A Probabilistic and Distributed Concurrent Constraint Programming

Luca Bortolussi\(^1\)  Herbert Wiklicky\(^2\)

\(^1\)Department of Mathematics and Computer Science
University of Udine, Italy.

\(^2\)Department of Computing
Imperial College, London, Uk.

ICLP 2005, Sitges, 2\(^{nd}\) October 2005
We wanted to model distributed optimization metaheuristics. We needed:

- **constraints** → *constraint based language*;
- **concurrency** → *CCP*;
- **distribution** → *distributed CCP*;
- **probabilities** → *probabilistic and distributed CCP*;
Store as a valuation replaced by a constraint store, modeled by a constraint system (extension of cylindric algebra). Computations evolve monotonically on the constraint system.

### Syntax of CCP

\[
\begin{align*}
\text{Program} &= \text{Decl}.A \\
D &= \varepsilon | \text{Decl.Decl} | p(x) : -A \\
A &= 0 \\
\text{tell}(c).A \\
\text{ask}(c_1).A_1 + \text{ask}(c_2).A_2 \\
A_1 | A_2 \\
\exists_x A \\
p(x)
\end{align*}
\]

### SOS for CCP

\[
\begin{align*}
\langle \text{tell}(c).A, d \rangle &\rightarrow \langle A, d \sqcup c \rangle \\
\langle \text{ask}(c_1).A_1 + \text{ask}(c_2).A_2, d \rangle &\rightarrow \langle A_1, d \rangle & \text{if } d \models c_1 \\
\langle A, d \rangle &\rightarrow_p \langle A', d' \rangle \\
\langle A | B, d \rangle &\rightarrow \langle A' | B, d' \rangle \\
\langle A, d \sqcup \exists_x c \rangle &\rightarrow \langle B, d' \rangle \\
\langle \exists_x^d A, c \rangle &\rightarrow \langle \exists_x^d B, c \sqcup \exists_x^{d'} \rangle \\
\langle p(y), c \rangle &\rightarrow \langle \Delta_x^y A, c \rangle & \text{if } p(x) : -A \in \text{Decl}
\end{align*}
\]
We introduce a KLAIM-like architecture. The basic distributed entity is a node, which is part of a network.

\[ n ::= s ::^p \varrho P \]

- \( s \in S \) is a **physical locality** or **site**, i.e. the address of the node \( n \).
- \( p \) is the **probability of execution** of the node.
- Nodes are linked by **communication channels**, among which messages can be exchanged (synchronously). They are indicated by \( \alpha, \beta \in \mathcal{L} \).
- \( \varrho \) is the **environment** of the node, i.e. a function associating probability distributions to communication channels: \( \varrho : \mathcal{L} \rightarrow \mathcal{D}(S) \) (\( \mathcal{D}(S) \) are probability distributions over \( S \)).
Nodes are connected by communication channels, and the topology can be changed dynamically. A network $N$ is defined as a parallel composition of a finite number of nodes:

$$N ::= \parallel_{i=1}^{m} n_i.$$
Each node of a network $N$ has its own constraint store. Therefore the local configuration of a node $n_i$ is a point in the space $\mathcal{P}_i \times \mathcal{C}_i$, i.e. a pair $\langle A_i, c_i \rangle$.

A configuration of the network is the cartesian product of local configurations, i.e. a point in the space $\prod_{i=1}^{m} \mathcal{P}_i \times \mathcal{C}_i$.

The product of the local constraint stores, $\mathcal{C} = \prod_{i=1}^{m} \mathcal{C}_i$ is referred as global constraint store, and it is still a cylindric algebra (as long as variables are independent among the different local constraint stores).
Operational semantics is given by a congruence relation and two labeled transition systems, one local and one global. Both are labeled with the probability associated to the particular transition.

- the local transition relation is indicated by $\rightarrow$ and it is a subset of $P \times C \times [0, 1] \times P \times C$

- the global transition relation is represented by $\Rightarrow$, and it is a subset of $P_1 \times \ldots \times P_m \times C_1 \times \ldots \times C_m \times [0, 1] \times P_1 \times \ldots \times P_m \times C_1 \times \ldots \times C_m$
Local Primitives

Local Transitions

In each node computations evolve locally according to probabilistic CCP rules.

Local Syntax

Program = Decl.A

D = ε | Decl.Decl | p(x) : ¬A

A = 0 | tell(c).A | ∃xA
   | p(x) | ∑₁≤i≤k qᵢ : ask(cᵢ).Aᵢ
   | ∏₁≤i≤k qᵢ : Aᵢ
Local Primitives

Local Transitions

In each node computations evolve locally according to probabilistic CCP rules.

