On the Precision of Abstract Interpretations *

Alessandra Di Pierro[1] and Herbert Wiklicky[2]

[1] Dipartimento di Informatica, Universitá di Pisa, Italy
[2] Department of Computing, Imperial College, London, UK

Abstract. We present a method for approximating the semantics of probabilistic programs to the purpose of constructing semantics-based analyses of various properties. The method is also suited for a probabilistic analysis of classical deterministic programs. The framework resembles the one based on Galois connection used in abstract interpretation, the main difference being the choice of linear space structures instead of order-theoretic ones as semantical (concrete and abstract) domains. The intrinsic quantitative nature of linear spaces makes the method suitable for investigations on the problem of a numerical comparison of abstract interpretations with respect to their precision.

1 Introduction

The construction of semantics-based program analysis algorithms or in general the specification and validation of the analyses of programs finds a good support in the theory of Abstract Interpretation, which has become very popular since it was introduced in 1977 by Patrick and Radhia Cousot [2,3].

Despite its appeal and generality there are still some aspects of semantic approximation within the abstract interpretation framework which seem to be in need of further investigation. One of such aspects is the problem of "measuring" the precision of an abstraction; for example in [8, Section 3.4] we find the following remark: "*One shortcoming of the development [...] is that a correct analysis may be so imprecise as to be practically useless. [...] The notion of correctness is topological in nature but we would ideally like something that was a bit more metric in nature so that we could express* how *imprecise a correct analysis is. Unfortunately no one has been able to develop an adequate metric for these purposes.*"

In this paper we propose and discuss a solution to this problem by means of a methodology for *probabilistic abstract interpretation*. The basic idea is that an appropriate re-formulation of the Cousot framework in a probabilistic setting provides us with a means to give an exact estimate of the approximation error introduced by a certain abstraction α and its corresponding concretisation γ.

* Extended Abstracts of LOPSTR 2000, Tenth International Workshop on Logic-based Program Synthesis and Transformation, 24-28 July 2000, London, UK. Technical Report Report Series, Department of Computer Science, University of Manchester, ISSN 1361-6161. Report number UMCS-00-6-1. http://www.cs.man.ac.uk/cstechrep/titles00.html

This allows us to "quantify" somehow the imprecision of the various abstract interpretations, thus making the notion of correctness more effective: for example, we could choose among the correct abstract interpretations for a given analysis the one which is "precise up to 35%".

2 Probabilistic Abstract Interpretation

2.1 Probabilistic Semantics

Probabilistic programs are programs which are formulated by reference to a "coin flipping" device (e.g. random choice or assignment) which determines the flow of information. As a consequence, a probabilistic program execution will deliver results with a *probability* attached to them expressing their likelihood of actually being computed. A denotational semantics for such programs must therefore contemplate structures which are able to reflect both the qualitative and the quantitative aspects of the information flow. There exist several approaches towards the semantics of probabilistic programming languages, starting with the contributions in the fundamental papers of Saheb-Djahromi [14] and Kozen [9]. More recently, the authors have developed a probabilistic version of the concurrent constraint programming paradigm in [5, 6], where other references to the literature on probabilistic semantics can be found.

For the purpose of this paper we will consider vector spaces as the prototypical semantic domain for probabilistic programs. Moreover, we will consider as a standard semantics a denotational semantics defined via a linear fix-point operator $\Phi : \mathcal{V} \mapsto \mathcal{V}$, on a vector space $\mathcal{V}$.

2.2 Abstract Interpretation

Classical abstract interpretation — as defined by Cousot&Cousot — is a method of designing *approximate* semantics of programs which can be used for analysis purposes [2, 3]. In order to prove the correctness of an abstract interpretation (that is the fact that the approximation does not affect the reliability of the analysis although this might be imprecise), it is required the existence of a reference *standard* semantics. Both the approximate (or abstract) and the standard (or concrete) semantics are given by specifying domains which are usually complete partial orders (lattices), and the relation between them is defined in terms of a Galois connection between the two partially ordered domains.

For probabilistic programs the standard semantics refers to domains which are usually structurally different from cpo's as explained in Section 2.1. In the following we will show how a notion similar to Galois connection can be defined for probabilistic domains (vector spaces). One advantage in doing so is that we can now use the real numbers expressing the probabilities to give a quantitative estimate of the precision of a (correct) abstract interpretation.

