Probabilistic λ-Calculus and Quantitative Program Analysis

Alessandra Di Pierro Chris Hankin Herbert Wiklicky

January 6, 2005

Abstract

We show how the framework of probabilistic abstract interpretation can be applied to statically analyse a probabilistic version of the λ-calculus. The resulting analysis allows for a more speculative use of its outcomes based on the consideration of statistically defined quantities. After introducing a linear operator based semantics for our probabilistic λ-calculus Λ_p, and reviewing the framework of abstract interpretation and strictness analysis, we demonstrate our technique by constructing a probabilistic (first-order) strictness analysis for Λ_p.

1 Introduction

In this paper we aim to show how probabilistic abstract interpretation [12, 13] can be used to analyse terms in a probabilistic λ-calculus. Our running example will be a simple strictness analysis [18, 2]. This analysis has been used in the non-probabilistic setting to optimise lazy functional languages by allowing lazy evaluation to be replaced by eager evaluation without compromising the semantics. We suggest that, in the probabilistic setting, strictness analysis might be used to perform a more speculative optimisation which replaces lazy by eager evaluation as long as the risk of introducing non-termination is sufficiently low.

In order to illustrate how quantitative elements change classical analysis, we will present an example borrowed from the theory of stochastic processes (see Example 2.1 of [6]), which is related to economics and in particular to risk management. Consider the situation in which a company starts with initial capital of $\mathbf{Cap}_0$, at each time step its income is $\mathbf{In}_i$ and its outlay to meet claims is $\mathbf{Out}_i$; the sequence of incomes and outlays are modelled by mutually independent and identically distributed variables. The fortunes of the company are modelled by a simple random walk with an absorbing barrier at zero and jumps $\mathbf{Step}_n = \mathbf{In}_n - \mathbf{Out}_n$:

$$\mathbf{Cap}_n = \begin{cases} \mathbf{Cap}_{n-1} + \mathbf{Step}_n & \text{if } \mathbf{Cap}_{n-1} > 0 \text{ and } \mathbf{Cap}_{n-1} + \mathbf{Step}_n > 0 \\ 0 & \text{otherwise} \end{cases}$$

Thus, at each step the capital of the company changes according to the profit and losses it makes. Once losses exceed the capital the company will go bankrupt and remain so from then on.

Qualitatively, we can analyse the random walk and just conclude that **Cap** ranges over the interval $[0, \infty)$; quantitatively, we can ask the more interesting question: *What is the probability of bankruptcy for a given statistical behaviour of claims and income?* Obviously, one can ask similar questions also with respect to computational processes which in one way or another use limited computational resources.

The probabilistic version of the λ-calculus we introduce is the base for the definition of a quantitative approach to a semantics based program analysis. We will define a probabilistic semantics for the λ-calculus which reflects the spirit of the example above by associating to a λ-program a linear operator specifying the dynamics of a Markov chain (a generalisation of a random walk). We will then show how this linear operator semantics can be used to apply probabilistic abstract interpretation techniques and obtain a quantitative version of the classical strictness analysis of the λ-calculus.

The rest of this paper is organised as follows. We start by introducing the probabilistic λ-calculus and its semantics. In the next Section we review the main features of classical abstract interpretation and show how the framework may be applied to produce a strictness analysis for a first-order fragment of an applied λ-calculus. The paper [2] shows how these ideas can be extended to the higher-order case. We then describe probabilistic abstract interpretation [12, 13]. The final main section returns to the problem of strictness analysis in the probabilistic λ-calculus.

2 Probabilistic λ-calculus

We introduce a probabilistic version of the untyped λ-calculus, that is the formal theory of functions where each expression can be considered as both a function and a function argument. This theory can be seen as the simplest programming language with higher-order objects. An extension of this basic theory with probabilistic features is at the basis of the quantitative approach to static program analysis we will introduce in Section 5.

Other works have introduced probabilistic features into the λ-calculus, mainly motivated by the design and implementation of probabilistic languages (see [22] and [21] for recent examples). In this work, we aim to introduce a probabilistic semantics for the λ-calculus by extending the classical theory with a notion of *probabilistic term*. This corresponds essentially to a probability distribution over classical λ-terms. We recall that the set Λ of classical λ-terms, is the set of all expressions generated by the syntax:

$$M ::= x \;\Big|\; \lambda x.M \;\Big|\; MN.$$

Formally, we define the class, Λ_P, of probabilistic λ-terms as follows. Let x range over a set Var of variables. Then the class of probabilistic λ-expressions (ranged over by P) conforms to the syntax:

$$P ::= x \;\Big|\; \lambda x.P \;\Big|\; PP' \;\Big|\; \bigoplus_{i=1}^{n} p_i : P_i.$$

We require that for each $\bigoplus$-construct the associated p_i form a probability distribution, i.e. p_i are real numbers in the interval $[0, 1]$ and they sum up to one[1] (if the sum is less or equal to one we will call it a sub-probability distribution).

[1]Alternatively, one could allow for any $p_i \in [0, \infty)$ and statically normalise all p_i to $\tilde{p}_i = p_i / \sum_j p_j$.

Danos and Harmer [8] show that most forms of probabilistic behaviour can be encoded using a coin flip, i.e. that a binary choice — which we also denote by $P_1 \oplus_p P_2$ instead of $(1-p) : P_1 \oplus p : P_2$ — is sufficient. This minimalist programme however introduces a number of technical complications (such as distributivity in a nested binary sum) which tend to obfuscate the basic structures. We therefore opted for an n-ary probabilistic choice.

Example 1 Using the usual representation of integers as Church numerals a simple example of a term in Λ_p is given by

$$(\frac{2}{3} : (\lambda x.x) \oplus \frac{1}{3} : (\lambda x.x + x))3$$

or equivalently

$$(\lambda x.x \oplus_{\frac{1}{3}} \lambda x.(x + x))3$$

Another probabilistic λ-term, which we will use as a running example, is

$$((\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x))(\perp \oplus_{\frac{3}{4}} 42)$$

where $\perp$ is a constant representing an undefined term.

Clearly, every classical λ-term can also be seen as a probabilistic λ-term. This can be interpreted as a probability distribution on Λ, i.e. a map $\Lambda \to [0, 1]$ which assigns 1 to term M and 0 to all the other terms.

Although the recursive definition of P makes it not immediately evident, we can give a similar interpretation to every probabilistic λ-term. In fact, we will show how we can define for each $P \in \Lambda_p$ a probability distribution on Λ corresponding to a vector $\overline{P}$ in the vector space $\mathcal{V}(\Lambda)$.

The *vector space* $\mathcal{V}(X, \mathbb{W})$ over a set X is the space of formal (potentially infinite) linear combinations of elements in X with coefficients in some field $\mathbb{W}$ (e.g. $\mathbb{W} = \mathbb{R}$), i.e.

$$\mathcal{V}(X, \mathbb{W}) = \left\{ \sum c_x \overline{x} \mid c_x \in \mathbb{W},\ x \in X \right\}.$$

Thus, we can interpret a probability distribution on Λ as a vector in $\mathcal{V}(\Lambda) = \mathcal{V}(\Lambda, \mathbb{R})$. Classical terms $M \in \Lambda$ correspond to the basic vectors of this space, i.e. vectors with $c_M = 1$ and $c_N = 0$ for all $N \in \Lambda$, $N \neq M$. We will denote such vectors by $\overline{M}$. Any term $P \in \Lambda_p$ corresponds to a linear combination of basic vectors which we will denote by

$$\overline{P} = \sum_{i=1}^{n} p_i \cdot \overline{M_i}.$$

It will be convenient to assume the existence of a (Gödel) enumeration of all classical λ-terms, i.e. a function: $\sharp. : \Lambda \to \mathbb{N}$ which is bijective, and its inverse function $\flat. : \mathbb{N} \to \Lambda$.

Then we can denote a probability distribution over the set Λ of classical λ-terms by $\sum_i p_i \cdot \overline{\flat(i)}$ (or, alternatively as a vector of real numbers $(p_i)_i$, or as a set of pairs $\{\langle \flat(i), p_i \rangle\}_i$), where $p_i \in [0, 1]$ for all i and $\sum_i p_i = 1$; p_i is intended to represent the probability associated to term $\flat(i) \in \Lambda$. It will also be convenient to introduce the following operations on vector distributions on Λ:

$$\lambda x. \left(\sum_i p_i \cdot \overline{M_i} \right) = \sum_i p_i \cdot \overline{(\lambda x. M_i)}$$

and

$$\left(\sum_i p_i \cdot \overline{M_i} \right) \otimes \left(\sum_j q_j \cdot \overline{N_j} \right) = \sum_{i,j} (p_i q_j \cdot \overline{M_i N_j})$$

Definition 2 The flattened version $\overline{P}$ of a probabilistic λ-term $P \in \Lambda_p$ is defined by:

$$\begin{array}{rcl} P = x & \mapsto & \overline{P} = \overline{x} \\ P = \lambda x. P' & \mapsto & \overline{P} = \lambda x. \overline{P'} \\ P = P_1 P_2 & \mapsto & \overline{P} = \overline{P_1} \otimes \overline{P_2} \\ P = \bigoplus_{i=1}^{n} p_i : P_i & \mapsto & \overline{P} = \sum_{i=1}^{n} p_i \cdot \overline{P_i} \end{array}$$

Proposition 3 For all $P \in \Lambda_p$, $\overline{P}$ is a probability distribution on Λ, i.e. $\overline{P} = \sum_i p_i \cdot \overline{M_i}$, $M_i \in \Lambda$ for all i, and $\sum_i p_i = 1$.

