Measuring 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 develop a methodology for constructing semantics-based analyses of programs by approximating their probabilistic semantics. The framework we introduce resembles the one based on Galois connections 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. After introducing the notion of probabilistic abstract interpretation, we define a measure of its precision by means of the norm of a linear operator which encodes the "incompleteness" of the abstraction. Finally we show the application of our results in a series of classical examples.

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 [5, 6].

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, as remarked for example in [14, Section 3.4]:

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

The essential element of a probabilistic abstract interpretation is the definition of the abstraction function $\boldsymbol{\alpha}$ as a linear operator between two vector spaces representing the concrete and abstract domains of the approximated function. The corresponding concretisation function $\boldsymbol{\gamma}$ is defined by using the notion of Moore-Penrose pseudo-inverse, which shows some important structural similarities with the Galois connection.

By interpreting in this setting the completeness conditions for classical abstract interpretation, we obtain a method which produces a numerical estimate of the (im-)precision of an abstraction. This method consists in the calculation of the norm of a linear operator comparing the two terms of the completeness conditions. We apply this method for calculating the precision of three abstract interpretations typical in the classical theory, namely the Rule of Signs for addition and multiplication and the Proof by Nine.

2 Probabilistic Abstract Interpretation

2.1 Probabilistic Semantics

Our approach towards a quantitative version of abstract interpretation has its origins in our investigation of the semantics of probabilistic languages. Although it can easily be applied also to conventional programs (e.g. the examples in Section 5) the need of introducing a quantitative semantics, in the form of a vector space based semantics, was first realised in that context.

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 of the fundamental papers by Saheb-Djahromi [20] and Kozen [15]. More recently, the authors have developed a probabilistic version of the concurrent constraint programming paradigm in [9, 10], 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 fixpoint operator $\Phi : \mathcal{V} \mapsto \mathcal{V}$, on a vector space $\mathcal{V}$ (for further details see e.g. [10, 11]).

2.2 Abstract Interpretation

Classical abstract interpretation — as defined by Cousot and Cousot — is a method for designing *approximate* semantics of programs which can be used for analysis purposes [5, 6]. 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 3.4. For the finite-dimensional case we exploit the fact that any n-dimensional vector space, with $n < \infty$, is isomorphic to $\mathbb{R}^n$ [19]. Therefore we can take, without loss of generality, the spaces $\mathbb{R}^n$ as a concrete domain and $\mathbb{R}^m$ as an abstract domain, for some positive integers n, m. 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. If there are two functions $\alpha : \mathcal{C} \mapsto \mathcal{D}$ and $\gamma : \mathcal{D} \mapsto \mathcal{C}$ such that for all $c \in \mathcal{C}$ and all $d \in \mathcal{D}$:

$$c \leq_{\mathcal{C}} \gamma(d) \text{ iff } \alpha(c) \sqsubseteq d,$$

then $(\mathcal{C}, \alpha, \gamma, \mathcal{D})$ forms a *Galois connection.*

An equivalent definition is as follows:

Definition 2. 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 analogues 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 3. 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 [19].

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 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 orthogonal 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 exactly define the so-called *Moore-Penrose pseudo-inverse*:

Definition 4. 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. $\boldsymbol{\gamma} \circ \boldsymbol{\alpha} = \pi_{\boldsymbol{\gamma}}$.

An alternative characterisation of the Moore-Penrose pseudo-inverse is given by:

Theorem 5. *Let $\mathcal{C}$ and $\mathcal{D}$ be two finite-dimensional vector spaces. For a linear map $\boldsymbol{\alpha} : \mathcal{C} \mapsto \mathcal{D}$ the Moore-Penrose pseudo-inverse $\boldsymbol{\gamma} : \mathcal{D} \mapsto \mathcal{C}$ is uniquely characterised by the following conditions:*

1. $\boldsymbol{\alpha} \circ \boldsymbol{\gamma} \circ \boldsymbol{\alpha} = \boldsymbol{\alpha}$,
2. $\boldsymbol{\gamma} \circ \boldsymbol{\alpha} \circ \boldsymbol{\gamma} = \boldsymbol{\gamma}$,
3. $(\boldsymbol{\alpha} \circ \boldsymbol{\gamma})^* = \boldsymbol{\alpha} \circ \boldsymbol{\gamma}$,
4. $(\boldsymbol{\gamma} \circ \boldsymbol{\alpha})^* = \boldsymbol{\gamma} \circ \boldsymbol{\alpha}$.

This theorem provides further evidence of the similarity between Galois connections and Moore-Penrose pseudo-inverse, since the following proposition holds for Galois connections:

Proposition 6. *Let $(\mathcal{C}, \alpha, \gamma, \mathcal{D})$ be a Galois connection. Then α and γ are* quasi-inverse*, i.e.*

1. $\alpha \circ \gamma \circ \alpha = \alpha$, *and*
2. $\gamma \circ \alpha \circ \gamma = \gamma$.

