Probabilistic Analysis of Programs: A Weak Limit Approach

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

[1] Dipartimento di Informatica, Università di Verona, Strada le Grazie, 15 – 37134 Verona, Italy
[2] Department of Computing, Imperial College London, 180 Queen's Gate – London SW7 2AZ, UK

1 Introduction

In probabilistic analysis we can distinguish two different situations, namely the case in which the program is deterministic and the input data varies according to some probability distribution, and the case in which the program is probabilistic and we evaluate its behaviour in different executions determined by the probability distribution on the same input data.

In this paper we present a general framework for the probabilistic static analysis of quantitative properties, that can cover both aspects and in particular includes the treatment of 'average case' analyses as a special instance of a wider range of applications centred on the quantification of resource consumption. This is similar to the algorithmic setting where probabilistic analysis, or average-case analysis, refers to the estimation of the computational complexity of an algorithm starting from an assumption about a probabilistic distribution of the set of all possible inputs and the analysis results are then used for the design of efficient algorithms. The approach we follow is a semantics based analysis where the programs semantics is defined formally so as to express the various resources via the observable behaviour of the program.

As the possible behaviours of a given program are often 'too many', the complexity of the analysis of the possible executions of some code often suffers from the problem of combinatorial explosion (even when un-decidability is not involved). At the centre of numerous approaches to program analysis is therefore the attempt of a "simplification" or "abstraction" of programs and their possible executions, in particular the abstraction of the concrete state space to a substantially simpler one.

In previous work [1–3] we have introduced a framework for probabilistic analysis based on least square approximation, which achieves such a simplification. We have called the general methodology Probabilistic Abstract Interpretation (PAI) for its strong analogy with the theory of Abstract Interpretation [4]. PAI aims in constructing statistical or average estimates of program properties which is usable for resource optimisation, in the same way as in algorithmic complexity a probabilistic analysis might aim at estimating the average-case computational complexity of an algorithm to the purpose of improving its efficiency.

Contrary to the classical abstract interpretation based static analysis, PAI-based analyses are not necessarily safe, i.e. it is not guaranteed that all accepted behaviours are conform to a given specification. They are rather guaranteed to be as close as possible to the concrete properties. This is because the objective of a PAI based analysis is 'performance' rather than 'correctness', and applications are the optimisation of resource usage rather than the construction of bounds for the (minimal or maximal) probability that something happens. As very often probability bounds tend to be 0 or 1 they are only of limited use in a context where optimisation is the main issue.

In previous work we have studied various aspects of the PAI framework for large but still *finite* state spaces. The central contribution of this paper is to extend PAI to *infinite* state spaces, by facing the mathematical problems which arise and showing how they can be overcome. In particular, we will present a weak limit construction for abstraction operators on infinite domains that will allow us to deal with the problem of the unboundedness of such operators.

Mathematical Background. For the mathematical notions and notation used in this paper we refer to the standard literature and in particular to the recent monograph by Kubrusly [5] with regard to functional analytical and operator algebraic concepts. We will only recall here some basic notions that are essential for the comprehension of the results we will present.

The concrete Banach and Hilbert spaces we consider here are spaces ℓ_p of infinite sequences of real numbers $(x_i)_{i\in\mathbb{N}}$ for which $(\sum_{i\in\mathbb{N}} |x_i|^p)^{1/p} < \infty$ with $p = 1$ and $p = 2$, respectively. The space ℓ_1 is the standard example of a Banach space; in particular it contains (as the set of positive normalised elements) all probability distributions on $\mathbb{N}$. The space ℓ_2 is the standard example of a real Hilbert space with inner product $\langle (x_i), (y_i)\rangle = \sum_i x_i y_i$.

We also recall the definition of the three main topologies or notions of convergence on Hilbert spaces, namely the norm or uniform convergence, denoted $\mathbf{A}_n \to \mathbf{A}$ or $\lim_n \mathbf{A}_n = \mathbf{A}$, the strong operator topology, $\mathbf{A}_n \overset{s}{\to} \mathbf{A}$ or $s\text{-}\lim_n \mathbf{A}_n = \mathbf{A}$ (e.g. [5, Def 4.45]) and the weak topology $\mathbf{A}_n \overset{w}{\to} \mathbf{A}$ or $w\text{-}\lim_n \mathbf{A}_n = \mathbf{A}$ defined by:

$$\begin{aligned}
&\mathbf{A}_n \to \mathbf{A} \text{ iff } \|\mathbf{A}_n - \mathbf{A}\| \to 0\\
&\mathbf{A}_n \overset{s}{\to} \mathbf{A} \text{ iff } \|(\mathbf{A}_n - \mathbf{A})(x)\| \to 0\\
&\mathbf{A}_n \overset{w}{\to} \mathbf{A} \text{ iff } \langle (\mathbf{A}_n - \mathbf{A})(x), y\rangle \to 0
\end{aligned}$$

for all $x, y \in \mathcal{H}$. Note that the norm topology is defined in terms of the operator norm, while the strong topology in terms of the vector norm on $\mathcal{H}$. We have $\mathbf{A}_n \to \mathbf{A}$ implies $\mathbf{A}_n \overset{s}{\to} \mathbf{A}$ and $\mathbf{A}_n \overset{s}{\to} \mathbf{A}$ implies $\mathbf{A}_n \overset{w}{\to} \mathbf{A}$, but not vice versa.

2 The Language

We will discuss our framework by referring to a simple core language. This is in essence the language used by Kozen in [6]. In this section we introduce both the syntax and the semantics for this language, which we call **pWhile**.

2.1 Syntax

In a style typical of static analysis [7], we introduce a labelled version of the **pWhile** language, where labels are used to identify the programs points that are crucial for defining our formal semantics.

$$\begin{aligned} S \texttt{ ::= } & [\texttt{skip}]^\ell \mid [x \texttt{ := } e]^\ell \mid [x \texttt{ ?= } \rho]^\ell \mid S_1 \texttt{;}\ S_2 \\ & \mid \texttt{if } [b]^\ell \texttt{ then } S_1 \texttt{ else } S_2 \texttt{ fi} \mid \texttt{while } [b]^\ell \texttt{ do } S \texttt{ od} \end{aligned}$$

We assume a unique labelling (by numbers $\ell \in \mathbf{Lab}$).

The statement `skip` does not have any operational effect but can be used, for example, as a placeholder in conditional statements. We have the usual (deterministic) assignment $x \texttt{ := } f(x_1, \ldots, x_n)$, sometimes also in the form $x \texttt{ := } e$.

Then we have the random assignment $x \texttt{ ?= } \rho$ where the value of a variable is set to a value according to some random distribution ρ. In [6] it is left open how to define or specify distributions ρ in detail. We will use occasionally an ad-hoc notation as sets of tuples $\{(v_i, p_i)\}$, i.e. expressing the fact that value v_i will be selected with probability p_i. It might be useful to assume that the random number generator or scheduler which implements this construct can only implement choices over finite ranges, but in principle we can also use distributions with infinite support. For the rest we have the usual sequential composition, conditional statement and loop. We leave the detailed syntax of functions f or expressions e open as well as for boolean expressions or test b in conditionals and loop statements.

