_j \;\middle|\; j \in \mathcal{J} \right\}$ and $q_{k_j} = p_j$ for every $j \in \mathcal{J}$. The sized walk induced by the distribution type μ is then the sized walk associated to $(\mathcal{I}, (q_i)_{i \in \mathcal{I}})$.*

Example 1. Let $\mu = \left\{ \left(\mathsf{Nat}^{\mathfrak{i}} \to \mathsf{Nat}^{\infty}\right)^{\frac{1}{2}}, \left(\mathsf{Nat}^{\widehat{\mathfrak{i}}^{2}} \to \mathsf{Nat}^{\infty}\right)^{\frac{1}{3}} \right\}$. Then the induced sized walk is the one associated to $\left(\{0, 2\}, \left(p_0 = \frac{1}{2}, p_2 = \frac{1}{3}\right)\right)$. In other words, it is the random walk on $\mathbb{N}$ which is stationary on 0, and which on non-null integers $i+1$ moves to i with probability $\frac{1}{2}$, to $i+2$ with probability $\frac{1}{3}$, and jumps to 0 with probability $\frac{1}{6}$. Note that the type μ, and therefore the associated sized walk, models a recursive function which calls itself on a size lesser by one unit with probability $\frac{1}{2}$, on a size greater by one unit with probability $\frac{1}{3}$, and which does not call itself with probability $\frac{1}{6}$.

Typing Rules. Judgements are of the shape $\Gamma \,|\, \Theta \vdash M \,:\, \mu$. When a distribution $\mu = \left\{ \sigma^1 \right\}$ is Dirac, we simply write it σ. The type system is defined in Figure 4. As earlier, we define sets of typable terms, and set $\Lambda_{\oplus}^{\mathfrak{s},V}\left(\Gamma \,|\, \Theta, \sigma\right) = \left\{ V \;\middle|\; \Gamma \,|\, \Theta \vdash V \,:\, \sigma \right\}$, and $\Lambda_{\oplus}^{\mathfrak{s}}\left(\Gamma \,|\, \Theta, \mu\right) = \left\{ M \;\middle|\; \Gamma \,|\, \Theta \vdash M \,:\, \mu \right\}$. We abbreviate $\Lambda_{\oplus}^{\mathfrak{s},V}\left(\emptyset \,|\, \emptyset, \sigma\right)$ as $\Lambda_{\oplus}^{\mathfrak{s},V}\left(\sigma\right)$ and $\Lambda_{\oplus}^{\mathfrak{s}}\left(\emptyset \,|\, \emptyset, \sigma\right)$ as $\Lambda_{\oplus}^{\mathfrak{s}}\left(\sigma\right)$.

This sized type system is a refinement of the affine simple type system for $\lambda_{\oplus}$: if $x_1 \,:\, \sigma_1, \ldots, x_n \,:\, \sigma_n \,|\, f \,:\, \mu \vdash M \,:\, \nu$, then it is easily checked that $x_1 :: \langle \sigma_1 \rangle, \ldots, x_n :: \langle \sigma_n \rangle, f :: \langle \mu \rangle \vdash M :: \langle \nu \rangle$.

Lemma 2 (Properties of Distribution Types).
- $\Gamma \,|\, \Theta \vdash V \,:\, \mu \quad \Longrightarrow \quad \mu$ *is Dirac.*
- $\Gamma \,|\, \Theta \vdash M \,:\, \mu \quad \Longrightarrow \quad \mu$ *is proper.*

$$\text{Var}\quad \frac{}{\Gamma, x : \sigma \,|\, \Theta \vdash x : \sigma} \qquad\qquad \frac{}{\Gamma \,|\, x : \sigma \vdash x : \sigma}\quad \text{Var'}$$

$$\text{Succ}\quad \frac{\Gamma \,|\, \Theta \vdash V : \mathsf{Nat}^{\mathfrak{s}}}{\Gamma \,|\, \Theta \vdash \mathsf{S}\, V : \mathsf{Nat}^{\widehat{\mathfrak{s}}}} \qquad\qquad \frac{}{\Gamma \,|\, \Theta \vdash \mathsf{0} : \mathsf{Nat}^{\widehat{\mathfrak{s}}}}\quad \text{Zero}$$

$$\lambda\quad \frac{\Gamma, x : \sigma \,|\, \Theta \vdash M : \mu}{\Gamma \,|\, \Theta \vdash \lambda x.M : \sigma \to \mu} \qquad\qquad \frac{\Gamma \,|\, \Theta \vdash M : \mu \qquad \mu \sqsubseteq \nu}{\Gamma \,|\, \Theta \vdash M : \nu}\quad \text{Sub}$$

$$\text{App}\quad \frac{\Gamma, \Delta \,|\, \Theta \vdash V : \sigma \to \mu \qquad \Gamma, \Xi \,|\, \Psi \vdash W : \sigma \qquad \langle \Gamma \rangle = \mathsf{Nat}}{\Gamma, \Delta, \Xi \,|\, \Theta, \Psi \vdash V\, W : \mu}$$

$$\text{Choice}\quad \frac{\Gamma \,|\, \Theta \vdash M : \mu \qquad \Gamma \,|\, \Psi \vdash N : \nu \qquad \langle \mu \rangle = \langle \nu \rangle}{\Gamma \,|\, \Theta \oplus_p \Psi \vdash M \oplus_p N : \mu \oplus_p \nu}$$

$$\text{Let}\quad \frac{\begin{array}{c}\Gamma, \Delta \,|\, \Theta \vdash M : \left\{ \sigma_i^{p_i} \;\middle|\; i \in \mathcal{I} \right\} \qquad \langle \Gamma \rangle = \mathsf{Nat} \\ \Gamma, \Xi, x : \sigma_i \,|\, \Psi_i \vdash N : \mu_i \quad (\forall i \in \mathcal{I})\end{array}}{\Gamma, \Delta, \Xi \,|\, \Theta, \left(\sum_{i \in \mathcal{I}} p_i \cdot \Psi_i\right) \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N : \sum_{i \in \mathcal{I}} p_i \cdot \mu_i}$$

$$\text{Case}\quad \frac{\Gamma \,|\, \emptyset \vdash V : \mathsf{Nat}^{\widehat{\mathfrak{s}}} \qquad \Delta \,|\, \Theta \vdash W : \mathsf{Nat}^{\mathfrak{s}} \to \mu \qquad \Delta \,|\, \Theta \vdash Z : \mu}{\Gamma, \Delta \,|\, \Theta \vdash \mathsf{case}\ V\ \mathsf{of}\ \{ \mathsf{S} \to W \;|\; \mathsf{0} \to Z \} : \mu}$$