It would be interesting to investigate the properties of Galois connections which correspond to conditions 3. and 4. in Theorem 5. We believe that an appropriate setting for these investigations is category theory, and we plan to develop such studies in a future work.

By using the notion Moore-Penrose pseudo-inverse, we can now define a *probabilistic abstract interpretation* as follows:

Definition 7. Let $\mathcal{C}$ an $\mathcal{D}$ be two probabilistic domains. A probabilistic abstract interpretation is a pair of linear maps, $\boldsymbol{\alpha} : \mathcal{C} \mapsto \mathcal{D}$ and $\boldsymbol{\gamma} : \mathcal{D} \mapsto \mathcal{C}$, between (the concrete domain) $\mathcal{C}$ and (the abstract domain) $\mathcal{D}$, such that $\boldsymbol{\gamma}$ is the Moore-Penrose pseudo-inverse of $\boldsymbol{\alpha}$, and vice versa.

Example 8. 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:

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

In a probabilistic abstract interpretation the notion of projection operators is essentially used to replace the notion of partial order in classical abstract interpretation. The orthogonality property of such projections allows us furthermore to conclude that the probabilistic abstract interpretation we define in Definition 7 are the *best* in the sense that the pair $(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ is among all the other pairs $(\boldsymbol{\alpha}, \boldsymbol{\gamma}')$ (which we could possibly define) the one mapping the abstraction $\boldsymbol{\alpha}(\boldsymbol{d})$ to the concrete vector closest to $\boldsymbol{d}$. This is justified by the fact that $\boldsymbol{\gamma} \circ \boldsymbol{\alpha}$ is an orthogonal projection and by the following proposition [2, Corr 2.1.1], [18, 44.2.10]:

Proposition 9. *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.*

There are various possible measures for the length, i.e. *norm* of a vector, and different norms would correspond to different metrics on $\mathcal{V}$. In general, a whole family of *p-norms* can be defined by:

$$\|\boldsymbol{x}\|_p = \|(x_c)_{c \in \mathcal{C}}\|_p = \left(\sum_{c \in \mathcal{C}} |x_c|^p \right)^{1/p}$$

Although in Proposition 9 we refer to a particular norm, namely the Euclidean norm or 2-norm, $\|\boldsymbol{x}\|_2 = (|x_1|^2 + \ldots + |x_n|^2)^{(1/2)}$, in a finite-dimensional space the choice of one norm or another is not relevant for topological considerations, as they all induce the same topology [19].

3 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. [17, 7].

3.1 Probabilistic Induced Analysis

The construction we discuss first is that of a fixpoint abstraction using Moore-Penrose pseudo-inverses (for further details see [11]). The same construction using Galois connections gives in the classical case so called *induced analysis*.

We assume that we have a probabilistic semantics in terms of a (concrete) fixpoint operator Φ on some vector space $\mathcal{V}$, as mentioned in Section 2.1. Furthermore, assume that there is a linear abstraction function $\boldsymbol{\alpha} : \mathcal{V} \mapsto \mathcal{W}$ from the concrete domain into an abstract domain $\mathcal{W}$.

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 fixpoint operator on the abstract domain $\mathcal{W}$, obtained by a formal computation starting from the concrete linear fixpoint operator $\Phi : \mathcal{V} \mapsto \mathcal{V}$ 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.

3.2 Vector Space Lifting

In order to apply our probabilistic abstract interpretation methodology we may need to recast a cpo based semantics in a vector space setting. A straightforward method to *lift* a cpo to a vector space — similar to lifting a semantics to a power space setting in the classical framework — is based on a free vector space construction.

Given a classical semantic domain (cpo) $\mathcal{C}$, we define 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. [17, 4.3]), can be then *lifted* to a linear map $\boldsymbol{\alpha} : \mathcal{V}(\mathcal{C}) \mapsto \mathcal{V}(\mathcal{D})$ as follows:

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

Remark Note that the probabilistic version of an induced fixpoint semantics in Section 3.1 can also be seen as arising from a lifting construction. In fact, the domains $\mathcal{V}$ and $\mathcal{W}$ we investigated in [11] are indeed constructed as free vector spaces on the underlying abstract and and concrete lattices of constraints.