2.2 Linear Operator Semantics

We will base our probabilistic analysis on a syntax-based semantics which represents the executions of a program P as a Discrete Time Markov Chain (DTMC). More precisely, we associate to P a linear operator $\mathbf{T}(P)$ corresponding to the generator of the DTMC associated to P. This is a possibly infinite matrix whose domain is the set of *probabilistic configurations*, i.e. distributions on classical configurations $(x_1, \ldots, x_v, \ell)$, which record the value of all variables and the current label, defined by $\mathbf{Dist}(\mathbf{Conf}) = \mathbf{Dist}(\mathbb{X}^v \times \mathbf{Lab}) \subseteq \ell_2(\mathbb{X}^v \times \mathbf{Lab})$. Using the *tensor product* (e.g. [8, Chap. 14] or [9, Chap. 2.6]) we can exploit the fact that $\ell_2(\mathbb{X}^v \times \mathbf{Lab}) = \ell_2(\mathbb{X})^{\otimes v} \otimes \ell_2(\mathbf{Lab})$. We refer to $s \in \mathbf{Var} \to \mathbb{X} = \mathbb{X}^v$ as a *classical state* and to $\sigma \in \mathbf{Dist}(\mathbf{Var} \to \mathbb{X}) \subseteq \ell_2(\mathbb{X})^{\otimes v}$ as *probabilistic state*. In this definition we assume that that variables occurring in a **pWhile** program can take values in some countable set $\mathbb{X}$ that might be finite (e.g. Booleans) or infinite (typically $\mathbb{Z}$ or $\mathbb{N}$).

The labelled version of the syntax introduced in Section 2.1 allows us to use labels as a kind of program counter. The control flow $\mathcal{F}(S)$ in a program P is then defined via a function *flow* : $\mathbf{Stmt} \to \mathcal{P}(\mathbf{Lab} \times \mathbf{Lab})$ which maps statements to sets of pairs which represent the control flow graph. This only records that a certain control flow step is possible. For tests $[b]^\ell$ in conditionals and loops we indicate the branch corresponding to the case when the test succeeds

$$
\begin{aligned}
\mathbf{P}(s) &= \bigotimes_{i=1}^{v} \mathbf{P}(s(\mathtt{x}_i)) & \qquad \mathbf{U}(\mathtt{x}_k \leftarrow c) &= \bigotimes_{i=1}^{k-1} \mathbf{I} \otimes \mathbf{U}(c) \otimes \bigotimes_{i=k+1}^{v} \mathbf{I} \\
\mathbf{P}(e = c) &= \sum_{\mathcal{E}(e)s=c} \mathbf{P}(s) & \qquad \mathbf{U}(\mathtt{x}_k \leftarrow e) &= \sum_{c} \mathbf{P}(e = c)\mathbf{U}(\mathtt{x}_k \leftarrow c)
\end{aligned}
$$

$$
\begin{gathered}
[\![[x \texttt{ := } e]^{\ell]}]\!] = \mathbf{U}(x \leftarrow e) \qquad [\![[v \texttt{ ?= } \rho]^{\ell}]\!] = \textstyle\sum_{c \in \mathbb{X}} \rho(c)\mathbf{U}(x \leftarrow c) \\
[\![[b]^{\ell}]\!] = \mathbf{P}(b = \mathtt{false}) \qquad \underline{[\![[b]^{\ell}]\!]} = \mathbf{P}(b = \mathtt{true}) \\
[\![[\mathtt{skip}]^{\ell}]\!] = \underline{[\![[\mathtt{skip}]^{\ell}]\!]} = \underline{[\![[x \texttt{ := } e]^{\ell]}]\!]} = \underline{[\![[v \texttt{ ?= } \rho]^{\ell}]\!]} = \mathbf{I}
\end{gathered}
$$

Table 1. Elements of the LOS

by underlining it. As our semantics is ultimately modelling the semantics of a program via the generator of a DTMC we are also confronted with the fact that such processes never terminate. For this we will add a single final loop via a virtual label ℓ^* at the end of the program.

The construction of $\mathbf{T}(P)$ is done compositionally by using among its building blocks simple operators such as the *identity matrix* $\mathbf{I}$ and the *matrix units* $\mathbf{E}_{ij}$ containing only a single non zero entry $(\mathbf{E}_{ij})_{ij} = 1$. We also define for any Boolean expression b on $\mathbb{X}$ a diagonal *projection matrix* $\mathbf{P}(b)$ with $(\mathbf{P}(b))_{ii} = 1$ if $b(i)$ holds and 0 otherwise. The operator $\mathbf{P}(s)$ tests for a classical state s, i.e. if each variable $\mathtt{x}_k$ has the value $s(\mathtt{x}_i)$, and $\mathbf{P}(e = c)$ whether an expression e evaluates to a constant c. The *update* operator $\mathbf{U}$ implements state changes. The matrix $\mathbf{U}(c)$ implements the deterministic update of a variable to a constant c via $(\mathbf{U}(c))_{nm} = 1$ if $m = c$ and 0 otherwise. The operator $\mathbf{U}(x_k \leftarrow e)$ makes sure that the k^{th} variable $\mathtt{x}_k$ is assigned the value of the expression e

The matrix $\mathbf{T}(P)$ of the DTMC representing the program's executions is then defined as the sum of the effects of the individual control flow steps, i.e. the computational effect of each (labelled) block $[B]^{\ell}$ – with $B = \mathtt{skip}$, a test b or a (random) assignment – and a control flow step of the form $\mathbf{E}_{\ell\ell'}$.

$$
\mathbf{T}(P) = \sum_{(\ell,\ell') \in \mathcal{F}(P)} [\![[B]^{\ell}]\!] \otimes \mathbf{E}_{\ell,\ell'} + \sum_{(\ell,\underline{\ell'}) \in \mathcal{F}(P)} \underline{[\![[B]^{\ell}]\!]} \otimes \mathbf{E}_{\ell,\ell'}
$$

If we also consider the final loop we have to add the term $\mathbf{I} \otimes \mathbf{E}_{\ell^*,\ell^*}$. The definition of the semantics of the individual blocks is given in Table 2.2.

3 Probabilistic Abstract Interpretation

Classically the correctness of a program analysis is asserted with respect to the semantics in terms of a correctness relation. The theory of Abstract Interpretation allows for constructing analyses that are automatically correct without having to prove it a posteriori [4]. This is possible because of the conditions defining a Galois connection which guarantee that we do not lose safety by going back and forth between the two lattices of the concrete and the analysis domains – although we may lose precision.

We have shown in a number of papers how the PAI technique can be used for performing the probabilistic analysis of many problems typical of classical program analysis, such as data-flow and pointer analyses [2], and of security problems [10]. The treatment in this previous work is restricted to the special case of finite state spaces. In this paper we generalise the theory to infinite-dimensional Hilbert spaces, where it is necessary the consideration of some appropriate topological notions in order to correctly deal with possibly infinite abstraction operators (and their generalised inverses).