Proof By structural induction and the fact the the two operations λx and $\otimes$ transform distributions on Λ into distributions on Λ. $\square$

Example 4 The flattened version of the second probabilistic λ-term in Example 1 is constructed in the following way:

$$\begin{array}{rl} & \overline{((\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x))\ (\bot \oplus_{\frac{3}{4}} 42)} \mapsto \\ \mapsto & \overline{((\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x))} \otimes \overline{(\bot \oplus_{\frac{3}{4}} 42)} \mapsto \\ \mapsto & (\frac{1}{2}\overline{(\lambda x.0)} + \frac{1}{2}\overline{(\lambda x.x)}) \otimes (\frac{1}{4}\overline{\bot} + \frac{3}{4}\overline{42}) \mapsto \\ \mapsto & \frac{1}{2}\frac{1}{4}\overline{(\lambda x.0)\bot} + \frac{1}{2}\frac{1}{4}\overline{(\lambda x.x)\bot} + \frac{1}{2}\frac{3}{4}\overline{(\lambda x.0)42} + \frac{1}{2}\frac{3}{4}\overline{(\lambda x.x)42}) = \\ = & \frac{1}{8}\overline{(\lambda x.0)\bot} + \frac{1}{8}\overline{(\lambda x.x)\bot} + \frac{3}{8}\overline{(\lambda x.0)42} + \frac{3}{8}\overline{(\lambda x.x)42} \end{array}$$

We also see that the sum of all probabilities add up to one, i.e. we have indeed a probability distribution over classical λ-terms.

2.1 Probabilistic β-Reduction

The notion of *β-reduction* expresses the computational aspect of the classical λ-calculus. Informally, it consists in the evaluation of the β-normal form of a term by means of the application of the β-rule

$$(\lambda x.M)N \to_\beta M[x := N].$$

If this process is identified with function application in programming languages, then β-reduction is actually the execution of λ-term programs.

For our probabilistic λ-calculus, we need a notion of *probabilistic β-reduction* which is able to take into account the different alternatives of a probabilistic choice represented by a probabilistic λ-term. We will introduce two different versions of an operational semantics for our probabilistic λ-calculus: the first one uses distributions over classical λ-terms as operational states and a non-probabilistic transition relation $\Rightarrow_\beta$ on them which can be seen as a 'lifting' of the classical β-reduction $\rightarrow_\beta$; the second one is based on a probabilistic transition relation $\rightarrow^p$ on probabilistic λ-terms defined in the SOS style typically adopted in the semantics of programming languages. We will then introduce in Section 2.2 a linear representation of the latter semantics inspired by the theory of stochastic processes and Markov chains. This will represent the base of our probabilistic analysis.

All the three semantics give equivalent meanings to probabilistic λ-terms although using different representations which gradually lift the classical semantics of the λ-calculus to a non-standard semantics based on linear operators via an operational semantics as a kind of halfway house semantics.

2.1.1 Distribution-Based Semantics

Formally, we introduce the probabilistic one-step β-reduction $\Rightarrow_\beta$ for probabilistic λ-terms as an extension of the classical one-step β-reduction relation $\rightarrow_\beta$ on the classical terms Λ, which we will briefly recall in the following [1, 15].

Definition 5 The binary relation $\rightarrow_\beta$ on Λ is defined by:

$$(\lambda x.M)N \rightarrow_\beta M[x := N]$$

$$\frac{M \rightarrow_\beta N}{MZ \rightarrow_\beta NZ}$$

$$\frac{M \rightarrow_\beta N}{ZM \rightarrow_\beta ZN}$$

$$\frac{M \rightarrow_\beta N}{\lambda x.M \rightarrow_\beta \lambda x.N}$$

We can now define a β-reduction relation on probabilistic λ-terms as a lifting of the classical one to distributions over classical λ-terms.

Definition 6 The binary relation $\Rightarrow_\beta$ on $\mathcal{V}(\Lambda)$ is defined by:

$$\frac{M \rightarrow_\beta N}{\overline{M} \Rightarrow_\beta \overline{N}}$$

$$\frac{M \not\rightarrow_\beta}{\overline{M} \Rightarrow_\beta \overline{M}}$$

$$\frac{\overline{M_i} \Rightarrow_\beta \overline{N_i}}{\overline{\bigoplus_i p_i : M_i} \Rightarrow_\beta \overline{\bigoplus_i p_i : N_i}}$$

Classically, we say that term M β-reduces to term N if (M, N) is in the reflexive, transitive closure, $\rightarrow^*_\beta$, of $\rightarrow_\beta$. Analogously, we define the relation $\Rightarrow^*_\beta$ on $\mathcal{V}(\Lambda)$ as the reflexive, transitive closure of $\Rightarrow_\beta$.

Definition 7 The binary relation $\Rightarrow^*_\beta$ on $\mathcal{V}(\Lambda)$ is defined by:

$$\overline{P} \Rightarrow^*_\beta \overline{P}$$

$$\frac{\overline{P} \Rightarrow_\beta \overline{P'}}{\overline{P} \Rightarrow^*_\beta \overline{P'}}$$

$$\frac{\overline{P_1} \Rightarrow_\beta \overline{P_2}, \overline{P_2} \Rightarrow_\beta \overline{P_3}}{\overline{P_1} \Rightarrow^*_\beta \overline{P_3}}$$

The computational idea behind the classical β-reduction is to calculate for a term M its *β-normal form* (if it exists), that is a term N with no sub-expressions of the form $(\lambda x.N')N''$. In fact, if M has a β-normal form N, then this is unique and $M \rightarrow^*_\beta N$ (by the Church-Rosser property [1]). Definition 7 implies that for classical terms we also have that $\overline{M} \Rightarrow^*_\beta \overline{N}$ holds.

The probabilistic β-reduction allows us to compute β-normal forms for classical terms together with information on the probability of actually achieving them by β-reduction.

Definition 8 A probabilistic λ-term P is a probabilistic β-normal form if its flattened version is a probability distribution on λ-terms which are β-normal forms, i.e. $\overline{P} = \sum_i p_i \cdot \overline{N_i}$ and $N_i \in \Lambda$ are β-normal forms for all i.

Intuitively, a probabilistic λ-term has a probabilistic β-normal form if and only if all its possible "branches" reduce to a classical β-normal form (with some probability). We can show that the uniqueness property of classical β-normal forms still holds for probabilistic β-normal forms.

Proposition 9 For a probabilistic λ-term P there is at most one probabilistic term P_β such that P_β is a probabilistic β-normal form and $\overline{P} \Rightarrow^*_\beta \overline{P_\beta}$.

Proof Consider the flattened form $\overline{P} = \sum_{i=1}^n p_i \cdot \overline{M_i}$ of P with $M_i \in \Lambda$. Then M_i either has a β-normal form N_i, i.e. $M_i \rightarrow^*_\beta N_i$, or diverges in $\rightarrow^*_\beta$. Therefore, $\overline{P} \Rightarrow^*_\beta \sum_i p_i \cdot \overline{N_i} = \overline{P_\beta}$ where the sum is over all i such that $M_i \rightarrow^*_\beta N_i \not\rightarrow_\beta$. By Corollary 3.1.13 in [1] the N_i's are unique (if they exist); thus P_β is also unique. $\square$

Example 10 Starting from the flattened version of our running example (see Example 4) we get the following reductions

$$\begin{aligned}
&\frac{1}{8}\overline{(\lambda x.0)\bot} + \frac{1}{8}\overline{(\lambda x.x)\bot} + \frac{3}{8}\overline{(\lambda x.0)42} + \frac{3}{8}\overline{(\lambda x.x)42} \Rightarrow_\beta \\
\Rightarrow_\beta \quad &\frac{1}{8}\overline{0} + \frac{1}{8}\overline{(\lambda x.x)\bot} + \frac{3}{8}\overline{(\lambda x.0)42} + \frac{3}{8}\overline{(\lambda x.x)42} \Rightarrow_\beta \\
\Rightarrow_\beta \quad &\frac{1}{8}\overline{0} + \frac{1}{8}\overline{\bot} + \frac{3}{8}\overline{(\lambda x.0)42} + \frac{3}{8}\overline{(\lambda x.x)42} \Rightarrow_\beta
\end{aligned}$$

$$\begin{array}{ll}
\Rightarrow_\beta & \frac{1}{8}\overline{0} + \frac{1}{8}\overline{\perp} + \frac{3}{8}\overline{0} + \frac{3}{8}\overline{(\lambda x.x)42} \Rightarrow_\beta \\
\Rightarrow_\beta & \frac{1}{8}\overline{0} + \frac{1}{8}\overline{\perp} + \frac{3}{8}\overline{0} + \frac{3}{8}\overline{42} = \\
= & \frac{1}{2}\overline{0} + \frac{1}{8}\overline{\perp} + \frac{3}{8}\overline{42}.
\end{array}$$

2.1.2 Term-Based Semantics

A more programming language oriented definition of the reduction of probabilistic λ-terms is based on the definition of a probabilistic transition relation $\rightarrow^p \subseteq \Lambda_p \times [0,1] \times \Lambda_p$ on probabilistic λ-terms.