3.3 General Construction

In the construction of Section 3.1 we examined the special case of a fixpoint abstraction, which represents a typical framework for the analyses of programming languages. The same construction applies also to the the general case of semantical functions $f : \mathcal{A} \mapsto \mathcal{B}$. In this case, a classical abstract interpretation is given by two abstraction functions, α and α', mapping the input domain $\mathcal{A}$ and the output domain $\mathcal{B}$ into an abstract input domain $\mathcal{A}^\#$ and an abstract output domain $\mathcal{B}^\#$ respectively, and by an abstract semantical functions $f^\# : \mathcal{A}^\# \mapsto \mathcal{B}^\#$. The following diagram illustrates this case in the classical cpo-based framework.

$$\begin{tikzcd}
\mathcal{A} \arrow[r, shift left, "\alpha"] \arrow[d, "f"'] & \mathcal{A}^\# \arrow[l, shift left, "\gamma"] \arrow[d, "f^\#"] \\
\mathcal{B} \arrow[r, shift left, "\alpha'"] & \mathcal{B}^\# \arrow[l, shift left, "\gamma'"]
\end{tikzcd}$$

In the general scenario described above (which includes fixpoint abstractions as particular cases) an abstract function $f^\# : \mathcal{A}^\# \mapsto \mathcal{B}^\#$ is a *correct* approximation of a concrete function $f : \mathcal{A} \mapsto \mathcal{B}$ if

$$\alpha' \circ f \leq_{\mathcal{B}^\#} f^\# \circ \alpha,$$

where $\leq_{\mathcal{B}^\#}$ is the order relation on the abstract domain $\mathcal{B}^\#$.

Given two abstract domains $\mathcal{A}^\#$ and $\mathcal{B}^\#$, there always exists a *best correct approximation* of a concrete function $f : \mathcal{A} \mapsto \mathcal{B}$. This is defined by $f^\# = \alpha' \circ f \circ \gamma$.

However, the choice of the best correct approximation of a function does not guarantee in general that the results of an abstract computation are best approximations of the corresponding concrete ones. Whenever this happens then the abstract interpretation is complete. Formally, an abstract function $f^\# : \mathcal{A}^\# \mapsto \mathcal{B}^\#$ is a *complete* approximation of a concrete function $f : \mathcal{A} \mapsto \mathcal{B}$ if

$$\alpha' \circ f = f^\# \circ \alpha.$$

Unlike the correctness condition, completeness is not an essential requirement, but rather an ideal situation which does not occur very often in practice. In the literature it is often also referred to as *exactness* [4] or *optimality* [8].

By applying the same lifting technique described in Section 3.2 to both abstract domains $\mathcal{A}^\#$ and $\mathcal{B}^\#$ we can construct a probabilistic abstract interpretation in this general case too. Obviously, we will have now to compute two Moore-Penrose pseudo-inverses, namely $\boldsymbol{\gamma} = \boldsymbol{\alpha}^\dagger$ for $\boldsymbol{\alpha}$, and $\boldsymbol{\gamma}' = \boldsymbol{\alpha}'^\dagger$ for $\boldsymbol{\alpha}'$. In the following we will denote a probabilistic abstract interpretation of a function f by specifying the abstract domains and function, $(\mathcal{A}^\#, \mathcal{B}^\#, f^\#)$.

3.4 Infinite-Dimensional Abstractions

The construction of the Moore-Penrose pseudo-inverse, as presented here, is essentially based on *finite-dimensional* vector spaces. Although this gives an elegant quantitative analysis for finite domains, one is more than often confronted with an infinite concrete domain. For example a simple parity analysis 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 by defining our abstractions as *Fredholm operators* [1]. We just sketch here the basic argument and definitions.

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.

$$\ker(\alpha) = \{x \in \mathcal{C} \mid \alpha(x) = o\}.$$

The *image* of α is the set of vectors in $\mathcal{D}$ which are images under α of vectors in $\mathcal{C}$:

$$\mathrm{img}(\alpha) = \{y \in \mathcal{D} \mid \exists x \in \mathcal{C} : \alpha(x) = y\}.$$

The complement of $\mathrm{img}(\alpha)$ is called the *co-kernel* $\mathrm{coker}(\alpha)$.

Definition 10. [16, 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 the following theorem holds [16, Thm 1.4.15]:

Theorem 11. *Let α be a Fredholm operator α between two Banach spaces $\mathcal{C}$ and $\mathcal{D}$, then α admits a pseudo-inverse.*

It is relatively easy to guarantee that a lifted abstraction $\boldsymbol{\alpha} : \mathcal{V}(C) \mapsto \mathcal{V}(D)$ stemming from an extraction function $\alpha : C \mapsto D$ (see Section 3) is a Fredholm operator and thus possesses a pseudo-inverse. If the original extraction function α is defined for all elements in C, i.e. if α is a total function, then the kernel is always the empty set, as no element in $\mathcal{V}(C)$ will be mapped to the null vector. If this is not the case, a partial α can be made total in the usual way, namely by adding an "undefined" element $\perp$ to D. On the other hand, if α is a surjective function on D then the co-kernel of $\boldsymbol{\alpha}$ is empty. Otherwise, we can replace D by the image of C, $D' = \alpha(D) \subseteq D$. In other words, for lifted abstractions it is always possible to define $\boldsymbol{\alpha}$ such that $\text{img}(\boldsymbol{\alpha}) = \emptyset$ and $\text{coker}(\boldsymbol{\alpha}) = \emptyset$.

