λ-calculus and Quantitative Program Analysis (Extended Abstract)

Chris Hankin
Department of Computing
Imperial College London
London, UK

Herbert Wiklicky
Department of Computing
Imperial College London
London, UK

Abstract

In this paper we show how the framework of probabilistic abstract interpretation can be applied to statically analyse a probabilistic λ-calculus. We start by reviewing the classical framework of abstract interpretation. We choose to use (first-order) strictness analysis as our running example. We present the definition of probabilistic abstract interpretation and use it to construct a probabilistic strictness analysis.

1 Introduction

In this paper we aim to show how probabilistic abstract interpretation [6,7] can be used to analyse terms in a probabilistic λ-calculus. Our running example will be a simple strictness analysis [11,2]. 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 [4]), which is related to economics and in particular to risk management.

Example 1 [Random Walk] A company starts with initial capital of \( \text{Cap}_0 \), at each time step its income is \( \text{In}_i \) and its outlay to meet claims is \( \text{Out}_i \); 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 0 and jumps \( \text{Step}_n = \text{In}_n - \text{Out}_n \):

\[
\text{Cap}_n = \begin{cases} 
\text{Cap}_{n-1} + \text{Step}_n & \text{Cap}_{n-1} > 0, \text{Cap}_{n-1} + \text{Step}_n > 0 \\
0 & \text{otherwise}
\end{cases}
\]

Qualitatively, we can analyse the random walk and just conclude that \( \text{Cap} \) ranges over the interval \([0, \infty)\); quantitatively, we can ask the more interesting question: \textit{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 rest of this paper is organised as follows. We start by introducing the probabilistic \( \lambda \)-calculus. 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 \( \lambda \)-calculus. The paper [2] shows how these ideas can be extended to the higher-order case. We then present our approach to semantics based on linear operators and describe probabilistic abstract interpretation [6,7]. The final main section returns to the problem of strictness analysis in the probabilistic \( \lambda \)-calculus.

2 Probabilistic \( \lambda \)-calculus

We define, \( \Lambda_P \), the class of probabilistic \( \lambda \)-terms to be the least class defined by:

- Each variable \( x \) is a term.
- Each constant, including \( \bot \), is a term.
- For \( M, N \in \Lambda_P \), \((MN) \in \Lambda_P \) and \((\lambda x.M) \in \Lambda_P \).
- For \( M, N \in \Lambda_P \), \((M \oplus_p N) \) for some probability \( p \).

This is the usual \( \lambda \)-terms plus terms of the form \( e_1 \oplus_p e_2 \) (indicating a probabilistic choice, the right hand summand being chosen with probability it \( p \)).

We assume a leftmost reduction strategy. We write \( e_1 \rightarrow_p e_2 \) to mean that \( e_1 \) reduces to \( e_2 \) with probability \( p \). Configurations, \( e \), are a pair of a term and an environment; \( \rho \) maps \( \text{Var} \) to \( \Lambda_P \). The semantics of terms are given by the
following reduction system:

$$
\begin{align*}
(\text{var}) & \quad (x, \rho) \rightarrow_1 \rho(x) \\
(\text{app}) & \quad (M, \rho) \rightarrow_p (P, \rho') \\
& \quad ((MN), \rho) \rightarrow_p ((PN), \rho') \\
(\beta) & \quad ((\lambda x. M) N, \rho) \rightarrow^\beta_1 (M, \rho[x := N]) \\
(\delta_1) & \quad ((M \oplus_p N), \rho) \rightarrow^\delta_{1-p} (M, \rho) \\
(\delta_2) & \quad ((M \oplus_p N), \rho) \rightarrow^\delta_p (N, \rho)
\end{align*}
$$

Terminal configurations have a term which is a $\lambda$-term or a constant. The \texttt{app} and $\beta$ together enforce the leftmost reduction strategy.

3 Classical Abstract Interpretation

We start by sketching the classical approach to semantics-based program analysis: abstract interpretation [3,12]. The semantics of a program $p$ identifies some set $V$ of values and specifies how the program transforms one value $v_1$ to another $v_2$:

$$p \vdash v_1 \rightarrow v_2$$

In a similar way, a program analysis identifies the set $L$ of properties and specifies how a program $p$ transforms one property $l_1$ to another $l_2$:

$$p \vdash l_1 \triangleright l_2$$

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:

$$R : V \times L \rightarrow \{\text{true}, \text{false}\}$$

The intention is that $v R l$ formalises our claim that the value $v$ is described by the property $l$.

