Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 161 (2006) 131–150

www.elsevier.com/locate/entcs

Operator Algebras and the Operational Semantics of Probabilistic Languages

Alessandra Di Pierro[1]

Dipartimento di Informatica, Universitá di Pisa, Italy

Herbert Wiklicky[1]

Department of Computing, Imperial College London, United Kingdom

Abstract

We investigate the construction of linear operators representing the semantics of probabilistic programming languages expressed via probabilistic transition systems. Finite transition relations, corresponding to finite automata, can easily be represented by finite dimensional matrices; for the infinite case we need to consider an appropriate generalisation of matrix algebras. We argue that C*-algebras, or more precisely Approximately Finite (or AF) algebras, provide a sufficiently rich mathematical structure for modelling probabilistic processes.
We show how to construct for a given probabilistic language a unique AF algebra $\mathcal{A}$ and how to represent the operational semantics of processes within this framework: finite computations correspond directly to operators in $\mathcal{A}$, while infinite processes are represented by elements in the so-called strong closure of this algebra.

Keywords: Linear operator, C*-algebra, AF-algebra, operational semantics, probabilistic programming language,

1 Introduction

The operational semantics of programming languages is usually phrased in terms of the notion of *transition system*. Transition systems can be seen as abstract machines which specify an interpreter for the programming language via a binary relation on some state space.

We present a general method for transforming transition systems into an equivalent operator-algebraic semantics. More precisely, we show how to construct the semantics of a programming language as a continuous linear operator on a suitable space. This allows us to study and analyse programs using powerful tools

[1] The authors are partly funded by the EPSRC project S77066A "Quantitative Analysis of Computational Resources".

doi:10.1016/j.entcs.2006.04.029

from functional analysis. For example, we can use an appropriate operator norm (which depends on the specific application) to introduce a "measure" for the program semantics. Possible uses of such a measure are the definition of approximate programs properties (e.g. probabilistic termination [21], security properties [17]) and a quantitative comparison between programs, where the result needs not be boolean (programs are equivalent or not) and can be used to define approximative notions of process equivalence [16].

Intuitively, the idea of interpreting transition systems as matrices is very simple. In the classical case transition relations on a set X, representing the set of states or configurations, can be expressed by 0/1 matrices, that is linear operators on a space representing the elements in X. By considering matrices with generic (numerical) entries, we obtain the more general notion of a *quantitative relation*, of which probabilistic relations are typical examples.

We will show that probabilistic relations which are defined on (at most countable) infinite sets can be modelled via continuous linear operators, which represent the elements of a C*-algebra. The theory of C*-algebras, introduced by Gelfand and Naimark in the 1940s, offers an appropriate setting to deal with recursive definitions on infinite sets, where topological considerations are essential for the construction of a consistent general model.

We will define the semantics of probabilistic programming languages as follows. Starting from the probabilistic transition system defining the operational semantics of the language we construct a corresponding continuous linear operator on a Hilbert space built out of the computational states. We will illustrate our approach by showing the construction of the operators representing the operational semantics of a simple concrete probabilistic concurrent language, namely Probabilistic Concurrent Constraint Programming (PCCP) [18,21].

2 Preliminaries

Basic concepts in *functional analysis* and *operator theory* can be found in [45], [38], [6], [24]. To simplify our treatment we consider here only complex vector spaces and algebras, i.e. we assume that the base field is $\mathbb{C}$. We denote by $\overline{\cdot}$ the complex conjugation in $\mathbb{C}$, i.e. $\overline{x+iy} = x - iy$.

An *algebra* is a vector space $\mathcal{A}$ together with a map $\mathcal{A} \times \mathcal{A} \to \mathcal{A}$ denoted by $(a, b) \mapsto a \cdot b = ab$, which is bi-linear — i.e. $a(\alpha b) = \alpha ab$, $(\alpha a)b = \alpha ab$ for $\alpha \in \mathbb{C}$, and $(a + b)c = ac + bc$, $a(b + c) = ab + ac$ — such that $a(bc) = (ab)c$. An algebra with a norm which is also sub-multiplicative, i.e. $\|ab\| \leq \|a\|\|b\|$, is called a *normed algebra*. A normed algebra which is complete — with respect to the metric topology induced by the norm $d(x, y) = \|x - y\|$ — is called a *Banach algebra*. An *involutive algebra* or a **-algebra* is an algebra $\mathcal{A}$ together with a conjugate-linear — i.e. $(\alpha a)^* = \overline{\alpha} a^*$ for $\alpha \in \mathbb{C}$, and $(a + b)^* = a^* + b^*$ — map $\mathcal{A} \to \mathcal{A}$ denoted by $a \mapsto a^*$, such that $a^{**} = a$ and $(ab)^* = b^* a^*$. A *Banach *-algebra* is a complete normed involutive algebra such that $\|a^*\| = \|a\|$.

Definition 2.1 A *C*-algebra* is a Banach *-algebra such that: $\|a^* a\| = \|a\|^2$.

A norm (seminorm) which fulfills the condition $\|a^*a\| = \|a\|^2$ is called a *C*-norm* (*C*-seminorm*).

A simple example of a C*-algebra is the set $\mathcal{M}_n$ of finite dimensional $n \times n$ complex matrices. The scalar multiplication, addition and algebra product are the usual ones for matrices. The C*-norm of $a \in \mathcal{M}_n$ is given by the square root of the *spectral radius* ρ — i.e. the largest eigenvalue — of a^*a: $\|a\|^2 = \rho(a^*a)$.

Other examples of C*-algebras include the complex numbers, $\mathbb{C}$, the algebra of complex-valued continuous functions on a compact space X with pointwise operations, $C(X)$, and the algebra $\mathcal{B}(\mathcal{H})$ of bounded linear operators on a Hilbert space $\mathcal{H}$. A linear operator $\mathbf{T}$ on a Hilbert space $\mathcal{H}$ is *bounded* if its *operator norm* is bounded, i.e. if we have $\|\mathbf{T}\| = \sup_{\|x\|=1} \|\mathbf{T}(x)\| < \infty$, where the supremum is over all $x \in \mathcal{H}$ with norm 1. It can be shown that a linear operator $\mathbf{T}$ on $\mathcal{H}$ is continuous if and only if it is bounded [6, Prop 1.1]. The involution on the C*-algebra $\mathcal{B}(\mathcal{H})$ is the *adjoint* operation on $\mathcal{B}(\mathcal{H})$; given an operator $\mathbf{T}$ there exists a unique element $\mathbf{T}^* \in \mathcal{B}(\mathcal{H})$ such that $\langle \mathbf{T}^* x, y\rangle = \langle x, \mathbf{T}y\rangle$, with $x, y \in \mathcal{H}$ and $\langle \cdot, \cdot \rangle$ is the inner product in $\mathcal{H}$ [33, Thm 2.4.2]. $\mathbf{T}^*$ is called the adjoint of $\mathbf{T}$.

C*-algebras are particularly well behaved operator algebras from a topological viewpoint; in fact they are all isomorphic to a sub-algebra of $\mathcal{B}(\mathcal{H})$ (e.g. [24, Thm 2.2.1 & 5.4.1]).

Proposition 2.2 (Gelfand-Naimark) *Any C*-algebra is isometrically *-isomorphic to a C*-subalgebra of some $\mathcal{B}(\mathcal{H})$. If the C*-algebra is separable then $\mathcal{H}$ can be taken to be separable.*

A topological space is separable if it contains a countable dense subset, that is a set with a countable number of elements whose closure is the entire space. For normed spaces $\mathcal{A}$ such as C*-algebras and Hilbert spaces separability means in particular that there exists a countable set $\mathcal{S} \subseteq \mathcal{A}$ such that for all $\varepsilon > 0$ and all $a \in \mathcal{A}$, there exists $b \in \mathcal{S}$ such that $\|a - b\| < \varepsilon$.

All infinite dimensional separable C*-algebras can therefore be represented as C*-subalgebras of $\mathcal{B}(\ell^2)$ since every separable Hilbert space is isomorphic to the "standard" Hilbert space (see e.g. [33, Cor 2.2.13]) of infinite vectors:

$$\ell^2 = \ell^2(\mathbb{N}) = \{(x_i)_{i\in\mathbb{N}} \mid x_i \in \mathbb{C} : \sum_{i\in\mathbb{N}} |x_i|^2 < \infty\},$$

with standard norm defined as $\|x\|_2 = \|(x_i)_{i\in\mathbb{N}}\|_2 = \sqrt{\sum_{i\in\mathbb{N}} |x_i|^2}$.

It is common to distinguish between *abstract C*-algebras* which we denote by $\mathcal{A}$, $\mathcal{B}$, etc. with elements $a, b, \ldots \in \mathcal{A}$ and *concrete C*-algebras*, i.e. C*-algebras which are given as C*-subalgebras of some $\mathcal{B}(\mathcal{H})$ and whose elements are linear bounded operators denoted by $\mathbf{A}, \mathbf{B}, \ldots \in \mathcal{B}(\mathcal{H})$.