A common case is when only one of the two domains is infinite-dimensional, namely the concrete one. One example is the parity test, where we abstract from an infinite domain to a finite one containing just two representatives for the even and the odd numbers respectively, and the undefined element. This situation is especially simple. Since the abstract domain is finite-dimensional the co-kernel condition is trivially fulfilled. On the other hand, the extraction function is total — e.g. every natural number has a well-defined parity, either even or odd — and thus the kernel is empty.

4 Measuring Probabilistic Abstract Interpretations

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 3.1, the difference between $\boldsymbol{\gamma} \circ (\lim \Psi)(\boldsymbol{d})$ and $(\lim \Phi) \circ \boldsymbol{\gamma}(\boldsymbol{d})$ 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*, which allows us to estimate the worst approximation error made. This norm is defined for an operator $\boldsymbol{\beta}$ as:

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

i.e. as the length of the "longest" vector a unit vector may be mapped to. The meaning of an operator norm for a linear map $\boldsymbol{\beta} : V \mapsto W$ is thus the largest value by which $\boldsymbol{\beta}$ stretches an element of V. Depending on the application we could choose a particular vector norm $\|x\|$. In the examples below we will use $\|x\|_1$ for calculating some numerical results, as it corresponds to a kind of simple average.

As already pointed out, the assumption of a finite-dimensional vector space makes it not relevant the choice of a particular vector norm $\|x\|$. In the examples below we will use $\|x\|_1$ for calculating some numerical results, but the choice of any other p-norm would lead us essentially to the same conclusions.

In the general case of a generic semantic function $f : \mathcal{A} \mapsto \mathcal{B}$ described in Section 3.3, it seems reasonable to measure the precision of a probabilistic abstract interpretation defined by $(\mathcal{A}^{\#}, \mathcal{B}^{\#}, f^{\#})$ by means of the norm of the "pseudo-quotient":

$$\Delta = (\boldsymbol{\alpha}' \circ f) \circ (f^{\#} \circ \boldsymbol{\alpha})^{\dagger}.$$

An estimate of the precision of the abstraction is then given by calculating the norm $\|\Delta\|$ of the linear operator Δ.

The operator $\Delta : \mathcal{B}^{\#} \mapsto \mathcal{B}^{\#}$ is an operator on $\mathcal{B}^{\#}$, mapping an abstract vector $\boldsymbol{b}_1 \in \mathcal{B}^{\#}$ into another abstract vector $\boldsymbol{b}_2$ resulting from abstracting the result of $f(\boldsymbol{a})$, where $\boldsymbol{a}$ is the concrete vector such that $f^{\#} \circ \boldsymbol{\alpha}(a) = \boldsymbol{b}_1$. Therefore, the norm $\|\Delta\|$ gives us a measure of the approximation introduced by $(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ in the form of an expansion factor expressing how much the abstraction $\boldsymbol{b}_1$ can be bigger than $\boldsymbol{b}_2$.

According to this notion of precision we can define correctness and completeness of probabilistic abstract interpretations as follows:

Definition 12. Let $(\mathcal{A}^{\#}, \mathcal{B}^{\#}, f^{\#})$ be a probabilistic abstract interpretation, and let Δ be the associated pseudo-quotient operator. Then

Correctness: $(\mathcal{A}^{\#}, \mathcal{B}^{\#}, f^{\#})$ is *correct* iff $\|\Delta\| \geq 1$;
Completeness: $(\mathcal{A}^{\#}, \mathcal{B}^{\#}, f^{\#})$ is *complete* iff $\|\Delta\| = 1$.

This definition is intuitively justified by re-interpreting the correctness and completeness conditions of the classical theory in the probabilistic setting.

As mentioned at the beginning of this section, for the fixpoint abstraction the approximation error is measured by $\|\boldsymbol{\gamma} \circ \boldsymbol{\alpha}\|$, and we would expect that such an abstract interpretation is complete when $\|\boldsymbol{\gamma} \circ \boldsymbol{\alpha}\| = 1$ The following proposition shows that this is coherent with Definition 12.

Proposition 13. *Let $(\mathcal{A}^{\#}, \Psi)$ be a probabilistic abstract interpretation as in Section 3.1. Then,*

$$\|\boldsymbol{\gamma} \circ \boldsymbol{\alpha}\| = 1 \text{ iff } \|\Delta\| = 1,$$

where Δ is the pseudo-quotient $(\boldsymbol{\alpha}' \circ f) \circ (f^{\#} \circ \boldsymbol{\alpha})^{\dagger}$.