Definition 11 The probabilistic transition relation $\rightarrow^p \subseteq \Lambda_p \times [0,1] \times \Lambda_p$ is defined inductively by the following rules:

$$(\mathbf{app}_1) \qquad \frac{P \rightarrow^p P' \quad Q \rightarrow^q Q'}{\begin{array}{c} PQ \rightarrow^{rp} P'Q \\ PQ \rightarrow^{(1-r)q} PQ' \end{array}}$$

$$(\mathbf{app}_2) \qquad \frac{P \rightarrow^p P' \quad Q \not\rightarrow^q}{PQ \rightarrow^p P'Q} \quad \frac{P \not\rightarrow^p \quad Q \rightarrow^q Q'}{PQ \rightarrow^q PQ'}$$

$$(\beta) \qquad (\lambda x.P)Q \rightarrow^1 P[x := Q]$$

$$(\delta) \qquad (\textstyle\bigoplus_i p_i : P_i) \rightarrow^{p_i} P_i$$

where $P, Q \in \Lambda_p$ are probabilistic terms and $r \in [0,1]$ is a parameter representing a probabilistic scheduler.

The first rule in the above definition states that whenever it is possible to use a left- or right-most scheduling this will be resolved by a probabilistic choice between them. The probabilities of choosing one of these will be r and $1-r$ respectively. The second rule covers the case when only one strategy is possible which will then be applied with probability one. This transition relation $\rightarrow^p$ is parametric in r but we will usually omit this dependency in the notation (cf. Proposition 13).

We write $P_1 \rightarrow^p P_2$ to mean that P_1 is transformed into P_2 with probability p in one step, i.e. by application of one of the above transition rules. We denote by $P \not\rightarrow$ the fact that there are no probabilistic transitions from P. If the reduction process is finite, then it terminates with a term which is a λ-term or a constant.

Example 12 For our running example, using the short-hand notation

$$\begin{array}{rcl}
P & \equiv & (\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x) \\
Q & \equiv & \perp \oplus_{\frac{3}{4}} 42
\end{array}$$

we get the derivations depicted in Figure 1.

The transitive closure $\rightarrow^{*p}$ of $\rightarrow^p$ is defined in the usual way, but with some care for the interpretation of the label p; this is calculated by taking the product of

$$\begin{tikzcd}
 & & & & & PQ \arrow[dllll, "\frac{1}{2}r"'] \arrow[dl, "\frac{1}{2}r"'] \arrow[dr, "\frac{1}{4}(1-r)"] \arrow[drrr, "\frac{3}{4}(1-r)"] & & & & \\
 & (\lambda x.0)Q \arrow[ddl, "r"'] \arrow[d, "\frac{1}{4}(1-r)"] \arrow[dr, "\frac{3}{4}(1-r)"] & & & (\lambda x.x)Q \arrow[ddl, "r"'] \arrow[d, "\frac{1}{4}(1-r)"] \arrow[dr, "\frac{3}{4}(1-r)"] & & P\bot \arrow[d, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & & P42 \arrow[d, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & \\
 & (\lambda x.0)\bot \arrow[d, "1"'] & (\lambda x.0)42 \arrow[d, "1"'] & & (\lambda x.x)\bot \arrow[d, "1"'] & (\lambda x.x)42 \arrow[d, "1"'] & (\lambda x.0)\bot \arrow[d, "1"'] & (\lambda x.x)\bot \arrow[d, "1"'] & (\lambda x.0)42 \arrow[d, "1"'] & (\lambda x.x)42 \arrow[d, "1"'] \\
0 & 0 & 0 & Q \arrow[d, "\frac{1}{4}"'] \arrow[dr, "\frac{3}{4}"] & \bot & 42 & 0 & \bot & 0 & 42 \\
 & & & \bot & 42 & & & & &
\end{tikzcd}$$

Figure 1: Derivations of $((\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x))(\bot \oplus_{\frac{3}{4}} 42)$

probabilities along the paths and then summing up the probabilities along different paths which reach the same probabilistic λ-term (see e.g. [10]).

In the example above there are three paths leading from PQ to $\bot$ with probability $\frac{1}{2}r \cdot r \cdot \frac{1}{4} = \frac{1}{8}r^2$, $\frac{1}{2}r \cdot \frac{1}{4}(1-r) \cdot 1 = \frac{1}{8}r(1-r)$, and $\frac{1}{4}(1-r) \cdot \frac{1}{2} \cdot 1 = \frac{1}{8}(1-r)$, respectively. Thus we have $PQ \rightarrow^{*p} \bot$ with $p = \frac{1}{8}r^2 + \frac{1}{8}(r-r^2) + \frac{1}{8}(1-r) = \frac{1}{8}r + \frac{1}{8}(1-r) = \frac{1}{8}$. Note that the final probability of reaching $\bot$ does not depend on r. This is no coincidence as shown by the following proposition.

Proposition 13 (Independence of the scheduling) For all parameters r_1 and r_2 and all probabilistic λ-terms $P \in \Lambda_p$ with $P \rightarrow^{*p(r_1)} N \not\rightarrow$ and $P \rightarrow^{*p(r_2)} N \not\rightarrow$ we have $p(r_1) = p(r_2)$.

Proof (sketch) The probabilistic reduction relation $\rightarrow^{p(r)}$ is independent of the value r when it comes to probabilistic choices, i.e. terms of the form $\bigoplus_i p_i : P_i$, cf rule (δ). The dependency on r is thus essentially a problem of classical λ-terms which get reduced according to the probabilistic relation $\rightarrow^{p(r)}$ rather than by classical β-reduction.

If we therefore consider the *probabilistic* reduction of *classical* terms M we can see that the only difference between classical and probabilistic reduction is the annotation or decoration of transitions with probabilities: If there is a classical reduction of a term $M \rightarrow_\beta M'$ then there is also a probabilistic reduction $M \rightarrow^{p(r)} M'$ with $p(r) > 0$. Furthermore, the sum of all probabilities associated with a term M sum up to 1. In the classical case we know, by the Church-Rosser property, that terms with normal form have a unique normal form, i.e. all derivations of classical terms are confluent. If we decorate the classical derivations with probabilities according to the rules for $\rightarrow^{p(r)}$ we see that the probabilities of all paths from M to its normal form must sum up to one.

In particular, every time an application term gets reduced there is a left-most path with probability r and a right-most path with probability $1 - r$. This "splitting" of paths can be repeated recursively, but we will always end up with the same normal form with combined probability 1. For example, a classical reduction and its probabilistic counterpart will be of the following forms:

$$\begin{tikzcd}
 & M \arrow[dl] \arrow[dr] & & \\
M' \arrow[dr] & & M'' \arrow[dl] \arrow[dr] & \\
 & M''' \arrow[dr] & & M'''' \arrow[dl] \\
 & & N &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & M \arrow[dl, "r"'] \arrow[dr, "(1-r)"] & & \\
M' \arrow[dr, "1"'] & & M'' \arrow[dl, "r"'] \arrow[dr, "(1-r)"] & \\
 & M''' \arrow[dr, "1"'] & & M'''' \arrow[dl, "1"] \\
 & & N &
\end{tikzcd}$$

We can lift this argument to general probabilistic λ-terms, i.e. terms which contain (possibly) nested choices $\bigoplus$, by structural induction. This exploits the fact that classical terms reduce probabilistically only to classical terms. The probabilistic reduction of a term of the form $\bigoplus_i p_i : M_i$ (with M_i classical terms M_i) can be therefore constructed out of the probabilistic reduction of the M_i's and their probabilities p_i. $\square$

The probabilities of reaching β-normal forms is also independent from dynamically changing r's, i.e. if the choice between left- and right-most reduction is made with different probabilities in each step. This independence of the normal forms and their probabilities from the application (probability) of different reduction strategies can be seen as a kind of probabilistic analogue of the classical Church-Rosser property.