$$\mathsf{letrec}\quad \frac{\begin{array}{c}\langle \Gamma \rangle = \mathsf{Nat} \\ \mathfrak{i} \notin \Gamma \text{ and } \mathfrak{i} \text{ positive in } \nu \text{ and } \forall j \in \mathcal{J},\ \mathrm{spine}(\mathfrak{s}_j) = \mathfrak{i} \\ \left\{ (\mathsf{Nat}^{\mathfrak{s}_j} \to \nu[\mathfrak{s}_j/\mathfrak{i}])^{p_j} \;\middle|\; j \in \mathcal{J} \right\} \text{induces an AST sized walk} \\ \Gamma \,|\, f : \left\{ (\mathsf{Nat}^{\mathfrak{s}_j} \to \nu[\mathfrak{s}_j/\mathfrak{i}])^{p_j} \;\middle|\; j \in \mathcal{J} \right\} \vdash V : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \nu[\widehat{\mathfrak{i}}/\mathfrak{i}]\end{array}}{\Gamma, \Delta \,|\, \Theta \vdash \mathsf{letrec}\ f = V : \mathsf{Nat}^{\mathfrak{r}} \to \nu[\mathfrak{r}/\mathfrak{i}]}$$

Fig. 4. Affine distribution types for $\lambda_{\oplus}$.

Subject Reduction for Monadic Affine Sized Types. The type system enjoys a form of subject reduction adapted to the probabilistic case and more specifically to the fact that terms reduce to *distributions* of terms. Let us sketch the idea of this adapted subject reduction property on an example. Remark that the type system allows us to derive the sequent

$$\emptyset \,|\, \emptyset \vdash \mathsf{0} \oplus \mathsf{0} : \left\{ \left(\mathsf{Nat}^{\widehat{\mathfrak{s}}}\right)^{\frac{1}{2}}, \left(\mathsf{Nat}^{\widehat{\widehat{\mathfrak{r}}}}\right)^{\frac{1}{2}} \right\} \tag{9}$$

where this distribution type is formed by typing a copy of $\mathsf{0}$ with $\mathsf{Nat}^{\widehat{\mathfrak{s}}}$ and the other with $\mathsf{Nat}^{\widehat{\widehat{\mathfrak{r}}}}$. Then, the term $\mathsf{0} \oplus \mathsf{0}$ reduces to $\left\{ \mathsf{0}^{\frac{1}{2}} \right\} + \left\{ \mathsf{0}^{\frac{1}{2}} \right\} = \left\{ \mathsf{0}^{1} \right\} = [\![\mathsf{0} \oplus \mathsf{0}]\!]$: the operational semantics collapses the two copies of $\mathsf{0}$ appearing during the reduction. However, in the spirit of the usual subject reduction for deterministic languages, we would like to type the two copies of $\mathsf{0}$ appearing during the reduction with different types. We therefore use the notion of *pseudo-representation*: $\left[\mathsf{0}^{\frac{1}{2}}, \mathsf{0}^{\frac{1}{2}}\right]$ is a pseudo-representation of $[\![\mathsf{0} \oplus \mathsf{0}]\!]$, and we attribute the type $\mathsf{Nat}^{\widehat{\mathfrak{s}}}$ to the first element of this pseudo-representation and the type $\mathsf{Nat}^{\widehat{\widehat{\mathfrak{t}}}}$ to the other, obtaining the following *closed distribution of typed terms*:

$$\left\{ \left(\mathsf{0} \,:\, \mathsf{Nat}^{\widehat{\mathfrak{s}}}\right)^{\frac{1}{2}}, \left(\mathsf{0} \,:\, \mathsf{Nat}^{\widehat{\widehat{\mathfrak{t}}}}\right)^{\frac{1}{2}} \right\} \tag{10}$$

We can then compute the *average* type of (10), which we call the *expectation type* of this closed distribution of typed terms:

$$\frac{1}{2} \cdot \left\{ \left(\mathsf{Nat}^{\widehat{\mathfrak{s}}}\right)^{1} \right\} + \frac{1}{2} \cdot \left\{ \left(\mathsf{Nat}^{\widehat{\widehat{\mathfrak{t}}}}\right)^{1} \right\} \;=\; \left\{ \left(\mathsf{Nat}^{\widehat{\mathfrak{s}}}\right)^{\frac{1}{2}}, \left(\mathsf{Nat}^{\widehat{\widehat{\mathfrak{t}}}}\right)^{\frac{1}{2}} \right\}$$

Remark that it coincides with the type of the initial term (9). This will be our result of subject reduction: when a closed term M of distribution type μ reduces to a distribution $\mathscr{D}$ of terms, we can type all the terms appearing in a pseudo-representation of $\mathscr{D}$ to obtain a closed distribution of typed terms whose expectation type is μ. Let us now introduce the definitions necessary to the formal statement of the subject reduction property.

Definition 13 (Distributions of Distribution Types, of Typed Terms).

- *A* distribution of distribution types *is a distribution $\mathscr{D}$ over the set of distribution types, and such that $\mu,\, \nu \in \mathsf{S}(\mathscr{D}) \;\Rightarrow\; \langle \mu \rangle \;=\; \langle \nu \rangle$.*
- *A* distribution of typed terms*, or* typed distribution*, is a distribution of typing sequents which are derivable in the monadic, affine sized type system. The representation of such a distribution has thus the following form: $\left\{ (\Gamma_i \,|\, \Theta_i \vdash M_i \,:\, \mu_i)^{p_i} \;\middle|\; i \in \mathcal{I} \right\}$. In the sequel, we restrict to the* uniform *case in which all the terms appearing in the sequents are typed with distribution types of the same fixed underlying type.*
- *A* distribution of closed typed terms*, or* closed typed distribution*, is a typed distribution in which all contexts are $\emptyset \,|\, \emptyset$. In this case, we simply write the representation of the distribution as $\left\{ (M_i \,:\, \mu_i)^{p_i} \;\middle|\; i \in \mathcal{I} \right\}$, or even as $(M_i \,:\, \mu_i)^{p_i}$ when the indexing is clear from context. We write pseudo-representations in a similar way.*

Definition 14 (Expectation Types). *Let $(M_i \,:\, \mu_i)^{p_i}$ be a closed typed distribution. We define its* expectation type *as the distribution type $\mathbb{E}\left((M_i \,:\, \mu_i)^{p_i}\right) = \sum_{i \in \mathcal{I}} \, p_i \mu_i$.*

We can now state the main lemma of subject reduction:

Lemma 3 (Subject Reduction, Fundamental Lemma). *Let $M \in \Lambda_{\oplus}^{\mathfrak{s}}(\mu)$ and $\mathscr{D}$ be the unique closed term distribution such that $M \to_v \mathscr{D}$. Then there exists a closed typed distribution $\left\{ (L_j \,:\, \nu_j)^{p_j} \;\middle|\; j \in \mathcal{J} \right\}$ such that*

- $\mathbb{E}\left((L_j : \nu_j)^{p_j}\right) \ = \ \mu$,
- $\left[\,(L_j)^{p_j} \ \ \middle|\ \ j \in \mathcal{J}\,\right]$ *is a pseudo-representation of* $\mathscr{D}$.