5 Examples

In this section we illustrate our approach by means of some examples, which are well understood in the classical case, namely the "Rule of Sign for Multiplications" (Example 5.1), the "Rule of Sign for Additions" (Example 5.2), and the "Proof by Nine" or "Cast out of Nine" (Example 5.3). We refer to [7] for the classical counterpart of these examples.

5.1 Rule of Sign: Multiplication

The Setting The problem of multiplying two integers a and b is abstracted to the problem of determining the sign of $a \times b$, given the sign of its factors.

$$\begin{tikzcd}
\mathcal{V}(\mathbb{Z}^2) \arrow[r, "\boldsymbol{\alpha}^2"] \arrow[d, "\times"'] & \mathcal{V}(\mathrm{Sign}^2) \arrow[d, "\times^\#"] \\
\mathcal{V}(\mathbb{Z}) \arrow[r, "\boldsymbol{\alpha}"] & \mathcal{V}(\mathrm{Sign})
\end{tikzcd}$$

Fig. 1. Lifting and Abstracting Multiplication

$\times^\#$	$\bot$	$-$	0	$+$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$-$	$\bot$	$+$	0	$-$	$\top$
0	$\bot$	0	0	0	$\top$
$+$	$\bot$	$-$	0	$+$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$

$+^\#$	$\bot$	$-$	0	$+$	$\top$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
$-$	$\bot$	$-$	$-$	$\top$	$\top$
0	$\bot$	$-$	0	$+$	$\top$
$+$	$\bot$	$\top$	$+$	$+$	$\top$
$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$

Fig. 2. Abstract Multiplication and Addition

The abstract domain of signs is given by $\mathrm{Sign} = \{\bot, -, 0, +, \top\}$, where $\bot$ can be read as "undefined" and $\top$ as "any sign". The abstract multiplication $\times^\# : \mathrm{Sign} \mapsto \mathrm{Sign}$ is defined in Figure 2.

The problem is therefore to relate the concrete multiplication $\times$ with the abstract multiplication $\times^\#$, where the abstraction function $\alpha : \mathbb{Z} \mapsto \mathrm{Sign}$ is simply the sign of an integer.

Numerical Results The situation — lifted to the vector space level — is depicted in Figure 1. As $\bot$ and $\top$ are largely irrelevant for our considerations we will ignore them in the following and identify $\mathcal{V}(\mathrm{Sign})$ with $\mathbb{R}^3$. The basis vectors for the vector space $\mathcal{V}(\mathrm{Sign}) = \mathbb{R}^3$ are identified with the signs $-, 0, +$:

$$\begin{aligned}
- &\longrightarrow (1,0,0)\\
0 &\longrightarrow (0,1,0)\\
+ &\longrightarrow (0,0,1)
\end{aligned}$$

Analogously, pairs of signs form a basis for the vector space $\mathcal{V}(\mathrm{Sign}^2) = \mathbb{R}^6$:

$$\begin{aligned}
- \times - &\longrightarrow (1,0,0,0,0,0)\\
- \times 0 &\longrightarrow (0,1,0,0,0,0)\\
- \times + &\longrightarrow (0,0,1,0,0,0)\\
0 \times 0 &\longrightarrow (0,0,0,1,0,0)\\
0 \times + &\longrightarrow (0,0,0,0,1,0)\\
+ \times + &\longrightarrow (0,0,0,0,0,1).
\end{aligned}$$

Given the concrete representation of the vector spaces, we can construct the matrices representing $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$, etc. The problem is that we actually had to consider

$$\begin{tikzcd}
\mathcal{V}(\mathbb{Z}^2) \arrow[r, "\boldsymbol{\alpha}^2"] \arrow[d, "+"'] & \mathcal{V}(\mathrm{Sign}^2) \arrow[d, "+^{\#}"] \\
\mathcal{V}(\mathbb{Z}) \arrow[r, "\boldsymbol{\alpha}"] & \mathcal{V}(\mathrm{Sign})
\end{tikzcd}$$

Fig. 3. Lifting and Abstracting Addition

infinite-dimensional vector spaces $\mathcal{V}(\mathbb{Z})$ and $\mathcal{V}(\mathbb{Z}^2)$, but one can easily see that any "truncation", i.e. replacing $\mathcal{V}(\mathbb{Z})$ by $\mathcal{V}(\{-n, -n+1\ldots, 0, \ldots, n-1, n\}) = \mathbb{R}^{(2n+1)}$ and $\mathcal{V}(\mathbb{Z}^2)$ by $\mathbb{R}^{(2n+1)^2}$, will lead to the same pseudo-quotient of $\boldsymbol{\alpha} \circ \times$ and $\times^{\#} \circ \boldsymbol{\alpha}^2$, namely:

$$(\boldsymbol{\alpha} \circ \times) \circ (\times^{\#} \circ \boldsymbol{\alpha}^2)^{\dagger} = \begin{pmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{pmatrix}$$