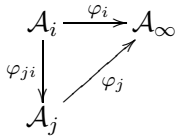
Apart from the norm topology there are several other important topologies on the concrete C*-algebra $\mathcal{B}(\ell^2)$ [9, Sect I.6]). In the norm topology a sequence of operators $(\mathbf{A}_n)_n$ in $\mathcal{B}(\ell^2)$ converges *uniformly* if there exists an operator $\mathbf{A} \in \mathcal{B}(\ell^2)$ such that $\lim_{n\to\infty} \|\mathbf{A}_n - \mathbf{A}\| = 0$. In the strong operator topology a sequence of operators $(\mathbf{A}_n)_n$ converges *strongly* if there exists an $\mathbf{A} \in \mathcal{B}(\ell^2)$ such that for all $x \in \ell^2$: $\lim_{n\to\infty} \|\mathbf{A}_n x - \mathbf{A}x\| = 0$. The strong operator topology is weaker than

the uniform or norm topology, i.e. convergence in the norm implies convergence in the strong topology but not vice versa. We write $\lim \mathbf{A}_n$ for the uniform limit and s-lim $\mathbf{A}_n$ for the strong limit. We denote by $\overline{\mathcal{A}}^s$ the strong closure of $\mathcal{A}$, i.e. the smallest strongly closed set containing $\mathcal{A}$, see e.g. [9, Section I.6].

Given an operator $\mathbf{M} \in \mathcal{B}(\ell^2)$, consider a sequence of (orthogonal) projections $\mathbf{P}_n : \ell^2 \to \ell^2$ onto the first n coordinates of ℓ^2, that is operators such that $\mathbf{P}_n^2 = \mathbf{P}_n = \mathbf{P}_n^*$. We call $\mathbf{M}_n = \mathbf{P}_n \mathbf{M} \mathbf{P}_n$ a *finite section* of $\mathbf{M}$. It corresponds effectively to taking the $n \times n$ sub-matrix in the upper left corner of the matrix representing $\mathbf{M}$. The sequence $(\mathbf{M}_n)_n$ is an *approximating sequence* for $\mathbf{M}$ in the sense that $\mathbf{M}$ is the strong limit of this sequence, i.e. $\mathbf{M} = \text{s-lim}\, \mathbf{M}_n = \text{s-lim}_{n\to\infty} \mathbf{P}_n \mathbf{M} \mathbf{P}_n$ (see e.g. [2, Sect 2.1]. This so called *finite section method* can be utilised in the analysis of infinite dimensional operators via finite approximations.

A special class of C*-Algebras are the Almost Finite C*-Algebras or AF algebras, first introduced by Bratteli [3]. AF algebra are separable C*-algebras constructed as the inductive limit of a sequence of finite dimensional C*-algebras. By Theorem III.1.1 in [9], every finite dimensional C*-algebra is a *unital* C*-algebra. A unital C*-algebra is a C*-algebra with a unit, i.e. an element e such that $a \cdot e = e \cdot a = a$ for all $a \in \mathcal{A}$. It is always possible to embed a non-unital C*-algebra in a unital one by considering the vector space $\mathcal{A}^+ = \mathcal{A} \oplus \mathbb{C}$ and defining: (i) the algebra product by $(a_1, \lambda_1)(a_2, \lambda_2) = (a_1 a_2 + \lambda_1 a_2 + \lambda_2 a_1, \lambda_1 \lambda_2)$, (ii) the involution by: $(a, \lambda)^* = (a^*, \overline{\lambda})$, and (iii) a C*-norm $\|(a, \lambda)\| = \sup_{\|b\| \le 1} \|ab + \lambda b\|$. The unit for $\mathcal{A} \oplus \mathbb{C}$ is given by $e = (o, 1)$, where o is the neutral element of the addition in $\mathcal{A}$. The algebra $\mathcal{A}^+$ is referred to as the *unitisation* of $\mathcal{A}$. The map $\mathcal{A} \to \mathcal{A}^+$ defined by $a \mapsto (a, 0)$ is an injective homomorphism which identifies $\mathcal{A}$ as an ideal of $\mathcal{A}^+$ [38, Sec 1.2].

The construction of the inductive limit of C*-algebras is given for example in [24, Sect 3.10] or [44, Appendix I]. Consider a sequence of finite dimensional C*-algebras $\{\mathcal{A}_n\}_{n\in\mathbb{N}}$ and unital *-homomorphisms $\varphi_{ij} : \mathcal{A}_j \to \mathcal{A}_i$ such that $\varphi_{ik} \circ \varphi_{kj} = \varphi_{ij}$ for $j < k < i$. Then, ignoring the topological structure of each $\mathcal{A}_n$ in the sequence and considering them as simple *-algebras, there exists a universal algebraic object $\mathcal{A}_\infty = \varinjlim \mathcal{A}_n$ of the same type as all $\mathcal{A}_n$'s, called the algebraic inductive limit of this sequence of algebras, and canonical morphisms $\varphi_i : \mathcal{A}_i \to \mathcal{A}_\infty$ such that the following diagram commutes whenever $i < i$:



and $\mathcal{A}_\infty = \bigcup_{n\in\mathbb{N}} \varphi_n(A_n)$. This object is universal in the category of *-algebras.

A concrete construction of the algebraic inductive limit $\mathcal{A}_\infty$ can be done as follows:

- construct $\mathcal{A}_\infty$ as the set of all sequences of operators $(a_i)_i$ where (i) $a_i \in \mathcal{A}_i$, (ii) there exists an i_0 such that for all $i > i_0$ we have $a_i = \varphi_{i_0 i}(a_{i_0})$, i.e. the sequence *eventually stabilises*,
- define the *-algebra operations on this set as follows: $\alpha(a_i)_i = (\alpha a_i)_i$, $(a_i)_i +$

$(b_i)_i = (a_i + b_i)_i$, $(a_i)_i \cdot (b_i)_i = (a_i \cdot b_i)_i$, and $(a_i)_i^* = (a_i^*)_i$.

We can now turn this *-algebra into a C*-algebra by defining an appropriate C*-norm. In general there is no immediate norm on $\mathcal{A}_\infty$, but we can define a C*-seminorm as follows:

$$\|(a_i)_i\| = \lim_i \|a_i\|.$$

The existence of this limit is guaranteed by the fact that *-homomorphisms between C*-algebras are norm-decreasing hence continuous (cf. e.g. Corollary 2.1.5 in [24]). A C*-algebra can be now constructed in two steps:

- Consider the set $\mathcal{N} = \{(a_i)_i \mid \|(a_i)_i\| = 0\}$, and construct the quotient *-algebra $\mathcal{A}_\infty/\mathcal{N}$. On this quotient the above C*-seminorm defines a C*-norm.
- construct the completion of $\mathcal{A}_\infty/\mathcal{N}$ with respect to this C*-norm.

In other words, an AF algebra is the completion of an algebra of sequences of elements in finite dimensional algebras. The inductive limit $\varinjlim \mathcal{A}_i$ is sometimes also denoted by $\overline{\bigcup_i \mathcal{A}_i}$, or by $\mathcal{A}_1 \to_{\varphi_{12}} \mathcal{A}_2 \to_{\varphi_{23}} \mathcal{A}_3 \to_{\varphi_{34}} \cdots$

Example 2.3 The *compact operators* or *completely continuous operators* $\mathcal{K} = \mathcal{K}(\ell^2)$ on the Hilbert space ℓ^2 form a non-unital C*-algebra. The unitisation $\mathcal{K}^+ = \mathcal{K} \oplus \mathbb{C}$ of $\mathcal{K}$ results in an AF algebra.

As in [9, Ex III.2.3] this can be constructed as the direct limit, $\varinjlim (\mathcal{M}_n \oplus \mathbb{C})$, of the algebras $\mathcal{M}_n \oplus \mathbb{C}$ where the connecting unital *-homomorphisms $\varphi_{n+1,n} : \mathcal{M}_n \oplus \mathbb{C} \to \mathcal{M}_{n+1} \oplus \mathbb{C}$ are given by:

$$\varphi_{n+1,n}(a \oplus \lambda) = \begin{pmatrix} a & 0 \\ 0 & \lambda \end{pmatrix} \oplus \lambda = \begin{pmatrix} a & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix}.$$

3 Linear Representations of Transition Relations

A transition relation is a binary relation $\to \subseteq S \times S$ on the set S of the program states, aka configurations. This notion is at the base of the definition of the operational semantics of programming languages, e.g. Plotkin's SOS semantics [39]. We will concentrate on probabilistic transition relations which are used in the context of probabilistic programming languages.