Note that the condition on expectations implies that $\bigcup_{j\in\mathcal{J}} \ \mathsf{S}(\nu_j) \ = \ \mathsf{S}(\mu)$.

The proof of this result, and its generalization to the iterated reduction of closed typed distributions, appear in the long version. They allow us to deduce the following property on the operational semantics of $\lambda_\oplus$-terms:

Theorem 1 (Subject Reduction). *Let* $M \ \in \ \Lambda_\oplus^{\mathfrak{s}}(\mu)$. *Then there exists a closed typed distribution* $\left\{\,(W_j : \sigma_j)^{p_j} \ \ \middle|\ \ j \in \mathcal{J}\,\right\}$ *such that*
- $\mathbb{E}\left((W_j : \sigma_j)^{p_j}\right) \ \preccurlyeq \ \mu$,
- *and that* $\left[\,(W_j)^{p_j} \ \ \middle|\ \ j \in \mathcal{J}\,\right]$ *is a pseudo-representation of* $[\![\, M \,]\!]$.

Note that $\mathbb{E}\left((W_j : \sigma_j)^{p_j}\right) \ \preccurlyeq \ \mu$ since the semantics of a term may not be a proper distribution at this stage. In fact, it will follow from the soundness theorem of Section 5 that the typability of M implies that $\sum [\![\, M \,]\!] = 1$ and thus that the previous statement is an equality.

5 Typability Implies Termination: Reducibility Strikes Again

This section is technically the most advanced one of the paper, and proves that the typing discipline we have introduced indeed enforces almost sure termination. As already mentioned, the technique we will employ is a substantial generalisation of Girard-Tait's reducibility. In particular, reducibility must be made quantitative, in that terms can be said to be reducible *with a certain probability*. This means that reducibility sets will be defined as sets parametrised by a real number p, called the *degree of reducibility* of the set. As Lemma 4 will emphasize, this degree of reducibility ensures that terms contained in a reducibility set parametrised by p terminate with probability at least p. These "intermediate" degrees of reducibility are required to handle the fixpoint construction, and show that recursively-defined terms that are typable are indeed AST — that is, that they belong to the appropriate reducibility set, parametrised by 1.

The first preliminary notion we need is that of a size environment:

Definition 15 (Size Environment). *A* size environment *is any function* ρ *from* $\mathcal{S}$ *to* $\mathbb{N} \cup \{\infty\}$. *Given a size environment* ρ *and a size expression* $\mathfrak{s}$, *there is a naturally defined element of* $\mathbb{N} \cup \{\infty\}$, *which we indicate as* $[\![\mathfrak{s}]\!]_\rho$:
- $[\![\widehat{\mathfrak{i}}^{\,n}]\!]_\rho \ = \ \rho(\mathfrak{i}) + n$,
- $[\![\infty]\!]_\rho \ = \ \infty$.

In other words, the purpose of size environments is to give a semantic meaning to size expressions. Our reducibility sets will be parametrised not only on a probability, but also on a size environment.

Definition 16 (Reducibility Sets).

- *For values of simple type* Nat*, we define the reducibility sets*

$$\mathsf{VRed}^{p}_{\mathsf{Nat}^{\mathfrak{s}},\rho} \quad = \quad \left\{\mathsf{S}^{n}\ 0 \;\middle|\; p > 0 \Longrightarrow n < [\![\mathfrak{s}]\!]_{\rho}\right\}.$$

- *Values of higher-order type are in a reducibility set when their applications to appropriate values are reducible terms, with an adequate degree of reducibility:*

$$\begin{aligned}\mathsf{VRed}^{p}_{\sigma\to\mu,\rho} \quad = \quad \Big\{V \in \Lambda^{V}_{\oplus}(\langle\sigma\to\mu\rangle) \;\Big|\; \forall q \in (0,1], \quad & \forall W \in \mathsf{VRed}^{q}_{\sigma,\rho}, \\ & V\ W \in \mathsf{TRed}^{pq}_{\mu,\rho}\Big\}\end{aligned}$$

- *Distributions of values are reducible with degree p when they consist of values which are themselves globally reducible "enough". Formally,* $\mathsf{DRed}^{p}_{\mu,\rho}$ *is the set of finite distributions of values – in the sense that they have a finite support – admitting a pseudo-representation* $\mathscr{D} = \left[(V_i)^{p_i} \;\middle|\; i \in \mathcal{I}\right]$ *such that, setting* $\mu = \left\{(\sigma_j)^{p'_j} \;\middle|\; j \in \mathcal{J}\right\}$*, there exists a family* $(p_{ij})_{i\in\mathcal{I},j\in\mathcal{J}} \in [0,1]^{|\mathcal{I}|\times|\mathcal{J}|}$ *of probabilities and a family* $(q_{ij})_{i\in\mathcal{I},j\in\mathcal{J}} \in [0,1]^{|\mathcal{I}|\times|\mathcal{J}|}$ *of degrees of reducibility, satisfying:*
  1. $\forall i \in \mathcal{I},\ \ \forall j \in \mathcal{J},\ \ V_i \in \mathsf{VRed}^{q_{ij}}_{\sigma_j,\rho}$,
  2. $\forall i \in \mathcal{I},\ \ \sum_{j\in\mathcal{J}} p_{ij} \ = \ p_i$,
  3. $\forall j \in \mathcal{J},\ \ \sum_{i\in\mathcal{I}} p_{ij} \ = \ \mu(\sigma_j)$,
  4. $p \leq \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} q_{ij}p_{ij}$.

