Probabilistic Concurrent Constraint Programming: Towards a Fully Abstract Model

Alessandra Di Pierro and Herbert Wiklicky

{adp,herbert}@cs.city.ac.uk
City University, Northampton Square, London EC1V OHB

Abstract. This paper presents a Banach space based approach towards a denotational semantics of a probabilistic constraint programming language. This language is based on the concurrent constraint programming paradigm, where randomness is introduced by means of a probabilistic choice construct. As a result, we obtain a declarative framework, in which randomised algorithms can be expressed and formalised. The denotational model we present is constructed by using functional-analytical techniques. As an example, the existence of fixed-points is guaranteed by the Brouwer-Schauder Fixed-Point Theorem. A concrete fixed-point construction is also presented which corresponds to a notion of observables capturing the exact results of both finite and infinite computations.

1 Introduction

Probabilistic Concurrent Constraint Programming (PCCP) was introduced in [4] in order to allow the formulation of randomised algorithms within the declarative framework of Concurrent Constraint Programming (CCP) [13]. The main feature of this language is a construct for probabilistic choice expressing a kind of nondeterminism which allows a program to make stochastic moves during its execution. An operational semantics describing such a behaviour was also given in [4]. The ultimate aim of this work is to provide PCCP with a denotational semantics which is fully abstract with respect to the notion of observables introduced in [4], and corresponding to the exact results of both finite and infinite computations.

One major problem that makes this task difficult is the presence in the language of nondeterminism (though in its *probabilistic*, thus more refined, version) in combination with synchronisation. In fact, any model reflecting these two aspects cannot be 'too abstract'; information about the branching structure and the synchronisation cannot be ignored, for which relatively complex structures are usually required like, for instance, the *reactive sequences* [3] or the *bounded traces operators* [13] adopted for CCP.

Another problem, somehow orthogonal to the first one, arises from the combination of (probabilistic) nondeterminism and infinite computations, in that it is difficult to find the appropriate structure of the domain where limits can be characterised by a fixed-point operator.

As a first step, we concentrate in this paper on the second problem and we abstract from the problem of synchronisation. We therefore define a denotational semantics capturing both probabilistic nondeterminism and infinite limit results for a sub-language of PCCP which has no suspension mechanism (all the guards are *true*). This language corresponds to Constraint Logic Programming where the or-nondeterminism is replaced by a probabilistic choice among the input clauses. Thus we call it Probabilistic Constraint Logic Programming (PCLP).

The domains we will consider for the semantics of PCLP are based on linear structures, that is on vector spaces and their structure preserving morphisms: linear mappings and operators. Vector spaces provide a common and most widely used model for various sciences, ranging from physics to economics, but they are much less popular in computer science. In the context of our investigations they come into considerations because they combine quantitative and qualitative concepts. This is useful in PCCP (PCLP), as a computation in this paradigm incorporates some quantitative information, besides the usual qualitative one, in the form of probabilities associated to the choice. Furthermore, vector spaces are very well studied mathematical structures, which makes it possible to utilise a great number of well established results.

We argue that the introduction of quantitative aspects in the semantics of CCP plays a fundamental role in modelling the program behaviour. Thanks to the ability to measure the "strength" of a constraint (quality) by means of the probability (quantity) assigned to it, our denotational model succeeds in capturing some observable behaviours that the more classical powerdomain or metric based approaches fail to capture. More specifically, while the Smyth powerdomain and the metric approaches to the semantics of constraint programming have been shown unable to model the *exact* (infinite) results of a computation [1], the probabilistic model we define in this paper perfectly matches this behaviour. Moreover, it can be used also for the standard (non-probabilistic) version of Constraint Logic Programming.

2 Probabilistic Constraint Logic Programming

In [4] we introduce the language PCCP, which is essentially CCP where the nondeterministic choice is replaced by a probabilistic one[1]. This allows us to see the execution of a program as a random walk on the transition graph. Probabilistic Constraint Logic Programming is the sub-language of PCCP obtained by replacing all the guards in the probabilistic choice construct by *true*. This eliminates the aspect of synchronisation from the language, thus allowing us to abstract (for the time being) from this problem.