To be useful one has to prove that the correctness relation $R$ is preserved under computation: if the relation holds between the initial value and the initial
property then it also holds between the final value and the final property. This may be formulated as the implication

\[ v_1 R l_1 \land p \vdash v_1 \rightarrow v_2 \land p \vdash l_1 \triangleright l_2 \Rightarrow v_2 R l_2 \]

The most common scenario in abstract interpretation is when both \( V \) and \( L \) are complete lattices. We then impose the following relationship between \( R \) and \( L \):

\[ v R l_1 \land l_1 \sqsubseteq l_2 \Rightarrow v R l_2 \quad (1) \]

\[ (\forall l \in L' \subseteq L : v R l) \Rightarrow v R \left( \bigcap L' \right) \quad (2) \]

The correctness relation is often achieved via a Galois connection: \((V, \alpha, \gamma, L)\) 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:

\[ \gamma \circ \alpha \supseteq \lambda v. v \quad \text{and} \quad \alpha \circ \gamma \subseteq \lambda l.l \]

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 [12] as long as we use interpretations of language operators that satisfy:

\[ F_{\text{abs}} \supseteq \alpha \circ F \circ \gamma \]

Since interesting languages involve iteration or recursion we also have to construct efficient implementations; a generic solution to this problem is the theory of widenings and narrowings [3].

3.1 Strictness Analysis

Strictness analysis [11,2] aims to answer for some function \( f \): Does \( f \perp = \perp \)? An affirmative answer would mean that arguments can be passed by value rather than using (the more costly) lazy evaluation. We will restrict ourselves to a first order functional language with integers as the only data type.

We can construct a Galois connection \((\mathcal{P}_H(Z_\perp^\top), \alpha, \gamma, \text{Two})\) where \( \mathcal{P}_H \) is the Hoare Powerdomain construction and \( \text{Two} \) 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.
We define:

\[
\alpha(Z) = \begin{cases} 
0 & \text{if } Z = \{\bot\} \\
1 & \text{otherwise}
\end{cases}, \quad 
\gamma(S) = \begin{cases} 
\{\bot\} & \text{if } S = 0 \\
Z^\perp & \text{if } S = 1
\end{cases}
\]

We can construct the induced operations that correspond to the operations in this first-order applied \(\lambda\)-calculus:

<table>
<thead>
<tr>
<th>Concrete operation</th>
<th>Induced operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants</td>
<td>1</td>
</tr>
<tr>
<td>(if ; x ; then ; y ; else ; z)</td>
<td>(x \sqcap (y \sqcup z))</td>
</tr>
<tr>
<td>(x ; op ; y)</td>
<td>(x \sqcap y)</td>
</tr>
</tbody>
</table>

Thus the abstract interpretation of

\((\lambda \; x. \; if \; x = 0 \; then \; 15 \; else \; 42)\)

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 \(\lambda\)-calculus. We could just apply the classical framework to this new setting [9,10]; instead we will apply the techniques of \textit{Probabilistic Abstract Interpretation} [6,7].

4 Linear Representations

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

\[
\mathcal{V}(X) = \left\{ \sum c_x \vec{x} \mid c_x \in \mathbb{W}, \; x \in X \right\}.
\]

The semantics of terms in our extended calculus can be represented by a probabilistic \textit{reduction graph} where edges are labelled with probabilities. We associate to each quantitative relation \(R \subseteq X \times \mathbb{W} \times X\) a matrix, i.e. a linear
operator $M_R$ on $\mathcal{V}(X)$ defined by:

$$(M_R)_{ij} = \begin{cases} w & \text{iff } \sum_{R(x_i,w',x_j)} w' = w \\ 0 & \text{otherwise} \end{cases}$$

For probabilistic relations this gives a Stochastic matrix.

### 4.1 Linear Semantics for the Probabilistic $\lambda$-calculus

In this section we follow a similar development to that of [5].

We start by defining three operators which represent transitions corresponding to $\beta$-reduction, $\delta$-reduction (for the probabilistic choice) and idling (for terms in normal form) respectively. Each operator is of type $\mathcal{V}(\Lambda_P) \rightarrow \mathcal{V}(\Lambda_P)$.

Before defining the operator for one-step $\beta$-reduction, we define an appropriate notion of active context – these are $\lambda$-terms with a single hole which determines where the next redex to be reduced is to be found. Such contexts are used in the definition of compatible closure in the standard construction of one-step reductions [1]; since we are interested in call-by-name evaluation, we do not reduce redexes in the argument or under $\lambda$s. Given this intuition, $C[\ ]$, the class of active contexts is the least class such that:

- $[\ ] \in C[\ ]$.
- $[\ ]N \in C[\ ]$ for any $N \in \Lambda_P$.