  Note that (2) *and* (3) *imply that* $\sum \mathscr{D} = \sum \mu$*. We say that* $\left[(V_i)^{p_i} \;\middle|\; i \in \mathcal{I}\right]$ witnesses *that* $\mathscr{D} \in \mathsf{DRed}^{p}_{\mu,\rho}$*.*

- *A term is reducible with degree p when its finite approximations compute distributions of values of degree of reducibility arbitrarily close to p:*

$$\begin{aligned}\mathsf{TRed}^{p}_{\mu,\rho} \quad = \quad \Big\{M \in \Lambda_{\oplus}(\langle\mu\rangle) \;\Big|\; \forall 0 \leq r < p, \quad & \exists \nu_r \preccurlyeq \mu, \quad \exists n_r \in \mathbb{N}, \\ & M \Rrightarrow^{n_r}_{v} \mathscr{D}_r \ \text{and}\ \mathscr{D}_r \in \mathsf{DRed}^{r}_{\nu_r,\rho}\Big\}\end{aligned}$$

  Note that here, unlike to the case of DRed*, the fact that* $M \in \Lambda_{\oplus}(\langle\mu\rangle)$ *implies that* μ *is proper.*

The first thing to observe about reducibility sets as given in Definition 16 is that they only deal with closed terms, and not with arbitrary terms. As such, we cannot rely *directly* on them when proving AST for typable terms, at least if we want to prove it by induction on the structure of type derivations. We will therefore define in the sequel an extension of these sets to open terms, which will be based on these sets of closed terms, and therefore enjoy similar properties. The following lemma, relatively easy to prove, is crucial for the understanding of the reducibility sets, for that it shows that the degree of reducibility of a term gives information on the sum of its operational semantics:

Lemma 4 (Reducibility and Termination).
- *Let* $\mathscr{D} \in \mathsf{DRed}^{p}_{\mu,\rho}$*. Then* $\sum \mathscr{D} \geq p$*.*
- *Let* $M \in \mathsf{TRed}^{p}_{\mu,\rho}$*. Then* $\sum [\![\, M \,]\!] \geq p$*.*

It follows from this lemma that terms with degree of reducibility 1 are AST:

Corollary 1 (Reducibility and AST). *Let* $M \in \mathsf{TRed}^{1}_{\mu,\rho}$*. Then* M *is AST.*

Fundamental Properties. Before embarking in the proof that typability implies reducibility, it is convenient to prove some fundamental properties of reducibility sets, which inform us about how these sets are structured, and which will be crucial in the sequel. First of all, if the degree of reducibility p is 0, then no assumption is made on the probability of termination of terms, distributions or values. It follows that the three kinds of reducibility sets collapse to the set of all affinely simply typable terms, distributions or values:

Lemma 5 (Candidates of Null Reducibility).
- *If $V \in \Lambda_{\oplus}^{V}(\kappa)$, then $V \in \mathsf{VRed}^{0}_{\sigma,\rho}$ for every σ such that $\langle\sigma\rangle = \kappa$ and every size environment ρ.*
- *Let $\mathscr{D} = \left\{ (V_i)^{p_i} \;\middle|\; i \in \mathcal{I} \right\}$ be a finite distribution of values. If $\forall i \in \mathcal{I},\ V_i \in \Lambda_{\oplus}^{V}(\kappa)$, then $\mathscr{D} \in \mathsf{DRed}^{0}_{\mu,\rho}$ for every μ such that $\langle\mu\rangle = \kappa$ and $\sum \mu = \sum \mathscr{D}$ and every ρ.*
- *If $M \in \Lambda_{\oplus}(\kappa)$, then $M \in \mathsf{TRed}^{0}_{\mu,\rho}$ for μ such that $\langle\mu\rangle = \kappa$ and every ρ.*

As p gives us a lower bound on the sum of the semantics of terms, it is easily guessed that a term having degree of reducibility p must also have degree of reducibility $q < p$. The following lemma makes this statement precise:

Lemma 6 (Downward Closure). *Let σ be a sized type, μ be a distribution type and ρ be a size environment. Let $0 \leq q < p \leq 1$. Then:*
- *For any value V, $V \in \mathsf{VRed}^{p}_{\sigma,\rho} \implies V \in \mathsf{VRed}^{q}_{\sigma,\rho}$,*
- *For any finite distribution of values $\mathscr{D}$, $\mathscr{D} \in \mathsf{DRed}^{p}_{\mu,\rho} \implies \mathscr{D} \in \mathsf{DRed}^{q}_{\mu,\rho}$,*
- *For any term M, $M \in \mathsf{TRed}^{p}_{\mu,\rho} \implies M \in \mathsf{TRed}^{q}_{\mu,\rho}$.*

To analyse the letrec construction, we will prove that, for every $\varepsilon \in (0,1]$, performing enough unfoldings of the fixpoint allows to prove that the recursively-defined term is in a reducibility set parametrised by $1-\varepsilon$. We will be able to conclude on the AST nature of recursive constructions using the following continuity lemma, proved using the theory of linear programming [18]:

Lemma 7 (Continuity). *Let σ be a sized type, μ be a distribution type and ρ be a size environment. Let $p \in (0,1]$. Then:*
- $\mathsf{VRed}^{p}_{\sigma,\rho} = \bigcap_{0<q<p} \mathsf{VRed}^{q}_{\sigma,\rho}$,
- $\mathsf{DRed}^{p}_{\mu,\rho} = \bigcap_{0<q<p} \mathsf{DRed}^{q}_{\mu,\rho}$,
- $\mathsf{TRed}^{p}_{\mu,\rho} = \bigcap_{0<q<p} \mathsf{TRed}^{q}_{\mu,\rho}$.

The last fundamental property about reducibility sets which will be crucial to treat the recursive case is the following, stating that the sizes appearing in a sized type may be recovered in the reducibility set by using an appropriate semantics of the size variables, and conversely:

Lemma 8 (Size Commutation). *Let $\mathfrak{i}$ be a size variable, $\mathfrak{s}$ be a size such that $\mathfrak{s} = \infty$ or that $\mathrm{spine}(\mathfrak{s}) \neq \mathfrak{i}$ and ρ be a size environment. Then:*
- $\mathsf{VRed}^{p}_{\sigma[\mathfrak{s}/\mathfrak{i}],\rho} = \mathsf{VRed}^{p}_{\sigma,\rho[\mathfrak{i}\mapsto[\![\mathfrak{s}]\!]_{\rho}]}$,
- $\mathsf{DRed}^{p}_{\mu[\mathfrak{s}/\mathfrak{i}],\rho} = \mathsf{DRed}^{p}_{\mu,\rho[\mathfrak{i}\mapsto[\![\mathfrak{s}]\!]_{\rho}]}$,
- $\mathsf{TRed}^{p}_{\mu[\mathfrak{s}/\mathfrak{i}],\rho} = \mathsf{TRed}^{p}_{\mu,\rho[\mathfrak{i}\mapsto[\![\mathfrak{s}]\!]_{\rho}]}$.

Unfoldings. The most difficult step in proving all typable terms to be reducible is, unexpectedly, proving that terms involving *recursion* are reducible whenever their respective *unfoldings* are. This very natural concept expresses simply that any term in the form $\mathsf{letrec}\ f\ =\ W$ is assumed to compute the fixpoint of the function defined by W.