The syntax of PCLP is given in Table 1. Successful termination is expressed by the agent **stop**; the agent **tell**, the hiding operator $\exists_x$ and the procedure call $p(x)$ are the usual ones (of CCP). The operator $\|$ expresses the parallel

[1] Another approach to incorporate probabilistic aspects into CCP languages was introduced later in [7]. It is based on the use of random variables and is substantially different in both the aim and the method from our approach.

$$P ::= D.A$$

$$D ::= \epsilon \mid D.D \mid p(x) : -A$$

$$A ::= \textbf{stop} \mid \textbf{tell}(c) \mid \square_{i=1}^{n}\ true \mid p_i \rightarrow A_i \mid A \parallel A \mid \exists_x A \mid p(x)$$

Table 1. The syntax for PCLP.

composition of two agents. Additionally we provide a "probabilistic" choice $\square$. Operationally this construct expresses the choice of one of the agents A_i according to the assigned probabilities p_i. The intended meaning of this is the usual interpretation of probability in probability theory: if the choice is repeated (under the same condition and sufficiently often) the relative frequency of executions of an agent A_i is exactly p_i. For the definition of the (cylindric) constraint system underlying the language we refer to [13].

3 Operational Semantics for PCLP

For the operational semantics of PCLP we essentially use the probabilistic transition system introduced in [4]. Randomness is expressed by labels representing the probability that a transition takes place.

A *configuration* represents the state of the system at a certain moment, namely the agent A which has still to be executed, and the current store d. We denote a configuration by $<A, d>$. The probabilistic transition system for PCLP consists of a pair $(Conf, \longrightarrow_p)$, where $Conf$ is a set of configurations and $\longrightarrow_p \subseteq Conf \times \mathbb{R} \times Conf$ is the transition relation defined in Table 2. We denote the transitive closure of transition relation $\longrightarrow_p$ by $\longrightarrow_{p'}^*$, where p' is the product of the probabilities associated to each single step.

3.1 The Observables

The notion of *observables* we consider captures the *exact* results of both finite and infinite computations together with their associated probabilities.

Given a program P, we define the result $\mathcal{R}_P$ of an agent A and an initial store d as the (multi-)set of all pairs $<c, p>$, where c is the least upper bound of the partial constraints accumulated during a computation starting from d; and p is the probability of reaching that result.

$$\begin{aligned} \mathcal{R}_P(A, d) = &\{<c, p> \mid <A, d> \longrightarrow_p^* <B, c> \not\longrightarrow\} \cup \\ &\{<\textstyle\bigsqcup_i d_i, \prod_i p_i> \mid <A, d_0> \longrightarrow_{p_0} \ldots\}. \end{aligned}$$

The first term describes the results of finite computations, where the least upper bound of the partial store corresponds to the final store. The second term

$$\textbf{R1}\ <\mathbf{tell}(c), d> \rightarrow_1 <\mathbf{stop}, c \sqcup d>$$

$$\textbf{R2}\ <\Box_{i=1}^{n}\ \ \mathit{true} \mid p_i \rightarrow A_i, d> \rightarrow_{p_j} <A_j, d> \qquad j \in [1, n]$$

$$\textbf{R3}\ \frac{<A, c> \rightarrow_p <A', c'>}{\begin{array}{c}<A \parallel B, c> \rightarrow_p <A' \parallel B, c'> \\ <B \parallel A, c> \rightarrow_p <B \parallel A', c'>\end{array}}$$

$$\textbf{R4}\ \frac{<A, d \sqcup \exists_x c> \rightarrow_p <B, d'>}{<\exists_x^d A, c> \rightarrow_p <\exists_x^{d'} B, c \sqcup \exists_x d'>}$$

$$\textbf{R5}\ <p(y), c> \rightarrow_1 <\exists_\alpha (\delta_{y\alpha} \wedge \exists_x (\delta_{\alpha x} \wedge A)), c> \qquad p(x) : -A \in P$$

Table 2. The transition system for PCLP.

covers the infinite results. The probability of obtaining a certain result depends on the probabilities p associated to the possible paths which lead to it. To capture the true behaviour of an agent we have to identify different computational paths leading to the same result as well as to collect the accumulated probabilities associated with different interleavings. In order to do this in a precise way we define the following operation. By c_ij we denote the jth occurrence of the constraint c_i in the multi-set of all results.