Local Transition System

1. \((LR1)\) \(\langle \text{tell}(c).A, d \rangle \rightarrow_1 \langle A, d : c \rangle\)
2. \((LR2)\) \(\langle \sum_{i=1}^k q_i : \text{ask}(c_i).A_i, d \rangle \rightarrow_{\bar{q}_i} \langle A_i, d \rangle\) if \(d \vdash c_j\)
3. \((LR3)\) \(\langle A, d \rangle \rightarrow_p \langle A', d' \rangle\)
   \[\langle q_1 : A \mid q_{i=2}^k q_i : B_i, d \rangle \rightarrow_{p \bar{q}_1} \langle q_1 : A' \mid q_{i=2}^k q_i : B_i, d' \rangle\]
4. \((LR4)\) \(\langle A, d : \exists_x c \rangle \rightarrow_p \langle B, d' \rangle\)
   \[\langle \exists_x^d A, c \rangle \rightarrow_p \langle \exists_x^{d'} B, c : \exists_x^{d'} \rangle\]
5. \((LR5)\) \(\langle p(\overline{x}), c \rangle \rightarrow_1 \langle \overline{A} x A, c \rangle\) if \(p(\overline{x}) : -A \in \text{Decl}\)
Local transitions are lifted to the global level by the rule:

\[
(\text{GR1}) \quad \langle A, d \rangle \longrightarrow_p \langle A', d' \rangle
\]

\[
\frac{\langle A, d \rangle \omega_i \parallel N \Rightarrow p \cdot \tilde{p}_i \langle A', d' \rangle \omega_i \parallel N}{s_i} \quad \frac{p_i}{p_i} \quad \frac{p_i}{p_i}
\]

- Local probability of execution at node \(i\) is multiplied by the (normalized) probability of choosing that node for global execution.

- We can think the the choice of the node to be executed globally is governed by a global scheduler, which selects a node according to its global probability (priority).
Communicating Constraints

**Constraint Abstractions**

- Constraints can be sent through communication channels. Communication is synchronous and involves two processes because the free variables of a constraint must be replaced by the receiving agent with some variables of its constraint store (*constraint stores are independent*).

- So we can communicate **constraint abstractions**, denoted by \( \lambda \overrightarrow{x} c \). They are templates where all variables different from \( \overrightarrow{x} \) are existentially quantified.

- We can the **projection** over \( \overrightarrow{x} \) of \( c \) as \( \Pi_{\overrightarrow{x}}(c) = \exists_{fV(c) \setminus \overrightarrow{x}} c \).
The instruction to communicate constraint abstractions are:

\[
A := \text{out}_c(\lambda \overline{x} \cdot c)@\alpha. A \\
A := \text{in}_c(\overline{y})@\alpha. A
\]

The transition rule is

\[
| \overline{x} | = | \overline{y} | \text{ and } \varrho_i(\alpha)(s_j) > 0 \\
(\text{GR2}) \quad \frac{\text{s}_j \langle q_i : \text{out}_c(\lambda \overline{x} \cdot c)@\alpha. A_i | A_i', d_i \rangle_{\varrho_i} \parallel \text{s}_j \langle q_j : \text{in}_c(\overline{y})@\alpha. A_j | A_j', d_j \rangle_{\varrho_j} \parallel N}{\Rightarrow \tilde{a}_i \tilde{a}_j \tilde{p}_i \cdot \tilde{e}_i(\alpha)(s_j) \text{ s}_j \langle q_i : A_i | A_i', d_i \rangle_{\varrho_i} \parallel \text{s}_j \langle q_j : A_j | A_j', d_j \parallel (\prod x \cdot c)[\overline{y} / \overline{x}] \rangle_{\varrho_j} \parallel N}
\]
Global Primitives

Communicating Constraints

A Simple Example

\[ P_1 = \text{out}_c(\lambda x(x = 1))@\alpha \]
\[ P_2 = 1 : (\text{in}_c(y)@\alpha) | 1 : (1 : \text{ask}(y = 1) . \text{tell}(z = 1)) \]
\[ N = s_1 ::^1 \rho_1 P_1 \parallel s_2 ::^1 \rho_2 P_2 \]

\[ \langle \text{out}_c(\lambda x(x = 1))@\alpha, \text{true} \rangle \parallel \]
\[ \langle 1 : (\text{in}_c(y)@\alpha) | 1 : (1 : \text{ask}(y = 1) . \text{tell}(z = 1)), \text{true} \rangle \]
\[ \Longrightarrow_1 \langle 0, \text{true} \rangle \parallel \langle 1 : (0) | 1 : (1 : \text{ask}(y = 1) . \text{tell}(z = 1)), (y = 1) \rangle \]
\[ \Longrightarrow_1^* \langle 0, \text{true} \rangle \parallel \langle 0, (y = 1) \sqcup (z = 1) \rangle . \]
Exchanging Channels

The channels can be managed with the following instructions, which work in a $\pi$-calculus fashion.