Definition 17 (n-Unfolding). *Suppose that $V = (\mathsf{letrec}\ f\ =\ W)$ is closed, then the* n-unfolding *of V is:*

- *V if $n = 0$;*
- *$W[Z/f]$ if $n = m + 1$ and Z is the m-unfolding of V.*

We write the set of unfoldings of V as $Unfold\,(V)$. Note that if V admits a simple type, then all its unfoldings have this same simple type as well. In the sequel, we implicitly consider that V is simply typed.

Any unfolding of $V = (\mathsf{letrec}\ f\ =\ W)$ should behave like V itself: all unfoldings of V should be equivalent. This, however, cannot be proved using simply the operational semantics. It requires some work, and techniques akin to logical relations, to prove (see [18]) this behavioural equivalence between a recursive definition and its unfoldings.

Proposition 2 (Reducibility is Stable by Unfolding). *Let $n \in \mathbb{N}$ and $V = (\mathsf{letrec}\ f\ =\ W)$ be a closed value. Suppose that Z is the n-unfolding of V. Then $V \in \mathsf{VRed}^{p}_{\mathsf{Nat}^s \to \mu,\rho}$ if and only if $Z \in \mathsf{VRed}^{p}_{\mathsf{Nat}^s \to \mu,\rho}$.*

Extension to Open Terms. We are now ready to extend the notion of reducibility set from the realm of *closed* terms to the one of *open* terms. This turns out to be subtle. The guiding intuition is that one would like to define a term M with free variables in $\overrightarrow{x}$ to be reducible iff any closure $M[\overrightarrow{V}/\overrightarrow{x}]$ is itself reducible in the sense of Definition 16. What happens, however, to the underlying degree of reducibility p? How do we relate the degrees of reducibility of $\overrightarrow{V}$ with the one of $M[\overrightarrow{V}/\overrightarrow{x}]$? The answer is contained in the following definition:

Definition 18 (Reducibility Sets for Open Terms). *Suppose that Γ is a sized context in the form $x_1 : \sigma_1, \ldots, x_n : \sigma_n$, and that y is a variable distinct from $x_1, \ldots, x_n$. Then we define the following sets of terms and values:*

$$\mathsf{OTRed}^{\Gamma\,|\,\emptyset}_{\mu,\rho} = \Big\{ M \ \Big|\ \forall (q_i)_i \in [0,1]^n,\ \ \forall\,(V_1,\,\ldots,\,V_n) \in \textstyle\prod_{i=1}^{n}\ \mathsf{VRed}^{q_i}_{\sigma_i,\rho},\ \ M[\overrightarrow{V}/\overrightarrow{x}] \in \mathsf{TRed}^{\prod_{i=1}^{n} q_i}_{\mu,\rho} \Big\}$$

$$\mathsf{OVRed}^{\Gamma\,|\,\emptyset}_{\mu,\rho} = \Big\{ W \ \Big|\ \forall (q_i)_i \in [0,1]^n,\ \ \forall\,(V_1,\,\ldots,\,V_n) \in \textstyle\prod_{i=1}^{n}\ \mathsf{VRed}^{q_i}_{\sigma_i,\rho},\ \ W[\overrightarrow{V}/\overrightarrow{x}] \in \mathsf{VRed}^{\prod_{i=1}^{n} q_i}_{\mu,\rho} \Big\}$$

$$\mathsf{OTRed}_{\mu,\rho}^{\Gamma \,|\, y\,:\,\{\tau_j^{p_j}\}_{j\in\mathcal{J}}} \quad = \quad \Big\{ M \;\; \Big| \;\; \forall (q_i)_i \in [0,1]^n, \;\; \forall \overrightarrow{V} \in \textstyle\prod_{i=1}^n \; \mathsf{VRed}_{\sigma_i,\rho}^{q_i}, \\ \forall \left(q'_j\right)_j \in [0,1]^{\mathcal{J}}, \;\; \forall W \in \textstyle\bigcap_{j\in\mathcal{J}} \; \mathsf{VRed}_{\tau_j,\rho}^{q'_j}, \\ M[\overrightarrow{V}, W/\overrightarrow{x}, y] \in \mathsf{TRed}_{\mu,\rho}^{\alpha} \Big\}$$

$$\mathsf{OVRed}_{\mu,\rho}^{\Gamma \,|\, y\,:\,\{\tau_j^{p_j}\}_{j\in\mathcal{J}}} \quad = \quad \Big\{ Z \;\; \Big| \;\; \forall (q_i)_i \in [0,1]^n, \;\; \forall \overrightarrow{V} \in \textstyle\prod_{i=1}^n \; \mathsf{VRed}_{\sigma_i,\rho}^{q_i}, \\ \forall \left(q'_j\right)_j \in [0,1]^{\mathcal{J}}, \;\; \forall W \in \textstyle\bigcap_{j\in\mathcal{J}} \; \mathsf{VRed}_{\tau_j,\rho}^{q'_j}, \\ Z[\overrightarrow{V}, W/\overrightarrow{x}, y] \in \mathsf{VRed}_{\mu,\rho}^{\alpha} \Big\}$$

where $\alpha = \left(\prod_{i=1}^n q_i\right)\left(\left(\sum_{j\in\mathcal{J}} p_j q'_j\right) + 1 - \left(\sum_{j\in\mathcal{J}} p_j\right)\right)$ *is called the degree of reducibility. Note that these sets extend the ones for closed terms: in particular,* $\mathsf{OTRed}_{\mu,\rho}^{\emptyset\,|\,\emptyset} = \mathsf{TRed}_{\mu,\rho}^{1}$*.*

Lemma 9 (Reducible Values are Reducible Terms). *For every* Γ*,* Θ*,* σ *and* ρ*,* $V \in \mathsf{OVRed}_{\sigma,\rho}^{\Gamma\,|\,\Theta}$ *if and only if* $V \in \mathsf{OTRed}_{\{\,\sigma^1\,\},\rho}^{\Gamma\,|\,\Theta}$*. An immediate consequence is that* $\mathsf{OVRed}_{\sigma,\rho}^{\Gamma\,|\,\Theta} \subseteq \mathsf{OTRed}_{\{\,\sigma^1\,\},\rho}^{\Gamma\,|\,\Theta}$*.*

Reducibility and Sized Walks. To handle the fixpoint rule, we need to relate the notion of sized walk which guards it with the reducibility sets, and in particular with the degrees of reducibility we can attribute to recursively-defined terms.

Definition 19 (Probabilities of Convergence in Finite Time). *Let us consider a sized walk. We define the associated* probabilities of convergence in finite time $(Pr_{n,m})_{n\in\mathbb{N},m\in\mathbb{N}}$ *as follows:* $\forall n \in \mathbb{N}$*,* $\forall m \in \mathbb{N}$*, the real number* $Pr_{n,m}$ *is defined as the probability that, starting from* m*, the sized walk reaches* 0 *in* at most n *steps.*

The point is that, for an AST sized walk, the more we iterate, the closer we get to reaching 0 in finite time n with probability 1.