We will concentrate on the case of finite dimensional vector spaces, where we can represent all involved maps by matrices. Nevertheless, it is possible to generalise our approach to the infinite-dimensional case — i.e. Banach and Hilbert spaces, cf. Section 2.4.

For the finite-dimensional case we can take, without loss of generality, as concrete domain some $\mathbb{R}^n$ and as abstract domain a space $\mathbb{R}^m$ [13]. An abstraction function will then be a linear map $\boldsymbol{\alpha} : \mathbb{R}^n \mapsto \mathbb{R}^m$.

2.3 Galois Connections and Moore-Penrose Pseudo-Inverse

In the classical Cousot framework the relation between the abstraction function α and the concretisation function γ is that of a Galois connection:

Definition 1. Let $\mathcal{C} = (\mathcal{C}, \leq)$ and $\mathcal{D} = (\mathcal{D}, \sqsubseteq)$ be two partially ordered set together with two order preserving functions $\alpha : \mathcal{C} \mapsto \mathcal{D}$ and $\gamma : \mathcal{D} \mapsto \mathcal{C}$. Then $(\mathcal{C}, \alpha, \gamma, \mathcal{D})$ form a *Galois connection* iff

1. $\alpha \circ \gamma$ is reductive, and
2. $\gamma \circ \alpha$ is extensive.

The first condition means that for all $d \in D$, $\alpha \circ \gamma(d) \sqsubseteq d$. Analogously, the second condition means that for all $c \in C$, $c \leq \gamma \circ \alpha(c)$.

In a linear setting the condition of order preservation of the functions forming a Galois connection becomes linearity (i.e. general structure preservation). Thus we will define our probabilistic abstraction and concretisation functions as linear maps between the concrete and the abstract vector spaces. Conditions 1. and 2. find their linear analogs in two similar conditions which are expressed in terms of *projections*. Intuitively, a projection gives for each vector its "shadow" (on a screen represented by the image space of the projection). According to this intuition, the projection of a vector can be seen as an approximation of that vector. Formally, projections are defined as idempotent linear operators:

Definition 2. Given a finite dimensional (complex) vector space $\mathcal{V}$, a linear map $\pi : \mathcal{V} \mapsto \mathcal{V}$ is called a *projection* if $\pi^2 = \pi \circ \pi = \pi$. A projection is *orthogonal* if it coincides with its adjoint.

For finite dimensional vector spaces one can think of the adjoint of a linear map as the *transpose* of the matrix representing the map, i.e. the matrix where rows and columns are exchanged [13].

Given two finite dimensional vector spaces $\mathcal{V}$ and $\mathcal{W}$, and a linear map $\rho : \mathcal{V} \mapsto \mathcal{W}$, the image of ρ, $\rho(\mathcal{V}) \subseteq \mathcal{W}$, is a sub-space of $\mathcal{W}$. Associated to such a sub-space there is a unique orthogonal projection $\pi_{\rho(\mathcal{V})} : \mathcal{W} \mapsto \mathcal{W}$ mapping each vector in $\mathcal{W}$ into the sub-space $\rho(\mathcal{V})$. By using this notion of orthogonal projection, properties 1. and 2. of Galois connection can then be transferred into a vector space setting as follows. The condition $\alpha \circ \gamma(d) \sqsubseteq d$ becomes the condition that $\alpha \circ \gamma$ is the projection in the image of α, that is $\boldsymbol{\alpha} \circ \boldsymbol{\gamma} = \pi_\alpha$, and $\gamma \circ \alpha(c) \geq c$ turns into $\boldsymbol{\gamma} \circ \boldsymbol{\alpha} = \pi_\gamma$.