3.1 Basic Definitions

The theory of Probabilistic Abstract Interpretation relies on the notion of generalised (or pseudo-) inverse. This notion is well-known in mathematics where it is used for finding approximate solutions to integral and differential equations [11, Chap 9]. The abstract concept of a generalised or pseudo-inverse was introduced by Moore in the 1920s and was then rediscovered by Penrose in the 1950s. Generalised inverses for the infinite-dimensional case have been introduced by Tseng in the 1930s. Though there are a number of other definitions, it is the Moore-Penrose(-Tseng) pseudo-inverse which is usually considered to be *the* generalised inverse.

Definition 1. *Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two Hilbert spaces and $\mathbf{A} : \mathcal{H}_1 \mapsto \mathcal{H}_2$ a linear map between them. A linear map $\mathbf{A}^\dagger = \mathbf{G} : \mathcal{H}_2 \mapsto \mathcal{H}_1$ is the* Moore-Penrose pseudo-inverse *of $\mathbf{A}$ iff $\mathbf{A} \circ \mathbf{G} = \mathbf{P_A}$ and $\mathbf{G} \circ \mathbf{A} = \mathbf{P_G}$, where $\mathbf{P_A}$ and $\mathbf{P_G}$ denote orthogonal projections onto the ranges of $\mathbf{A}$ and $\mathbf{G}$.*

It is also possible to define the Moore-Penrose pseudo-inverse $\mathbf{A}^\dagger$ of an operator $\mathbf{A}$ without direct reference to orthogonal projections (cf. e.g. [12, Section 4.7]). For this we need the notion of the adjoint $\mathbf{A}^*$ of an operator $\mathbf{A} : \mathcal{H} \to \mathcal{H}$ on a Hilbert spaces which is defined via the condition $\langle \mathbf{A}(x), y \rangle = \langle x \mathbf{A}^*(y) \rangle$. Then the Moore-Penrose pseudo-inverse can be defined via the following four conditions: $\mathbf{A}\mathbf{A}^\dagger\mathbf{A} = \mathbf{A}$, $\mathbf{A}^\dagger\mathbf{A}\mathbf{A}^\dagger = \mathbf{A}^\dagger$, $(\mathbf{A}^\dagger\mathbf{A})^* = \mathbf{A}^\dagger\mathbf{A}$, and $(\mathbf{A}\mathbf{A}^\dagger)^* = \mathbf{A}\mathbf{A}^\dagger$. In this form it is also obvious that the Moore-Penrose pseudo-inverse (like a Galois connection) is indeed a pseudo-inverse.

If $\mathcal{C}$ an $\mathcal{D}$ are two Hilbert spaces, and $\mathbf{A} : \mathcal{C} \to \mathcal{D}$ and $\mathbf{G} : \mathcal{D} \to \mathcal{C}$ are linear operators between the concrete domain $\mathcal{C}$ and the abstract domain $\mathcal{D}$, such that $\mathbf{G}$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$, then we say that $(\mathcal{C}, \mathbf{A}, \mathcal{D}, \mathbf{G})$ forms a *probabilistic abstract interpretation.*

3.2 Correctness vs Best Solution

The Moore-Penrose pseudo-inverse plays a similar role in Probabilistic Abstract Interpretation as Galois connections in the theory of classical Abstract Interpretation in helping with the problem of combinatory explosion: it allows us to simplify the semantics $\mathbf{T}(S)$ of a program S by considering the abstract semantics $\mathbf{T}^{\#} = \mathbf{A}^{\dagger}\mathbf{T}(S)\mathbf{A}$ instead of the concrete one.

However, the properties of the Moore-Penrose pseudo-inverse (e.g. [13, 11]) guarantee a different form of optimality of the abstractions (abstract semantics) we can construct via PAI. In particular, these properties guarantee that PAI abstractions are the *closest* ones to the concrete semantics one can construct. Here closeness is defined via the distance induced by the norm on the Hilbert space, thus the name of '*least square approximation*' which is often used to refer to this approximation notion. As a consequence, in our probabilistic setting the main aim of the analysis is to reduce the error margin rather than to 'err on the safe side', which would lead to *safe* abstractions in the usual sense, i.e. over- or under-approximations of the concrete semantics.

The choice of the Hilbert space ℓ_2 as the domain for the LOS semantics is partly motivated by the symmetry between observables and states (we recall that ℓ_2 is self-dual) but also by the fact that it provides a simple and mathematically well understood theory of best approximations by Moore-Penrose pseudo-inverses. A similar well-behaved theory does not exist in general Banach spaces like ℓ_1 which do not have an inner product and a notion of an adjoint operator $\mathbf{A}^*$. As a result the theory of approximations in Banach spaces can, in general, not even guarantee the existence of unique optima (e.g. [14]).

3.3 Compositionality of PAI

Important for the applicability of PAI is the fact that it possesses some nice compositionality properties. These allows us to construct the abstract semantics by abstracting the single blocks of the concrete semantics $\mathbf{T}(S)$:

$$\mathbf{T}(S)^{\#} = \mathbf{A}^{\dagger}\mathbf{T}(S)\mathbf{A} = \mathbf{A}^{\dagger}\left(\sum [\![[B]^{\ell}]\!]\right)\mathbf{A} = \sum \left(\mathbf{A}^{\dagger}[\![[B]^{\ell}]\!]\mathbf{A}\right) = \sum [\![[B]^{\ell}]\!]^{\#}$$

The fact that we can work with the abstract semantics of individual blocks instead of the full operator obviously reduces the complexity of the analysis substantially.

Another important fact is that the Moore-Penrose pseudo-inverse of a tensor product can be computed as [11, 2.1,Ex 3]:

$$(\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \ldots \otimes \mathbf{A}_v)^{\dagger} = \mathbf{A}_1^{\dagger} \otimes \mathbf{A}_2^{\dagger} \otimes \ldots \otimes \mathbf{A}_v^{\dagger}.$$

We can therefore abstract properties of individual variables and then combine them in the global abstraction. This is also made possible by the definition of the concrete LOS semantics (cf. Section 2.2) which is heavily based on the use of tensor product. Typically we have $[\![[B]^{\ell}]\!] = (\bigotimes_{i=1}^{v} \mathbf{T}_{i\ell}) \otimes \mathbf{E}_{\ell\ell'}$ or a sum of a few of such terms. The $\mathbf{T}_{i\ell}$ represents the effect of $\mathbf{T}(S)$, and in particular of $[\![[B]^{\ell}]\!]$, on variable i at label ℓ (both labels and variables only form a finite set). For example, we can define an abstraction $\mathbf{A}$ for one variable and apply it individually to all variables (e.g. extracting their even/odd property), or use different abstractions for different variables (maybe even forgetting about some of them by using $\mathbf{A}_f = (1, 1, \ldots)^t$) and define $\mathbf{A} = \bigotimes_{i=1}^{v} \mathbf{A}_i$ such that $\mathbf{A}^\dagger = \bigotimes_{i=1}^{v} \mathbf{A}_i^\dagger$ in order to get an analysis on the full state space.