Lemma 10 (Finite Approximations of AST). *Let* $m \in \mathbb{N}$ *and* $\varepsilon \in (0,1]$*. Consider a sized walk, and its associated probabilities of convergence in finite time* $(Pr_{n,m})_{n\in\mathbb{N},m\in\mathbb{N}}$*. If the sized walk is AST, there exists* $n \in \mathbb{N}$ *such that* $Pr_{n,m} \geq 1 - \varepsilon$*.*

The following lemma is the crucial result relating sized walks with the reducibility sets. It proves that, when the sized walk is AST, and after substitution of the variables of the context by reducible values in the recursively-defined term, we can prove the degree of reducibility to be any probability $Pr_{n,m}$ of convergence in finite time.

Lemma 11 (Convergence in Finite Time and letrec). *Consider the distribution type* $\mu = \left\{ (\mathsf{Nat}^{\mathfrak{s}_j} \to \nu[\mathfrak{s}_j/\mathfrak{i}])^{p_j} \;\middle|\; j \in \mathcal{J} \right\}$*. Let* Γ *be the sized context* $x_1 : \mathsf{Nat}^{\mathfrak{r}_1}, \ldots, x_l : \mathsf{Nat}^{\mathfrak{r}_l}$*. Suppose that* $\Gamma \,|\, f : \mu \vdash V : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \nu[\widehat{\mathfrak{i}}/\mathfrak{i}]$ *and that*

μ induces an AST sized walk. Denote $(Pr_{n,m})_{n\in\mathbb{N},m\in\mathbb{N}}$ its associated probabilities of convergence in finite time. Suppose that $V \in \mathsf{OVRed}^{\Gamma\,|\,f\,:\,\mu}_{\mathsf{Nat}^{\widehat{\mathfrak{i}}}\to\nu[\widehat{\mathfrak{i}}/\mathfrak{i}],\rho}$ for every ρ. Let $\overrightarrow{W} \in \prod_{i=1}^{l} \mathsf{VRed}^{1}_{\mathsf{Nat}^{\mathfrak{r}_i},\rho}$, then for every $(n,m) \in \mathbb{N}^2$, we have that

$$\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}] \ \in \ \mathsf{VRed}^{Pr_{n,m}}_{\mathsf{Nat}^{\mathfrak{i}}\to\nu,\rho[\mathfrak{i}\mapsto m]}$$

Proof. We give a sketch of the proof, to be found in the long version [18]. The proof is by recurrence on n. The main case relies on the decomposition $Pr_{n+1,m'+1} \ = \ \sum_{j\in\mathcal{J}} \ p_j Pr_{n,m'+k_j} \ + \ 1 - \left(\sum_{j\in\mathcal{J}} \ p_j\right)$. The induction hypothesis allows then to state that for every $j \in \mathcal{J}$ we have $\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}] \ \in \ \mathsf{VRed}^{Pr_{n,m'+k_j}}_{\mathsf{Nat}^{\mathfrak{i}}\to\nu,\rho[\mathfrak{i}\mapsto m'+k_j]}$. We use the Size Commutation lemma (Lemma 8) to obtain that $\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}]$ is in an appropriate intersection of reducibility sets, and the hypothesis that $V \in \mathsf{OVRed}^{\Gamma\,|\,f\,:\,\mu}_{\mathsf{Nat}^{\widehat{\mathfrak{i}}}\to\nu[\widehat{\mathfrak{i}}/\mathfrak{i}],\rho[\mathfrak{i}\mapsto m']}$ then implies that $V[\overrightarrow{W}, \mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}]/\overrightarrow{x}, f] \ \in \ \mathsf{VRed}^{Pr_{n+1,m'+1}}_{\mathsf{Nat}^{\mathfrak{i}}\to\nu,\rho[\mathfrak{i}\mapsto m'+1]}$, using the Size Commutation lemma once again. As this term is an unfolding of $\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}]$, we conclude using Proposition 2. □

When $m = \infty$, the previous lemma does not allow to conclude, and an additional argument is required. Indeed, it does not make sense to consider a sized walk beginning from ∞: the meaning of this size is in fact *any integer*, not the ordinal ω. The following lemma justifies this vision by proving that, if a term is in a reducibility set for any finite interpretation of a size, then it is also in the set where the size is interpreted as ∞.

Lemma 12 (Reducibility for Infinite Sizes). *Suppose that $\mathfrak{i}$ pos ν and that W is the value $\mathsf{letrec}\ f \ = \ V$. If $W \in \mathsf{VRed}^{p}_{\mathsf{Nat}^{\mathfrak{i}}\to\nu,\rho[\mathfrak{i}\mapsto n]}$ for every $n \in \mathbb{N}$, then $W \in \mathsf{VRed}^{p}_{\mathsf{Nat}^{\mathfrak{i}}\to\nu,\rho[\mathfrak{i}\mapsto\infty]}$.*

All these fundamental lemmas allow us to prove the following proposition, which expresses that all typable terms are reducible and is the key step towards the fact that typability implies AST:

Proposition 3 (Typing Soundness). *If $\Gamma\,|\,\Theta \vdash M \, : \, \mu$, then $M \in \mathsf{OTRed}^{\Gamma\,|\,\Theta}_{\mu,\rho}$ for every ρ. Similarly, if $\Gamma\,|\,\Theta \vdash V \, : \, \sigma$, then $V \in \mathsf{OVRed}^{\Gamma\,|\,\Theta}_{\sigma,\rho}$ for every ρ.*

Proof. We proceed by induction on the derivation of the sequent $\Gamma\,|\,\Theta \vdash M \, : \, \mu$. When $M = V$ is a value, we know by Lemma 2 that $\mu = \{\, \sigma^1 \,\}$; and we prove that $V \in \mathsf{OVRed}^{\Gamma\,|\,\Theta}_{\sigma,\rho}$ for every ρ. By Lemma 9 we obtain that $V \in \mathsf{OTRed}^{\Gamma\,|\,\Theta}_{\mu,\rho}$ for every ρ. We proceed by case analysis on the last rule of the derivation:

- letrec: Suppose that $\Gamma, \Delta\,|\,\Theta \vdash \mathsf{letrec}\ f \ = \ V \ : \ \mathsf{Nat}^{\mathfrak{r}} \to \nu[\mathfrak{r}/\mathfrak{i}]$. We treat the case where $\Delta \ = \ \Theta \ = \ \emptyset$. The general case is easily deduced using the downward-closure of the reducibility sets (Lemma 6). Let $\Gamma \ = \ x_1 \ :$