In general, a probabilistic relation on a set X is a subset $R \subset X \times [0,1] \times X$ such that $pr(x) = 1$ for all $x \in X$, where $pr(x) = \sum\{p \mid (x,p,y) \in R \text{ and } y \in X\}$. We will assume that X is a countable set. The normalisation condition above makes a probabilistic transition relation closely related to the transition matrix of a discrete time Markov chain [43]. A generalisation of this notion to uncountable state spaces requires a measure-theoretical treatment as done in [23,11].

Probabilistic transition relations on a set X can be characterised in terms of linear operators as follows. We first have to lift X to a vector space.

Definition 3.1 The *vector space* $\mathcal{V}(X)$ over a set X is the set of formal linear combinations, $\sum_{x\in X} c_x x$, of elements in X with coefficients $c_x \in \mathbb{C}$ which we can represent as possibly infinite sequences in $\mathbb{C}$ indexed by elements in X:

$$\mathcal{V}(X) = \{(c_x)_{x\in X} \mid c_x \in \mathbb{C}\}\,.$$

We can interpret a probabilistic relation R on X, as a function $R : X \times X \to [0,1]$ by adding all the weights associated to the same pair $(x,y) \in X \times X$, i.e. $R(x,y) = \sum_{(x,p,y)\in R} p$.

The matrix representing a probabilistic relation $R \subseteq X \times [0,1] \times X$ is defined by:

$$(\mathbf{M}_R)_{xy} = \begin{cases} p \text{ iff } R(x,y) = p \\ 0 \text{ otherwise} \end{cases}$$

This is a *stochastic matrix*, that is a positive matrix where the entries in each row sum up to one.

For finite sets X of cardinality n, the representation of a probabilistic relation as a linear operator on $\mathcal{V}(X)$ is a $n \times n$ matrix. Since $\mathcal{V}(X)$ is isomorphic to the n-dimensional complex vector space $\mathbb{C}^n$, the topological structure of the space of $n \times n$ matrices is unique [25, 1.22] and every linear operator is automatically continuous.

For infinite (countable) sets, however, the algebra of infinite matrices which we obtain this way is topologically "unstable" [29]. The algebra of infinite matrices has no universal topological structure and the notions of linearity and continuity do not coincide. It is therefore difficult to define the limit of a sequence of infinite matrices in a general way. To overcome this problem, we will restrict our attention to relations which can be represented as elements of a C*-algebra.

By the Gelfand-Naimark theorem (cf. Theorem 2.2), we can consider concrete bounded operators on the standard Hilbert space $\ell^2(X) \subseteq \mathcal{V}(X)$. The algebraic structure of a C*-algebra allows for exactly one norm topology [38, Cor. 2.1.2], and thus offers the same advantages as the linear algebra of finite dimensional matrices.

Our aim is to identify probabilistic relations which can be represented not just as linear operators but also as *continuous* linear operators in $\mathcal{B}(\ell^2(X))$, since continuity is a usual requirement for a semantics.

Given a set X and a probabilistic relation $R \subseteq X \times [0,1] \times X$ on X, the *range* or *orbit* of an element $x \in X$ is the set, $\mathcal{R}(x) = \{y \mid \exists n \in \mathbb{N} : (x,p,y) \in R^n \text{ with } p > 0\}$, of elements which are related to x via R, R^2, R^3, etc. By R^i, with $i \in \mathbb{N}$ we denote the i-th iterative application of R. This is defined by

$$R^1 = \{(x,1,x) \mid x \in X\},$$
$$R^i = R^{i-1} \cdot R,$$

where '$\cdot$' is the composition of two probabilistic relations; this is defined for all relations R and Q on X by:

$$R \cdot Q = \{(x,p,y) \mid \exists\, z \in X, \exists\, q_1, q_2 \in [0,1] \text{ such that } (x,q_1,z) \in Q \text{ and } (z,q_2,y) \in R \text{ and } p = q_1 q_2\}.$$

In terms of the operational semantics R^i corresponds to all those states which are reachable in exactly i steps. Thus $\mathcal{R}(x)$ encodes the elements $y \in X$ which are in the reflexive transitive closure of R — i.e. are reachable from x via a path of any length in the transition system R. The *orbit restriction*, $R(x)$, of R to the range of x is given by $R(x) = R|_{\mathcal{R}(x)} = \{(u,p,v) \mid u,v \in \mathcal{R}(x)\}$. If R is a transition relation, then the sub-relation $R(x)$ encodes the possible transitions starting with x — i.e. the execution tree of x.

Proposition 3.2 *Given a countable set X and a probabilistic relation $R \subseteq X \times [0,1] \times X$, then we can construct an AF algebra $\mathcal{A}(R)$ such that the operator representations $\mathbf{M}_{R|_Y}$ of R restricted to any finite subset Y of X are elements in $\mathcal{A}(R)$.*

Proof. In order to construct the algebra $\mathcal{A}(R)$, we can follow the general construction of AF algebras as inductive limit of finite dimensional C*-algebras (cf. Section 2). In fact, the following construction corresponds exactly to the construction of the algebra of the compact operators $\mathcal{K}$ shown in Example 2.3.

The algebra $\mathcal{A}(R)$ is defined using *single step operators*, represented by *matrix units* $\mathbf{E}_{xy}$ in the space of the $n \times n$ matrices $\mathcal{M}_n$ defined by: $(\mathbf{E}_{xy})_{st} = 1$ iff $s = x$ and $t = y$, and $(\mathbf{E}_{xy})_{st} = 0$ otherwise.

- Consider an enumeration ι of X and define $X_n = \{x \in X \mid \iota(x) \leq n\}$.
- Define the C*-algebra, $\mathcal{A}_n \subseteq \mathcal{M}_n$, generated by the set: $\{\mathbf{E}_{xy} \mid (x,p,y) \in R|_{X_n} \text{ for some } p > 0\}$.
- Transform $\mathcal{A}_n$ in the unital C*-algebra $\mathcal{A}_n^+ \subseteq \mathcal{M}_n \oplus \mathbb{C}$.
- Construct the standard unital *-homomorphisms $\varphi_{n+1,n} : \mathcal{A}_n^+ \to \mathcal{A}_{n+1}^+$ as

$$\varphi_{n+1,n} : a \oplus \lambda \mapsto \begin{pmatrix} a & 0 \\ 0 & \lambda \end{pmatrix} \oplus \lambda.$$

- Define $\mathcal{A}(R)$ as the inductive limit of these finite dimensional algebras

$$\mathcal{A}(R) = \varinjlim \mathcal{A}_n^+.$$

As any finite subset Y of X is contained in some X_n, we have that

$$\mathbf{M}_{R|_Y} = \sum_{x,y \in Y} p_{xy} \cdot \mathbf{E}_{xy},$$

with $\langle x, p_{xy}, y \rangle \in R$. Finally, we note that $\mathbf{M}_{R|_Y} \in \mathcal{A}_n$ as all the $\mathbf{E}_{xy}$'s are in $\mathcal{A}_n$.□

Corollary 3.3 *Given a countable set X and a probabilistic relation $R \subseteq X \times [0,1] \times X$, then $\mathcal{A}(R)$ is a C*-subalgebra of $\mathcal{K}^+ = \mathcal{K} \oplus \mathbb{C}$.*

Proof. Since by [9, Ex III.2.3] $\varinjlim \mathcal{M}_n^+ = \mathcal{K} \oplus \mathbb{C}$, and by Proposition 3.2 we have that

$$\mathcal{A}(R) = \varinjlim \mathcal{A}_n^+ \subseteq \varinjlim \mathcal{M}_n^+ = \mathcal{K} \oplus \mathbb{C}.$$

□

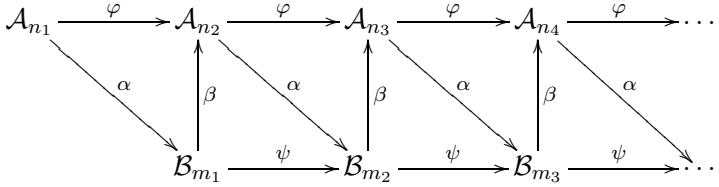
The construction of $\mathcal{A}(R)$ depends on a particular enumeration ι of X and we could obtain different algebras $\mathcal{A}(R)$ if we used a different enumeration. However,

we can show that the construction of $\mathcal{A}(R)$ is independent of the enumeration. There is a criterion to decide if two sequences generate the same AF algebra (cf. [3, Thm 2.7], [9, Thm III.3.5], [40, Ex 6.8]). Given two AF algebras

$$\mathcal{A} \simeq \mathcal{A}_1 \xrightarrow{\varphi_{21}} \mathcal{A}_2 \xrightarrow{\varphi_{32}} \mathcal{A}_3 \xrightarrow{\varphi_{43}} \cdots$$

$$\mathcal{B} \simeq \mathcal{B}_1 \xrightarrow{\psi_{21}} \mathcal{B}_2 \xrightarrow{\psi_{32}} \mathcal{B}_3 \xrightarrow{\psi_{43}} \cdots$$

if there exist *-homomorphisms $\alpha_{m_i,n_i} : \mathcal{A}_{n_i} \to \mathcal{B}_{m_i}$ and $\beta_{n_{i+1},m_i} : \mathcal{B}_{n_i} \to \mathcal{A}_{m_{i+1}}$ such that the diagram



commutes, then $\mathcal{A} \simeq \mathcal{B}$. For a probabilistic transition relation on a countable set we can always construct these *-homomorphisms.

