Constraint Logic Programming

Sylvain Soliman, François Fages and Nicolas Beldiceanu
{Sylvain.Soliman,Francois.Fages}@inria.fr

INRIA – Projet CONTRAINTES

MPRI C-2-4-1 Course – September-November, 2006

INRIA

Part I: CLP - Introduction and Logical Background

Part II: Constraint Logic Programs

Part III: Operational and Fixpoint Semantics

Part IV: Logical Semantics

Part V: Concurrent Constraint Programming

Part VI: CC - Denotational Semantics

Semantic Equations

Let $[\![]\!] : \mathcal{D} \times A \rightarrow \mathcal{P}(\mathcal{C})$ be a closure operator presented by the set of its fixpoints, and defined as the least fixpoint set of the equations:

$$
\begin{array}{lll}
[\![\mathcal{D}.tell(c)]\!] & = \uparrow c & (\simeq \lambda s.s \wedge c) \\
[\![\mathcal{D}.c \rightarrow A]\!] & = (\mathcal{C} \setminus \uparrow c) \cup (\uparrow c \cap [\![\mathcal{D}.A]\!]) & \\
 & & (\simeq \lambda s.\ \text{if } s \vdash_{\mathcal{C}} c \text{ then } [\![\mathcal{D}.A]\!]s \text{ else } s) \\
[\![\mathcal{D}.A||B]\!] & = [\![\mathcal{D}.A]\!] \cap [\![\mathcal{D}.B]\!] & (\simeq Y(\lambda s.[\![\mathcal{D}.A]\!][\![\mathcal{D}.B]\!]s)) \\
[\![\mathcal{D}.\exists xA]\!] & = \{d \mid c \in [\![\mathcal{D}.A]\!],\ \exists xc = \exists xd\} & (\simeq \lambda s.\exists x[\![\mathcal{D}.A]\!]\exists xs) \\
[\![\mathcal{D}.p(\vec{x})]\!] & = [\![\mathcal{D}.A[\vec{x}/\vec{y}]]\!] \text{ if } p(\vec{y}) = A \in \mathcal{D} & (\simeq \lambda s.[\![\mathcal{D}.A[\vec{x}/\vec{y}]]\!]s)
\end{array}
$$

Theorem ([SRP91])

For any deterministic process $\mathcal{D}.A$

$$
\mathcal{O}_{ts}(\mathcal{D}.A; c) = \begin{cases} \{min([\![\mathcal{D}.A]\!] \cap \uparrow c)\} & \text{if } [\![\mathcal{D}.A]\!] \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}
$$

Non-deterministic CC($\mathcal{X}$) with Local Choice (2)

Let $[\![]\!] : \mathcal{D} \times A \rightarrow \mathcal{P}(\mathcal{P}(\mathcal{C}))$ be the least fixpoint (for $\subseteq$) of

$$
\begin{array}{rcl}
[\![\mathcal{D}.c]\!] & = & \{\uparrow c\} \\
[\![\mathcal{D}.c \rightarrow A]\!] & = & \{\mathcal{C} \setminus \uparrow c\} \cup \{\uparrow c \cap X | X \in [\![\mathcal{D}.A]\!]\} \\
[\![\mathcal{D}.A || B]\!] & = & \{X \cap Y \mid X \in [\![\mathcal{D}.A]\!],\ Y \in [\![\mathcal{D}.B]\!]\} \\
[\![\mathcal{D}.A + B]\!] & = & [\![\mathcal{D}.A]\!] \cup [\![\mathcal{D}.B]\!] \\
[\![\mathcal{D}.\exists x A]\!] & = & \{\{d \mid \exists x c = \exists x d,\ c \in X\} | X \in [\![\mathcal{D}.A]\!]\} \\
[\![\mathcal{D}.p(\vec{x})]\!] & = & [\![\mathcal{D}.A[\vec{x}/\vec{y}]]\!]
\end{array}
$$

Theorem ([MFP97])

For any process $\mathcal{D}.A$,
$\mathcal{O}_{ts}(\mathcal{D}.A; c) = \{d |$ *there exists* $X \in [\![\mathcal{D}.A]\!]$ *s.t.* $d = min(\uparrow c \cap X)\}$.