$\mathsf{Nat}^{\mathfrak{r}_1}, \dots, x_n : \mathsf{Nat}^{\mathfrak{r}_n}$. We need to prove that, for every family $(q_i)_i \in [0,1]^n$ and every $(W_1, \dots, W_n) \in \prod_{i=1}^n \mathsf{VRed}^{q_i}_{\mathsf{Nat}^{\mathfrak{r}_i},\rho}$, we have

$$(\mathsf{letrec}\ f \ = \ V)\,[\overrightarrow{W}/\overrightarrow{x}] \ = \ \Big(\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}]\Big) \ \in \ \mathsf{VRed}^{\prod_{i=1}^n q_i}_{\mathsf{Nat}^{\mathfrak{r}} \to \nu[\mathfrak{r}/\mathfrak{i}],\rho}$$

If there exists $i \in \mathcal{I}$ such that $q_i = 0$, the result is immediate as the term is simply-typed and Lemma 5 applies. Else, for every $i \in \mathcal{I}$, we have by definition that $\mathsf{VRed}^{q_i}_{\mathsf{Nat}^{\mathfrak{r}_i},\rho} = \mathsf{VRed}^{1}_{\mathsf{Nat}^{\mathfrak{r}_i},\rho}$. Since the sets VRed are downward-closed (Lemma 6), it is in fact enough to prove that for every $(W_1, \dots, W_n) \in \prod_{i=1}^n \mathsf{VRed}^{1}_{\mathsf{Nat}^{\mathfrak{r}_i},\rho}$, we have

$$\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}] \ \in \ \mathsf{VRed}^{1}_{\mathsf{Nat}^{\mathfrak{r}} \to \nu[\mathfrak{r}/\mathfrak{i}],\rho}$$

Moreover, by size commutation (Lemma 8),

$$\mathsf{VRed}^{1}_{\mathsf{Nat}^{\mathfrak{r}} \to \nu[\mathfrak{r}/\mathfrak{i}],\rho} \ = \ \mathsf{VRed}^{1}_{\mathsf{Nat}^{\mathfrak{i}} \to \nu,\rho[\mathfrak{i} \mapsto [\![\mathfrak{r}]\!]_\rho]}$$

Let us therefore prove the stronger fact that, for *every* integer $m \in \mathbb{N} \cup \{\infty\}$,

$$\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}] \ \in \ \mathsf{VRed}^{1}_{\mathsf{Nat}^{\mathfrak{i}} \to \nu,\rho[\mathfrak{i} \mapsto m]}$$

Now, the typing derivation gives us that $\Gamma \,|\, f : \mu \vdash V : \mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \nu[\widehat{\mathfrak{i}}/\mathfrak{i}]$ and that μ induces an AST sized walk. Denote $(Pr_{n,m})_{n\in\mathbb{N},m\in\mathbb{N}}$ its associated probabilities of convergence in finite time. By induction hypothesis, $V \in \mathsf{OVRed}^{\Gamma \,|\, f : \mu}_{\mathsf{Nat}^{\widehat{\mathfrak{i}}} \to \nu[\widehat{\mathfrak{i}}/\mathfrak{i}],\rho}$ for every ρ and we can apply Lemma 11. It follows that, for every $(n,m) \in \mathbb{N}$,

$$\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}] \ \in \ \mathsf{VRed}^{Pr_{n,m}}_{\mathsf{Nat}^{\mathfrak{i}} \to \nu,\rho[\mathfrak{i} \mapsto m]}$$

Let $\varepsilon \in (0,1)$. By Lemma 10, there exists $n \in \mathbb{N}$ such that $Pr_{n,m} \geq 1 - \varepsilon$. Using downward closure (Lemma 6) and quantifying over all the ε, we obtain

$$\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}] \ \in \ \bigcap_{0<\varepsilon<1} \mathsf{VRed}^{1-\varepsilon}_{\mathsf{Nat}^{\mathfrak{i}} \to \nu,\rho[\mathfrak{i} \mapsto m]}$$

so that, by continuity of VRed (Lemma 7), we obtain

$$\mathsf{letrec}\ f \ = \ V[\overrightarrow{W}/\overrightarrow{x}] \ \in \ \mathsf{VRed}^{1}_{\mathsf{Nat}^{\mathfrak{i}} \to \nu,\rho[\mathfrak{i} \mapsto m]} \tag{11}$$

for every $m \in \mathbb{N}$, allowing us to conclude. It remains however to treat the case where $m = \infty$. Since $\mathfrak{i}$ pos ν and that (11) holds for every $m \in \mathbb{N}$, Lemma 12 applies and we obtain the result.

- **Other cases:** the other cases are treated in the long version [18]. □

This proposition, together with the definition of OTRed, implies the main result of the paper, namely that typability implies almost-sure termination:

Theorem 2. *Suppose that $M \in \Lambda^{\mathfrak{s}}_{\oplus}(\mu)$. Then M is AST.*

Proof. Suppose that $M \in \Lambda^{\mathfrak{s}}_{\oplus}(\mu)$, then by Proposition 3 we have $M \in \mathsf{OTRed}^{\emptyset \,|\, \emptyset}_{\mu,\rho}$ for every ρ. By definition, $\mathsf{OTRed}^{\emptyset \,|\, \emptyset}_{\mu,\rho} = \mathsf{TRed}^{1}_{\mu,\rho}$. Corollary 1 then implies that M is AST.

6 Conclusions and Perspectives

We presented a type system for an affine, simply-typed λ-calculus enriched with a probabilistic choice operator, constructors for the natural numbers, and recursion. This affinity constraint implies that a given higher-order variable may occur (freely) at most once in any probabilistic branch of a program. The type system we designed decorates the affine simple types with *size information*, allowing to incorporate in the types relevant information about the recursive behaviour of the functions contained in the program. A guard condition on the typing rule for letrec, formulated with reference to an appropriate Markov chain, ensures that typable terms are AST. The proof of soundness of this type system for AST relies on a quantitative extension of the reducibility method, to accommodate sets of candidates to the infinitary and probabilistic nature of the computations we consider.

A first natural question is the one of the decidability of type inference for our system. In the deterministic case, this question was only addressed by Barthe and colleagues in an unpublished tutorial [20], and their solution is technically involved, especially when it comes to dealing with the fixpoint rule. We believe that their approach could be extended to our system of monadic sized types, and hope that it could provide a decidable type inference procedure for it. However, this extension will certainly be challenging, as we need to appropriately infer distribution types associated with AST sized walks in the letrec rule.