According to the fact that the rule of sign, i.e. $\times^{\#}$, is a **complete** approximation of the multiplication $\times$, we obtain as a measure of precision:

$$\|(\boldsymbol{\alpha} \circ \times) \circ (\times^{\#} \circ \boldsymbol{\alpha}^2)^{\dagger}\|_1 = 1$$

5.2 Rule of Sign: Addition

The Setting The problem is very similar to the one above, the difference being that now our interest is in determining the sign of the addition of two integers instead of their multiplication. Again we have the abstract domain: $\mathrm{Sign} = \{\bot, -, 0, +, \top\}$.

Contrary to the previous example, we now do not have a complete abstraction: It is not possible to know the sign of a sum just by looking at the signs of the factors. Therefore, we get an abstract addition function $+^{\#}$ where we eventually have to leave it open which sign the sum will have, indicating this by using the abstract "sign" $\top$ in Figure 2. The undefined "sign" $\bot$ is irrelevant as we consider only well-defined concrete additions. The basis vectors for the vector space $\mathcal{V}(\mathrm{Sign}) = \mathbb{R}^4$ are identified with the signs $-, 0, +, \top$:

$$\begin{aligned} - &\longrightarrow (1,0,0,0) \\ 0 &\longrightarrow (0,1,0,0) \\ + &\longrightarrow (0,0,1,0) \\ \top &\longrightarrow (0,0,0,1). \end{aligned}$$

$$Q_2 = \begin{pmatrix} 1.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 1.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 1.00000 & 0.00000 \\ 0.25000 & 0.50000 & 0.25000 & 0.00000 \end{pmatrix}$$

$$Q_5 = \begin{pmatrix} 1.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 1.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 1.00000 & 0.00000 \\ 0.40000 & 0.20000 & 0.40000 & 0.00000 \end{pmatrix}$$

$$Q_{10} = \begin{pmatrix} 1.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 1.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 1.00000 & 0.00000 \\ 0.45000 & 0.10000 & 0.45000 & 0.00000 \end{pmatrix}$$

$$Q_{20} = \begin{pmatrix} 1.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 1.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 1.00000 & 0.00000 \\ 0.47500 & 0.05000 & 0.47500 & 0.00000 \end{pmatrix}$$

$$Q_{30} = \begin{pmatrix} 1.00000 & 0.00000 & 0.00000 & 0.00000 \\ 0.00000 & 1.00000 & 0.00000 & 0.00000 \\ 0.00000 & 0.00000 & 1.00000 & 0.00000 \\ 0.48333 & 0.03333 & 0.48333 & 0.00000 \end{pmatrix}$$

Fig. 4. The pseudo-quotients Q_n for $n = 2$, $n = 5$, $n = 10$, $n = 20$, $n = 30$.

Numerical Results Figure 3 again depicts the overall situation. In this case, considering a truncation instead of the actual infinite-dimensional vector spaces, $\mathcal{V}(\mathbb{Z})$ and $\mathcal{V}(\mathbb{Z}^2)$, does not lead to the same result. In fact, the "truncations" $\mathcal{V}([-n,n])$ and $\mathcal{V}([-n,n]^2)$ for different n lead to different pseudo-quotients. Therefore, we need to consider all of them and look at their limit. Some of these truncated pseudo-quotients Q_n for $n = 2$, $n = 5$, $n = 10$, $n = 20$, $n = 30$ are depicted in Figure 4. The sequence of the pseudo-quotients $Q_n = (\boldsymbol{\alpha} \circ +) \circ (+^{\#} \circ \boldsymbol{\alpha}^2)^{\dagger}$, where we consider only pairs in $[-n,n]$, converges to:

$$\lim_{n\to\infty} Q_n = \begin{pmatrix} 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 \\ 0.50 & 0.00 & 0.50 & 0.00 \end{pmatrix}$$

The 1-norm this time is larger than 1, namely:

$$\| \lim_{n\to\infty} Q_n \|_1 = 1.50 = 1\frac{1}{2},$$

which is indicating the fact that the rule of sign for the addition i.e. $+^{\#}$, is a (correct) non-complete abstraction of the concrete addition, i.e. $+$.

5.3 Cast out of Nine

As another example — actually a whole set of examples — we now investigate a probabilistic version of 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 'check-sums' 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 [7]. For our purposes we will need only a simplified and slightly changed presentation.