Part VII: CC and Linear Logic

Soundness

Theorem (Soundness of transitions)

Let $(X; c; \Gamma)$ and $(Y; d; \Delta)$ be CC configurations.
If $(X; c; \Gamma) \equiv (Y; d; \Delta)$ then $(X; c; \Gamma)^\dagger \dashv\vdash_{ILL(\mathcal{C},\mathcal{D})} (Y; d; \Delta)^\dagger$.
If $(X; c; \Gamma) \longrightarrow (Y; d; \Delta)$ then $(X; c; \Gamma)^\dagger \vdash_{ILL(\mathcal{C},\mathcal{D})} (Y; d; \Delta)^\dagger$.

Proof.

By induction on $\equiv$. Immediate.
By induction on $\longrightarrow$.
The choice operator $+$ is translated by the additive conjunction $\&$, which expresses "may" properties: $A \& B \vdash A$ and $A \& B \vdash B$. □

Completeness I

Theorem (Observation of successes)

Let A be a CC agent and c be a constraint.
If $A^\dagger \vdash_{ILL(\mathcal{C},\mathcal{D})} c$, then there exists a constraint d such that $(\emptyset; 1; A) \longrightarrow (X; d; \emptyset)$ and $\exists X d \vdash_{\mathcal{C}} c$.

Proof.

By induction on a sequent calculus proof π of ${A_1}^\dagger, \ldots, {A_n}^\dagger \vdash_{ILL(\mathcal{C},\mathcal{D})} \phi$,
where the A_i's are agents and ϕ is either a constraint or a procedure name.

□

Completeness II

Recall that $\top$ is the additive true constant neutral for $\&$.

Theorem (Observation of accessible stores)

Let A be a CC agent and c be a constraint.
If $A^\dagger \vdash_{ILL(\mathcal{C},\mathcal{D})} c \otimes \top$, then c is a store accessible from A, i.e. there exist a constraint d and a multiset Γ of agents such that $(\emptyset; 1; A) \longrightarrow (X; d; \Gamma)$ and $\exists X d \vdash_{\mathcal{C}} c$.

Proof.

The proof uses the first completeness theorem, and proceeds by an easy induction for the right introduction of the tensor connective in $c \otimes \top$. □

Encoding the π-calculus in LCC($\mathcal{H}$)

- Direct encoding of the asynchronous π-calculus:

$$
\begin{array}{lcl}
[0] & = & 1 \\
[(y)P] & = & \exists y[P] \\
[\overline{x}y.0] & = & \\
[x(y).P] & = & \\
[P|Q] & = & [P]|||[Q] \\
[[x=y]P] & = & (x=y) \rightarrow [P] \\
[P+Q] & = & [P]+[Q]
\end{array}
$$

- The usual (synchronous) π-calculus can be simulated with a synchronous communication protocol.

Encoding the π-calculus in LCC($\mathcal{H}$)

- Direct encoding of the asynchronous π-calculus:

$$\begin{array}{lcl} [0] & = & 1 \\ [(y)P] & = & \exists y[P] \\ [\overline{x}y.0] & = & tell(c(x,y)) \\ [x(y).P] & = & \\ [P|Q] & = & [P]|||[Q] \\ [[x=y]P] & = & (x=y) \rightarrow [P] \\ [P+Q] & = & [P]+[Q] \end{array}$$

- The usual (synchronous) π-calculus can be simulated with a synchronous communication protocol.

Encoding the π-calculus in LCC($\mathcal{H}$)

- Direct encoding of the asynchronous π-calculus:

$$
\begin{array}{lcl}
[0] & = & 1 \\
[(y)P] & = & \exists y[P] \\
[\overline{x}y.0] & = & \mathit{tell}(c(x,y)) \\
[x(y).P] & = & \forall y\ c(x,y) \rightarrow [P] \\
[P|Q] & = & [P]|||[Q] \\
[[x=y]P] & = & (x=y) \rightarrow [P] \\
[P+Q] & = & [P]+[Q]
\end{array}
$$

- The usual (synchronous) π-calculus can be simulated with a synchronous communication protocol.

Producer Consumer Protocol in LCC

$\texttt{P = dem} \longrightarrow (\texttt{pro} \parallel \texttt{P})$
$\texttt{C = pro} \longrightarrow (\texttt{dem} \parallel \texttt{C})$
$\texttt{init = dem}^n \parallel \texttt{P}^m \parallel \texttt{C}^k$

Deadlock-freeness: $\texttt{init} \not\longrightarrow_{LCC} \texttt{dem}^{n'} \parallel \texttt{P}^{m'} \parallel \texttt{C}^{k'} \parallel \texttt{pro}^{l'}$, with either $n' = l' = 0$ or $m' = 0$ or $k' = 0$