These new conditions fit very well with the properties of the so-called *Moore-Penrose pseudo-inverse*, which can be defined as follows:

Definition 3. Let $\mathcal{C}$ and $\mathcal{D}$ be two finite dimensional (complex) vector spaces and $\boldsymbol{\alpha} : \mathcal{C} \mapsto \mathcal{D}$ a linear map between them. A linear map $\boldsymbol{\alpha}^\dagger = \boldsymbol{\gamma} : \mathcal{D} \mapsto \mathcal{C}$ is the (unique) *Moore-Penrose pseudo-inverse* of $\boldsymbol{\alpha}$ iff

1. $\boldsymbol{\alpha} \circ \boldsymbol{\gamma} = \pi_{\boldsymbol{\alpha}}$, and
2. $\boldsymbol{\gamma} \circ \boldsymbol{\alpha} = \pi_{\boldsymbol{\gamma}}$.

By using this notion, we can therefore define a *probabilistic abstract interpretation* as a pair of linear maps, an abstraction function $\boldsymbol{\alpha} : \mathcal{C} \mapsto \mathcal{D}$ and a concretisation function $\boldsymbol{\gamma} : \mathcal{D} \mapsto \mathcal{C}$, between a concrete domain $\mathcal{C}$ and an abstract one $\mathcal{D}$, such that $\boldsymbol{\gamma}$ is the Moore-Penrose pseudo-inverse of $\boldsymbol{\alpha}$, and vice versa.

Example 4. Consider $\mathcal{C} = \mathbb{R}^3$ as a concrete probabilistic domain. We can define an abstraction function $\boldsymbol{\alpha} : \mathbb{R}^3 \mapsto \mathbb{R}^2$ by $\boldsymbol{\alpha}(c_1, c_2, c_3) = (c_1, c_3)$, i.e. we abstract in as far as we ignore the second component of a concrete (distribution) vector. The linear operator $\boldsymbol{\alpha}$ and its Moore-Penrose pseudo-inverse $\boldsymbol{\gamma} = \boldsymbol{\alpha}^\dagger$ can be represented by the following matrices:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \Gamma = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

The image of $\mathcal{D} = \mathbb{R}^2$ under $\boldsymbol{\gamma}$ in $\mathcal{C} = \mathbb{R}^3$ is therefore the c_1-c_3-plane (i.e. the two-dimensional sub-space generated by c_1 and c_3). Thus we get the following matrix, which indeed is a projection into the c_1-c_3-plane:

$$\Gamma \cdot A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2.4 Construction of Probabilistic Abstract Interpretations

We discuss here a method for constructing probabilistic abstract interpretations, which represents a first step towards a more complete and structured study of construction and combination methods similar to the catalogues in the classical case, e.g. [11, 4]. The construction we discuss is that of a fix-point abstraction using Moore-Penrose pseudo-inverses. The same construction using Galois connection gives in the classical case so called *induced analysis*.

Vector Space Lifting One way to lift a semantic domain $\mathcal{C}$ to a vector space is by defining the *free real vector space* $\mathcal{V}(\mathcal{C})$ on $\mathcal{C}$ as the set of all formal linear combinations:

$$\mathcal{V}(\mathcal{C}) = \left\{ \sum x_c c \mid x_c \in \mathbb{R},\ c \in \mathcal{C} \right\}.$$

An *extraction* function $\alpha : \mathcal{C} \mapsto \mathcal{D}$ of C into an abstract domain $\mathcal{D}$ (e.g. [11, 4.3]), can be then *lifted* to a linear map $\boldsymbol{\alpha} : \mathcal{V}(\mathcal{C}) \mapsto \mathcal{V}(\mathcal{D})$ as follows:

$$\boldsymbol{\alpha}(p_1 \cdot c_1 + p_2 \cdot c_2 + \ldots) = p_1 \cdot \alpha(c_1) + p_2 \cdot \alpha(c_2) \ldots$$

In order to construct a probabilistic abstract interpretation we now compute the (unique) Moore-Penrose pseudo-inverse $\boldsymbol{\gamma} = \boldsymbol{\alpha}^\dagger$ of $\boldsymbol{\alpha}$. The abstract semantics can then be defined as a linear fix-point operator on the abstract domain $\mathcal{V}(\mathcal{D})$, obtained by a formal computation starting from a concrete linear fix-point operator $\Phi : \mathcal{V}(\mathcal{C}) \mapsto \mathcal{V}(\mathcal{C})$ as follows:

$$\Psi = \boldsymbol{\alpha} \circ \Phi \circ \boldsymbol{\gamma}.$$

Since $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are linear (thus continuous) it is immediate that:

$$\boldsymbol{\gamma} \circ (\lim \Psi) = \boldsymbol{\gamma} \circ (\lim \boldsymbol{\alpha} \circ \Phi \circ \boldsymbol{\gamma}) = \boldsymbol{\gamma} \circ \boldsymbol{\alpha} \circ (\lim \Phi) \circ \boldsymbol{\gamma}.$$

Thus the operator $\boldsymbol{\gamma} \circ \boldsymbol{\alpha}$ exactly describes the "gap" between the concretisation of the abstract semantics and the concrete semantics.