$$\mathcal{K}(\{<c_{ij}, p_{ij}>\}_{i,j}) = \{<c_i, P_{c_i}> \mid\ P_{c_i} = \textstyle\sum_j p_{ij}\}_i.$$

Another operation normalises the probabilities. This is necessary as the probabilities in each interleaving add up to one such that the overall sum of probabilities is exactly the number of possible interleavings. This process of renormalisation effectively implies that all interleavings are equally likely.

$$\mathcal{N}(\{<c_i, p_i>\}_i) = \{<c_i, \tfrac{p_i}{P}> \mid\ P = \textstyle\sum_i p_i\}.$$

With these two operations we can define the observables associated to an agent A and an initial store d as:

$$\mathcal{O}_P(A, d) = \mathcal{N}(\mathcal{K}(\mathcal{R}_P(A, d))).$$

Note that this notion of observables differs from the classical notion of input/output behaviour in CCP. In the classical case a constraint c belongs to the input/output observables of a given agent A if *at least* one path leads from the initial store d to the final result c. In the probabilistic case we have to consider *all* possible paths leading to the same result c and combine the associated probabilities.

4 A Denotational Semantics for PCLP

To simplify our presentation we will assume in the following that the set of agents $\mathcal{A} = \{A_1, \ldots, A_{|\mathcal{A}|}\}$ is finite, and that the set of constraints $\mathcal{C} = \{c_i\}_{i=0}^{\infty}$ is countable. In the case of uncountable constraint systems we can generalise our approach, replacing sums by integrals, l^1 by L^1, etc. Then most of the results presented here can be transferred into an appropriate measure-theoretic setting.

We assign to each agent a (probability) distribution on the set of constraints.

Definition 1. *A distribution ρ on the set of constraints $\mathcal{C}$ is a map from $\mathcal{C}$ into the real interval $[0,1]$ satisfying the normalisation condition: $\sum_{c\in\mathcal{C}} \rho(c) = 1$. The set of distributions on $\mathcal{C}$ is denoted by $\mathcal{D}(\mathcal{C})$ or simply $\mathcal{D}$.*

We define an *interpretation* $I : \mathcal{A} \mapsto \mathcal{D}$ as a function from the set of agents $\mathcal{A}$ into the set of distributions $\mathcal{D}(\mathcal{C})$ on $\mathcal{C}$. The set of all possible interpretations is denoted by $\mathcal{I}$. For an agent $A \in \mathcal{A}$ we represent its interpretation by $I(A) = \{<c_i, p_i>\}_i$, where $c_i \in \mathcal{C}$ and $p_i = I(A)(c_i)$. We will omit those pairs where the probability vanishes.

The set of possible interpretations of an agent A, $\mathcal{I}(A)$, forms a subset of the (real) *Banach space* $l^1(\mathcal{C})$. The elements of the Banach space $l^1(\mathcal{C})$ are given by sequences of real numbers indexed by the elements of a (countable) constraint system such that the sum of their absolute values exists:

$$l^1(\mathcal{C}) = \{<x_i, c_i> \mid x_i \in \mathbb{R}, c_i \in \mathcal{C} \text{ and } \sum_{c_i\in\mathcal{C}} |x_i| < \infty\}.$$

On the space of sequences $l^1(\mathcal{C})$ we define a scalar product and vector addition *pointwise* and the norm as the usual l^1-norm (with $q, p \in \mathbb{R}$) by:

$$\begin{aligned} q \cdot \{<c_i, p_i>\}_i &= \{<c_i, qp_i>\}_i\,, \\ \{<c_i, p_i>\}_i + \{<c_i, q_i>\}_i &= \{<c_i, p_i + q_i>\}_i\,, \\ \| \{<c_i, p_i>\}_i \| &= \sum_{c_i\in\mathcal{C}} |p_i|. \end{aligned}$$

In order to model all constructs of our language we define two additional operations on this space: a tensor product $\otimes$ and a *pointwise* hiding operator:

$$\begin{aligned} \{<c_i, p_i>\}_i \otimes \{<c_j, q_j>\}_j &= \{<c_i \sqcup d_j, p_i q_j>\}_{i,j}\,, \\ \exists_x \{<c_i, p_i>\}_i &= \{<\exists_x c_i, p_i>\}_i\,. \end{aligned}$$

We can embed the set of all possible interpretations, $\mathcal{I}$, in a similarly defined Banach space $l^1(\mathcal{C})^{|\mathcal{A}|}$, i.e. the (finite) cartesian product of $|\mathcal{A}|$ copies of $l^1(\mathcal{C})$.