Number of units consumed always < number of units produced:
$\texttt{P = dem} \longrightarrow (\texttt{pro} \parallel \texttt{P} \parallel \forall \texttt{X} \ (\texttt{np=X} \longrightarrow \texttt{np=X+1}))$
$\texttt{C = pro} \longrightarrow (\texttt{dem} \parallel \texttt{C} \parallel \forall \texttt{X} \ (\texttt{nc=X} \longrightarrow \texttt{nc=X+1}))$
$\texttt{init = dem}^n \parallel \texttt{P}^m \parallel \texttt{C}^k \parallel \texttt{np=0} \parallel \texttt{nc=0}$
$\texttt{init} \not\longrightarrow_{LCC} \texttt{dem}^{n'} \parallel \texttt{pro}^{l'} \parallel \texttt{P}^m \parallel \texttt{C}^k \parallel \texttt{np=np}_0 \parallel \texttt{nc=nc}_0$
with $\texttt{nc}_0 > \texttt{np}_0$

Part VIII

LCC

Part VIII: LCC

LCC Operational Semantics

Tell $(X; c; tell(d), \Gamma) \longrightarrow (X; c \otimes d; \Gamma)$

Ask
$$\frac{c \vdash_{\mathcal{C}} d \otimes e[\vec{t}/\vec{y}]}{(X; c; \forall \vec{y}(e \rightarrow A), \Gamma) \longrightarrow (X; d; A[\vec{t}/\vec{y}], \Gamma)}$$

Hiding
$$\frac{y \notin X \cup fv(c, \Gamma)}{(X; c; \exists y A, \Gamma) \longrightarrow (X \cup \{y\}; c; A, \Gamma)}$$

Proc. call
$$\frac{(p(\vec{y}) = A) \in \mathcal{D}}{(X; c; p(\vec{y}), \Gamma) \longrightarrow (X; c; A, \Gamma)}$$

Choice
$$(X; c; A + B, \Gamma) \longrightarrow (X; c; A, \Gamma)$$
$$(X; c; A + B, \Gamma) \longrightarrow (X; c; B, \Gamma)$$

Congr.
$$\frac{z \notin fv(A)}{\exists y A \equiv \exists z A[z/y]} \quad A \parallel B \equiv B \parallel A \quad A \parallel (B \parallel C) \equiv (A \parallel B) \parallel C$$

An LCC($\mathcal{FD}$) program for the dining philosophers

An LCC($\mathcal{FD}$) program for the dining philosophers

```
Goal(N) = RecPhil(1,N).

RecPhil(M,P) =
      M ≠ P → (Philo(M,P) ∥ fork(M) ∥ RecPhil(M+1,P))
∥
      M = P → (Philo(M,P) ∥ fork(M)).

Philo(I,N) =
      (fork(I) ⊗ fork(I+1 mod N)) →
          (eat(I) ∥
           eat(I) → (fork(I) ∥ fork(I+1 mod N) ∥
Philo(I,N))).
```

CC($\mathcal{FD}$) in LCC($\mathcal{H}$)

```
fd(X) = tell(min(X,min_integer) ⊗ max(X,max_integer))

'x≥₁y+c'(X,Y,C) =
    min(X,MinX) ⊗ min(Y,MinY) ⊗ (MinX<MinY+C)
    → (tell(min(X,MinY+C) ⊗ min(Y,MinY))
       ∥ 'x≥₁y+c'(X,Y,C))

'x≥y+c'(X,Y,C) = 'x≥₁y+c'(X,Y,C) ∥ 'x≥₂y+c'(X,Y,C)

'ask(x≥y)→a'(X,Y,A) =
    min(X,MinX) ⊗ max(Y,MaxY) ⊗ (MinX>MaxY)
    → A ∥ tell(min(X,MinX) ⊗ max(Y,MaxY))
```

CC($\mathcal{FD}$) propagators, including indexicals, are now easily embedded in LCC.

Imperative variables allow a finer control, which is necessary for certain constraint solvers, see for instance the implementation of a Simplex solver in LCC [Sch99].

INRIA

Logical Semantics

Simple translation of LCC into ILL:

$$
\begin{array}{lr}
tell(c)^\dagger = c & p(\vec{x})^\dagger = p(\vec{x}) \\
\forall \vec{y}(c \rightarrow A)^\dagger = \forall \vec{y}\ (c \multimap A^\dagger) & (A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \\
(A + B)^\dagger = A^\dagger \ \&\ B^\dagger & (\exists x A)^\dagger = \exists x A^\dagger
\end{array}
$$

ILL$(\mathcal{C}, \mathcal{D})$ denotes the deduction system obtained by adding to intuitionistic linear logic the axioms:

- $c \vdash d$ for every $c \Vdash_{\mathcal{C}} d$ in $\Vdash_{\mathcal{C}}$,
- $p(\vec{x}) \vdash A^\dagger$ for every declaration $p(\vec{x}) = A$ in $\mathcal{D}$.