Infinite-Dimensional Abstractions. The construction of the Moore-Penrose pseudo-inverse is essentially based on *finite-dimensional* vector spaces. Although this alone gives an elegant quantitative analysis, one is more than often confronted with an infinite concrete domain; for example a simple parityanalysis starts with the set of all natural numbers for which we therefore have to construct an infinite-dimensional vector space like $\mathcal{V}(\mathbb{N})$. In this case we can still guarantee the existence of a pseudo-inverse. We just sketch here the basic argument.

The *kernel* of a map α between two vector spaces $\mathcal{C}$ and $\mathcal{D}$ is the set of all those elements of $\mathcal{C}$ which are mapped into the null vector in $\mathcal{D}$, i.e. $\ker(\alpha) = \{x \in \mathcal{C} | \alpha(x) = o\}$. The *image* of α is the set of vectors in $\mathcal{D}$ which are images under α of vectors in $\mathcal{C}$: $\operatorname{img}(\alpha) = \{y \in \mathcal{D} | \exists x \in \mathcal{C} : \alpha(x) = y\}$. The complement of $\operatorname{img}(\alpha)$ is called the *co-kernel* $\operatorname{coker}(\alpha)$.

Definition 5. [10, p23] A bounded linear operator α between two Banach spaces $\mathcal{C}$ and $\mathcal{D}$ is *Fredholm* iff the dimensions of kernel and co-kernel are finite.

For Fredholm operators we have this existence theorem [10, Thm 1.4.15]:

Theorem 6. *Let α be a Fredholm operator α between two Banach spaces $\mathcal{C}$ and $\mathcal{D}$, then α admits a pseudo-inverse.*

2.5 Measuring Probabilistic Abstract Interpretations

The probabilistic abstract interpretation outlined in Section 2 is essentially defined in terms of projection operators, which we use to replace the notion of partial order in classical abstract interpretation. In particular, the orthogonality property of such projections allows us to conclude that the probabilistic abstractions we define are *best approximations* in the sense expressed by the following proposition [1, Corr 2.1.1],[12, 44.2.10]:

Proposition 7. *Let $\pi_{\mathcal{W}}$ be the orthogonal projection on a finite dimensional vector space $\mathcal{V}$ associated to the sub-space $\mathcal{W} \subseteq \mathcal{V}$. Then for every vector $\boldsymbol{x} \in \mathcal{V}$ its image $\pi_{\mathcal{W}}(\boldsymbol{x})$ is the unique vector in $\mathcal{W}$ such that $\|\boldsymbol{x} - \pi_{\mathcal{W}}(\boldsymbol{x})\|_2$ is minimal.*

The norm $\|\cdot\|_2$ is the Euclidean norm $\|\boldsymbol{x}\|_2 = (|x_1|^2 + \ldots + |x_n|^2)^{(1/2)}$.

As already mentioned, in our probabilistic setting we have the possibility of measuring the precision of a probabilistic abstract interpretation in numerical terms. For abstract interpretations constructed as in Section 2.4, the difference between $\boldsymbol{\gamma} \circ (\lim \Psi)(\boldsymbol{d})$ and $\boldsymbol{\gamma} \circ \boldsymbol{\alpha} \circ (\lim \Phi) \circ \boldsymbol{\gamma}(\boldsymbol{d})$ describes the approximation error and is obviously determined by the "size" of the map (or matrix) $\boldsymbol{\gamma} \circ \boldsymbol{\alpha}$. This size can be measured by means of an *operator norm*. Depending on the application, various *operator norms* could be considered. For a very general measure, estimating the worst approximation error made we can use the *operator norm* which is defined for an operator $\boldsymbol{\alpha}$ as:

$$\|\boldsymbol{\alpha}\| = \sup_{\|x\|=1} \|\boldsymbol{\alpha}(x)\|$$

i.e. as the length of the "longest" vector a unit vector may be mapped to.