The operator $B$ has the following matrix representation for each variable $x$ and term $N \in \Lambda_P$:

$$B(x,N)_{t_1,t_2} = \begin{cases} 1 & \text{if } t_1 \equiv C[(\lambda x.M)N], t_2 \equiv C[M[x := N]] \\ 0 & \text{otherwise} \end{cases}$$

The operator $C$ for the probabilistic choice operator has the following matrix representation:

$$C_{t_1,t_2} = \begin{cases} p & \text{if } t_1 \equiv C[M \oplus_q N], t_1 \rightarrow_p t_2, ((p = q) \lor (p = 1 - q)) \\ 0 & \text{otherwise} \end{cases}$$
Finally, the idling operator, \( N \), has the following matrix representation:

\[
N_{t_1, t_2} = \begin{cases} 
1 & \text{if } t_1 \equiv t_2, \ t_1 \text{ is a } \beta\delta\text{nf} \\
0 & \text{otherwise}
\end{cases}
\]

The operator, \( T \), which describes the transitions available from a term is then defined as:

\[
T = C + N + \sum_{x,N} B(x, N)
\]

The semantics of a term is then recovered by iterated application of \( T \):

\[
\lim_{i \to \infty} T^i.
\]

In practice we will restrict to the reachable terms from some given term, \( M \):

\[
R(M) = \{N | M \rightarrow^*_{\beta\delta} N\}
\]

and work with the restricted transition operator:

\[
\overline{T} = \pi_M T \pi_M
\]

where \( \pi_M \) is the projection on to \( V(R(M)) \).

**Example 2** Consider the term:

\[
((\lambda x.0) \oplus_{\frac{1}{7}} (\lambda x.x))(\bot \oplus_{\frac{2}{7}} 42)
\]

An enumeration of the reachable terms is:

- \((\lambda x.0) \oplus_{\frac{1}{7}} (\lambda x.x))(\bot \oplus_{\frac{2}{7}} 42)
- \((\lambda x.0)(\bot \oplus_{\frac{2}{7}} 42)
- \((\lambda x.x)(\bot \oplus_{\frac{2}{7}} 42)
- 0
- \bot \oplus_{\frac{2}{7}} 42
- \bot
- 42

7
and we have:

\[
T = \begin{pmatrix}
0 & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & \frac{1}{2} & \frac{3}{4} & \frac{3}{4} \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\]

5 Probabilistic Abstract Interpretation

Given two probabilistic domains (i.e. vectors of λ-terms), \( \mathcal{C} \) and \( \mathcal{D} \), a **probabilistic abstract interpretation** is a pair of linear maps, \( A : \mathcal{C} \mapsto \mathcal{D} \) and \( G : \mathcal{D} \mapsto \mathcal{C} \), between the concrete domain \( \mathcal{C} \) and the abstract domain \( \mathcal{D} \), such that \( G \) is the **Moore-Penrose pseudo-inverse** of \( A \), and vice versa. Let \( \mathcal{C} \) and \( \mathcal{D} \) be two Hilbert spaces and \( A : \mathcal{C} \mapsto \mathcal{D} \) a bounded linear map between them. A bounded linear map \( A^\dagger = G : \mathcal{D} \mapsto \mathcal{C} \) is the Moore-Penrose pseudo-inverse of \( A \) iff

\[
A \circ G = P_A \quad \text{and} \quad G \circ A = P_G
\]

where \( P_A \) and \( P_G \) denote orthogonal projections onto the ranges of \( A \) and \( G \).

Alternatively, if \( A \) is Moore-Penrose invertible, its Moore-Penrose pseudoinverse, \( A^\dagger \) satisfies the following:

(i) \( AA^\dagger A = A \),
(ii) \( A^\dagger AA^\dagger = A^\dagger \),
(iii) \( (AA^\dagger)^* = AA^\dagger \),
(iv) \( (A^\dagger A)^* = A^\dagger A \).

where \( M^* \) is the adjoint of \( M \). It is instructive to compare these equations with the classical setting. For example, if \((\alpha, \gamma)\) is a Galois insertion: \( \alpha \circ \gamma \circ \alpha = \alpha \) and \( \gamma \circ \alpha \circ \gamma = \gamma \).