Same soundness/completeness as CC.

Phase Semantics

A phase space $\mathbf{P} = \langle P, \times, 1, \mathcal{F} \rangle$ is a structure such that:

1. $\langle P, \times, 1 \rangle$ is a commutative monoid.
2. the set of facts $\mathcal{F}$ is a subset of P such that: $\mathcal{F}$ is closed by arbitrary intersection, and for all $A \subset P$, for all $F \in \mathcal{F}$,

   $A \multimap F = \{x \in P : \forall a \in A, a \times x \in F\}$ is a fact.

We define the following operations:

$$A \,\&\, B = A \cap B$$

$$A \otimes B = \bigcap \{F \in \mathcal{F} : A \times B \subset F\} \qquad A \oplus B = \bigcap \{F \in \mathcal{F} : A \cup B \subset F\}$$

$$\exists x A = \bigcap \{F \in \mathcal{F} : (\bigcup_x A) \subset F\} \qquad \forall x A = \bigcap \{F \in \mathcal{F} : (\bigcap_x A) \subset F\}$$

We'll note $\top$ the fact P, $\mathbf{0} = \bigcap \{F \in \mathcal{F}\}$ and $\mathbf{1} = \bigcap \{F \in \mathcal{F} \mid 1 \in F\}$.

Interpretation

Let η be a valuation assigning a fact to each atomic formula such that: $\eta(\top) = \top$, $\eta(\mathbf{1}) = \mathbf{1}$ and $\eta(\mathbf{0}) = \mathbf{0}$.

We can now define inductively the interpretation of a sequent:

$$\eta(\Gamma \vdash A) = \eta(\Gamma) \multimap \eta(A) \qquad \eta(\Gamma) = \mathbf{1} \text{ if } \Gamma \text{ is empty}$$

$$\eta(\Gamma, \Delta) = \eta(\Gamma) \otimes \eta(\Delta) \qquad \eta(A \otimes B) = \eta(A) \otimes \eta(B)$$

$$\eta(A \,\&\, B) = \eta(A) \,\&\, \eta(B) \qquad \eta(A \multimap B) = \eta(A) \multimap \eta(B)$$

We then define the notion of validity as follows:
$\mathbf{P}, \eta \models (\Gamma \vdash A)$ iff $1 \in \eta(\Gamma \vdash A)$, thus $\eta(\Gamma) \subset \eta(A)$.

Soundness:

$$\Gamma \vdash_{ILL} A \text{ implies } \forall \mathbf{P}, \forall \eta, \mathbf{P}, \eta \models (\Gamma \vdash A).$$

Phase Counter-Models

We impose to every valuation η to satisfy the non-logical axioms of $\mathrm{ILL}_{\mathcal{C},\mathcal{D}}$:

- $\eta(c) \subset \eta(d)$ for every $c \Vdash_{\mathcal{C}} d$ in $\Vdash_{\mathcal{C}}$,
- $\eta(p) \subset \eta(A^\dagger)$ for every declaration $p = A$ in $\mathcal{D}$.

The contrapositive of the two soundness theorems becomes:

Theorem

to prove a safety property of the form

$$(X; c; A) \not\longrightarrow (Y; d; B)$$

It is enough to show

$$\exists \mathbf{P}, \exists \eta, \exists a \in \eta((X; c; A)^\dagger) \text{ such that } a \notin \eta((Y; d; B)^\dagger).$$

Producer Consumer Protocol in LCC

$\texttt{P = dem} \longrightarrow (\texttt{pro} \parallel \texttt{P})$
$\texttt{C = pro} \longrightarrow (\texttt{dem} \parallel \texttt{C})$
$\texttt{init = dem}^n \parallel \texttt{P}^m \parallel \texttt{C}^k$

Deadlock-freeness: $\texttt{init} \not\longrightarrow \texttt{dem}^{n'} \parallel \texttt{P}^{m'} \parallel \texttt{C}^{k'} \parallel \texttt{pro}^{l'}$, with either $n' = l' = 0$ or $m' = 0$ or $k' = 0$