The β-reduction of a classical λ-terms (if it terminates) leads to a unique result, namely its β-normal form (by the Church-Rosser theorem). The transition relation $\rightarrow^{*p}$ for probabilistic λ-terms leads in general to several final terms. The notion of observables will collect all these possible results together with their probabilities. Proposition 13 guarantees that the probability of reaching a final result is independent of the parameter r of the transition relation. Therefore, we can define a notion of observables as follows.

Definition 14 We define the observables of a probabilistic λ-term $P \in \Lambda_p$ as

$$\mathcal{O}(P) = \{\langle M_i, p_i\rangle \mid P \rightarrow^{*p_i} M_i \ \wedge \ M_i \not\rightarrow^p\}$$

Note that the definition of $\rightarrow^{*p}$ (and in particular of the label p in this relation) guarantees that

$$\sum\{ \ p_i \mid \ \langle M_i, p_i\rangle \in \mathcal{O}(P)\} \leq 1,$$

that is $\mathcal{O}(P) \in \mathcal{V}(\Lambda)$ is a sub-probability distribution for every $P \in \Lambda_p$. If all "branches" M_i of P terminate, then $\mathcal{O}(P)$ defines a probability distribution.

Example 15 The observables of the term P in Example 12 are

$$\mathcal{O}(P) = \{\langle 0, 1/2\rangle, \langle 42, 3/8\rangle, \langle \bot, 1/8\rangle\}.$$

As the observables are independent of the reduction strategy we can fix one by setting $r = 1$ (or $r = 0$) thus enforcing a left-most (right-most) scheduling strategy.

Example 16 The possible transitions of the probabilistic λ-term $P \equiv ((\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x))(\perp \oplus_{\frac{3}{4}} 42)$ with $r = 1$, i.e. using left-most scheduling, can be depicted as follows:

$$\begin{tikzcd}
 & ((\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x))(\perp \oplus_{\frac{3}{4}} 42) \arrow[dl, "\frac{1}{2}"'] \arrow[dr, "\frac{1}{2}"] & \\
(\lambda x.0)(\perp \oplus_{\frac{3}{4}} 42) \arrow[d, "1"'] & & (\lambda x.x)(\perp \oplus_{\frac{3}{4}} 42) \arrow[d, "1"] \\
0 & & \perp \oplus_{\frac{3}{4}} 42 \arrow[dl, "\frac{1}{4}"'] \arrow[d, "\frac{3}{4}"] \\
 & \perp & 42
\end{tikzcd}$$

The original term is of the form PQ, thus we apply the rule ($\mathbf{app}_1$) with $r = 1$, i.e. reduce the first term P first. This is of the form $P_1 \oplus_{\frac{1}{2}} P_2$, so we have to apply the (δ) rule and choose P_1 and P_2 with their respective probability $\frac{1}{2}$. After that we end up with two terms which are both of the form $(\lambda x.P)P'$; thus we can apply with probability 1 the (β) rule which gives us either the constant 0 or the choice $\perp \oplus_{\frac{3}{4}} 42$. The latter then reduces via the (δ) rule either to $\perp$ or 42 with the indicated probabilities.

2.1.3 Correspondence

The probabilistic β-reduction $\Rightarrow_\beta$ introduced in Section 2.1.1 can be seen as a "parallelised" version of the probabilistic transition relation $\rightarrow^p$ introduced in Section 2.1.2, where all sub-terms of a probabilistic term P are reduced simultaneously. Each computational path constructed in the probabilistic transition system for P leads to pairs $\langle M_i, p_i\rangle$ of a term $M_i \in \Lambda$ and a probability p_i which is calculated via the rules in the transition system. On the other hand, the flattening rules in Definition 2 calculate the probabilities p_i beforehand by transforming P into its flattened version $\overline{P}$ which is then reduced by $\Rightarrow_\beta$. Intuitively, we expect that the probabilities p_i be the same independently of whether we use one or the other procedure, i.e. we expect that the probabilistic observables $\mathcal{O}(P)$ defined via the probabilistic transition system correspond to the probabilistic β-normal forms that can be calculated via the probabilistic β- reduction.

The following proposition formally shows this correspondence.

Proposition 17 Let $P \in \Lambda_p$ be a probabilistic λ-term which admits a probabilistic β-normal form. Then we have:

$$\mathcal{O}(P) = \overline{P_\beta}$$

Proof We have to show that for all $P \in \Lambda_p$, $\mathcal{O}(P) = \{\langle N_i, p_i\rangle\}_i = \sum_i p_i \cdot \overline{N_i} = \overline{P_\beta}$, with $N_i \in \Lambda$ β-normal forms for all i.

($P = x$) We have that $\mathcal{O}(P) = \{\langle x, 1\rangle\} = \overline{x} = \overline{P_\beta}$.

($P = \lambda x.P'$) Let $\mathcal{O}(P) = \{\langle N_i, p_i\rangle\}_i$. By Proposition 13 we can use a right-most strategy without loss of generality, so that all derivations for P are of the form

$$\lambda x.P' \to^{p_i} \lambda x.N_i' \to^{*1} N_i \not\to^p,$$

with $\langle N_i', p_i\rangle \in \mathcal{O}(P')$, and $N_i' \in \Lambda$ for all i. By the induction hypothesis, we have that $\mathcal{O}(P') = \overline{P_\beta'} = \sum_i p_i \cdot \overline{N_i'}$. This means that

$$\overline{P'} \Rightarrow_\beta^* \sum_i p_i \cdot \overline{N_i'},$$

and so

$$\overline{P} = \lambda x.\overline{P'} \Rightarrow_\beta^* \lambda x.\left(\sum_i p_i \cdot \overline{N_i'}\right) = \sum_i p_i \cdot \overline{\lambda x.N_i'}.$$

We now observe that for all i, $\lambda x.N_i' \to^{*1} N_i$ coincides with the classical β-reduction $\lambda x.N_i' \to_\beta^* N_i$; thus by applying the first and third rule in Definition 6 and by the transitivity rule of Definition 7, we get

$$\sum_i p_i \cdot \overline{\lambda x.N_i'} \Rightarrow_\beta^* \sum_i p_i \cdot \overline{N_i}.$$

Since the N_i's are all β-normal forms, we then have that

$$\sum_{i=1}^n p_i \cdot \overline{N_i} = \overline{P_\beta}.$$

($P = P_1P_2$) Let $\mathcal{O}(P_1) = \{\langle M_i^1, p_i^1\rangle \mid i = 1, \ldots, m_1\}$ and $\mathcal{O}(P_2) = \{\langle M_j^2, p_j^2\rangle \mid j = 1, \ldots, m_2\}$. Then all derivations for P are of the form

$$P \to^{*p_i^1 p_j^2} M_i^1 M_j^2 \to^{*1} N_{ij} \not\to^p .$$

By the induction hypothesis we have that

$$\overline{P_1} \Rightarrow_\beta^* \overline{(P_1)_\beta} = \sum_{i=1}^{m_1} p_i^1 \cdot \overline{M_i^1}, \text{ and}$$

$$\overline{P_2} \Rightarrow_\beta^* \overline{(P_2)_\beta} = \sum_{j=1}^{m_2} p_j^2 \cdot \overline{M_j^2}.$$

Thus,

$$\begin{aligned}\overline{P} = \overline{P_1} \otimes \overline{P_2} \quad &\Rightarrow_\beta^* \quad \left(\sum_{i=1}^{m_1} p_i^1 \cdot \overline{M_i^1}\right) \otimes \left(\sum_{j=1}^{m_2} p_j^2 \cdot \overline{M_j^2}\right) \\ &= \sum_{i,j} p_i^1 p_j^2 \cdot \overline{M_i^1 M_j^2}.\end{aligned}$$

Now we have to show that $P_\beta = \sum_{ij} p^1_i p^2_j \cdot \overline{N_{ij}}$. This follows from observing again that the derivation $M^1_i M^2_j \rightarrow^{*1} N_{ij}$ corresponds to a classical β-reduction; we can then apply Definition 6 and Definition 7 to get

$$\sum_{i,j} p^1_i p^2_j \cdot \overline{M^1_i M^2_j} \Rightarrow^*_\beta \sum_{i,j} p^1_i p^2_j \cdot \overline{N_{ij}}.$$

Since $N_{ij} \in \Lambda$ are all β-normal forms, we conclude that $\overline{P_\beta} = \sum_{ij} p^1_i p^2_j \cdot \overline{N_{ij}}$.

($P = \bigoplus p_i : P_i$) Let $\mathcal{O}(P_i) = \{\langle N^i_j, q^i_j \rangle \mid j = 1, \ldots, m_i\}$. Then all derivations form P are of the form

$$P \rightarrow^{p_i} P_i \rightarrow^{*q^i_j} N^i_j \not\rightarrow^p,$$

and so $\mathcal{O}(P) = \{\langle N^i_j, p_i q^i_j \rangle \mid j = 1, \ldots, m_i\}_i$.