3 An Example

We illustrate our approach by means of an example of abstract interpretation which was presented in [4], namely the so called "Proof by Nine" or "Cast out of Nine". The idea behind such a proof is to check the plausibility of the result of a multiplication by comparing it with the corresponding multiplication of the 'checksums' modulo some digit d; in the classical case d is the digit 9. The question we want to address in our quantitative setting is: why is $d = 9$ "better" than for example $d = 2$. A complete formal presentation of this classical plausibility check was given in [4]. For our purposes we will need only a simplified and slightly changed presentation.

3.1 The Setting

The problem is to look at a multiplication of two natural numbers, i.e. an expression of the form "$a \times b = c$" and ask if it is "correct". We can identify the set of all expressions of this form with $\mathbb{N} \times \mathbb{N} \to \mathbb{N} \equiv \mathbb{N}^3$.

Not all possible (syntactically well-formed) expressions of this type are (semantically) correct, i.e. indeed represent a valid multiplication. For example "$(2 \times 2 = 4$" is correct but "$2 \times 2 = 5$" is not. We will express the correctness of an expression by a map $\alpha_1 : \mathbb{N}^3 \mapsto \mathbb{B}$, where $\mathbb{B} = \{\mathit{true}, \mathit{false}\}$ defined as:

$$\alpha_1((a,b,c)) = \begin{cases} \mathit{true} & \text{if } c = a \cdot b \\ \mathit{false} & \text{if } c \neq a \cdot b \end{cases}$$

We implement the plausibility check by mapping each (concrete) multiplication into a multiplication expression involving the 'check sums' modulo some d, i.e. we introduce an abstraction map $\alpha : \mathbb{N}^3 \mapsto \mathbb{D}^3$ defined by

$$\alpha((a,b,c)) = (\bar{a}, \bar{b}, \bar{c})$$

where $\bar{x} = x \bmod d$.

Again we can check the correctness of an abstract multiplication expression modulo d by a mapping $\alpha_2 \mathbb{D}^3 \mapsto \mathbb{B}$ defined by:

$$\alpha_2((\bar{a}, \bar{b}, \bar{c})) = \begin{cases} \mathit{true} & \text{if } \bar{c} = \bar{a} \cdot \bar{b} \bmod d \\ \mathit{false} & \text{if } \bar{c} \neq \bar{a} \cdot \bar{b} \bmod d \end{cases}$$

The relevant observation is that the correctness of the simplified multiplication expression (over $\mathbb{D}$) is a plausibility test for the correctness of the multiplication expression over $\mathbb{N}$: If the expression over $\mathbb{D}$ is incorrect then the original expression *is* incorrect too, but if $\alpha_2 \circ \alpha$ maps an expression into *true* we can only conclude that it *may* be correct (analogously, some expressions which are mapped into *false* under α_1 may be mapped into *true* under $\alpha_2 \circ \alpha$).

In this sense the abstraction is conservative: We can detect incorrect expressions, but we cannot be sure about the expressions which pass this plausibility test. However, we can try to estimate the proportion of incorrect expressions which pass the test. This would give us a measure of the precision of the abstraction. To this purpose we look at the lifting of the situation above to the free vector space setting, and consider two maps from $\mathcal{V}(\mathbb{B})$ to $\mathcal{V}(\mathbb{B})$, namely $\boldsymbol{\alpha}_2 \circ \boldsymbol{\alpha} \circ \boldsymbol{\gamma}_1$ and $\boldsymbol{\alpha}_1 \circ \boldsymbol{\gamma} \circ \boldsymbol{\gamma}_2$ where $\boldsymbol{\gamma} = \boldsymbol{\alpha}^\dagger$, $\boldsymbol{\gamma}_1 = \boldsymbol{\alpha}_1^\dagger$, and $\boldsymbol{\gamma}_2 = \boldsymbol{\alpha}_2^\dagger$, are the Moore-Penrose pseudo-inverses of $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}_1$, and $\boldsymbol{\alpha}_2$ respectively, i.e. the quantitative concretisation functions corresponding to each abstraction function. This situation is depicted in Figure 1.