Usually we do not abstract the control step $\mathbf{E}_{\ell\ell'}$, although this would also be possible. Very often most of the $\mathbf{T}_{i\ell}$'s are just the identity matrix – expressing the fact that the variable in question is not involved in whatever happens at label ℓ. In this case we have $\mathbf{A}_i^\dagger \mathbf{I} \mathbf{A}_i = \mathbf{I}$, where the two identity operators are to be intended with the appropriate dimensions (maybe an infinite matrix on the concrete space, and a mostly finite matrix on the abstract property space).

The abstractions of tests $[b]^\ell$, i.e. $\mathbf{P}(b)$, provide the basis for analysing *abstract branching probabilities.* This can be done by constructing $\mathbf{A}^\dagger [\![[b]^\ell]\!] \mathbf{A}$ and $\mathbf{A}^\dagger [\![[\neg b]^\ell]\!] \mathbf{A}$. As these operators are projections, we can exploit the fact that $[\![[b]^\ell]\!]^\# = \mathbf{I} - [\![[\neg b]^\ell]\!]$ and compute in practice only one of them. The abstract operator $[\![[\neg b]^\ell]\!]^\#$ in effect estimates the chance of the test b succeeding in terms of the abstract properties. For example, with a parity analysis we obtain a (least square) estimate for the chance that in an conditional statement, for an even number we take the `then` branch rather than the `else` branch (cf. Example in Section 4). If an analytical (statistical) construction of $[\![[b]^\ell]\!]^\#$ is infeasible one might consider profiling information as a replacement of the exact solution.

3.4 Bounded and Unbounded Abstractions

A necessary and sufficient condition for the existence of the Moore-Penrose inverse for a *bounded* linear operator $\mathbf{A} : \mathcal{H} \to \mathcal{H}$ on a Hilbert space $\mathcal{H}$ is that A is *normally solvable*, i.e. its range $R(\mathbf{A}) = \{\mathbf{A}x \mid x \in \mathcal{H}\}$ is closed. Thus, all operators on a finite dimensional Hilbert space are Moore-Penrose invertible.

Proposition 1. *[12, Thm 4.24] A bounded operator $\mathbf{A} \in \mathcal{B}(\mathcal{H}))$ is Moore-Penrose pseudo-invertible, i.e. a unique $\mathbf{A}^\dagger$ exists, if and only if it is normally solvable. In this case $(\mathbf{A}^*\mathbf{A} + \mathbf{P})$ is invertible and $\mathbf{A}^\dagger = (\mathbf{A}^*\mathbf{A} + \mathbf{P})^{-1}\mathbf{A}^*$ where $\mathbf{P}$ is the orthogonal projection of $\mathcal{H}$ onto $N(\mathbf{A}) = \{x \mid \mathbf{A}(x) = o\}$ with o being the null vector.*

Unfortunately, it is not possible to guarantee that all abstractions one might be interested in are bounded (cf. $\mathbf{A}_f$ of Example 1 below). However, the following theorem allows us to establish the existence of a Moore-Penrose pseudo-inverse also for unbounded operators. The notation $\overline{X}$ indicates the topological closure of $X \subseteq \mathcal{H}$. A linear map $\mathbf{T} \in \mathcal{L}(\mathcal{X}, \mathcal{Y})$ is called *closed* if its graph is closed in $\mathcal{X} \oplus \mathcal{Y}$. Equivalently, $\mathbf{T}$ is closed if and only if for any sequence $x_n \in \mathcal{D}(\mathbf{T}) \subseteq \mathcal{X}$, where $\mathcal{D}(\mathbf{T})$ is the domain where $\mathbf{T}$ is defined, with $x_n \to x$ and $\mathbf{T}(x_n) \to y$ this implies that $y = \mathbf{T}(x)$, cf. [5, p287] or [15, p31].

Theorem 1. *[15, Thm2.12] If $\mathbf{A} : \mathcal{D} \subseteq \mathcal{H}_1 \to \mathcal{H}_2$ is a closed densely defined linear operator then $\mathbf{A}^\dagger : \mathcal{D}(\mathbf{A}^\dagger) = R(\mathbf{A}) + R(\mathbf{A})^\perp \to \mathcal{D}(\mathbf{A}) \cap N(\mathbf{A})^\perp$ is closed and densely defined. Moreover,*

(i) *$\mathbf{A}^\dagger$ is bounded if and only if $R(\mathbf{A})$ is closed.*
(ii) *$\mathbf{A}\mathbf{A}^\dagger(y) = \mathbf{P}_{\overline{R(\mathbf{A})}}(y)$ for all $y \in \mathcal{D}(\mathbf{A}^\dagger)$.*
(iii) *$\mathbf{A}^\dagger\mathbf{A}(y) = \mathbf{P}_{N(\mathbf{A})^\perp}(y)$ for all $y \in \mathcal{D}(\mathbf{A})$.*

3.5 Abstraction Operators

In classical program analysis [7], an extraction or classification function $\alpha : C \to D$ is a function which associates to each element c of a concrete domain C an abstract description or property $d \in D$.

In our framework for probabilistic analysis, we will consider Hilbert spaces for both concrete and abstract domains. In particular, for an extraction function $\alpha : C \to D$ we will construct the Hilbert spaces $\ell_2(C)$ and $\ell_2(D)$ generated by the base vectors $\{\boldsymbol{c}\}_{c \in C}$ and $\{\boldsymbol{d}\}_{d \in D}$, respectively. For finite C and/or D we can identify them the with finite dimensional vector spaces $\mathbb{R}^{|C|}$ and/or $\mathbb{R}^{|D|}$. We then define a linear map $\mathbf{A}_\alpha = \mathbf{A} : \ell_2(C) \to \ell_2(D)$ by mapping every $\boldsymbol{x} = \sum_c x_c$ to $\mathbf{A}(\boldsymbol{x}) = \mathbf{A}(\sum_c x_c \boldsymbol{c}) = \sum_c x_c \mathbf{A}(\boldsymbol{c})$, with $(\mathbf{A}(\boldsymbol{c}))_{\alpha(c)} = 1$ and $(\mathbf{A}(\boldsymbol{c}))_d = 0$ for all $d \neq \alpha(c)$. In effect this means that we construct a matrix $\mathbf{A}$ with rows enumerated by elements in C and columns enumerated by elements in D such that: $(\mathbf{A})_{cd} = 1$ if $\alpha(c) = d$ and 0 otherwise.

Typically D will be finite but C could be countably infinite. In this case we are faced with the problem that $\mathbf{A}$ might not be bounded. Thus – though $\mathbf{A}$ represents a closed map because D is finite – we need to be more careful when we construct the Moore-Penrose pseudo-inverse as explained in Section 3.4.