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 check 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 $eval : \mathbb{N}^3 \mapsto \mathbb{B}$, where $\mathbb{B} = \{\bot = true, false = \top\}$ defined as:

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

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 $eval^{\#} : \mathbb{D}^3 \mapsto \mathbb{B}$ defined by:

$$eval^{\#}(\text{“}\bar{a} \cdot \bar{b} = \bar{c}\text{”}) = eval^{\#}((\bar{a}, \bar{b}, \bar{c})) = \begin{cases} true & \text{if } \bar{c} = \bar{a} \cdot \bar{b} \bmod d \\ false & \text{if } \bar{c} \neq \bar{a} \cdot \bar{b} \bmod d \end{cases}$$

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 $eval^{\#} \circ \alpha$ maps an expression into *true* we can only conclude that it *may* be correct (in other words, some expressions which are mapped into *false* under *eval* may be mapped into *true* under $eval^{\#} \circ \alpha$).

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

$$
\begin{tikzcd}
\mathcal{V}(\mathbb{N}^3) \arrow[r, "\boldsymbol{\alpha}"] \arrow[d, "\boldsymbol{eval}"'] & \mathcal{V}(\mathbb{D}^3) \arrow[d, "\boldsymbol{eval}^{\#}"] \\
\mathcal{V}(\mathbb{B}) \arrow[r, "\boldsymbol{id}"] & \mathcal{V}(\mathbb{B})
\end{tikzcd}
$$

Fig. 5. Abstraction and Concretisation of "Cast by d"

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. The measure of precision is again the pseudo-quotient S_d between $\boldsymbol{id} \circ \boldsymbol{eval} = \boldsymbol{eval}$ and $\boldsymbol{eval}^{\#} \circ \boldsymbol{\alpha}$. This situation is depicted in Figure 5.

Some Numerical Experiments For truncated — i.e. finite-dimensional — versions of the functions involved we can compute the matrices involved explicitly. Therefore we again have to overcome the problem that $\mathcal{V}(\mathbb{N})$ is infinite-dimensional. This time we do this by an experimental method, in the spirit of various Monte-Carlo techniques. Concretely, we have checked 3333 randomly chosen triples $a * b = c$ with $a, b, c \in [0, 10]$. This was implemented using the interactive numerical computation system octave [12].

The results show that the percentage of incorrect concrete multiplications which pass the test decreases when d becomes larger. That means that the precision of "Proof by d" increases as d gets larger. Each of the following matrices $S_d = (id \circ \boldsymbol{eval}) \circ (\boldsymbol{eval}^{\#} \circ \boldsymbol{\alpha})^{\dagger}$ for the cases $d = 2, 3, 5, 7, 9$ respectively, shows the results of our experiments.

$$
\|S_2\|_1 = \left\| \begin{pmatrix} 0.06392 & 0.93607 \\ 0.00000 & 1.00000 \end{pmatrix} \right\|_1 = 1.93
$$

$$
\|S_3\|_1 = \left\| \begin{pmatrix} 0.10072 & 0.89928 \\ 0.00000 & 1.00000 \end{pmatrix} \right\|_1 = 1.89
$$

$$
\|S_5\|_1 = \left\| \begin{pmatrix} 0.15156 & 0.84844 \\ 0.00000 & 1.00000 \end{pmatrix} \right\|_1 = 1.84
$$

$$
\|S_7\|_1 = \left\| \begin{pmatrix} 0.22178 & 0.77822 \\ 0.00000 & 1.00000 \end{pmatrix} \right\|_1 = 1.78
$$

$$
\|S_9\|_1 = \left\| \begin{pmatrix} 0.27792 & 0.72208 \\ 0.00000 & 1.00000 \end{pmatrix} \right\|_1 = 1.72
$$

The maps $S_d : \mathcal{V}(\mathbb{B}) \mapsto \mathcal{V}(\mathbb{B})$, are represented with respect to the following basis vectors for $\mathcal{V}(\mathbb{B})$:

$$\begin{array}{rl} \textit{true} & \longrightarrow (1,0) \\ \textit{false} & \longrightarrow (0,1) \end{array}$$

and with the following convention concerning the rows and columns of these matrices:

Row Indices correspond to the boolean values for the **abstract** results, and
Columns Indices correspond to the boolean values for the **concrete** results.

Each matrix S_d 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 S_d to these vectors, that is the multiplication $(1,0) \cdot S_d$ or $(0,1) \cdot S_d$, results in a vector which corresponds to the first row of S_d if the abstract vector is $(1,0)$, and to the second row of S_d 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 $d = 7$.

6 Conclusions and Related Work

We presented a quantitative version of the classical Cousot and Cousot framework for abstract interpretation in which the notion of a Galois connection (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 [11], but additionally introduces the possibility of measuring the precision of an abstract interpretation.