Proposition 1. *The space of interpretations $\mathcal{I}$ forms a convex, closed (non-empty) subset of the Banach space $l^1(\mathcal{C})^{|\mathcal{A}|}$.*

$\Phi(I)(\mathbf{stop})$	$= \{<true, 1>\}$
$\Phi(I)(\mathbf{tell}(c))$	$= \{<c, 1>\}$
$\Phi(I)(\square_{i=1}^{n}\ true \mid p_i \rightarrow A_i)$	$= \sum_{i=1}^{n}\ p_i \cdot \Phi(I)(A_i)$
$\Phi(I)(A_1 \parallel A_2)$	$= \Phi(I)(A_1) \otimes \Phi(I)(A_2)$
$\Phi(I)(\exists_x A)$	$= \exists_x \Phi(I)(A)$
$\Phi(I)(p(x))$	$= I(\Delta_y^x A)$

Table 3. The compositional definition of $\Phi : \mathcal{I} \rightarrow \mathcal{I}$.

On the set of interpretations $\mathcal{I}$ we define inductively the fixed-point operator Φ as in Table 3 (where $\Delta_y^x A$ is a shorthand notation for $\exists_\alpha(\delta_{y\alpha} \wedge \exists_x(\delta_{\alpha x} \wedge A))$ as in **R5** in Table 2). Some useful properties of this operator are stated in the following proposition.

Proposition 2. *The operator Φ is well-defined on $\mathcal{I} \subset l^1(\mathcal{C})^{|\mathcal{A}|}$ and has the following properties: (i) Φ is continuous, and (ii) Φ is compact, i.e. the closure of $\Phi(\mathcal{I})$ is compact.*

Proof. (Idea) Ad (i): Φ is linear and bound and therefore continuous, ad (ii): Φ is the limit of finite-dimensional operators as PCLP is finitely branching. □

To guarantee the existence of a fixed-point of Φ we use a classical theorem from functional analysis [5, Theorem 18.10'].

Theorem 1. (Brouwer-Schauder Theorem) *Let $F : K \mapsto K$ be a continuous mapping from a non-empty closed, convex set K in a Banach space into itself, with the closure of $F(K)$ compact. Then there exists a fixed-point of F, i.e. a point $c \in K$ such that $F(c) = c$.*

By Theorem 1 and Propositions 1 and 2, we can guarantee:

Theorem 2. *The operator Φ has a fixed-point.*

4.1 Construction of a Fixed-Point

In order to concretely construct a fixed-point of Φ we will mimic the classical fixed-point construction: Starting with the initial interpretation I_0 assigning to each agent A the distribution $I_0(A) = \{<true, 1>\}$, we iteratively apply Φ in order to construct a (pointwise) limit of the sequence of interpretations $\{I_n\}_n = I_0, \Phi(I_0), \Phi^2(I_0), \ldots, \Phi^n(I_0), \ldots$. This limit will be a fixed-point of Φ because of continuity. To show convergence we need some auxiliary constructions.

We introduce the notion of *volume* of a constraint with respect to a distribution. This is roughly the probability concentrated in the upward closure $\uparrow c$ of a constraint $c \in \mathcal{C}$ (with respect to $\sqsubseteq$ in the constraint system) and will be essential in the construction of the limit interpretation defining the meaning of our programs.

Definition 2. *Given a distribution $\rho \in \mathcal{D}$, we define the volume of a constraint c with respect to ρ as*

$$\mathrm{v}ol_\rho(c) = \sum_{d \in \uparrow c} \rho(d).$$

There is a one-to-one correspondence between the original distribution ρ and the distribution of volumes $\mathrm{v}ol_\rho$. Using a general inclusion-exclusion principle, e.g. [6, Eqn. 3.3], we can show the following lemma.