Example 1 (Forgetful Abstraction). Let us consider the abstraction into an abstract domain containing only a single property $*$, i.e. $C = \{*\}$ and $\alpha(s) = *$ for all concrete values s. Essentially this abstraction only tests for the existence of the system. It might seem rather pointless, but it turns out to be useful in various contexts, for example to analyse only a particular variables property ignoring other program variables, (see e.g. [3]).

The matrix $\mathbf{A}_f$ representing the forgetful abstraction in the PAI framework is a single column matrix containing only 1s. This corresponds to the map $x \mapsto (\|x\|_1) \in \mathbb{R}^1 = \ell_2(\{*\})$. This matrix represents clearly an unbounded map $\ell_2 \to \mathbb{R}$: the vector $x = (x_i)_{i=0}^\infty = (1, \frac{1}{2}, \frac{1}{3}, \ldots)$ is in ℓ_2 but $\mathbf{A}_f(x) = (\|x\|_1) = (\infty)$.

The central problem we aim to address in this paper is that for many infinite abstractions, like $\mathbf{A}_f$, the corresponding matrices do not represent bounded linear maps on ℓ_2. In order to nevertheless be able to construct our analyses we will use the weak limit notion (cf. [12], [11, Chapter 9], or [15]) to obtain useful approximations of their Moore-Penrose pseudo inverses.

For an abstraction operator $\mathbf{A}$ corresponding to a classification function α on a finite dimensional space $\ell_2(C) = \mathbb{R}^{|C|}$ we can construct the Moore-Penrose pseudo-inverse by just transposing $\mathbf{A}$ and row-normalising it. However we cannot use the same construction for infinite abstractions even if, as we will show later, we can guarantee the existence of $\mathbf{A}^\dagger$. In Example 1, it is clear that with a concrete finite space $C = \{c_1, \ldots, c_n\}$ we can construct

$$\mathbf{A}_f = (1, 1, \ldots, 1)^t \text{ and } \mathbf{A}_f^\dagger = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}).$$

However, if we extend this to the case of a countable infinite concrete space, e.g. $C = \mathbb{Z}$, then clearly the row-normalised transpose of an infinite version of $\mathbf{A}_f$ must be the zero (single-row) matrix.

Example 2 (Parity Abstraction). Consider as abstract and concrete domains $\mathcal{C} = \ell_2(\{0, \ldots, n\})$ and $\mathcal{D} = \ell_2(\{\text{even}, \text{odd}\})$. The abstraction operator $\mathbf{A}_p$ and its concretisation operator $\mathbf{G}_p = \mathbf{A}_p^\dagger$ corresponding to a *parity analysis* are represented by the following $(n+1) \times 2$ and $2 \times (n+1)$ matrices (assuming w.l.o.g. that n is odd)

$$\mathbf{A}_p = \begin{pmatrix} 1\,0\,1\,0\ldots 0 \\ 0\,1\,0\,1\ldots 1 \end{pmatrix}^t \qquad \mathbf{A}_p^\dagger = \begin{pmatrix} \frac{2}{n+1} & 0 & \frac{2}{n+1} & 0 & \cdots & 0 \\ 0 & \frac{2}{n+1} & 0 & \frac{2}{n+1} & \cdots & \frac{2}{n+1} \end{pmatrix}$$

The concretisation operator $\mathbf{A}_p^\dagger$ represents uniform distributions over the $\frac{n}{2}$ even numbers in the range $0, \ldots, n$ (as the first row) and the n odd numbers in the same range (in the second row). Clearly, if we increase the dimension n we encounter the same problems as with $\mathbf{A}_f$.

3.6 Weak and Strong Approximations

For practical and computational purposes we need not only to guarantee the existence of $\mathbf{A}^\dagger$ but also to construct $\mathbf{A}^\dagger$ for bounded and unbounded abstractions. In numerical mathematics [16, Sect 12.1], we have a general theory of the so-called *finite sections* or *projection* methods which aims in approximating infinite dimensional operators on Hilbert spaces by finite approximations, i.e. finite dimensional abstractions. We used this theory as a main inspiring source for our approach, and in particular for constructing approximations of abstraction operators. Given one such operator $\mathbf{A}$, its *finite sections* $\mathbf{A}_n = \mathbf{P}_n\mathbf{A}\mathbf{P}_n$ are finite dimensional matrices (thus bounded operator) for which it is easily possible to construct the Moore-Penrose pseudo-inverse $\mathbf{A}_n^\dagger$. In fact, for bounded $\mathbf{A}$ it is known that we can approximate $\mathbf{A}^\dagger$ as the *strong* limit of the approximating Moore-Penrose sequence $\mathbf{A}_n^\dagger$, cf. [12, Corr. 4.34].

Proposition 2. *[12, Cor. 4.34] Let $\mathbf{A} \in \mathcal{B}(\ell_2(\mathbb{X}))$ be a bounded operator with an approximating sequence $\{\mathbf{A}_n\}$ such that $\{\mathbf{A}_n\}$ and $\{\mathbf{A}_n^*\}$ converge strongly, and $\mathbf{A}_n \to \mathbf{A}$. Then the following are equivalent:*

(i) $\{\mathbf{A}_n\}$ *is an* exact Moore-Penrose sequence*.*
(ii) $\{\mathbf{A}_n^\dagger\}$ *converges strongly.*
(iii) $\mathbf{A}$ *is normally solvable and* $\mathbf{A}_n^\dagger \to \mathbf{A}^\dagger$ *strongly.*

$\{\mathbf{A}_n\} \in \mathcal{F}$ is said to be an exact Moore-Penrose sequence iff $\{\mathbf{A}_n\}$ is Moore-Penrose invertible in $\mathcal{F}$, where $\mathcal{F}$ is the space of sequences $\{\mathbf{A}_n\}$ of $n \times n$ matrices for which $\sup_n \|\mathbf{A}_n\| < \infty$, see [12, p46].

Example 3 (Approximation of $\mathbf{A}_f^\dagger$). If we construct the weak limit of finite approximations to $\mathbf{A}_f$ and consider its effect only in the context of a concrete state σ then we get a useful result even if we do not effectively construct the norm or uniform limit of $\mathbf{A}_{fn}$. For any probabilistic state $\sigma \in \ell_1(C)$ with $\sigma_i \geq 0$ and $\|\sigma\|_1 = 1$, i.e. a probability distribution, we get for all $x \in \ell_2(C)$

$$\lim_{n\to\infty} \langle \mathbf{A}_{fn}(\sigma), x \rangle = \mathbf{E}(x, \sigma) \quad \text{and} \quad \lim_{n\to\infty} \left\langle \mathbf{A}_{fn}^\dagger(*), x \right\rangle = \mathbf{E}(x, \nu),$$

where $\mathbf{E}(x, d)$ is the expectation value of x with respect to the distribution d and ν is the uniform distribution. In other words, the weak limit $\mathbf{A}_f^\dagger = w\text{-}\lim \mathbf{A}_{fn}$ represents the effect of an eventually non-existing "uniform measure" on all of C, even when C is an infinite set. In measure theoretic terms this means that we represent a uniform measure (on all of C) which per se cannot be expressed as a distribution as the limit of distributions.