Proposition 3.4 *Given a countable set X and a probabilistic relation $R \subseteq X \times [0,1] \times X$ on X, the AF algebra $\mathcal{A}(R)$ is unique (up to *-isomorphism).*

In other words, the AF algebra $\mathcal{A}(R)$ characterises the relation R. The AF algebra obtained this way also allows us to describe the expressiveness of programming languages (preliminary ideas on this were presented in [5]).

Under certain conditions, we can represent all orbit restrictions $R(x)$ of a probabilistic transition relation R (including the infinite ones) by operators in the strong closure, $\overline{\mathcal{A}(R)}^s$, of $\mathcal{A}(R)$. In order to show this formally in Theorem 3.6 we need some preliminary definitions and results which we state in the following.

For a state $x \in X$ in a probabilistic transition relation R on a countable set X we denote by out-deg(x) the number of *successors* of x, i.e. the cardinality of the set:

$$\{y \mid (x,p,y) \in R \text{ with } p > 0\},$$

and by in-deg(x) the number of *predecessors* of x, i.e. the cardinality of the set:

$$\{y \mid (y,p,x) \in R \text{ with } p > 0\}.$$

A probabilistic transition relation R with finite out-deg(x) for all $x \in X$ is usually called a *finitely branching* probabilistic transition relation. We will require that a transition systems satisfy a stronger property, namely that $\sup_{y \in X}$ in-deg$(y) < \infty$. We will call such systems *strongly finitely branching*.

Proposition 3.5 *Let X be a countable set and let R be a probabilistic transition relation on X such that* $\sup_{y \in X}$ in-deg$(y) < \infty$ *and* $\sup_{y \in X}$ out-deg$(y) < \infty$. *Then the operator representation $\mathbf{M}_R$ of R defines a bounded linear operator, i.e. we have $\mathbf{M}_R \in \mathcal{B}(\ell^2(X))$.*

Proof. In the following we will use the shortcut $\mathbf{M}$ for $\mathbf{M}_R$. We show that for all $v = (v_s)_{s\in X} \in \ell^2(X))$ such that $\|v\|_2 = 1$, we have $\|\mathbf{M}(v)\|_2 < \infty$. We have that

$$\|\mathbf{M}(v)\|_2^2 = \sum_{j=1}^{\infty}(\sum_{i=1}^{\infty}\mathbf{M}_{ij}v_i)^2.$$

Let $m = \sup_{s\in X}\text{in-deg}(s)$ and $n = \sup_{s\in X}\text{out-deg}(s)$. This means that in each column i of $\mathbf{M}$ there are at most $m(i) \leq m$ non-zero entries $\mathbf{M}_{f_1(i)i}, \mathbf{M}_{f_2(i)i}, \ldots \mathbf{M}_{f_{m(i)}i}$. The functions $f_1, \ldots, f_m$ are functions picking out the non-zero entries in each column i in decreasing order, i.e. we assume that $v_{f_1(i)} \geq v_{f_2(i)} \geq \ldots \geq v_{f_{m(i)}(i)}$. Since $\mathbf{M}_{ij} \leq 1$ for all i, j, we get:

$$\|\mathbf{M}(v)\|_2^2 = \sum_{j=1}^{\infty}(\sum_{i=1}^{m(j)}\mathbf{M}_{f_i(j)j}v_{f_i(j)})^2 \leq \sum_{j=1}^{\infty}(\sum_{i=1}^{m(j)}v_{f_i(j)})^2 \leq \sum_{j=1}^{\infty}(mv_{f_1(j)})^2$$

Since $\sup_{s\in X}\text{out-deg} = n < \infty$, we have that for every row k the number of i's such that $f_1(i) = k$ cannot be greater than n. We therefore have:

$$\|\mathbf{M}(v)\|_2^2 \leq m^2\sum_{j=1}^{\infty}v_{f_1(j)}^2 \leq m^2n\sum_{k=1}^{\infty}v_k^2 = nm^2\|v\|_2^2.$$

Therefore, $\|\mathbf{M}\|_2 = \sup_{\|v\|_2=1}\|\mathbf{M}(v)\|_2 \leq nm^2 < \infty$. □

Theorem 3.6 *Given a countable set X and a strongly finitely branching probabilistic transition relation $R \subseteq X \times [0,1] \times X$ such that $\sup_{y\in X}\text{in-deg}(y) < \infty$, let $\mathcal{A}(R)$ be the AF algebra associated to R. Then the operator representations $\mathbf{M}_{R(x)}$ of all orbit restrictions $R(x)$ of R, for $x \in X$ are elements in the strong closure $\overline{\mathcal{A}(R)}^s$ of $\mathcal{A}(R)$.*

Proof. By Proposition 3.5, $\mathbf{M}_{R(x)}$ is bounded. Therefore, by applying the finite section method we can construct $\mathbf{M}_{R(x)}$ as the strong limit of its finite sections $(\mathbf{M}_{R(x)})_n$. Since each finite section is of the form $\mathbf{M}_{R|_Y}$ for some finite set subset Y of X, by Proposition 3.2 we have that $(\mathbf{M}_{R(x)})_n \in \mathcal{A}(R)$ for all n, and therefore $\mathbf{M}_{R(x)} = \text{s-lim}(\mathbf{M}_{R(x)})_n \in \overline{\mathcal{A}(R)}^s$. □

This result introduces a new way to look at the semantics of programming languages which lends itself to a more "quantitative" approach towards program analysis and reasoning about programs. Current work has already shown this view to be particularly appropriate in many areas such as concurrency and security analysis. For example, the use of a linear operator based operational semantics allows us to provide quantitative estimations of the result of process equivalences and program properties as well as of the result of a given static analysis [16,15,13,14,17].

This general method of defining an algebra of operators for expressing the meaning of a program can be instantiated to any particular programming language or process calculus whose operational semantics can be defined via a transition system (e.g. in the Plotkin SOS style). Adequacy results can then be established showing the relationship with the original standard operational semantics. We will illustrate this in the next section where we consider as an example a declarative constraint-based programming language.

4 A Probabilistic Language and its Operator-algebraic Semantics

We apply the approach described in the previous section to the definition of an operator semantics for a simple probabilistic language. The language we consider is Probabilistic Concurrent Constraint Programming (PCCP), which was introduced in [18,21] as a probabilistic version of the Concurrent Constraint Programming (CCP) paradigm [41]. This language can be seen as a kind of "process algebra" enhanced with a notion of "computational state", referred to as "store". These states are ordered by an entailment relation $\vdash$ (also denoted by $\sqsupseteq$), and all computations lead to sequences of stores which are *monotone* with respect to $\vdash$. Other systems for probabilistic constraint programming can be found in the literature, e.g. [27,28], for which a linear operator semantics can be defined in a similar way as for PCCP.

The syntax and the basic execution model of PCCP are based on the central notion of a generic *constraint system* $\mathcal{C}$. Following [42], a constraint system is modelled as a complete algebraic lattice in which the ordering $\sqsubseteq$ is the reverse of the the entailment relation ($c \sqsubseteq d$ means that d contains "more information" than c). The top element *false* represents inconsistency, the bottom element *true* is the empty constraint, and the *least upper bound* (lub) $\sqcup$ represents the join of information, i.e. the logical *and*. In order to model *hiding* of local variables and *parameter passing* in constraint programming, in [42] the notion of constraint system is enriched with *cylindrification operators* and *diagonal elements*, concepts borrowed from the theory of cylindric algebras [30] (see [41,10] for more details). The combined use of the cylindrification operators and diagonal elements allow us to model variable renaming by representing $\phi[y/x]$ as the formula $\exists_x(\delta_{xy} \sqcup \phi)$. In fact, if cylindrification is interpreted as the first-order existential operator, and δ_{xy} as equality between x and y, then $\exists_x(\delta_{xy} \sqcup \phi)$ has precisely the meaning of the formula derived from ϕ by replacing all the free occurrences of x by y.

The syntax of a PCCP agent is given by the following grammar, where c and c_i are *finite* constraints in $\mathcal{C}$, and p_i and q_i are real numbers representing probabilities:

$$A ::= \mathbf{stop} \;\Big|\; \mathbf{tell}(c) \;\Big|\; \textstyle\prod_{i=1}^{n} \mathbf{ask}(c_i) \to p_i : A_i \;\Big|\; \|_{i=1}^{n}\, q_i : A_i \;\Big|\; \exists_x A \;\Big|\; p(x).$$

The two probabilistic operators of PCCP are represented by the probabilistic choice construct and a form of probabilistic parallelism, which replaces the pure nondeterministic scheduler by a probabilistic one in the interleaving semantics of CCP. The language also provides a construct, $\exists_x A$, for expressing locality.