$$\begin{tikzcd}
\mathcal{V}(\mathbb{N}^3) \arrow[rr, shift left, "\boldsymbol{\alpha}"] \arrow[d, shift right, "\boldsymbol{\alpha}_1"'] & & \mathcal{V}(\mathbb{D}^3) \arrow[ll, shift left, "\boldsymbol{\gamma}"] \arrow[d, shift left, "\boldsymbol{\alpha}_2"] \\
\mathcal{V}(\mathbb{B}) \arrow[u, shift right, "\boldsymbol{\gamma}_1"'] \arrow[rr, shift left, "\boldsymbol{\alpha}_2 \circ \boldsymbol{\alpha} \circ \boldsymbol{\gamma}_1"] & & \mathcal{V}(\mathbb{B}) \arrow[u, shift left, "\boldsymbol{\gamma}_2"] \arrow[ll, shift left, "\boldsymbol{\alpha}_1 \circ \boldsymbol{\gamma} \circ \boldsymbol{\gamma}_2"]
\end{tikzcd}$$

Fig. 1. Probabilistic Abstraction and Concretisation Functions

The two maps, $\boldsymbol{\alpha}_2 \circ \boldsymbol{\alpha} \circ \boldsymbol{\alpha}_1^\dagger$, and $\boldsymbol{\alpha}_1 \circ \boldsymbol{\alpha}^\dagger \circ \boldsymbol{\alpha}_2^\dagger$ allow us to measure and compare the precision of plausibility checks with different base d, as shown in the following section.

3.2 Some Numerical Experiments

For truncated — i.e. finite dimensional — versions of the functions and matrices involved we can compute the matrices for $\boldsymbol{\alpha}_2 \circ \boldsymbol{\alpha} \circ \boldsymbol{\alpha}_1^\dagger$ and $\boldsymbol{\alpha}_1 \circ \boldsymbol{\alpha}^\dagger \circ \boldsymbol{\alpha}_2^\dagger$ explicitly.

We have done this by using the interactive numerical computation system octave [7]. We have checked 3333 randomly chosen triples $a * b = c$ with $a, b, c \in [0, 10]$. The results show that the percentage of incorrect concrete multiplications which pass the test decreases when d becomes bigger.

Each of the following matrices $\Gamma = \boldsymbol{\alpha}_1 \circ \boldsymbol{\alpha}^\dagger \circ \boldsymbol{\alpha}_2^\dagger$ represents the concretisation map for the cases $d = 2, 3, 5, 7, 9$ respectively.

For $d = 2$ $$\boldsymbol{\alpha}_1 \circ \boldsymbol{\alpha}^\dagger \circ \boldsymbol{\alpha}_2^\dagger = \begin{pmatrix} 0.06392 & 0.93607 \\ 0.00000 & 1.00000 \end{pmatrix}$$

For $d = 3$ $$\boldsymbol{\alpha}_1 \circ \boldsymbol{\alpha}^\dagger \circ \boldsymbol{\alpha}_2^\dagger = \begin{pmatrix} 0.10072 & 0.89928 \\ 0.00000 & 1.00000 \end{pmatrix}$$

For $d = 5$ $$\boldsymbol{\alpha}_1 \circ \boldsymbol{\alpha}^\dagger \circ \boldsymbol{\alpha}_2^\dagger = \begin{pmatrix} 0.15156 & 0.84844 \\ 0.00000 & 1.00000 \end{pmatrix}$$

For $d = 7$ $$\boldsymbol{\alpha}_1 \circ \boldsymbol{\alpha}^\dagger \circ \boldsymbol{\alpha}_2^\dagger = \begin{pmatrix} 0.22178 & 0.77822 \\ 0.00000 & 1.00000 \end{pmatrix}$$

For $d = 9$ $$\boldsymbol{\alpha}_1 \circ \boldsymbol{\alpha}^\dagger \circ \boldsymbol{\alpha}_2^\dagger = \begin{pmatrix} 0.27792 & 0.72208 \\ 0.00000 & 1.00000 \end{pmatrix}$$