Various authors have investigated the problem of comparing abstract interpretations with respect to their precision, starting from Cousot and Cousot, who in [5] introduce a partial ordering relation to encode the levels of abstraction of the interpretations. Successive work has concentrated on the study of abstract domains as representatives of their corresponding abstract interpretations, and operators on domains have been defined which allow for combining domains in order to enhance their precision (see [13] and references there in). Also in this case the comparison is expressed by means of a partial ordering (in fact a complete lattice) on the set of the domains. A slightly different approach is the one in [3] where domains are compared by means of a notion of *quotient* with respect to the property which is to be analysed. This approach seems to be closer to ours in as far as it is also based on the idea that the level of precision of an abstract interpretation can be estimated by measuring the "gap" missing for completeness. Contrary to our pseudo-quotient, their notion is still only qualitative and cannot give a numerical estimate as a result of such a measuring.

The possibility of deciding which of various available abstract interpretations is most appropriate for a given problem is of course an important advantage for those who have to analyse the problem. This explains the reason why so much work has been devoted to the study of the comparison of abstract interpretations. A common feature of all these previous approaches is the characterisation of the precision of an abstract interpretation or (equivalently) an abstract domain from a purely qualitative viewpoint. The significance of our work in this framework is in the consideration of a *quantitative* estimate of the precision of an abstract interpretation as opposed to the qualitative one considered up to now. This numerical quantification of the precision is an additional information which the user can effectively exploit in the choice of the appropriate abstract interpretation in order to optimise the balance accuracy/cost of an analysis. For example, if an analysis which requires a considerable implementational cost is just 0.02% more precise than another which is easier or less expensive then she/he could opt for the latter. Note that in a classical qualitative framework the first analysis would be regarded as "just" more precise than the other ignoring "how much" more precise it is. This could force the choice of the more expensive one when accuracy is a primary issue even thought the actual loss of precision is insignificant.

More work needs to be devoted to the construction and combination of probabilistic abstract interpretations along the lines of [17, 7], 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. Frederick J. Beutler. The operator theory of the pseudo-inverse. *Journal of Mathematical Analysis and Applications*, 10:451–470,471–493, 1965.
2. Stephen L. Campbell and D. Meyer. *Generalized Inverse of Linear Transformations*. Constable and Company, London, 1979.
3. A. Cortesi, G. File', and W. Winsborough. The Quotient of an Abstract Interpretation. *Theoretical Computer Science*, 202(1–2):163–192, 1998.
4. Patrick Cousot. Constructive design of a Hierarchy of Semantics of a Transition System by Abstract Interpretation. In S. Brooks and M. Mislove, editors, *13th International Symposium on Mathematical Foundations of Programming Semantics (MFPS97)*, volume 6 of *Electronic Notes in Theoretical Computer Science*, Amsterdam, 1997. Elsevier.
5. 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.
6. 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.

7. Patrick Cousot and Radhia Cousot. Abstract Interpretation and Applications to Logic Programs. *Journal of Logic Programming*, 13(2-3):103–180, July 1992.
8. D. Dams, R. Gerth, and O. Grumberg. Abstract Interpretations of Reactive Systems. *ACM Trans. Program. Lang. Syst.*, 19(2):253–291, 1997.
9. 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.
10. 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.
11. Alessandra Di Pierro and Herbert Wiklicky. Concurrent Constraint Programming: Towards Probabilistic Abstract Interpretation. In M. Gabbrielli and F. Pfenning, editors, *Proceedings of PPDP'00 – Priciples and Practice of Declarative Programming*, pages 127–138, Monteéal, Canada, September 2000. ACM SIGPLAN, Association of Computing Machinery.
12. John W. Eaton. *Octave*. Boston, MA, 1997.
13. R. Giacobazzi and F. Ranzato. Completeness in abstract interpretation: A domain perspective. In M. Johnson, editor, *Proc. of the 6th International Conference on Algebraic Methodology and Software Technology (AMAST'97)*, volume 1349 of *Lecture Notes in Computer Science*, pages 231–245. Springer-Verlag, Berlin, 1997.
14. 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.
15. Dexter Kozen. Semantics for probabilistic programs. *Journal of Computer and System Sciences*, 22:328–350, 1981.
16. Gerard J. Murphy. *C^*-Algebras and Operator Theory*. Academic Press, San Diego, 1990.
17. Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Aanalysis*. Springer Verlag, Berlin – Heidelberg, 1999.
18. Viktor V. Prasolov. *Problems and Theorems in Linear Algebra*, volume 134 of *Translation of Mathematical Monographs*. American Mathematical Society, Providence, Rhode Island, 1994.
19. Walter Rudin. *Functional Analysis*. International Series in Pure and Applied Mathematics. McGraw-Hill, 1991. Second Edition.
20. N. Saheb-Djahromi. CPO's of measures for nondeterminism. *Theoretical Computer Science*, 12:19–37, 1980.