4.1 Operational semantics

The operational model of PCCP can be intuitively described as follows: All processes share a common store consisting of the least upper bound, denoted by $\sqcup$, (with respect to the inverse $\sqsubseteq$ of the entailment relation) of all the constraints established up to that moment by means of **tell** actions. These actions allow for communication. Synchronisation is achieved via an **ask** guard which tests whether the store entails a given constraint. The probabilistic choice construct allows for

a random selection of one of the different possible synchronisations making the program similar to a stochastic process like a *random walk* [26].

The operational semantics of PCCP is defined in terms of a probabilistic transition system, $(\mathrm{Conf}, \longrightarrow_p)$, where Conf is the set of configurations $\langle A, d\rangle$ representing the state of the system at a certain moment and the transition relation $\longrightarrow_p$ is defined in Table 1. The state of the system is described by the agent A which has still to be executed, and the common store d. The index p in the transition relation indicates the probability of the transition to take place. The rules in Table 1 are closely related to the ones for nondeterministic CCP [10], and we refer to [21] for a more detailed description. The rules **R2** and **R3** for probabilistic choice and prioritised parallelism involve a normalisation process needed to re-distribute the probabilities among those agents A_i which can actually be chosen for execution. Such agents must be enabled (i.e. the corresponding guards $\mathbf{ask}(c_i)$ succeed) or active (i.e. able to make a transition). We will consider the agent **stop** to be not active, although in our operational model we will define it as an agent looping on itself (see Section 4.2). The probability after normalisation is denoted by $\tilde{p}_j$ and is defined by

$$\tilde{p}_j = \frac{p_j}{\sum_i p_i},$$

where the sum $\sum_{p_i}$ is over all enabled agents (i.e. those for which c_j is entailed by the store d) for **R2**, while for **R3** it is over all the active ones (i.e. those which can make a transition and are not the **stop** agent. There might occur the situation where all enabled/active agents have probability zero; in this case the normalisation will consist in the assignment of a uniform distribution to the enabled/active agents (see [21] for further details). The meaning of rule **R4** is intuitively explained by saying that the agent $\exists_x^d A$ behaves "almost" like A, with the difference that the variable x which is possibly present in A must be considered local, and that the information present in d has to be taken into account. The semantics of a procedure call $p(x)$, modelled by Rule **R5**, consists in the execution of the agent A defining $p(x)$ with a parameter passing mechanism similar to call-by-reference: the formal parameter x is linked to the actual parameter y in such a way that y inherits the constraints established on x and vice-versa. This is realised in a way to avoid clashes between the formal parameter and occurrences of y in the agent via the operator Δ_y^x defined by: $\Delta_y^x A = \exists_y^{d_{xy}} A$ if $x \neq y$ and $\Delta_y^x A = A$ if $x = y$.

4.2 Observables

The operational semantics introduced above allows us to define various notions of observables. We will consider observables which correspond to the I/O behaviour of a program. More precisely, we will consider for an agent A, the results of all finite and infinite computations starting with configuration $\langle A, d\rangle$ (typically $\langle A, \mathit{true}\rangle$, where *true* is the constraint representing the empty store) with their associated probability. Formally, these results can be represented as pairs $\langle c, p\rangle$, where c is the "final" store (i.e. the least upper bound of the partial constraints established at each step of the computation) and p is the product of the single step probabilities.

R1	$\langle \mathbf{tell}(c), d\rangle \longrightarrow_1 \langle \mathbf{stop}, c \sqcup d\rangle$	
R2	$\langle \Box_{i=1}^{n} \mathbf{ask}(c_i) \to p_i : A_i, d\rangle \longrightarrow_{\tilde{p}_j} \langle A_j, d\rangle$	$j \in [1,n]$ and $d \vdash c_j$
R3	$\dfrac{\langle A_j, c\rangle \longrightarrow_p \langle A'_j, c'\rangle}{\langle \|_{i=1}^{n}\ p_i : A_i, c\rangle \longrightarrow_{p\cdot\tilde{p}_j} \langle \|_{j\neq i=1}^{n}\ p_i : A_i \parallel p_j : A'_j, c'\rangle}$	$j \in [1,n]$
R4	$\dfrac{\langle A, d \sqcup \exists_x c\rangle \longrightarrow_p \langle A', d'\rangle}{\langle \exists_x^d A, c\rangle \longrightarrow_p \langle \exists_x^{d'} A', c \sqcup \exists_x d'\rangle}$	
R5	$\langle p(y), c\rangle \longrightarrow_1 \langle \Delta_y^x A, c\rangle$	with $p(x) : -A \in P$

Table 1
The Transition System for PCCP

If there is more than one computation leading to the same constraint, then we have to sum up all the probabilities coming from each such computation.

We will treat finite and infinite computations in a uniform way by defining a finite computation as an infinite one where there is a loop on the last configuration $\langle \mathbf{stop}, c_n\rangle \longrightarrow_1 \langle \mathbf{stop}, c_n\rangle$, for some $n < \infty$.

Definition 4.1 Let A be a PCCP agent. A computational path π for A in store d is defined by $\pi \equiv \langle A_0, c_0\rangle \longrightarrow_{p_0} \langle A_1, c_1\rangle \longrightarrow_{p_1} \ldots \longrightarrow_{p_{i-1}} \langle A_i, c_i\rangle \longrightarrow \ldots$,where $A_0 = A$, $c_0 = d$. A path π is finite if for some $n < \infty$, we have that $A_j = \mathbf{stop}$ and $c_j = c_n$ for all $j \geq n$.

We denote by π_n the n-step prefix of a path π and by $\mathrm{Comp}(A, d)$ the set of all computational paths for A in the initial store d.

Definition 4.2 Given an agent A and an initial store d, we define the n-step partial results $Res_n(A, d)$ of A in store d as the multiset:

$$\left\{ \left\langle \bigsqcup_{i=0}^{n} c_i, \prod_{i=0}^{n} p_i \right\rangle \;\middle|\; \begin{array}{l} \text{there exists } \pi \in \mathrm{Comp}(A,d) \text{ such that:} \\ \pi_n = \langle A, d\rangle \longrightarrow_{p_0} \langle A_1, c_1\rangle \ldots \longrightarrow_{p_{n-1}} \langle A_n, c_n\rangle \end{array} \right\}.$$

The n-step observables for A in d are then defined as the aggregated n-step partial results:

$$\mathcal{O}_n(A, d) = \{\langle c, p\rangle \mid c \in \mathcal{C} \text{ and } p = \sum_{\langle c, p_j\rangle \in Res_n(A,d)} p_j\}.$$

Since PCCP programs evolve monotonically, for finite computations $\bigsqcup_{i=0}^{n} c_i$ corresponds exactly to the final constraint c_n. For infinite computation we are interested in observing the behaviour of programs which do not terminate and yet compute more and more refined approximations to a (infinite) limit constraint. This is

captured by the definition of a notion of operational *observables* in terms of the limit of finite approximations in the norm topology of the Hilbert space on the constraint system.

Lemma 4.3 *Each $\mathcal{O}_n(A,d)$ is a vector in the Hilbert space $\ell^2(\mathcal{C})$ on the constraint system.*

Proof. By definition a vector in $\ell^2(\mathcal{C})$ must satisfy $\sum_{c\in\mathcal{C}} p_c^2 < \infty$. since every $\mathcal{O}_n$ has only finitely many non-zero probabilities, the sum of the squares of these probabilities is therefore finite. □

Definition 4.4 Given an agent A and an initial store d, we define the observables of A in store d as the partial function $\mathcal{O} : \mathrm{Conf} \to \ell^2(\mathcal{C})$

$$\mathcal{O}(A,d) = \lim_{n\to\infty} \mathcal{O}_n(A,d),$$

where lim is the limit in the Hilbert space $\ell^2(\mathcal{C})$ i.e. the vector which satisfies $\lim_{n\to\infty}\|\mathcal{O}_n(A,d) - \mathcal{O}(A,d)\| = 0$.

Note that there are programs for which this limit does not exist, that is programs whose infinite behaviour does not represent the gradual construction of an infinite object. Typical examples are infinite deterministic programs (see Example 4.5). Note also that this notion of observables does not in general correspond to the pointwise limit of the distributions $\mathcal{O}_n(A,d)$. For *finite* computations — i.e. results obtained by finite observation — ℓ^2 and pointwise convergence always coincide; however, in the infinite case we obtain two different notions of infinite observables.