Producer Consumer Protocol in LCC

$\texttt{P} = \texttt{dem} \longrightarrow (\texttt{pro} \parallel \texttt{P})$
$\texttt{C} = \texttt{pro} \longrightarrow (\texttt{dem} \parallel \texttt{C})$
$\texttt{init} = \texttt{dem}^n \parallel \texttt{P}^m \parallel \texttt{C}^k$

Deadlock-freeness: $\texttt{init} \not\longrightarrow \texttt{dem}^{n'} \parallel \texttt{P}^{m'} \parallel \texttt{C}^{k'} \parallel \texttt{pro}^{l'}$, with either $n' = l' = 0$ or $m' = 0$ or $k' = 0$

Let us consider the structure $(\mathbb{N}, \times, 1, \mathcal{P}(\mathbb{N}))$, it is obviously a phase space.

Producer Consumer Protocol in LCC

$\texttt{P = dem} \longrightarrow (\texttt{pro} \parallel \texttt{P})$
$\texttt{C = pro} \longrightarrow (\texttt{dem} \parallel \texttt{C})$
$\texttt{init = dem}^n \parallel \texttt{P}^m \parallel \texttt{C}^k$

Deadlock-freeness: $\texttt{init} \not\longrightarrow \texttt{dem}^{n'} \parallel \texttt{P}^{m'} \parallel \texttt{C}^{k'} \parallel \texttt{pro}^{l'}$, with either $n' = l' = 0$ or $m' = 0$ or $k' = 0$

Let us consider the structure $(\mathbb{N}, \times, 1, \mathcal{P}(\mathbb{N}))$, it is obviously a phase space.

Let us define the following valuation:

$$\eta(\texttt{P}) = \{2\} \;\; \eta(\texttt{C}) = \{3\} \;\; \eta(\texttt{dem}) = \{5\} \;\; \eta(\texttt{pro}) = \{5\}$$

$$\eta(\texttt{init}) = \{2^m \cdot 3^k \cdot 5^n\}$$

INRIA

Proof

- We have to check the correctness of η:
  $\forall p_1 \in \eta(\mathtt{P}), \exists p_2 \in \eta(\mathtt{P}), \mathit{dem} \cdot p_1 = \mathit{pro} \cdot p_2$, hence
  $\eta(\mathtt{P}) \subset \eta(\text{body of } \mathtt{P})$.
  The same for C, and $\eta(\mathtt{init}) = \eta(\text{body of } \mathtt{init})$.
- Instead of exhibiting a counter-example, we will prove *Ab absurdum* that the inclusion
  $\eta(\mathtt{init}) \subset \eta(\mathtt{dem}^{n'} \parallel \mathtt{P}^{m'} \parallel \mathtt{C}^{k'} \parallel \mathtt{pro}^{l'})$ is impossible.
  Suppose $\eta(\mathtt{init}) \subset \{5^{n'} \cdot 2^{m'} \cdot 3^{k'} \cdot 5^{l'}\}$ Comparing the power of 5, 3 and 2, anything else than: $n' + l' = n$ and $m' = m$ and $k' = k$ is impossible, and therefore if there is a deadlock ($n' + l' = 0 \neq n$, or $m' = 0 \neq m$, or $k' = 0 \neq k$) $\eta(\mathtt{init})$ is not a subset of its interpretation and thus init does not reduce into it, qed.

Automatization

The search for a phase space can be automatized, if one accepts some restrictions:

- always use the structure $(\mathbb{N}, \times, 1, \mathcal{P}(\mathbb{N}))$;
- always look for simple (singleton/doubleton/finite) interpretations.

Automatization

The search for a phase space can be automatized, if one accepts some restrictions:

- always use the structure $(\mathbb{N}, \times, 1, \mathcal{P}(\mathbb{N}))$;
  [*be careful that integers are invertible*]
- always look for simple (singleton/doubleton/finite) interpretations.
  [*might lead to confusions*]

Bibliography I

Kim Marriott Moreno Falaschi, Maurizio Gabbrielli and Catuscia Palamidessi.
Confluence in concurrent constraint programming.
Theoretical Computer Science, 183(2):281–315, 1997.

Vincent Schächter.
Programmation concurrente avec contraintes fondée sur la logique linéaire.
PhD thesis, Université d'Orsay, Paris 11, 1999.

Vijay A. Saraswat, Martin C. Rinard, and Prakash Panangaden.
Semantic foundations of concurrent constraint programming.
In *POPL'91: Proceedings of the 18th ACM Symposium on Principles of Programming Languages*, 1991.