Lemma 1. *Given the volume $\mathrm{v}ol_\rho(c)$ of each constraint $c \in \mathcal{C}$ with respect to a distribution $\rho \in \mathcal{D}$ it is possible to reconstruct the distribution ρ uniquely, by*

$$\rho(c) = \mathrm{v}ol(c) - \sum_{d>c} \mathrm{v}ol(d) + \sum_{d>e>c} \mathrm{v}ol(d \sqcup e) - \sum_{f>d>e>c} \mathrm{v}ol(d \sqcup e \sqcup f) + \ldots$$

The sequence $\{I_n\}_n$ in general is not pointwise monotone (e.g. example 2 below), therefore it is not obvious how to prove its convergence directly. However, it is easy to see that the corresponding sequence of volume distributions does converge.

Lemma 2. *Let $A \in \mathcal{A}$, $c \in \mathcal{C}$ and $\{I_n\}_n$ the sequence defined above. Then the sequence $\left\{\mathrm{v}ol_{I_n(A)}(c)\right\}_n$ converges.*

Proof. (Sketch) For each constraint $c \in \mathcal{C}$ the following holds $\forall n \in \mathbb{N}$:

- $\mathrm{v}ol_{I_n(A)}(c) \leq 1$, i.e. the volume of each constraint is bound by one in each interpretation, because Φ is “normalised”, i.e. maps $\mathcal{I}(A)$ into $\mathcal{I}(A)$.
- $\mathrm{v}ol_{I_n(A)}(c) \leq \mathrm{v}ol_{I_{n+1}(A)}(c)$, i.e. the sequence of volumes is monotone (increasing).

Therefore, the limit $\lim_{n\to\infty} \mathrm{v}ol_{I_n(A)}(c)$ of a monotone and bound sequence of real numbers $\mathrm{v}ol_{I_n(A)}(c)$ exists. $\square$

By Lemma 1 and continuity of Φ we can reconstruct the pointwise limit of distributions from the pointwise limit of volumes of the constraints.

Theorem 3. *The sequence $\{<c, I_n(A)(c)>\}_n$ converges pointwise to a fixed-point of Φ.*

We are now in a position to define a semantics for PCLP.

Definition 3. *For each agent $A \in \mathcal{A}$ we define its semantics $Q(A)$ as the pointwise limit of $\{I_n(A)\}_n$,*

$$Q(A) = \lim_{n\to\infty} I_n(A) = \lim_{n\to\infty} \Phi^n(I_0(A)).$$

For this semantics we can establish the correspondence with the observables defined in Section 3 by structural induction.

Theorem 4. *For all agents $A \in \mathcal{A}$ the fixed-point semantics $Q(A)$ coincides with the observables*

$$\mathcal{O}_P(A, true) = Q(A).$$

We would like to point out that alternative fixed-point constructions can be defined which model different notions of observables.

4.2 Examples

Example 1. Consider the following PCLP program for computing the natural numbers:

$$\begin{array}{l} nat(x) :- \ true|\tfrac{1}{2} \rightarrow \mathbf{tell}(x = 0) \\ \qquad\qquad\qquad \Box \ true|\tfrac{1}{2} \rightarrow \exists_y(\mathbf{tell}(x = s(y)) \parallel nat(y)) \end{array}$$

The sequence of interpretations $I_n(nat(x))$ converges pointwise to

$$Q(nat(x)) = \left\{<x = 0, 1/2>, <x = s(0), 1/4>, \ldots, <x = s^n(0), 1/2^{n+1}>, \ldots\right\}.$$

This clearly coincides with the observables $\mathcal{O}_P(nat(x), true)$. Note that, contrary to the classical approach, these observables not only tell us that all numbers may be computed but also that the probability of computing larger numbers decreases.