Example 4.5 Consider the following PCCP program:

$$p(x) :- 1 : \exists_y(\mathbf{tell}(x = s(y)) \parallel 0 : p(x)).$$

We have that for all $n \in \mathbb{N}$, $\mathcal{O}_n(p(x), \mathit{true}) = \{\langle \cup_{i=1}^{n} \exists_{y_i}(x = s(y_i)), 1\rangle\}$, and it is easy to check that the sequence $(\mathcal{O}_n(p(x), \mathit{true}))_{n\in\mathbb{N}}$ does not converge in the ℓ^2 norm topology, while its pointwise limit is the zero vector $(0, 0, 0, \ldots)$.

4.3 Linear Representation

The probabilistic transition relation $\longrightarrow_p$ on the countable set of configurations Conf corresponds to an AF algebra $\mathcal{A}(\longrightarrow_p)$ which we can construct in the way described in Section 3. From Proposition 3.2 we see immediately that the execution tree of a terminating program can be represented by an operator in $\mathcal{A}(\longrightarrow_p)$. For infinite programs we can construct an operator in the strong closure of $\mathcal{A}(\longrightarrow_p)$ by Proposition 3.6, provided that the conditions of finite $\sup_{y\in X}$ in-deg(y) and $\sup_{y\in X}$ out-deg(y) are satisfied.

We have to distinguish the construction of $\mathbf{F} \in \varinjlim \mathcal{A}_i = \mathcal{A}(\longrightarrow_p)$ representing the full SOS semantics of PCCP, i.e. the relation $\longrightarrow_p$, and $\mathbf{F}^{\langle A,d\rangle} \in \varinjlim \mathcal{A}_i^{\langle A,d\rangle}$ representing only the transition tree for A executed in d, or in other words, the restriction of $\longrightarrow_p$ to the orbit of $\langle A, d\rangle$ (i.e. configurations reachable from $\langle A, d\rangle$).

Each $\mathcal{A}_i^{\langle A,d\rangle}$ is the algebra generated by the matrix units representing the transitions occurring in the computational tree for $\langle A, d\rangle$ including only the configurations with index $j \leq i$ in a fixed enumeration of configurations. The restriction of $\longrightarrow_p$ to configurations reachable from $\langle A, d\rangle$ and with index $j \leq i$ is represented by the operator $\mathbf{F}_i^{\langle A,d\rangle} \in \mathcal{A}_i^{\langle A,d\rangle}$. Each such operator can be represented by an $i \times i$ matrix and implements a linear map $\mathbf{F}_i^{\langle A,d\rangle} : \ell^2(\mathcal{C}_i) \to \ell^2(\mathcal{C}_i)$ with $\mathcal{C}_i = \{c \in \mathcal{C} \mid \iota(\langle A', c\rangle) \leq i, \langle A', c\rangle \in \mathrm{Conf}\}$. Note that $\ell^2(\mathcal{C}_i) \simeq \mathcal{V}(\mathcal{C}_i)$ as $|\mathcal{C}_i|$ is finite and by Corollary 2.2.13 in [33] all Hilbert spaces with finite dimension n are isomorphic to $\mathbb{C}^n$. By choosing an enumeration ι we can show the following:

Proposition 4.6 *Let $\pi \in \mathrm{Comp}(A, d)$ be a computational path for $\langle A, d\rangle$, i.e. $\pi = \langle A, d\rangle \longrightarrow_{p_1} \langle A_1, c_1\rangle \longrightarrow_{p_2} \ldots \longrightarrow_{p_i} \langle A_i, c_i\rangle \longrightarrow_{p_{i+1}} \ldots$, and let $\mathbf{F}_i^{(A,d)} \in \mathcal{A}_i^{(A,d)}$. Then for all i, we have:*

$$\mathcal{P}_{\langle A_i,c_i\rangle}(\mathbf{F}_i^{\langle A,d\rangle}(\widehat{\langle A_{i-1}, c_{i-1}\rangle})) = p_i,$$

where $\widehat{\langle A, c\rangle}$ represents the vector in $\ell^2(\mathit{Conf})$ with all coefficients zero except an entry one for the $\langle A, c\rangle$-th coefficient and $\mathcal{P}_{\langle A,c\rangle} : \ell^2(\mathit{Conf}) \to \mathbb{C}$ extracts the $\iota(\langle A, c\rangle)$-th coordinate of a vector in $\ell^2(\mathit{Conf})$.

Intuitively this means that we can reconstruct every path in $\mathrm{Comp}(A, d)$ from the sequence of the $\mathbf{F}_i^{\langle A,d\rangle}$.

We denote by $\mathcal{V}c : \ell^2(\mathrm{Conf}) \to \ell^2(\mathcal{C})$ the map extracting the constraint part of configurations, i.e. $\mathcal{V}c : (\langle\langle A_k, c_k\rangle, p_k\rangle)_k \to (\left\langle c_l, \sum_{c_k=c_l} p_k \right\rangle)_l$. It follows that the i-th front of the execution of A in d is given by $\mathcal{O}_i \langle A, d\rangle = \mathcal{V}c((\mathbf{F}_i^{\langle A,d\rangle})^i \langle A, d\rangle)$. Therefore, we have the following result.

Theorem 4.7 (Correspondence) *For all agents A and initial stores d:*

$$\mathcal{O}(A, d) = \lim_{n\to\infty} \mathcal{V}c(\mathbf{F}^{(A,d)})^n(\widehat{\langle A, d\rangle})).$$

We illustrate these constructions in the following example.

Example 4.8 [Infinite computations] Consider the following program:

$$\begin{aligned} nat(x) :- \ & true \to \tfrac{1}{2} : \mathbf{tell}(x = 0) \\ [] \quad & true \to \tfrac{1}{2} : \exists_y(\tfrac{1}{2} : \mathbf{tell}(x = s(y)) \parallel \tfrac{1}{2} : nat(y)). \end{aligned}$$

We will use the following shorthand notation for the constraints involved $x = 0 \equiv 0$, $x = s(0) \equiv 1$, $x = s(s(0)) \equiv 2$, $\ldots \exists_y x = s(y) \equiv *$, and we enumerate the relevant configurations as follows: $\iota(\langle nat(x), true\rangle) = 1$, $\iota(\langle nat(y_n), *\rangle) = 2n + 1$ for $n \geq 1$, and $\iota(\langle \mathbf{stop}, n\rangle) = 2n + 2$ for $n \geq 0$ (with y_n denoting the n-th local y, so that $y_0 \equiv x$).

To construct an operator representing the operational semantics starting from the initial configuration $\langle nat(x), true\rangle$ we consider the following sequence of $n \times n$

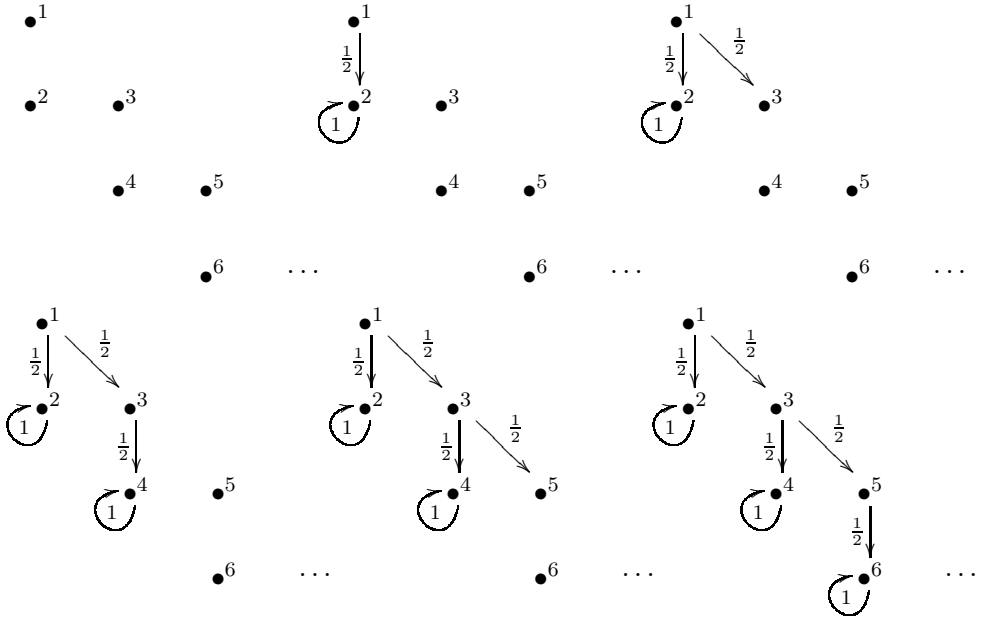
matrices F_n, $n \geq 1$:

$$\left(0\right), \begin{pmatrix} 0 & \frac{1}{2} \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \ldots$$

In general we have:

$$(\mathbf{F}_n)_{i,j} = \begin{cases} \frac{1}{2} \text{ for } i = 2k+1 \ \wedge \ (j = 2k+2 \vee j = 2k+3) \text{ with } 0 \leq k \leq n/2 \\ 1 \text{ for } i = 2k = j \text{ with } 0 \leq k \leq n/2 \\ 0 \text{ otherwise} \end{cases}$$

This corresponds to an iterative expansion of the infinite derivation tree:



We construct for each of these matrices the corresponding finite dimensional unital C*-algebras $\mathcal{A}_i^{\langle nat,true\rangle +} \simeq \mathcal{A}_i^{\langle nat,true\rangle} \simeq \mathcal{M}_i$. The inductive limit is given by:

$$\mathcal{A}^{\langle nat,true\rangle} = \varinjlim \mathcal{A}_i^{\langle nat,true\rangle +} = \varinjlim \mathcal{M}_i = \mathcal{K}.$$

Importantly, the sequence $(\mathbf{F}_i)_i$ does not eventually stabilise, i.e. the sequence $(\mathbf{F}_i)_i$ is not itself an element in the algebraic direct limit $\mathcal{A}_\infty^{\langle nat,true\rangle}$. The sequence $(\mathbf{F}_i)_i$ also does not converge in the uniform topology. This sequence thus does not represent (i.e. converge to) an operator in the (norm) closure $\overline{\mathcal{A}}_\infty^{\langle nat,true\rangle} = \mathcal{A}^{\langle nat,true\rangle}$.