A simple method to construct a probabilistic abstract interpretation is as follows: Given a linear operator \( \Phi \) on some vector space \( \mathcal{V} \) expressing the probabilistic semantics of a concrete system, and a linear abstraction function \( A : \mathcal{V} \mapsto \mathcal{W} \) from the concrete domain into an abstract domain \( \mathcal{W} \), we compute the (unique) Moore-Penrose pseudo-inverse \( G = A^\dagger \) of \( A \). The abstract
semantics can then be defined as the linear operator on the abstract domain \( \mathcal{W} \):

\[
\Psi = A \circ \Phi \circ G.
\]

6 Probabilistic Strictness Analysis

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

We call an \( n \times m \)-matrix \( K \) 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. 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 \( K \in \mathcal{K}(n, m) \) and vice versa.

The pseudo-inverse of a classification matrix \( K \in \mathcal{K}(n, m) \) corresponds to its normalised transpose or adjoint (these coincide for real \( K \)).

\[
K^\dagger = \mathcal{N}(K^T) = \mathcal{N}(K^*)
\]

where the normalisation operation \( \mathcal{N} \) is defined for a matrix \( A \) by:

\[
\mathcal{N}(A)_{ij} = \begin{cases} 
\frac{A_{ij}}{a_j} & \text{if } a_j = \sum_i A_{ij} \neq 0 \\
0 & \text{otherwise.}
\end{cases}
\]

A suitable abstraction for probabilistic strictness analysis classifies terms as undefined, don’t know or defined. This abstraction is achieved by a classification operator.
Example 3 A suitable classification matrix for our running example is

\[
K = \begin{bmatrix}
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\]

which has Moore-Penrose pseudoinverse

\[
K^+ = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\
\end{bmatrix}
\]

The abstract semantics of our original program is

\[
K^+TK = \begin{bmatrix}
1 & 0 & 0 \\
\frac{1}{16} & \frac{7}{16} \\
0 & 0 & 1 \\
\end{bmatrix}
\]

The middle row and column represent the don’t know value. Iterating this abstract operator causes the probability of a transition from don’t know to don’t know to decrease rapidly; for example after three iterations we have:

\[
(K^+TK)^3 = \begin{bmatrix}
1 & 0 & 0 \\
\frac{7}{64} & \frac{1}{8} & \frac{49}{64} \\
0 & 0 & 1 \\
\end{bmatrix}
\]

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 interpreting relations as linear operators allows us to measure them. The standard way to measure the “size” of a linear operator is via an operator norm which in turn may have its origins in a vector norm:

- \(\|x\| \geq 0\)
• \( \|\vec{x}\| = 0 \iff \vec{x} = \vec{0} \)
• \( \|\alpha \vec{x}\| = |\alpha| \|\vec{x}\| \)
• \( \|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\| \)

For example, we could use the 1-norm (sum of absolute values), euclidean norm (square root of the sum of squares of absolute values) or the supremum norm (supremum of absolute values).

**Example 4** An accurate abstraction of the original program, computed from the reduction graph, is:

\[
\mathbf{T^\#} = \begin{pmatrix}
1 & 0 & 0 \\
\frac{1}{8} & 0 & \frac{7}{8} \\
0 & 0 & 1
\end{pmatrix}
\]

Considering the difference between this and our first abstraction we get:

\[
\|\mathbf{T^\#} - \mathbf{K}^\dagger \mathbf{T} \mathbf{K}\|_\infty = \frac{1}{2}
\]

whilst

\[
\|\mathbf{T^\#} - (\mathbf{K}^\dagger \mathbf{T} \mathbf{K})^3\|_\infty = \frac{1}{8}
\]

We have abstracted \( \mathbf{T} \) but we could also iterate this operator.

**Example 5** We find that:

\[
\lim_{i \to \infty} \mathbf{T}^i = \mathbf{T}^3 = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

11
The abstraction of this is:

\[
(K^\dagger T^3 K) = \begin{pmatrix}
1 & 0 & 0 \\
5/32 & 0 & 27/32 \\
0 & 0 & 1
\end{pmatrix}
\]

and

\[
\|T^\# - K^\dagger T^3 K\|_\infty = \frac{1}{32}
\]

Finally, it should also be noted that:

\[
\|K^\dagger T^3 K - (K^\dagger TK)^3\|_\infty = \frac{1}{8}
\]

7 Conclusions

We have reviewed the classic approach to abstract interpretation and also shown that Di Pierro and Wiklicky’s notion of probabilistic abstract interpretation is a natural analogue of the classical framework. We have illustrated the approach for the \(\lambda\)-calculus in the context of a simple strictness analysis. A present shortcoming of our work is that neither the linear semantics nor the strictness analysis are defined in a compositional way. If we were able to give a compositional linear semantics, we would expect that compositional strictness analysis would be straightforward. Unfortunately this has to remain work for the future but it is possible that earlier work on the relationship between \(\lambda\)-calculus and operator algebras [8] could help in this endeavour.

References