By the induction hypothesis we have that for all i

$$\overline{P_i} \Rightarrow^*_\beta \overline{(P_i)_\beta} = \sum_{j=1}^{m_i} q^i_j \cdot \overline{N^i_j}.$$

Thus we have

$$\overline{P} = \sum_i p_i \cdot \overline{P_i} \Rightarrow^*_\beta \sum_i \sum_{j=1}^{m_i} p_i q^i_j \cdot \overline{N^i_j}$$

Since $N^i_j \in \Lambda$ are all in β-normal form, we have that $\overline{P_\beta} = \sum_i \sum_{j=1}^{m_i} p_i q^i_j \cdot \overline{N^i_j} = \mathcal{O}(P)$.

$\square$

2.2 Linear Representation

In this section we introduce a linear operator on the vector space $\mathcal{V}(\Lambda)$ as a representation of the the semantics $\mathcal{O}(P)$ for a probabilistic λ-term P which is more suitable as a base for the probabilistic analysis we will present later. For this purpose we will use the distribution-based semantics introduced in Section 2.1.1 as an intermediate step, and define a linear operator representing the one-step reduction relation $\Rightarrow_\beta$. We first define a linear operator for the β-reduction of the classical λ-calculus, i.e. for terms $M, N \in \Lambda$,

$$\mathbf{T} : \mathcal{V}(\Lambda) \rightarrow \mathcal{V}(\Lambda)$$

This is defined by

$$(\mathbf{T})_{MN} = \begin{cases} 1 & \text{if } M \rightarrow_\beta N \\ 1 & \text{if } M \not\rightarrow_\beta \;\wedge\; M = N \\ 0 & \text{otherwise} \end{cases}$$

and then the restriction for each classical λ-term $M \in \Lambda$:

$$\mathbf{T}_M = \pi_{\mathcal{R}(M)} \mathbf{T} \pi_{\mathcal{R}(M)},$$

where $\pi_{\mathcal{R}(M)}$ is the projection onto the reachable terms $\mathcal{R}(M) = \{M' \in \Lambda \mid M \rightarrow^*_\beta M'\}$, i.e. the sub-vector space $\mathcal{V}(\mathcal{R}(M))$ of $\mathcal{V}(\Lambda)$.

For a probabilistic term $P \in \Lambda_p$ we consider its linear representation, i.e. its flattened form $\overline{P} \in \mathcal{V}(\Lambda)$:

$$\overline{P} = \sum_i p_i \cdot \overline{M_i}$$

and the restriction of $\mathbf{T}$ to the union of reachable terms, i.e.

$$\mathbf{T}_P = \pi_{\mathcal{R}(P)} \mathbf{T} \pi_{\mathcal{R}(P)}$$

with $\mathcal{R}(P) = \bigcup_i \mathcal{R}(M_i)$. We can then express the semantics of P via the iteration of the operator $\mathbf{T}_P$.

Definition 18 For a probabilistic λ-term $P \in \Lambda_p$ we define its semantics as:

$$[\![P]\!] = \overline{P} \cdot \lim_{i \to \infty} \mathbf{T}_P^i$$

Note that for probabilistic λ-terms $P \in \Lambda_p$ which admit a probabilistic β-normal form the set of reachable states $\mathcal{R}(P)$ is finite as each of its (finitely many) "branches" M_i will reduce to a normal form (in finitely many steps). The vector space $\mathcal{V}(\mathcal{R}(P))$, on which $\mathbf{T}_P$ acts, is therefore finite dimensional. This guarantees that the limit $\lim_{i\to\infty} \mathbf{T}_P^i$ exists and is well defined as there is only one (standard) topology on finite dimensional vector spaces.

Proposition 19 For every probabilistic λ-term $P \in \Lambda_p$ which admits a probabilistic β-normal form we have:

$$[\![P]\!] = \mathcal{O}(P).$$

Proof By the construction of the operator $\mathbf{T}_P$ we have that $[\![P]\!] = P_\beta$. Then by the correspondence shown in Proposition 17 we conclude that $[\![P]\!] = \mathcal{O}(P)$. □

Example 20 Consider again the probabilistic λ-term from Example 16:

$$P \equiv ((\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x))(\bot \oplus_{\frac{3}{4}} 42)$$

After the flattening procedure we get the probability distribution

$$\begin{aligned} \overline{P} &= (\frac{1}{2}\overline{\lambda x.0} + \frac{1}{2}\overline{\lambda x.x}) \otimes (\frac{1}{4}\overline{\bot} + \frac{3}{4}\overline{42}) \\ &= \frac{1}{8}\overline{(\lambda x.0)\bot} + \frac{3}{8}\overline{(\lambda x.0)42} + \frac{1}{8}\overline{(\lambda x.x)\bot} + \frac{3}{8}\overline{(\lambda x.x)42} \end{aligned}$$

An enumeration of the reachable terms is as follows:

1 $(\lambda x.0)\bot$	**3** $(\lambda x.x)\bot$	**5** 0
2 $(\lambda x.0)42$	**4** $(\lambda x.x)42$	**6** $\bot$
		7 42.

According to this enumeration we can represent $\overline{P}$ by the vector

$$\overline{P} = \begin{pmatrix} \frac{1}{8} & \frac{3}{8} & \frac{1}{8} & \frac{3}{8} & 0 & 0 & 0 \end{pmatrix},$$

and we have:

$$\lim_{i\to\infty} \mathbf{T}_P^i = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We can then calculate the linear semantics of P:

$$[\![P]\!] = \overline{P} \cdot \lim_{i\to\infty} \mathbf{T}_P^i = \lim_{i\to\infty} \overline{P} \cdot \mathbf{T}_P^i = \begin{pmatrix} 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{8} & \frac{3}{8} \end{pmatrix},$$

which coincides with the observables $\mathcal{O}(P)$ in Example 15.

3 Classical Abstract Interpretation

Program analysis aims to determine some property of a program without running it. A classical example is the *Reaching Definitions* analysis which determines, for each node in a flowchart, which definitions (assignments) reach it [19]. The results of this analysis might be used to perform a constant folding transformation of the program. Such transformations should be semantics preserving and it is therefore important that the analysis gives correct information about the program. Often the properties that we are interested in are undecidable and so correctness is replaced by some approximation notion.

We start by sketching the classical approach to semantics-based program analysis: *abstract interpretation* [4, 19]. The *semantics* of a program f identifies some set V of values and specifies how the program transforms one value v_1 to another v_2.

In a similar way, a *program analysis* identifies the set L of properties and specifies how a program f transforms one property l_1 to another l_2: $f \vdash l_1 \rhd l_2$.

As we have seen, every program analysis should be correct with respect to the semantics. For first-order program analyses, i.e. those that abstract properties of values, this is established by directly relating properties to values using a *correctness relation*: $R : V \times L \rightarrow \{\mathit{true}, \mathit{false}\}$. The intention is that $v \; R \; l$ formalises our claim that the value v is described by the property l.

The correctness relation is often achieved via a *Galois connection*: (V, α, γ, L) is a Galois connection between the complete lattices $(V, \sqsubseteq)$ and $(L, \sqsubseteq)$ if and only if $\alpha : V \rightarrow L$ and $\gamma : L \rightarrow V$ are monotone functions that satisfy: $\gamma \circ \alpha \sqsupseteq \lambda v.v$ and $\alpha \circ \gamma \sqsubseteq \lambda l.l$.

Having defined a suitable "set" of properties we then define suitable interpretations of program operations. The framework of abstract interpretation guarantees that the analysis will be safe as long as we use an interpretation, F_{abs}, of each language operator, F, that satisfies: $F_{\text{abs}} \sqsupseteq \alpha \circ F \circ \gamma$.

Since interesting languages involve iteration or recursion we also have to construct efficient implementations; a generic solution to this problem is the theory of widening and narrowing [4].

3.1 Strictness Analysis

Strictness analysis [18, 2] aims to answer for some function *f*: Does $f\ \bot = \bot$? If the function has this property then it either uses its argument or is the bottom function. In either case, an affirmative answer would mean that arguments can be passed by value rather than using (the more costly) lazy evaluation. For illustration, we will restrict ourselves to a first-order applied λ-calculus with integers as the only data type.

We can construct a Galois connection $(\mathcal{P}_H(\mathbf{Z}_\bot), \alpha, \gamma, \mathbf{Two})$ where $\mathcal{P}_H$ is the *Hoare Powerdomain* construction on the integers $\mathbf{Z}$ and $\mathbf{Two}$ is $\{0,1\}$ ordered by $0 \sqsubseteq 1$. The elements of the Hoare Powerdomain in this case are just down-closed sets ordered by subset inclusion, so that every set contains $\bot$.