However, the sequence $(\mathbf{F}_i)_i$ converges in the strong topology, i.e. there exists an operator $\mathbf{F} \in \mathcal{B}(\ell^2(\mathrm{Conf}))$ such that for all vectors $\vec{x} \in \ell^2(\mathrm{Conf})$ we have: $\lim_{i\to\infty} \|\mathbf{F}(\vec{x}) - \mathbf{F}_i(\vec{x})\|_2 = 0$. The operator $\mathbf{F}$ in $\mathcal{B}(\ell^2(\mathrm{Conf}))$ is represented by the infinite matrix:

$$(\mathbf{F})_{i,j} = \begin{cases} \frac{1}{2} \text{ for } i = 2k+1 \ \wedge \ (j = 2k+2 \vee j = 2k+3) \text{ with } k = 0,1,2,\ldots \\ 1 \text{ for } i = 2k = j \text{ with } k = 0,1,2,\ldots \\ 0 \text{ otherwise} \end{cases}$$

In other words, the operational semantics of $\langle nat, true\rangle$ is represented by the strong limit:

$$\text{s-}\lim_{i\to\infty} \mathbf{F}_i = \mathbf{F} \in \overline{\mathcal{A}^{\langle nat,true\rangle}}^{s}.$$

The initial configuration $\langle nat, true\rangle$ is represented by $\vec{x}_0 = (1,0,0,0,\ldots) \in \ell^2(\mathrm{Conf})$. The iterations of $\mathbf{F}$ give us the following sequence of vectors:

$$\begin{aligned} \mathbf{F}^1(\vec{x}_0) &= (0,1/2,1/2,0,0,0,0,0,0,\ldots) \\ \mathbf{F}^2(\vec{x}_0) &= (0,1/2,0,1/4,1/4,0,0,0,\ldots) \\ \mathbf{F}^3(\vec{x}_0) &= (0,1/2,0,1/4,0,1/8,1/8,0,\ldots) \\ &\ldots \quad \ldots \end{aligned}$$

It is easy to see that this sequence converges in the norm topology on $\ell^2(\mathrm{Conf})$; in fact, we have that $\lim_{i\to\infty} \mathbf{F}^i(\vec{x}_0) = \vec{x} = (x_i)_i$

with $x_{2k} = \frac{1}{2^k}$ and $x_{2k+1} = 0$ for all $k = 1,2,\ldots$. From this vector $\vec{x}$ in $\ell^2(\mathrm{Conf})$ we can obtain a vector $\mathcal{V}c(\vec{x}) \in \ell^2(\mathcal{C})$ which represents exactly the operational observables:

$$\mathcal{O}(nat, true) = \{\langle 0,1/2\rangle, \langle 1,1/4\rangle, \langle 2,1/8\rangle, \ldots\}.$$

Example 4.9 Consider the program $p(x)$ of Example 4.5. By fixing an enumeration of the constraint system $\mathcal{C}$ in which constraints $\{\cup_{i=1}^{n} \exists_{y_i}(x = s(y_i))\}_{n\in\mathbb{N}}$ get successive numbers, the linear operator associated to $p(x)$ is the unital shift on $\mathbb{N}$ whose matrix representation is given by:

$$\mathbf{S}_{ij} = \begin{cases} 1 \text{ if } j = i+1 \\ 0 \text{ otherwise.} \end{cases}$$

It is easy to check that $\|\mathbf{S}\| = 1$, as we have for all vectors $\vec{x} \in \ell^2(\mathrm{Conf})$, $\|\mathbf{S}(\vec{x})\| = \|\vec{x}\|$, and thus $\mathbf{S} \in \mathcal{B}(\ell^2)$. By Proposition 3.6, we have therefore that $\mathbf{S}$ is in the strong closure of $\mathcal{A}(\longrightarrow_p)$.

However, the sequence obtained by iteratively applying $\mathbf{S}$ does not converge in the ℓ^2 norm. In fact, starting for example from vector $\vec{x}_0 = (1,0,0,0,0,\ldots)$,

representing constraint *true*, we get the following sequence of vectors:

$$\begin{aligned}
\mathbf{S}^0(\vec{x}_0) &= (1,0,0,0,0,\ldots) = \vec{x}_0 \\
\mathbf{S}^1(\vec{x}_0) &= (0,1,0,0,0,\ldots) = \vec{x}_1 \\
\mathbf{S}^2(\vec{x}_0) &= (0,0,1,0,0,\ldots) = \vec{x}_2 \\
\mathbf{S}^3(\vec{x}_0) &= (0,0,0,1,0,\ldots) = \vec{x}_3 \\
&\ldots \quad \ldots
\end{aligned} \qquad \text{i.e.} \qquad (\vec{x}_i)_j = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$$

This sequence does not form a Cauchy sequence, as $\|\vec{x}_i - \vec{x}_k\|_2 = \sqrt{2}$ whenever $i \neq j$, that is there is no vector $\vec{x} \in \ell^2$ such that $\lim \|\vec{x} - \vec{x}_i\|_2 = 0$. This reflects the fact that, as shown in Example 4.5, $p(x)$ does not have infinite observables as $\mathcal{O}(p(x), \mathit{true})$ is not defined.

5 Conclusions and Related Work

We presented a characterisation of probabilistic transition relations in terms of linear operators. We identified AF algebras as suitable domains for representing relations when the state spaces are countably infinite. The framework presented here allows for the application of novel techniques and methods — borrowed from diverse areas like symbolic dynamics [36], algebraic graph theory [1] and in particular (topological) Markov Chains [8] — in order to understand and analyse the semantics of probabilistic languages. This framework constitutes a base for the application of operator algebraic techniques to language semantics and program analysis.

Our work on linear structures in semantics has originally been motivated by an attempt to develop a *quantitative* version of static program analysis extending the classical Cousot & Cousot approach of Abstract Interpretation [7]. The resulting framework, Probabilistic Abstract Interpretation (PAI,[20,22]), is based on the notion of the so called Moore-Penrose pseudo inverse which replaces the order theoretic concept of a Galois connection. In particular, within the PAI setting we are able to analyse the precision of an abstraction in quantitative terms [22]. The PAI framework allowed us to relate probabilistic bisimulation with probabilistic abstractions [16] and to recast it in a way which closely resembles the original notion of "lumpability" for Markov chains [34]. While the original formulation of PAI had been given in terms of finite dimensional vector spaces and matrices, it is the C*-algebra setting presented here which provides a convenient setting for PAI in infinite dimensions [2].

Our linear operator approach proved useful in the area of security where we were able to introduce approximate notions of confinement based on various notions of observations [17,13].

One can argue that C*-algebras constitute a much too large mathematical structure than actually needed, in particular because of the use of complex numbers as a base field. Even for probabilistic languages one obviously needs only real numbers as weights to indicate transition probabilities; thus the infinitely many non-selfadjoint operators in a complex C*-algebra do not correspond in any way to probabilistic programs. Furthermore, in effect only weights in the interval $[0,1]$ are of relevance

and we have to consider only stochastic operators.

However, our aim in this paper has been the identification of a "working" infinite dimensional generalisation of finite dimensional matrix algebras in order to provide a mathematical framework for quantitative reasoning about general, i.e. recursive, programs. Arguably, complex C*-algebras constitute the most extensively studied class of such operator algebras. Furthermore, their theory is commonly considered to be much simper than that of their real counterparts; this is essentially due to the "algebraic closedness" of $\mathbb{C}$ established by the fundamental theorem of algebra: every complex polynomial of degree n has exactly n (not necessarily different) complex roots.