Another perspective would be to study the general, *non-affine* case. This is challenging, for two reasons. First, the system of size annotations needs to be more expressive in order to distinguish between various occurrences of a same function symbol in a same probabilistic branch. A solution would be to use the combined power of dependent types – which already allowed Xi to formulate an interesting type system for termination in the deterministic case [23] – and of linearity: we could use *linear dependent types* [34] to formulate an extension of the monadic sized type system keeping track of *how many* recursive calls are performed, and of the size of *each* recursive argument. The second challenge would then be to associate, in the typing rule for letrec, this information contained in linear dependent types with an appropriate random process. This random process should be kept decidable to guarantee that at least *derivation* checking can be automated, and there will probably be a trade-off between the duplication power we allow in programs and the complexity of deciding AST for the guard in the letrec rule.

The extension of our type system to deal with general inductive datatypes is essentially straightforward. Other perspectives would be to enrich the type system so as to be able to treat coinductive data, polymorphic types, or ordinal sizes, three features present in most system of sized types dealing with the traditional deterministic case, but which we chose not to address in this paper to focus on the already complex task of accommodating sized types to a probabilistic and higher-order framework.

References

1. Manning, C.D., Schütze, H.: Foundations of statistical natural language processing. MIT Press (2001)
2. Pearl, J.: Probabilistic reasoning in intelligent systems - networks of plausible inference. Morgan Kaufmann series in representation and reasoning. (1989)
3. Thrun, S.: Robotic mapping: A survey. In: Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann (2002)
4. de Leeuw, K., Moore, E.F., Shannon, C.E., Shapiro, N.: Computability by probabilistic machines. Automata Studies **34** (1956) 183–212
5. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)
6. Barthe, G., Grégoire, B., Béguelin, S.Z.: Formal certification of code-based cryptographic proofs. In Shao, Z., Pierce, B.C., eds.: POPL 2009, ACM (2009) 90–101
7. Barthe, G., Grégoire, B., Heraud, S., Béguelin, S.Z.: Computer-aided security proofs for the working cryptographer. In Rogaway, P., ed.: CRYPTO 2011. Volume 6841 of LNCS., Springer (2011) 71–90
8. Goodman, N.D., Mansinghka, V.K., Roy, D.M., Bonawitz, K., Tenenbaum, J.B.: Church: a language for generative models. In McAllester, D.A., Myllymäki, P., eds.: UAI 2008, AUAI Press (2008) 220–229
9. Bournez, O., Kirchner, C.: Probabilistic rewrite strategies. applications to ELAN. In Tison, S., ed.: RTA 2002. Volume 2378 of LNCS., Springer (2002) 252–266
10. Esparza, J., Gaiser, A., Kiefer, S.: Proving termination of probabilistic programs using patterns. In Madhusudan, P., Seshia, S.A., eds.: CAV 2012. Volume 7358 of LNCS., Springer (2012) 123–138
11. Fioriti, L.M.F., Hermanns, H.: Probabilistic termination: Soundness, completeness, and compositionality. In Rajamani, S.K., Walker, D., eds.: POPL 2015, ACM (2015) 489–501
12. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. In Bodík, R., Majumdar, R., eds.: POPL 2016, ACM (2016) 327–342
13. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through positivstellensatz's. In Chaudhuri, S., Farzan, A., eds.: CAV 2016. Volume 9779 of LNCS., Springer (2016) 3–22
14. Hughes, J., Pareto, L., Sabry, A.: Proving the correctness of reactive systems using sized types. In Boehm, H., Jr., G.L.S., eds.: POPL'96, ACM Press (1996) 410–423
15. Hofmann, M.: A mixed modal/linear lambda calculus with applications to Bellantoni-Cook safe recursion. In Nielsen, M., Thomas, W., eds.: CSL '97. Volume 1414 of LNCS., Springer (1997) 275–294
16. Girard, J.Y., Taylor, P., Lafont, Y.: Proofs and Types. Cambridge University Press, New York, NY, USA (1989)
17. Dal Lago, U., Hofmann, M.: Realizability models and implicit complexity. Theor. Comput. Sci. **412**(20) (2011) 2029–2047
18. Dal Lago, U., Grellois, C.: Probabilistic termination by monadic affine sized typing (long version). https://arxiv.org/abs/1701.04089 (2016)
19. Barthe, G., Frade, M.J., Giménez, E., Pinto, L., Uustalu, T.: Type-based termination of recursive definitions. MSCS **14**(1) (2004) 97–141
20. Barthe, G., Grégoire, B., Riba, C.: A tutorial on type-based termination. In Bove, A., Barbosa, L.S., Pardo, A., Pinto, J.S., eds.: ALFA Summer School 2008. Volume 5520 of LNCS., Springer (2008) 100–152

21. Barthe, G., Grégoire, B., Riba, C.: Type-based termination with sized products. In Kaminski, M., Martini, S., eds.: CSL 2008. Volume 5213 of LNCS., Springer (2008) 493–507
22. Abel, A.: Termination checking with types. ITA **38**(4) (2004) 277–319
23. Xi, H.: Dependent types for program termination verification. Higher-Order and Symbolic Computation **15**(1) (2002) 91–131
24. Amadio, R.M., Coupet-Grimal, S.: Analysis of a guard condition in type theory. In Nivat, M., ed.: FoSSaCS'98. Volume 1378 of LNCS., Springer (1998) 48–62
25. Bournez, O., Garnier, F.: Proving positive almost-sure termination. In Giesl, J., ed.: RTA 2005. Volume 3467 of LNCS., Springer (2005) 323–337
26. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In Sharygina, N., Veith, H., eds.: CAV 2013. Volume 8044 of LNCS., Springer (2013) 511–526
27. Cappai, A., Dal Lago, U.: On equivalences, metrics, and polynomial time. In Kosowski, A., Walukiewicz, I., eds.: FCT 2015. Volume 9210 of LNCS., Springer (2015) 311–323
28. Dal Lago, U., Parisen Toldin, P.: A higher-order characterization of probabilistic polynomial time. Inf. Comput. **241** (2015) 114–141
29. Dal Lago, U.: A short introduction to implicit computational complexity. In: ESSLLI 2010. (2011) 89–109
30. Dal Lago, U., Zorzi, M.: Probabilistic operational semantics for the lambda calculus. RAIRO - Theor. Inf. and Applic. **46**(3) (2012) 413–450
31. Brázdil, T., Brožek, V., Etessami, K., Kučera, A., Wojtczak, D.: One-counter markov decision processes. 21st ACM-SIAM Symposium on Discrete Algorithms (2010)
32. Dal Lago, U.: The geometry of linear higher-order recursion. In: LICS 2005, IEEE Computer Society (2005) 366–375
33. Sabry, A., Felleisen, M.: Reasoning about programs in continuation-passing style. Lisp and Symbolic Computation **6**(3-4) (1993) 289–360
34. Dal Lago, U., Gaboardi, M.: Linear dependent types and relative completeness. In: LICS 2011, IEEE Computer Society (2011) 133–142