The following two results generalise the properties of the forgetful abstraction in Example 1 to any abstraction onto a finite dimensional space.

Lemma 1. *For any countable set C, finite set D and classification map $\alpha : C \to D$ the corresponding map $\mathbf{A} : \ell_2(C) \to \ell_2(D)$ is a densely defined closed linear map.*

Proof. The abstraction operator $\mathbf{A}$ for any classification function α is given by a *stochastic* matrix, i.e. the row sums are all 1. Thus its matrix represents a bounded operator on ℓ_1 with respect to the 1-norm on ℓ_1, in fact they preserve the 1-norm $\|\mathbf{A}(x)\|_1 = \|x\|_1$.

Clearly, $\mathbf{A}$ is in general not defined for all vectors in ℓ_2, see for example $\mathbf{A}_f(x)$ for which we have vectors with $\mathbf{A}_f(x) = (\infty)$. However, all abstractions $\mathbf{A}$ are well-defined on ℓ_1 – they are even elements in $\mathcal{B}(\ell_1, \mathbb{R}^d)$ – which is a dense sub-space of ℓ_2. Regarding closedness: we rely on the fact that we have in general $\|x\|_2 \leq \|x\|_1$, cf. [17, Exercise 1.15]. If we assume convergence of x_n in the 2-norm, i.e. $\|x_n - x\|_2 \to 0$ and convergence $\mathbf{A}(x_n) \to y$ in $\mathbb{R}^d$, then – because $\mathbf{A} \in \mathcal{B}(\ell_1, \mathbb{R}^d)$ is continuous with respect to the 1-norm – we have $\|\mathbf{A}(x_n) - \mathbf{A}(x)\|_1 \to 0$, i.e. $\mathbf{A}(x_n) \to \mathbf{A}(x)$. Therefore, the two limits in $\mathbb{R}^d$ (on which all norms are equivalent) must agree and we have $\mathbf{A}(x) = y$. $\square$

Proposition 3. *For any countable set C and finite set D and any classification map $\alpha : C \to D$, the corresponding abstraction $\mathbf{A} : \ell_2(C) \to \ell_2(D)$ has a Moore-Penrose pseudo-inverse $\mathbf{A}^\dagger : \mathcal{D} \subseteq \ell_2(D) \to \ell_2(C)$.*

This follows from the fact that $\mathbf{A}$ is densely defined and closed (from Lemma 1) and Theorem 1.

Now that we are guaranteed that for all our abstractions there exists a Moore-Penrose pseudo inverse, we address the problem of effectively constructing one. The approach we take is to avoid the explicit construction of (an infinite representation of) $\mathbf{A}^\dagger$ and instead introduce a way of approximating the effect of the abstract semantics on a given abstract input distribution.

Regarding the construction of approximations of $\mathbf{A}^\dagger$ one can find a large number of so-called regularisation approaches in the mathematical literature [18–21] which give computationally efficient ways to approximate the (effect of the) Moore-Penrose pseudo-inverse for bounded and unbounded linear maps. For unbounded, densely defined closed abstractions $\mathbf{A}$ we consider here only the effect of finite sections on distributions, i.e. we are interested in the weak limit of $\mathbf{A}^\dagger$, $\mathbf{A}^\dagger\mathbf{A}$, or $\mathbf{A}^\dagger\mathbf{T}(P)\mathbf{A}$.

We can guarantee that the abstract semantics $\mathbf{T}^\#(P) = \mathbf{A}^\dagger\mathbf{T}(P)\mathbf{A}$ can be approximated via a weak limit as established by the following proposition.

Proposition 4. *For any countable set C and finite set D and any classification map $\alpha : C \to D$, let $\mathbf{A}$ be the corresponding linear map $\mathbf{A} : \ell_2(C) \to \ell_2(D)$. Then for any program P and its LOS operator $\mathbf{T}(P)$, we have that for all abstract distributions $x, y \in \mathbf{Dist}(\mathbf{Conf}) \subset \ell_2(D)$,*

$$\lim_{n\to\infty} \left\langle x \cdot \mathbf{A}_n^\dagger \mathbf{T}(P)\mathbf{A}_n, y \right\rangle < \infty.$$

These results follow from, for example, [22, Thm. 3.3] and [15, Chapter 5]. An extensive comparison of related results on weak and strong approximations of the Moore-Penrose pseudo-inverse can be found, for example, in [21].

This proposition guarantees the existence (rather than the precision) of program analyses as the weak limits of finite dimensional approximations of an infinite dimensional and possibly unbounded semantics and abstraction. In the next section will illustrate this with a concrete example.

4 An Example

We conclude by discussing in detail an example which illustrates how probabilistic abstraction allows us to analyse the properties of programs. We will also discuss the efficiency of the analysis, i.e. how PAI can be deployed in order to beat the combinatorial explosion or the curse of dimensionality.

It is easy to observe that the factorial function $n!$ "almost always" returns an even number (except for 0! and 1!). If we perform a classical abstraction we cannot justify this intuition as in oder to be safe we can only obtain a guarantee that the result may be *even* or *odd*.

In order to provide a formal analysis of $n!$ let us first consider the concrete semantics of the program F using labelling:

$$[m \texttt{ := } 1]^1;\ \texttt{while}\ [n > 1]^2\ \texttt{do}\ [m \texttt{ := } m \times n]^3;\ [n \texttt{ := } n - 1]^4\ \texttt{od};\ [\texttt{skip}]^5$$

The flow is $\mathcal{F}(F) = \{(1,2),(2,\underline{3}),(3,4),(4,2),(2,5),(5,5)\}$ which includes a looping on the final `skip` statement. The operator $\mathbf{T}(F)$ is then:

$$\begin{aligned}\mathbf{T}(F) = {} & \mathbf{U}(\texttt{m} \leftarrow \texttt{1}) \otimes \mathbf{E}(1,2) + \mathbf{P}(\texttt{(n > 1)}) \otimes \mathbf{E}(2,3) + \mathbf{U}(\texttt{m} \leftarrow \texttt{(m * n)}) \otimes \mathbf{E}(3,4) + {}\\ & \mathbf{U}(\texttt{n} \leftarrow \texttt{(n - 1)}) \otimes \mathbf{E}(4,2) + \mathbf{P}(\texttt{(n <= 1)}) \otimes \mathbf{E}(2,5) + \mathbf{I} \otimes \mathbf{E}(5,5)\end{aligned}$$

If we just consider the factorials 0!, 1! and 2! then we can restrict ourselves to values $m, n \in \{0,1,2\}$. In this case the semantics of each block is given by a $3 \cdot 3 \times 3 \cdot 3 = 9 \times 9$ matrix.