It would of course be possible to look for minimal (C*-algebraic) structures, for example, by considering so-called *real* C*-algebras, or by allowing just rational numbers as the base field, as suggested in [37], where the authors investigated such a kind of "commutative C*-algebras". Nevertheless, all these structures are naturally embedded in general complex C*-algebras. Complex C*-algebras are even general enough to accommodate quantum computational models, i.e. complex "probability amplitudes". We leave it to future work to identify smaller and more specialised C*-models needed.

The linear operator approach outlined here is intended as a quantitative alternative to order or domain theoretic approaches commonly considered in semantics and program analysis. Our aim was to establish a relatively simple mathematical model for computational processes in probabilistic languages. As our focus was on the *operational* and not on the *denotational* semantics of probabilistic programs, we did not consider C*-algebras as probabilistic domains in the sense of, for example, [31] and [32]. However, it might be worth noting that strongly closed C*-algebras, i.e. von Neumann or W*-algebras, also carry an interesting order theoretic structure (cf. [4, Thm 2.4.21]), which we aim to investigate as a future work. We are confident that this study will clarify the relation between classical domain theory and our operator algebraic setting. In particular, it provides a base for a comparison with recent approaches applying domain theoretic concepts to Labelled Markov Processes, e.g. [12,11].

We also aim to investigate in future how to formulate a *compositional* semantics in an AF algebraic setting. Furthermore, we would like to shed more light on the problem of how a *Hilbert space* setting, i.e. $\ell^2(S)$, is related to a *Banach space setting*, i.e. $\ell^1(S)$ — which a priori might seem to provide a more appropriate structure for investigating probabilistic languages [35,19]. An important feature of the $\ell^2(S)$ framework is the fact that it is self-dual (reflexive), which means that we can investigate observables and computational states as elements of the same space; in the ℓ^1 approach observables are naturally located in the dual space ℓ^∞. One would expect that the differences between the two approaches will become important only when it comes to infinite observables given that, contrary to the infinite case, we have a unique topology for finite dimensional vector spaces.

References

[1] Biggs, N., "Algebraic Graph Theory," Cambridge University Press, 1993.

[2] Böttcher, A. and B. Silbermann, "Introduction to Large Truncated Toeplitz Matrices," Springer Verlag, 1999.

[3] Bratteli, O., *Inductive limits of finite dimensional C*-algebras*, Transactions of the AMS **171** (1972), pp. 195–234.

[4] Bratteli, O. and D. Robinson, "Operator Algebras and Quantum Statistical Mechanics," Springer Verlag, 1979.

[5] Brogi, A., A. Di Pierro and H. Wiklicky, *Linear embedding for a quantitative comparison of language expressiveness*, in: *Proceedings of QAPL'01*, ENTCS **59:3** (2002).

[6] Conway, J., "A Course in Functional Analysis," Springer Verlag, 1990.

[7] Cousot, P. and R. Cousot, *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints*, in: *Proceedings of POPL'77*, 1977, pp. 238–252.

[8] Cuntz, J. and W. Krieger, *A class of C*-algebras and topological Markov chains*, Inventiones Mathematicae **56** (1980), pp. 251–268.

[9] Davidson, K., "C*-Algebras by Example," AMS, 1996.

[10] de Boer, F., A. Di Pierro and C. Palamidessi, *Nondeterminism and Infinite Computations in Constraint Programming*, Theoretical Computer Science **151** (1995), pp. 37–78.

[11] Desharnais, J., A. Edalat and P. Panangaden, *Bisimulation for labelled markov processes*, Information and Computation **179** (2002), pp. 163–193.

[12] Desharnais, J., R. Jagadeesan, V. Gupta and P.Panangaden, *The metric analogue of weak bisimulation for probabilistic processes*, in: *Proceedings of LICS'02*, 2002, pp. 413–422.

[13] Di Pierro, A., C. Hankin and H. Wiklicky, *Measuring the confinement of probabilistic systems*, Theoretical Computer Science To appear.

[14] Di Pierro, A., C. Hankin and H. Wiklicky, *Quantitative static analysis of distributed systems*, Journal of Functional Programming To appear.

[15] Di Pierro, A., C. Hankin and H. Wiklicky, *Approximate confinement under uniform attacks*, in: *Proceedings of SAS'02*, LNCS **2477** (2002), pp. 310–325.

[16] Di Pierro, A., C. Hankin and H. Wiklicky, *Quantitative relations and approximate process equivalences*, in: *Proceedings of CONCUR'03*, LNCS **2761** (2003), pp. 508–522.

[17] Di Pierro, A., C. Hankin and H. Wiklicky, *Approximate Non-Interference*, Journal of Computer Security **12** (2004), pp. 37–81.

[18] Di Pierro, A. and H. Wiklicky, *An operational semantics for Probabilistic Concurrent Constraint Programming*, in: *ICCL'98* (1998), pp. 174–183.

[19] Di Pierro, A. and H. Wiklicky, *Probabilistic Concurrent Constraint Programming: Towards a fully abstract model*, in: *Proceedings of MFCS'98*, LNCS **1450** (1998), pp. 446–455.

[20] Di Pierro, A. and H. Wiklicky, *Concurrent Constraint Programming: Towards Probabilistic Abstract Interpretation*, in: *Proceedings of PPDP'00*, 2000, pp. 127–138.

[21] Di Pierro, A. and H. Wiklicky, *Quantitative observables and averages in Probabilistic Concurrent Constraint Programming*, in: *New Trends in Constraints*, LNCS **1865** (2000), pp. 212–236.

[22] Di Pierro, A. and H. Wiklicky, *Measuring the precision of abstract interpretations*, in: *Proceedings of LOPSTR'00*, LNCS **2042** (2001), pp. 147–164.

[23] Doberkat, E.-E., *The converse of a stochastic relation.*, J. Log. Algebr. Program. **62** (2005), pp. 133–154.

[24] Fillmore, P., "A User's Guide to Operator Algebras," John Wiley & Sons, 1996.

[25] Greub, W., "Linear Algebra," Springer Verlag, 1967, third edition.

[26] Grinstead, C. and J. Snell, "Introduction to Probability," AMS, 1997.

[27] Gupta, V., R. Jagadeesan and P. Panangaden, *Stochastic processes as concurrent constraint programs*, in: *Proceedings of POPL'99*, 1999, pp. 189–202.

[28] Gupta, V., R. Jagadeesan and V. Saraswat, *Probabilistic concurrent constraint programming*, in: *Proceedings of CONCUR '97*, LNCS **1243** (1997), pp. 243–257.

[29] Halmos, P., "A Hilbert Space Problem Book," Springer Verlag, 1982.

[30] Henkin, L., J. Monk and A. Tarski, "Cylindric Algebras (Part I)," North-Holland, 1971.

[31] Jones, C. and G. Plotkin, *A probabilistic powerdomain of evaluations*, in: *Proceedings of LICS'89*, 1989, pp. 186–195.

[32] Jung, A. and R. Tix, *The troublesome probabilistic powerdomain*, in: *Third Workshop on Computation and Approximation*, ENTCS **13** (1998), p. 23.

[33] Kadison, R. and J. Ringrose, "Fundamentals of the Theory of Operator Algebras: Volume I – Elementary Theory," AMS, 1997.

[34] Kemeny, J. G. and J. L. Snell, "Finite Markov Chains," D. Van Nostrand, 1960.

[35] Kozen, D., *Semantics for probabilistic programs*, Journal of Computer and System Sciences **22** (1981), pp. 328–350.

[36] Lind, D. and B. Marcus, "An Introduction to Symbolic Dynamics and Coding," Cambridge University Press, 1995.

[37] Mislove, M., J. Ouaknine, D. Pavlovic and J. B. Worrell, *Duality for labelled Markov processes*, in: *Proceedings of FOSSACS'04*, LNCS **2987** (2004), pp. 393–407.

[38] Murphy, G., "C^*-Algebras and Operator Theory," Academic Press, 1990.

[39] Plotkin, G., *A structured approach to operational semantics*, Technical Report DAIMI FN-19, Computer Science Department, Aarhus University (1981).

[40] Rørdam, M., F. Larsen and N. Lausten, "An Introduction to K-Theory for C*-algebras," LMS Student Texts **49**, Cambridge University Press, 2000.

[41] Saraswat, V., M. Rinard and P. Panangaden, *Semantics foundations of concurrent constraint programming*, in: *Proceedings of POPL'91*, 1991, pp. 333–353.

[42] Saraswat, V. A. and M. Rinard, *Concurrent constraint programming*, in: *Proceedings of POPL* (1990), pp. 232–245.

[43] Seneta, E., "Non-negative Matrices and Markov Chains," Springer Verlag, 1981.

[44] Wegge-Olsen, N., "K-Theory and C*-Algebras," Oxford University Press, 1993.

[45] Yosida, K., "Functional Analysis," Springer Verlag, 1980.