Each matrix Γ must be applied to an abstract result (i.e. a vector in $\mathcal{V}(\mathbb{B})$) to get its concretisation (i.e. again a vector in $\mathcal{V}(\mathbb{B})$). The outcome of the plausibility check is represented by the vector $(1, 0)$ if it is *true* and $(0, 1)$ if it is *false*. The application of Γ to these vectors, that is the multiplication $(1, 0) \cdot \Gamma$ or $(0, 1) \cdot \Gamma$, results in a vector which corresponds to the first row of Γ if the abstract vector is $(1, 0)$, and to the second row of Γ if the abstract vector is $(0, 1)$. It is then easy to see that for all d, if the outcome of the plausibility test is *false*, then it indeed corresponds (with probability 1) to an incorrect concrete multiplication; however, if a multiplication passes the plausibility test — i.e. the result is *true* — the original multiplication still might be incorrect; in the particular case $d = 2$ there is a 93% chance of such an incorrect classification.

We see furthermore, that casting out of nine is "better" than casting out of two, as only 72% of the incorrect expressions pass undetected, compared to 89% for $d = 3$, 84% for $d = 5$ and 77% for $d = 7$.

4 Conclusions

We presented a quantitative version of the classical Cousot framework for abstract interpretation in which the notion of a Galois connection (which is based on order-theoretic concepts) is replaced by the notion of Moore-Penrose pseudo-inverse (of a linear map between vector spaces). This framework not only allows for the analysis of probabilistic programs, but additionally introduces the possibility of measuring the precision of an abstract interpretation.

More work needs to be devoted to the construction and combination of probabilistic abstract interpretations, in particular in the infinite-dimensional case. Not the least reason for this is that a direct computation of the Moore-Penrose pseudo-inverse can be numerically very expensive.

References

1. Stephen L. Campbell and D. Meyer. *Generalized Inverse of Linear Transformations*. Constable and Company, London, 1979.
2. Patrick Cousot and Radhia Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Symposium on Principles of Programming Languages (POPL)*, pages 238–252, Los Angeles, 1977.
3. Patrick Cousot and Radhia Cousot. Systematic Design of Program Analysis Frameworks. In *Symposium on Principles of Programming Languages (POPL)*, pages 269–282, San Antonio, Texas, 1979.
4. Patrick Cousot and Radhia Cousot. Abstract Interpretation and Applications to Logic Programs. *Journal of Logic Programming*, 13(2-3):103–180, July 1992.
5. Alessandra Di Pierro and Herbert Wiklicky. An operational semantics for Probabilistic Concurrent Constraint Programming. In P. Iyer, Y. Choo, and D. Schmidt, editors, *ICCL'98 – International Conference on Computer Languages*, pages 174–183. IEEE Computer Society Press, 1998.
6. Alessandra Di Pierro and Herbert Wiklicky. Probabilistic Concurrent Constraint Programming: Towards a fully abstract model. In L. Brim, J. Gruska, and J. Zlatuska, editors, *MFCS'98 – Mathematical Foundations of Computer Science*, volume 1450 of *Lecture Notes in Computer Science*, pages 446–455, Berlin – New York, August 1998. Springer Verlag.
7. John W. Eaton. *Octave*. Boston, MA, 1997.
8. Neil D. Jones and Flemming Nielson. Abstract interpretation: A semantic-based tool for program analysis. In S.Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Semantic Modelling*, volume 4 of *Handbook of Logic in Computer Science*, pages 527–636. Clarendon Press, Oxford, 1995.
9. Dexter Kozen. Semantics for probabilistic programs. *Journal of Computer and System Sciences*, 22:328–350, 1981.
10. Gerard J. Murphy. *C^*-Algebras and Operator Theory*. Academic Press, San Diego, 1990.
11. Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Aanalysis*. Springer Verlag, Berlin – Heidelberg, 1999.
12. Viktor V. Prasolov. *Problems and Theorems in Linear Algebra*, volume 134 of *Translation of Mathematical Monographs*. American Mathematical Society, Providence, Rhode Island, 1994.
13. Walter Rudin. *Functional Analysis*. International Series in Pure and Applied Mathematics. McGraw-Hill, 1991. Second Edition.
14. N. Saheb-Djahromi. CPO's of measures for nondeterminism. *Theoretical Computer Science*, 12:19–37, 1980.