For the updates in label 3 and 4 we have "empty rows", i.e. rows where we have no non-zero entries. These correspond to over- and under-flows as we are dealing only with finite values in $\mathbb{Z}$, e.g. the product for $m = 2$ and $n = 2$ is not in $\{0,1,2\}$. We could clarify the situation in various ways, e.g. by introducing an additional value $\perp$ for undefined (concrete) values of variables, or by introducing an `error` configuration. In the analysis we present here these over- and underflows do not play any relevant role and we therefore leave things as they are.

The full operator representing the LOS semantics of the factorial program is given by a $(3 \cdot 3 \cdot 5) \times (3 \cdot 3 \cdot 5) = 45 \times 45$ matrix.

We can construct an abstract version $\mathbf{T}^{\#}(F) = \mathbf{A}^{\dagger}\mathbf{T}(F)\mathbf{A}$ of $\mathbf{T}(F)$ by recording only the parity of m as even and odd. We will not abstract n nor the labels defining the current configuration during the execution. We thus get the $(2 \cdot 3 \cdot 5) \times (2 \cdot 3 \cdot 5) = 30 \times 30$ matrix

$$\mathbf{T}^{\#}(F) = (\mathbf{A}_p \otimes \mathbf{I} \otimes \mathbf{I})^{\dagger}\mathbf{T}(F)(\mathbf{A}_p \otimes \mathbf{I} \otimes \mathbf{I}),$$

Though this abstract semantics does have some interesting properties, it appears to be only a minor improvement with regard to the concrete semantics: We managed to reduce the dimension only from 45 to 30. However, the simplification becomes substantially more dramatic once we increase the possible values of `m` and `n`, and combinatorial explosion really takes a hold. If we allow `n` to take values between 0 and n then we must allow for `m` values between 0 and $n!$. Concrete values of the dimensions of $\mathbf{T}(F)$ and $\mathbf{T}^{\#}(F)$ for $n = 1, 2, \ldots, 9$ are $\dim(\mathbf{T}(F)) = 45, 140, 625, 3630, 25235, 201640, 1814445, 18144050$ but for the abstract semantics only $\dim(\mathbf{T}^{\#}(F)) = 30, 40, 50, 60, 70, 80, 90, 100$.

The problem is that the size of $\mathbf{T}(F)$ explodes so quickly that it is impossible to simulate it for values of n much larger than 5 on a normal PC. If we want to analyse the abstract semantics, things remain much smaller. Importantly, we can construct the abstract semantics in the same way as the concrete one, just using "smaller" matrices:

$$\mathbf{T}^{\#}(F) = \mathbf{U}^{\#}(\texttt{m} \leftarrow \texttt{1}) \otimes \mathbf{E}(1,2) + \mathbf{P}^{\#}(\texttt{(n > 1)}) \otimes \mathbf{E}(2,3) + \mathbf{U}^{\#}(\texttt{m} \leftarrow \texttt{(m * n)}) \otimes \mathbf{E}(3,4) +$$
$$\mathbf{U}^{\#}(\texttt{n} \leftarrow \texttt{(n - 1)}) \otimes \mathbf{E}(4,2) + \mathbf{P}^{\#}(\texttt{(n <= 1)}) \otimes \mathbf{E}(2,5) + \mathbf{I}^{\#} \otimes \mathbf{E}(5,5)$$

Fortunately, most of the operators $\mathbf{T}^{\#}(\ell, \ell')$ are very easy to construct. These matrices are $2 \cdot (n+1) \cdot 5 \times 2 \cdot (n+1) \cdot 5 = 10(n+1) \times 10(n+1)$ matrices if we consider the control transfer, and only $2(n+1) \times 2(n+1)$ matrices if we deal only with the update of the current state.

Except for label 3 only either `m` or `n` but never both are involved in each statement: We thus can express the $\mathbf{T}^{\#}(\ell, \ell')$'s as tensor products of a 2×2 and a $(n+1) \times (n+1)$ matrix.

Finally, we need to construct the update for label 3. It is easy to see that for even `m` the result is again even and for odd `m` the parity of `n` determines the parity of the resulting `m`. We can thus write this update at label 3 as:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \mathbf{I} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \otimes \mathbf{P}(\mathbf{even}) + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \mathbf{P}(\mathbf{odd})$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{P}(\mathbf{even})$ and $\mathbf{P}(\mathbf{odd})$ are the indicator projections for even and odd, e.g. $(\mathbf{P}(\mathbf{even}))_{ij} = 1$ if $i = j$ is even and 0 otherwise (with $i, j \in \{0, 1, \ldots\}$.

With this we can now approximate the probabilistic properties of the factorial function. In particular, if we look at the terminal configurations with the initial abstract configuration:

$$\boldsymbol{x}_0 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{n+1} & \cdots & \frac{1}{n+1} \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

which corresponds to a uniform distribution over all possible abstract values for our variables m and n (in fact, the part describing m could be any other distribution), then we get as final probabilistic configuration:

$$\boldsymbol{x} = \begin{pmatrix} \frac{n-1}{n+1} & \frac{2}{n+1} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{n+1} & \frac{n}{n+1} & 0 \ldots 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

This expresses the fact that indeed in most cases (with probability $\frac{n-1}{n+1}$) we get an even factorial – only in two cases out of $n+1$ (for 0 and 1) we get an odd result (namely 1). The final value of n is nearly always 1 except when we start with 0 and we always reach the final statement with label 5.

If we start with the abstract initial state $\boldsymbol{x}_0$ above and execute $\mathbf{T}^{\#}(F)$ until we get a fixpoint $\boldsymbol{x}$ we can use abstractions not to simplify the semantics but instead in order to extract the relevant information. Concretely we can use:

$$\mathbf{A} = \mathbf{I} \otimes \mathbf{A}_f \otimes \mathbf{A}_f$$

i.e. once we reached the terminal configuration (of the abstract execution) we ignore the value of n and the final label ℓ and only concentrate on the abstract, i.e. parity, values of m. Concretely we have to compute:

$$(\lim_{i\rightarrow\infty} \boldsymbol{x}_0 \cdot (\mathbf{T}^{\#}(F))^i) \cdot \mathbf{A}$$

Note that we always reach the fixpoint after a finite number of iterations (namely at most n) so this can be computed in finite time. The concrete probabilities we get for various n are:

n	**even**	**odd**
10	0.81818	0.18182
100	0.98019	0.019802
1000	0.99800	0.0019980
10000	0.99980	0.00019998

We can easily compute the final distribution on $\{\mathbf{even}, \mathbf{odd}\}$ for quite large n despite the fact that, as said, it is virtually impossible to compute the explicit representation of the concrete semantics $\mathbf{T}(F)$ already for $n = 6$.