We define:

$$\alpha(Z) \quad = \quad \begin{cases} 0 & \text{if } Z = \{\bot\} \\ 1 & \text{otherwise} \end{cases} \qquad \gamma(S) \quad = \quad \begin{cases} \{\bot\} & \text{if } S = 0 \\ \mathbf{Z}_\bot & \text{if } S = 1 \end{cases}$$

We can construct the induced operations that correspond to the operations in this first-order applied λ-calculus:

Concrete operation	Induced operation
base type constants	1
if x then y else z	$x \sqcap (y \sqcup z)$
$x\ op\ y$	$x \sqcap y$

The conditional takes three arguments (*x, y, z*); the predicate must be defined and then the result is at least as defined as either of the branches. Thus the abstract interpretation of

$$(\lambda\ x.\mathit{if}\ x = 0\ \mathit{then}\ 15\ \mathit{else}\ 42)$$

is $(\lambda\ x.(x \sqcap 1) \sqcap (1 \sqcup 1)) \equiv \lambda x.x$. Since $(\lambda\ x.\ x)\ 0 = 0$, this tells us that our original function is strict. We now extend this approach to a probabilistic λ-calculus by applying techniques of *Probabilistic Abstract Interpretation* [12, 13]. This will allow us to solve one important problem that despite decades of research into strictness analysis methods based on abstract interpretation has remained open, namely the precise and formal characterisation of the information loss.

4 Probabilistic Abstract Interpretation

Probabilistic Abstract Interpretation (PAI) was introduced in [12, 13] as a method for approximating the semantics of probabilistic programs which re-formulates the classical theory of Abstract Interpretation [4] in a setting suitable for a quantitative program analysis. The PAI framework is based on the notion of *probabilistic domains* which we will identify with Hilbert spaces over some set representing the computational states and linear operators.

A *Hilbert space* $\mathcal{H}$ is a complex vector space together with an *inner product* $\langle .,. \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$ with (i) $\langle x, y \rangle = \overline{\langle y, x \rangle}$, (ii) $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$, (iii) $\langle x + z, y \rangle = \langle x, y \rangle + \langle z, y \rangle$, and (iv) $\langle x, x \rangle = 0 \iff x = o$ for all $\alpha \in \mathbb{C}$ and $x, y, z \in \mathcal{H}$ (where $\bar{c}$ denotes the complex conjugation in $\mathbb{C}$, and o is the null vector in $\mathcal{H}$) such that the topology induced by the *norm* $\|x\| = \sqrt{\langle x, x\rangle}$ is complete. It is easy to show that this is indeed a *vector norm* on $\mathcal{H}$, i.e. that for all $x, y \in \mathcal{H}$ and $\alpha \in \mathbb{C}$ the following holds:

(i) $\|x\| \geq 0$ **(iii)** $\|\alpha x\| = |\alpha|\|x\|$

(ii) $\|x\| = 0 \Leftrightarrow x = o$ **(iv)** $\|x + y\| \leq \|x\| + \|y\|$

All finite dimensional complex vector spaces are isomorphic to a n-dimensional complex Hilbert space $\mathbb{C}^n$, for some n. The treatment based on complex numbers is technically somewhat simpler than when based on real numbers as, by the fundamental theorem of algebra, polynomials of order n over the complex numbers have exactly n (not necessarily distinct) roots. For finite sets of computational states X we thus can take the Hilbert space $\mathcal{V}(X, \mathbb{C}) \cong \mathbb{C}^n$ with the standard inner product $\langle (c_i)_{i=1}^n, (d_i)_{i=1}^n \rangle = \sum_{i=1}^n c_i \cdot \overline{d_i}$.

In fact, for (complex) finite dimensional vector spaces the algebraic structure is essentially forcing a unique topological structure and the Hilbert space with its Euclidean norm $\|.\|_2$ is only one way to introduce this (metric) topology. We would obtain essentially the same topology by using the 1-norm $\|.\|_1$ (sum of absolute values) or the supremum norm $\|.\|_\infty$ (supremum of absolute values). In infinite dimensions, Hilbert spaces are only one possible, although very important, example of defining a topology.

In finite dimensions all linear maps between two Hilbert spaces, represented usually by matrices, are automatically continuous, i.e. preserve the topological structure. For infinite dimensional Hilbert spaces a linear map is continuous if and only if it is bounded; a linear map $\mathbf{T} : \mathcal{H}_1 \to \mathcal{H}_2$ is bounded if its *operator norm*

$$\|\mathbf{T}\| = \sup_{x \in \mathcal{H}_1} \frac{\|\mathbf{T}(x)\|}{\|x\|}.$$

is bounded, i.e. $\|\mathbf{T}\| < \infty$. This norm allows us to measure operators in a canonical way which will be important in quantifying the precision of an abstraction.

On a Hilbert space $\mathcal{H}$ we can define the so called *adjoint* $\mathbf{T}^*$ of an operator $\mathbf{T} : \mathcal{H} \to \mathcal{H}$ via the requirement: $\langle \mathbf{T}(x), y\rangle = \langle x, \mathbf{T}^*(y)\rangle$ for all elements $x, y \in \mathcal{H}$. One can show that the adjoint operator for bounded linear operators on a Hilbert space always exists and that it is unique. For finite dimensional operators with a matrix representation $\mathbf{M} \in \mathcal{M}$ the matrix representation of the adjoint is $\mathbf{M}^* = (\overline{\mathbf{M}})^t$, i.e. the transpose complex conjugate matrix.

Probabilistic Abstract Interpretation is defined in general for infinite-dimensional Hilbert spaces. We recall here the general definition, although in this paper we will only consider the finite dimensional case. Given two probabilistic domains $\mathcal{C}$ and $\mathcal{D}$, a *probabilistic abstract interpretation* is defined by a pair of linear maps, $\mathbf{A} : \mathcal{C} \mapsto \mathcal{D}$ and $\mathbf{G} : \mathcal{D} \mapsto \mathcal{C}$, between the concrete domain $\mathcal{C}$ and the abstract domain $\mathcal{D}$, such that $\mathbf{G}$ is the *Moore-Penrose pseudo-inverse* of $\mathbf{A}$, and vice versa. Let $\mathcal{C}$ and $\mathcal{D}$ be two Hilbert spaces and $\mathbf{A} : \mathcal{C} \mapsto \mathcal{D}$ a bounded linear map between them. A bounded linear map $\mathbf{A}^\dagger = \mathbf{G} : \mathcal{D} \mapsto \mathcal{C}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$ iff

$$\mathbf{A} \circ \mathbf{G} = \mathbf{P}_A \text{ and } \mathbf{G} \circ \mathbf{A} = \mathbf{P}_G$$

where $\mathbf{P}_A$ and $\mathbf{P}_G$ denote orthogonal projections (i.e. $\mathbf{P}_A^* = \mathbf{P}_A = \mathbf{P}_A^2$ and $\mathbf{P}_G^* = \mathbf{P}_G = \mathbf{P}_G^2$) onto the ranges of $\mathbf{A}$ and $\mathbf{G}$.

Alternatively, if $\mathbf{A}$ is Moore-Penrose invertible, its Moore-Penrose pseudo-inverse, $\mathbf{A}^\dagger$ satisfies the following:

$$\textbf{(i)}\ \mathbf{A}\mathbf{A}^\dagger\mathbf{A} = \mathbf{A}, \qquad \textbf{(iii)}\ (\mathbf{A}\mathbf{A}^\dagger)^* = \mathbf{A}\mathbf{A}^\dagger,$$

$$\textbf{(ii)}\ \mathbf{A}^\dagger\mathbf{A}\mathbf{A}^\dagger = \mathbf{A}^\dagger, \qquad \textbf{(iv)}\ (\mathbf{A}^\dagger\mathbf{A})^* = \mathbf{A}^\dagger\mathbf{A}.$$

where $\mathbf{M}^*$ is the adjoint of $\mathbf{M}$. It is instructive to compare these equations with the classical setting. For example, if (α, γ) is a Galois insertion we similarly have $\alpha \circ \gamma \circ \alpha = \alpha$ and $\gamma \circ \alpha \circ \gamma = \gamma$.

As in the classical framework, given a concrete semantics we can always construct a *best correct approximation* for this semantics, although the notions of correctness and optimality assume a different connotation in our linear setting as explained in the following.