Example 2. The following declarations have been used in [1] (in their CCP formulation) as an example of the inapplicability of metric and order-theoretic approaches to modelling the exact results observables in constraint programming.

$$\begin{array}{l} p(x) :- \ q(x) \\ q(x) :- \ true|\tfrac{1}{2} \rightarrow p(x) \\ \qquad\qquad \Box \ true|\tfrac{1}{2} \rightarrow (r(x) \parallel \mathbf{tell}(c)) \\ r(x) :- \ \mathbf{tell}(\mathit{false}). \end{array}$$

In [1] it was shown that the interpretations defined by using an analogous of the Φ operator do not converge with respect to any metric or order. In our quantitative semantics we get convergence as the limit $\lim_n I_n$ exists for all three agents,

$$Q(p(x)) = Q(q(x)) = Q(r(x)) = \{<\mathit{false}, 1>\}.$$

5 Future Work

We plan to extend the denotational semantics developed here for PCLP to the full language PCCP. To this purpose it will be necessary to add an appropriate encoding of the branching structure and the synchronisation for dealing with global choice. It seems that we can still use an underlying Banach space structure; however it will be necessary to replace vectors (distributions) by matrices (operators) in order to keep track of the computational traces. We expect this model to be the 'quantitative' counterpart of the various (equivalent) fully abstract models developed until now for CCP. The only two attempts to describe the exact results of infinite computations in CCP we are aware of are [11] and [2], whereas the two approaches [3, 13] we already mentioned in the introduction have been shown to be fully abstract only with respect to the results of finite computations.

Additional investigations will compare our construction to other approaches towards the semantics of probabilistic programming languages [12, 9, 8], probabilistic predicate transformers [10] and logics and stochastic processes.

References

1. F. S. de Boer, A. Di Pierro, and C. Palamidessi. Nondeterminism and Infinite Computations in Constraint Programming. *Theoretical Computer Science*, 151(1), 1995. Selected Papers of the Workshop on Topology and Completion in Semantics, Chartres, France.
2. F. S. de Boer and M. Gabbrielli. Infinite Computations in Concurrent Constraint Programming. *Electronic Notes in Theoretical Computer Science*, 6:16, 1997.
3. F.S. de Boer and C. Palamidessi. A Fully Abstract Model for Concurrent Constraint Programming. In S. Abramsky and T.S.E. Maibaum, editors, *TAPSOFT/CAAP*, volume 493, pages 293–319. Springer Verlag, 1991.
4. A. Di Pierro and H. Wiklicky. An operational semantics for Probabilistic Concurrent Constraint Programming. In P. Iyer, Y. Choo, and D. Schmidt, editors, *ICCL'98 – International Conference on Computer Languages*, pages 174–183. IEEE Computer Society and ACM SIGPLAN, IEEE Computer Society Press, May 1998.
5. K. Goebel and W.A. Kirk. *Topics in Metric Fixed Point Theory*, volume 28 of *Cambridge studies in advanced mathematics*. Cambridge University Press, Cambridge, 1990.
6. C. M. Grinstead and J. L. Snell. *Introduction to Probability*. American Mathematical Society, Providence, Rhode Island, second revised edition, 1997.
7. V. Gupta, R. Jagadeesan, and V. A. Saraswat. Probabilistic concurrent constraint programming. In *Proceedings of CONCUR 97*. Springer Verlag, 1997.
8. Claire Jones. *Probabilistic Non-Determinism*. PhD thesis, University of Edinburgh, Edingburgh, 1993.
9. Dexter Kozen. Semantics for probabilistic programs. *Journal of Computer and System Sciences*, 22:328–350, 1981.
10. C. Morgan, A. McIver, K. Seidel, and J.W. Sanders. Probabilistic predicate transformers. Technical Report PRG-TR-4-95, Programming Research Group, Oxford University Computing Laboratory, 1995.
11. S. O. Nyström and B. Jonsson. Indeterminate Concurrent Constraint Programming: A Fixpoint Semantics for Non-Terminating Computations. In D. Miller, editor, *Proc. of the 1993 International Logic Programming Symposium*, Series on Logic Programming, pages 335–352. The MIT Press, 1993.
12. N. Saheb-Djahromi. CPO's of measures for nondeterminism. *Theoretical Computer Science*, 12:19–37, 1980.
13. V. A. Saraswat, M. Rinard, and P. Panangaden. Semantics foundations of concurrent constraint programming. In *Proceedings of POPL*, pages 333–353. ACM, 1991.