Considering the factorial only for a limited number of inputs – here for n less than 10, 100, 1000, and 10000 – means in effect considering the finite approximations for the full factorial function F with any input $n \in \mathbb{Z}$. We are essentially computing

$$(\lim_{i\rightarrow\infty} \boldsymbol{x}_0 \cdot (\mathbf{T}_n^{\#}(F))^i) \cdot \mathbf{A} = (\lim_{i\rightarrow\infty} \boldsymbol{x}_0 \cdot (\mathbf{P}_n\mathbf{T}(F)\mathbf{P}_n)^i) \cdot \mathbf{A}$$

for $n = 10$, 100, 1000, and 10000. Numerically the last table of results shows that that for $n \rightarrow \infty$ we obviously get $P(\mathbf{even}) = 1$, and $P(\mathbf{odd}) = 0$.

We see that with a uniform distribution for n we get a vanishing probability $P(\mathbf{odd}) = 0$ for m. But we could also execute the abstract program with different initial distributions to get different estimates for $P(\mathbf{odd})$. If we take, for example, as a distribution for n something like $(\frac{1}{2}, \frac{1}{2}, 0, 0, \ldots)$ – which would mean that we only need to compute 0! and 1! – we obviously get the result $P(\mathbf{even}) = 0$ for m. A worst-case, safe analysis would thus result in $0 \leq P(\mathbf{even}) \leq 1$ – which is certainly correct but trivial – while by an average case analysis providing instead optimal estimates (for different initial distributions) we can achieve more useful results.

5 Related Work and Conclusions

The aim of this work is to provide a mathematically sound framework for probabilistic program analysis. The two main elements for this are (i) a compositionally defined semantics, called LOS, and (ii) a way to reduce the concrete semantics in order to obtain a more manageable abstract one via PAI. The concepts of a linear operator semantics and probabilistic abstract interpretation have been used before in the setting of *finite* domains in [3, 2, 10] for the analysis of programs and security properties. This paper extends PAI to infinite abstract domains.

Our LOS is closely related to a number of model which are popular in, for example, performance analysis, like Stochastic Automata Networks (SAN) [23, 24]. The idea of reducing the complexity of dynamical systems via least square approximations, which is at the core of our PAI approach, can also be found in various approaches ranging from Kalman filters, to model order reduction [25] and aggregation of Markov models [26]. However, to the best of our knowledge most of these models are finite dimensional.

While the theoretical framework of generalised inverses for finite-dimensional and bounded operators is well understood and relatively straight forward (e.g. [13, 11]) it is less well developed for the infinite dimensional unbounded case, see for example [15]. In the area of program analysis the well-known semantics for probabilistic programs by Kozen [6] and the application of classical abstract interpretation [27] to probabilistic languages are perhaps the closest alternatives to our LOS and PAI approach. The main differences are however (i) that the LOS – in contrast to Kozen's semantics – is not only able to capture the input/output behaviour but rather, as defined in terms of the generator of a Markov Chain, defines the details and intermediate steps of a computational process, and (ii) that PAI provides closest estimates to probabilities rather than worst case bounds for them. This makes PAI a more useful alternative in all those cases where the expected outcomes of a static analysis is not a yes-or-no answer but some estimates on which to base a speculative optimisation (compiler design, tradeoffs and cost analysis, etc).

References

1. Di Pierro, A., Wiklicky, H.: Concurrent Constraint Programming: Towards Probabilistic Abstract Interpretation. In: PPDP'00, ACM (2000) 127–138
2. Di Pierro, A., Hankin, C., Wiklicky, H.: A systematic approach to probabilistic pointer analysis. In: APLAS'07. Volume 4807 of LNCS., Springer (2007) 335–350

3. Di Pierro, A., Sotin, P., Wiklicky, H.: Relational analysis and precision via probabilistic abstract interpretation. In: QAPL'08. Volume 220(3) of ENTCS., Elsevier (2008) 23–42
4. Cousot, P., Cousot, R.: Systematic Design of Program Analysis Frameworks. In: Proceedings of POPL'79. (1979) 269–282
5. Kubrusly, C.S.: The Elements of Operator Theory. second edn. Birkhäuser (2011)
6. Kozen, D.: Semantics of probabilistic programs. J. Comput. Syst. Sci. **22**(3) (1981) 328–350
7. Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis. Springer (1999)
8. Roman, S.: Advanced Linear Algebra. second edn. Springer (2005)
9. Kadison, R., Ringrose, J.: Fundamentals of the Theory of Operator Algebras: Elementary Theory. AMS (1997) reprint from Academic Press edition 1983.
10. Di Pierro, A., Hankin, C., Wiklicky, H.: Measuring the confinement of probabilistic systems. Theoretical Computer Science **340**(1) (2005) 3–56
11. Ben-Israel, A., Greville, T.N.E.: Gereralized Inverses – Theory and Applications. second edn. CMS Books in Mathematics. Springer, New York (2003)
12. Böttcher, A., Silbermann, B.: Introduction to Large Truncated Toeplitz Matrices. Springer (1999)
13. Deutsch, F.: Best Approximation in Inner-Product Spaces. Springer (2001)
14. Pinkus, A.M.: On L^1-Approximation. Cambridge University Press (1989)
15. Groetsch, C.W.: Stable Approximate Evaluation of Unbounded Operators. Volume 1894 of Lecture Notes in Mathematics. Springer (2007)
16. Atkinson, K., Han, W.: Theoretical Numerical Analysis – A Functional Analysis Framework. third edn. Volume 39 of Texts in Applied Mathematics. Springer (2009)
17. Fabian, M., Habala, P., Hájek, P., Montesinos, V., Zizler, V.: Banach Space Theory – The Basis for Linear and Nonlinear Analysis. Springer (2011)
18. Groetsch, C.: Spectral methods for linear inverse problems with unbounded operators. Journal of Approximation Theory **70** (1992) 16–28
19. Groetsch, C.: Dykstra's algorithm and a representation of the Moore-Penrose inverse. Journal of Approximation Theory **117** (2002) 179–184
20. Groetsch, C.: An iterative stabilization method for the evaluation of unbounded operators. Proceedings of the AMS **134** (2005) 1173–1181
21. Nailin, D.: Finite-dimensional approximation settings for infinite-dimensional Moore-Penrose inverses. SIAM Journal of Numerical Analysis **46**(3) (2008) 1454–1482
22. Kulkarni, S., Ramesh, G.: Projection methods for computing Moore-Penrose inverses of unbounded operators. Indian Journal of Pure and Applied Mathematics **41**(5) (2010) 647–662
23. Plateau, B., Atif, K.: Stochastic automata network of modeling parallel systems. IEEE Trans. Softw. Eng. **17**(10) (1991) 1093–1108
24. Fourneau, J.M., Plateau, B., Stewart, W.: Product form for stochastic automata networks. In: Proceedings of ValueTools '07, ICST (2007) 32:1–32:10
25. Gugercin, S., Antoulas, A.: Model reduction of large-scale systems by least squares. Linear Algebra and its Applications **415** (2006) 290–321
26. Buchholz, P., Kriege, J.: Aggregation of markovian models - an alternating least squares approach. In: QEST. (2012) 43–52
27. Cousot, P., Monerau, M.: Probabilistic abstract interpretation. In Seidel, H., ed.: ESOP12. Volume 7211 of LNCS., Springer (2012) 166–190