If Φ is a linear operator on some vector space $\mathcal{V}$ expressing the probabilistic semantics of a concrete system, and $\mathbf{A} : \mathcal{V} \mapsto \mathcal{W}$ is a linear abstraction function from the concrete domain into an abstract domain $\mathcal{W}$, we can compute the (unique) Moore-Penrose pseudo-inverse $\mathbf{G} = \mathbf{A}^\dagger$ of $\mathbf{A}$. An abstract semantics can then be defined as the linear operator on the abstract domain $\mathcal{W}$:

$$\Psi = \mathbf{A} \circ \Phi \circ \mathbf{G} = \mathbf{G}\Phi\mathbf{A}.$$

In the case of classical abstract interpretation the abstract semantics constructed in this way (called the *induced semantics in* [5]) is guaranteed to be the *best correct approximation* of the concrete semantics, meaning that it is the most precise among all correct approximation (the relative precision being left unquantified). In the linear space based setting of PAI where the order of the classical domains is replaced by some notion of metric distance, the induced abstract semantics is the *closest* one to the concrete semantics. This "closeness" property expresses both the "safety" of the approximation and its optimality, which comes from the following properties of the Moore-Penrose pseudo-inverse. The theory of the least-square approximation [9, 7] tells us that if $\mathcal{C}$ and $\mathcal{D}$ be two finite dimensional vector spaces, $\mathbf{A} : \mathcal{C} \mapsto \mathcal{D}$ a linear map between them, and $\mathbf{A}^\dagger = \mathbf{G} : \mathcal{D} \mapsto \mathcal{C}$ its Moore-Penrose pseudo-inverse, then the vector $x_0 = y\mathbf{G}$ is the one minimising the distance between $x\mathbf{A}$, for any vector x in $\mathcal{C}$, and y, i.e.

$$\inf_{x \in \mathcal{C}} \|x\mathbf{A} - y\| = \|x_0\mathbf{A} - y\|.$$

In other words, if we consider the equation $x\mathbf{A} = y$ we can identify a (exact) solution x_* as a vector for which $\|x_*\mathbf{A} - y\| = 0$. In particular in the case that no such solution vector x_* exists we can generalise the concept of a exact solution to that of a "pseudo-solution", i.e. we can look for a x_0 such that $x_0\mathbf{A}$ is the *closest* vector to y we can construct. This closest approximation to the exact solution is now constructed using the Moore-Penrose pseudo-inverse, i.e. take $x_0 = y\mathbf{A}^\dagger$.

Returning to our program analysis setting, suppose that we have an operator Φ and a vector x. We can apply Φ to x and abstract the result giving $x\Phi\mathbf{A}$ or we can apply the abstract operator to an abstract vector giving $x\mathbf{A}\mathbf{A}^\dagger\Phi\mathbf{A}$. Ideally, we would like these to be equal. If $\mathbf{A}$ is invertible then its Moore-Penrose pseudo-inverse is identical to the inverse and we are done. In program analysis $\mathbf{A}$ is never a square matrix and thus $\mathbf{A}\mathbf{A}^\dagger$ in $x\mathbf{A}\mathbf{A}^\dagger\Phi\mathbf{A}$ will lead to some loss of precision. The Moore-Penrose pseudo-inverse is as close as possible to an inverse if the matrix is not invertible and thus for the particular choice of $\mathbf{A}$, $\mathbf{A}^\dagger\Phi\mathbf{A}$ is the best approximation of Φ that we can have. Moreover, by choosing an appropriate notion of distance we can measure this closeness to get a quantitative estimate of the information lost in the abstraction.

5 Probabilistic Strictness Analysis

In this section we will use the PAI technique previously introduced to define a probabilistic abstraction of the linear operator expressing the semantics of a probabilistic λ-term (see Section 2.2). This will allow us to construct a probabilistic strictness analysis from which we will be able to extract quantitative information such as the probability that a given reduction will not terminate.

In many cases, and particularly in strictness analysis, the abstraction is a surjective function. An alternative view of abstraction in this case is that it maps concrete values to equivalence classes. Equivalence relations can be represented by a particular kind of operators, namely classification operators.

We call an $n \times m$-matrix $\mathbf{K}$ a *classification* matrix if it is a 0/1-matrix, where every row has exactly one non-zero entry and columns have at least one non-zero entry. Classification matrices are thus particular kinds of stochastic matrices. We denote by $\mathcal{K}(n, m)$ the set of all $n \times m$-classification matrices ($m \leq n$). Let $X = \{x_1, \ldots, x_n\}$ be a finite set. Then for each equivalence relation $\approx$ on X with $|X/_\approx| = m$, there exists a classification matrix $\mathbf{K} \in \mathcal{K}(n, m)$ and vice versa. Each column in the classification matrix represents a (non-empty) equivalence class.

The pseudo-inverse of a classification matrix $\mathbf{K} \in \mathcal{K}(n, m)$ corresponds to its normalised transpose or adjoint $\mathbf{K}^\dagger = \mathcal{N}(\mathbf{K}^T)$ (these coincide for real-valued $\mathbf{K}$), where the *normalisation* operation $\mathcal{N}$ is defined for a matrix $\mathbf{A}$ by:

$$\mathcal{N}(\mathbf{A})_{ij} = \begin{cases} \frac{\mathbf{A}_{ij}}{a_i} & \text{if } a_i = \sum_j \mathbf{A}_{ij} \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

A suitable abstraction for probabilistic strictness analysis classifies terms as undefined, don't know or defined. We abstract every term in the enumeration to one of these values. The don't know value is used to classify terms whose "definedness" is not yet determined. Classically 0 represents definite non-termination whilst 1 represents *possible* termination. The use of three values here allows for a more informative analysis – the defined value means *definitely* terminating. This abstraction is achieved by the classification operator corresponding to the partition of the set of terms into the three classes represented by 0, ? (unknown) and 1. We will demonstrate the method by the following examples.

Example 21 Consider again Example 16:

$$P \equiv ((\lambda x.0) \oplus_{\frac{1}{2}} (\lambda x.x))(\perp \oplus_{\frac{3}{4}} 42)$$

and its flattened version $P \mapsto \tilde{P}$:

$$\tilde{P} \equiv \frac{1}{8} : ((\lambda x.0)\perp) \oplus \frac{3}{8} : ((\lambda x.0)42) \oplus \frac{1}{8} : ((\lambda x.x)\perp) \oplus \frac{3}{8} : ((\lambda x.x)42).$$

Using the following abstraction operator and its pseudo-inverse

$$\mathbf{K} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{K}^\dagger = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}$$

we get the abstract (induced) semantical operator $\mathbf{T}_P^\# : \mathcal{V}(\{0, ?, 1\}) \rightarrow \mathcal{V}(\{0, ?, 1\})$:

$$\mathbf{T}_P^\# = \mathbf{K}^\dagger \mathbf{T}_P \mathbf{K} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{6} & 0 & \frac{5}{6} \\ 0 & 0 & 1 \end{pmatrix}$$

As the flattened version $\overline{P}$ is constructed only of classical λ-terms which can be reduced in a single step to their normal form the abstract semantics turns out to be an exact abstraction in the sense that we have:

$$\overline{P} \cdot \mathbf{T}_P \cdot \mathbf{K} = \begin{pmatrix} \frac{1}{8} & 0 & \frac{7}{8} \end{pmatrix} = \overline{P} \cdot \mathbf{K} \cdot \mathbf{T}_P^\#$$

i.e. the abstracted concrete semantics coincides with the abstract semantics.

In the last example the analysis is optimal as the abstraction does not lose any information. However, in general we will encounter a loss of precision when we move from the concrete to the abstract semantics, as the following example illustrates:

Example 22 Consider the probabilistic term

$$\begin{aligned} P \quad \equiv \quad & \frac{4}{12} : (\lambda x.x)(\lambda x.x)(42) \ \oplus \ \frac{2}{12} : (\lambda x.x)(\bot) \ \oplus \\ & \frac{3}{12} : (\lambda x.x)(\lambda x.x)(\bot) \ \oplus \ \frac{3}{12} : (\lambda x.x)(\lambda x.0)(\bot) \end{aligned}$$

and the enumeration of the reachable(classical) λ terms:

1 $(\lambda x.x)(\lambda x.x)(42)$

2 $(\lambda x.x)(\bot)$

3 $(\lambda x.x)(\lambda x.x)(\bot)$

4 $(\lambda x.x)(\lambda x.0)(\bot)$

5 $(\lambda x.x)(42)$

6 $\bot$

7 $(\lambda x.x)(\bot)$

8 $(\lambda x.0)(\bot)$

9 42

10 0

Then we can construct the restricted one-step reduction operator for P and a suitable classification matrix for P as follows:

$$\mathbf{T}_P = \begin{pmatrix} 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&1 \end{pmatrix} \qquad \mathbf{K} = \begin{pmatrix} 0&1&0 \\ 0&1&0 \\ 0&1&0 \\ 0&1&0 \\ 0&1&0 \\ 1&0&0 \\ 0&1&0 \\ 0&1&0 \\ 0&0&1 \\ 0&0&1 \end{pmatrix}.$$

We can then construct the abstract one-step reduction operator:

$$\mathbf{T}_P^{\#} = \mathbf{K}^{\dagger}\mathbf{T}_P\mathbf{K} = \begin{pmatrix} 1 & 0 & 0 \\ 0.29 & 0.43 & 0.29 \\ 0 & 0 & 1 \end{pmatrix}$$

The middle row and column represent the don't know value. The value in the middle row, middle column gives a bound on how much the other two values in that row might change when we iterate – in this sense, it gives a measure of the precision of the current abstract operator. Iterating this abstract operator causes the probability of a transition from don't know to don't know to decrease rapidly; for example after two iterations we have:

$$(\mathbf{T}_P^{\#})^2 = \begin{pmatrix} 1 & 0 & 0 \\ 0.41 & 0.18 & 0.41 \\ 0 & 0 & 1 \end{pmatrix}$$

Achieving a defined outcome becomes more and more likely. This result could be used to support the decision to speculatively evaluate the argument.