\[
A := \text{out}_{\text{loc}}(\beta)@\alpha
\]

\[
A := \text{in}_{\text{loc}}(\beta)@\alpha
\]

\[
A := \text{new}(\beta).A
\]
Similarly to constraints, also agent abstractions can be communicated among the network:

\[
\text{out}_A(\lambda \vec{x} A, \vec{x}_0) \circ_{\alpha} A \\
\text{in}_A(\vec{y}) \circ_{\alpha} A \\
(\vec{x}_0 \subseteq \vec{x} \subseteq \text{fv}(A))
\]
Migrating Agents

A Simple Example

\[ Q = \text{tell}(x = 1) \]
\[ P_1 = \text{out}_A(\lambda xQ, \emptyset)@\alpha \]
\[ P_2 = \text{in}_A(y)@\alpha \]
\[ N = s_1 ::^1_{\varepsilon_1} P_1 \parallel s_2 ::^1_{\varepsilon_2} P_2 \]

\[ \langle \text{out}_A(\lambda xQ, \emptyset)@\alpha, \text{true} \rangle \parallel \langle \text{in}_A(y)@\alpha, \text{true} \rangle \implies_1 \]
\[ \langle 0, \text{true} \rangle \parallel \langle \frac{1}{2} 0 | \frac{1}{2} \text{tell}(y = 1), \text{true} \rangle \implies_1 \langle 0, \text{true} \rangle \parallel \langle 0, (y = 1) \rangle. \]
A different form of communication can be obtained by linking together variables belonging to different constraint stores.

If variable $x$ at node $i$ is linked to variable $y$ in node $j$, then every time a constraint involving $x$ is posted, the same constraint (existentially quantified over all other variables) is posted automatically also on $y$.

Information about linked variables is stored in a set $\mathbb{L}$, containing elements of the form $(\vec{x}, \vec{y}, i, j)$. This is added to the system configuration.

$$A = \text{in}_{\text{link}}(\vec{x})@\alpha \mid \text{out}_{\text{link}}(\vec{y})@\alpha \mid \text{remove}_{\text{link}}(\vec{y})$$
Global Primitives

Linking Constraint Stores

An Example

\[
P = \text{out}_{\alpha}(x) . \text{tell}(x = 1)
\]

\[
Q = \text{in}_{\alpha}(y) . (1 : \text{ask}(y = 1) . \text{tell}(z = 1))
\]

\[P \parallel Q\]

\[
\begin{align*}
\langle \text{out}_{\alpha}(x) . \text{tell}(x = 1), \text{true} \rangle & \parallel \\
\langle \text{in}_{\alpha}(y) . (1 : \text{ask}(y = 1) . \text{tell}(z = 1)), \text{true}, \emptyset \rangle
\end{align*}
\]

\[
\implies_1 (\langle \text{tell}(x = 1), \text{true} \rangle \parallel \langle 1 : \text{ask}(y = 1) . \text{tell}(z = 1), \text{true} \rangle, \{(x, y, 1, 2)\})
\]

\[
\implies_1 (\langle 0, (x = 1) \rangle \parallel \langle 1 : \text{ask}(y = 1) . \text{tell}(z = 1), (y = 1) \rangle, \{(x, y, 1, 2)\})
\]

\[
\implies^*_1 (\langle 0, (x = 1) \rangle \parallel \langle 0, (y = 1) \sqcup (z = 1) \rangle, \{(x, y, 1, 2)\}).
\]
The probability of a trace is the product of the probabilities associated to each transition.

The probability $p$ of the star closure $\langle \overrightarrow{A}, \overrightarrow{c} \rangle \xrightarrow{\star}_p \langle \overrightarrow{B}, \overrightarrow{d} \rangle$, is computed by summing over all possible execution paths leading from $\langle \overrightarrow{A}, \overrightarrow{c} \rangle$ to $\langle \overrightarrow{B}, \overrightarrow{d} \rangle$.

The observable is a probability distribution over the constraint store, defined as

$$O_D((\overrightarrow{A}, \overrightarrow{c})) = \left\{ (\overrightarrow{d}, p) \mid (\overrightarrow{A}, \overrightarrow{c}) \xrightarrow{\star}_p (\overrightarrow{0}, \overrightarrow{d}) \right\}.$$
We designed a language with several communication primitives, with the aim of modeling concurrent optimization algorithms. The language is based on CCP, with probabilistic and distributed features.

We have also an implementation (a meta-interpreter in SICStus Prolog), for now working only on a subset of the language (no agent migration and no linking).

We want to design a continuous time version, and to improve the dynamic features of the network topology, by allowing a variable number of nodes and variable probability distributions.

We want to design methodologies to analyze features of programs written in the language.
THANKS FOR THE ATTENTION!

QUESTIONS?