One advantage of the use of linear operators is that we can measure them. The standard way to measure the "size" of a linear operator is via their operator norm.

Example 23 For the example above, an accurate abstraction of the original program can be achieved by taking the limit of the iterations of the abstract operator (numerically computed as $(\mathbf{T}_P^{\#})^{100}$):

$$\lim_{i\to\infty}(\mathbf{T}_P^{\#})^i = \begin{pmatrix} 1 & 0 & 0 \\ 0.50 & 0 & 0.50 \\ 0 & 0 & 1 \end{pmatrix}.$$

The limit of T_P gives instead the concrete semantics of P:

$$\lim_{i\to\infty} \mathbf{T}_P^i = \mathbf{T}_P^2 = \begin{pmatrix} 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&1 \end{pmatrix}$$

from which we can directly construct the abstract semantical operator

$$\mathbf{K}^\dagger(\lim_{i\to\infty} \mathbf{T}_P^i)\mathbf{K} = \mathbf{K}^\dagger\mathbf{T}_P^2\mathbf{K} = \begin{pmatrix} 1&0&0\\ 0.43&0&0.57\\ 0&0&1 \end{pmatrix}.$$

By comparing the two operators $\mathbf{K}^\dagger[\![P]\!]\mathbf{K}$ and $\lim_{i\to\infty}(\mathbf{T}_P^\#)^i$ we get

$$\|\mathbf{K}^\dagger(\lim_{i\to\infty} \mathbf{T}_P^i)\mathbf{K} - \lim_{i\to\infty}(\mathbf{K}^\dagger\mathbf{T}_P\mathbf{K})^i\|_\infty = 0.143$$

and when we compare the abstract and concrete semantics after two iterations (when the concrete semantics converges to its limit) we have:

$$\|\mathbf{K}^\dagger\mathbf{T}_P^2\mathbf{K} - (\mathbf{T}_P^\#)^2\|_\infty = 0.367.$$

These norm differences measure the "precision" of the abstract semantics. In the first case we see that the abstract semantics $\lim_{i\to\infty}(\mathbf{T}_P^\#)^i$ and the abstracted semantics $\mathbf{K}^\dagger(\lim_{i\to\infty}\mathbf{T}_P^i)\mathbf{K}$ differ by about 14%, i.e. the true chance of obtaining $\perp$ or a constant and the estimated probabilities are within $\pm 7\%$; and indeed we see that, for example, the true chance of getting $\perp$ is about 43%, while the abstract semantics forecasts a 50% chance of obtaining this result.

6 Conclusions

In this paper we introduced a probabilistic version of the λ-calculus and provided it with three different semantics. One is based on a *classical*, i.e. non-probabilistic, reduction relation on probability distributions on *classical* terms; a second one is defined via a *probabilistic* transition relation on *probabilistic* λ-terms which subsumes the classical β-reduction; and a third one is a *linear operator* semantics which specifies the computational dynamics in the form of so-called Markov chains [20]. All the three semantics are equivalent in the sense that they assign the same meaning to a probabilistic λ-term P. This is essentially a distribution on the β-normal forms to which the classical sub-terms of P can be β-reduced. The second semantics can be seen as an intermediate step between the first one and the linear operator semantics, which is the most appropriate as a base for quantitative analysis.

We have also reviewed the classic approach to abstract interpretation and strictness analysis and shown how Di Pierro and Wiklicky's notion of probabilistic abstract interpretation [12, 13] is a natural analogue of the classical framework in a probabilistic setting. We have illustrated the approach for the probabilistic λ-calculus in the context of a simple strictness analysis. This quantitative version of strictness analysis is based on the linear operator semantics for the λ-calculus and aims in constructing an abstract semantics which is as close as possible to the abstraction of the concrete semantics. The final outcome of this analysis are quantitative estimates on the probability that a probabilistic term reduces to a constant normal form or to $\perp$.

In our formal treatment of the semantics and analysis of the probabilistic λ-calculus we assumed that the λ-terms we consider all have normal forms, i.e. terminating reduction sequences. This allows us to use straight forward constructions from linear algebra. In related work we have shown that in principle it is possible to accommodate also infinite reduction sequences, e.g. [11]. However, this requires a recasting of our framework in a functional analytical setting, using results and concepts from the theory of Hilbert spaces and C*-algebras (see e.g. [3, 16]), which would deserve a separate treatment.

A present shortcoming of our work is that the strictness analysis is not defined in a compositional way. However, the intermediate, distribution-based semantics can be seen as a kind of 'lifting' of the classical β-rule and it seems therefore possible to import results from the semantics of the classical λ-calculus and classical strictness analysis in order to define a compositional linear semantics and to refine the simple probabilistic strictness analysis presented in this paper.

References

[1] H. P. Barendregt. The Lambda Calculus: Its Syntax and Semantics. North-Holland, 1981.

[2] G. Burn, C. Hankin and S. Abramsky. The Theory and Practice of Higher-order Strictness Analysis. *Science of Computer Programming*, 1986.

[3] J.B. Conway. *A Course in Functional Analysis*, volume 96 of *Graduate Texts in Mathematics*. Springer Verlag, New York, second edition, 1990.

[4] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixedpoints. In *Proceedings of POPL*, pages 238–252, 1977. ACM.

[5] P. Cousot and R. Cousot. Systematic Design of Program Analysis Frameworks. In *Proceedings of POPL*, pages 269–282, 1979. ACM.

[6] D. Cox and H. R. Miller. The Theory of Stochastic Processes. Chapman and Hall. 1965.

[7] Ben-Israel, A., Greville, T.: Generalised Inverses — Theory and Applications. second edn. Springer Verlag, New York — Berlin (2003)

[8] V. Danos and R. Harmer. Probabilistic Game Semantics. *ACM Transactions on Computational Logic*, 2001.

[9] F. Deutsch. *Best Approximation in Inner Product Spaces*, volume 7 of *CMS Books in Mathematics.* Springer Verlag, New York — Berlin, 2001.

[10] A. Di Pierro and H. Wiklicky. Quantitative observables and averages in Probabilistic Concurrent Constraint Programming. In K.R. Apt, T. Kakas, E. Monfroy, and F. Rossi, editors, *New Trends in Constraints*, number 1865 in Lecture Notes in Computer Science, Springer Verlag, New York — Berlin, 2000.

[11] A. Di Pierro and H. Wiklicky, *Linear structures for concurrency in probabilistic programming languages*, in: *Proceedings of MFCSIT00*, Electronic Notes in Theoretical Computer Science 40 (2001).

[12] A. Di Pierro and H. Wiklicky. Concurrent Constraint Programming: Towards Probabilistic Abstract Interpretation. In *Proceedings of PPDP'00*, 2000. ACM.

[13] A. Di Pierro and H. Wiklicky. Measuring the precision of abstract interpretations. In *Proceedings of LOPSTR'00*, LNCS 2042, 2001. Springer Verlag.

[14] W.H. Greub. *Linear Algebra*, volume 97 of *Grundlehren der mathematischen Wissenschaften.* Springer Verlag, New York, third edition, 1967.

[15] C. Hankin. *Lambda Calculi*, volume 3 of *Graduate Texts in Computer Science.* Clarendon Press, Oxford, 1994.

[16] R.V. Kadison and J.R. Ringrose. *Fundamentals of the Theory of Operator Algebras: Volume I & II*, volume 15 of *Graduate Studies in Mathematics.* AMS, Providence, Rhode Island, 1997.

[17] P. Malacaria and L. Regnier. Some results on the Interpretation of λ-calculus in Operator Algebra. In *Proceedings of LICS*, pages 63–72, 1991, IEEE Press.

[18] A. Mycroft. Abstract Interpretation and Optimising Transformations for Applicative Programs. Ph.D. Thesis, University of Edinburgh, 1981.

[19] F. Nielson, H. Riis Nielson and C. Hankin. *Principles of Program Analysis.* Springer Verlag, 1999.

[20] J.R. Norris. *Markov Chains. Cambidge Series in Statistical and Probabilistic Mathematics.* Cambridge University Press, Cambridge, 1997.

[21] Sungwoo Park. *A calculus for probabilistic languages.* ACM SIGPLAN Notices, 37(9), pages 206–217, 2002. ACM.

[22] N. Ramsey and A. Pfeffer. *Stochastic lambda calculus and monads of probability distributions.* In *Proceedings of POPL*, pages 154–165, 2002. ACM.