October 13, 1999

BRICS, University of Aarhus
Frank D. Valencia

Saraswat,

Concurrent Constraint Programming & La Semantic Foundations of Determinate
Concurrent Programming (CP):

Motivation
Operations Read/Write replaced by Ask/Tell.

von Neumann's store replaced by store with partial information.

Basic Ideas of CCP:

Programming (ConLP). CCP generalizes the basic ideas of ConLP.

Extending Logic Programming (LP) gave rise to Concurrent Logic Programming (CCP).

Motivation
Motivation: CCP Scenario & Concurrency

\[ tell(X > 10) \quad \text{ask}(X < 50) \rightarrow P \]

\[ tell(X < 20) \quad \text{ask}(X = 15) \rightarrow Q \]

▷ Concurrent Executions of Agents.

▷ Synchronization: Via Blocking-Ask.
\((Z)f = X \forall \in \mathcal{E} \leftarrow ((X)f = X^X)_{\forall \mathcal{E} \in \mathcal{E}} = \emptyset \) and \(((W)f = X) \forall \in \mathcal{E} = \mathcal{P} \)

\(W = Z \) then \((Z)f = X : Z^X \) if \((W)f = X \)

Suppose \(f \) is injective: Given \(f \) is injective send message \(d \) through channel \(W \)

Communication channels are embeddable: e.g. sending message \(d \) through channel \(W \)

Open Communication: By adding information to the store.

Motivation: Concurrency.
Agenda
Syntax for the cc-model.

Let $X, \ldots, \psi$ be variables and $\chi, \ldots, \eta$ be subsets of atomic propositions.

Agent definition

Definition call

Guarded choice (\(*\))

Parallel execution

Hide $X$

Ask $\alpha$

Tell $\rho$

do nothing

true

\[ \begin{align*}
\text{Progs} & \colon= \emptyset \\
\text{Agents} & \colon= D \triangleq A \land D
\end{align*} \]
\[(\ast) \text{ If } X \text{ is fresh } \quad \frac{\forall (X \alpha)}{\forall (X \alpha)} \quad \text{HIDE}\]

\[
\frac{\forall (X \alpha) \land \forall (Y \beta)}{\forall (X \alpha) \land \forall (Y \beta)}
\quad \frac{\forall (X \alpha) \land \forall (Y \beta)}{\forall (X \alpha) \land \forall (Y \beta)}
\quad \text{PAIR}
\]

\[(\ast) \quad \frac{\forall (X \alpha)}{\forall (X \alpha)} \quad \text{ASK}\]

\[
\forall (\text{true}) \quad \text{TELL}
\]

\[
\forall \text{ means: } A \land (\alpha \land \beta) \Rightarrow \text{Initiated in store } \beta
\]
is eventually executed.

\( X = Z \) \( \forall t \in T \) \( \wedge (\exists agent\ as\ \text{initiated\ in\ any\ store}) \)

\( \forall \) \( \forall t \in T \)

\( (Z', X)_{\text{max}} \mid (10 > \bar{X}) \forall t \Rightarrow (20 < X) \forall t \) = \( \forall \)

\( (Z', X)_{\text{max}} \mid (10 < \bar{X}) \forall t \Rightarrow (20 < X) \forall t \) = \( \forall \)

\( ((\bar{X} = Z) \forall t \Rightarrow (X < \bar{X}) \forall s) \mid ((X = Z) \forall t \Rightarrow (\bar{X} < X) \forall s) \) \( \Rightarrow (Z, X)_{\text{max}} \) = \( \forall \)

A program example

A program example
Coming up: A formal definition of Constraint Systems.

- CCP languages are parameterized in an underlying Constraint System.
- These assumptions.
- Notion of Constraint System generalizes and provides conditions for making
- Assumptions: Built-in relations on integers like $<$ and their decidability.

Roadmap I.
Relation \( \rightarrow \) is extended to \( (d)^f \times (d)^f \) in the obvious way.

\[ q \rightarrow q \quad \text{if} \quad q \perp a \quad \text{for all} \quad a \in \perp \quad \text{and} \quad \perp \rightarrow \quad q \]

\[ q \rightarrow q \quad \text{if} \quad a \in \perp \]

validating primitive constraints and \( (d)^f \rightarrow (d)^f \) is a compact entailment relation

A constraint system is a structure \( \langle D, \rightarrow \rangle \) where \( D \) is set of constraint systems à la Scott.
\{X > 7, X > 6, X > 5\} ... is represented as \{X > 7, X > 6, X > 5\}.

Example: Provided a suitable class of natural numbers with \(N\), constraint \(X > 5\) is denoted by \(|D|\).

Definition 2: Constrained constraints are subsets of \(\mathbb{R}^n\). The set of all constraints is denoted by \(|D|\).
The Lattice

Structure \((\mathcal{P}(D), \subseteq)\) is a complete (algebraic) lattice:

\[
\begin{array}{c}
\text{false} = D \\
\vdots \\
\text{true} = \{a \in D | \emptyset \vdash a\}
\end{array}
\]

- Glbs (\(\sqcap\)) is given by intersection.
- Lubs (\(\sqcup\)) is given by closure of the union.

\(\uparrow\text{Set } |D|_0 \subseteq |D|\) denotes the set of finite elements of the lattice.
A Constraint System Example
Constraint systems for arithmetics.

The Herbrand constraint system.

Constraint systems of data structures (trees, arrays, enumerated types).

Constraint systems of names (e.g. \( \emptyset \neq a \) for any two different names).

Any first-order language vocabulary with countable many variables and countable entailment relation.
Towards Hiding in Constraint Systems
Semantics of the Process.

The underlying complete lattice $|\mathcal{D}|$ will be used as the domain from which elements $|\mathcal{D}|^0$. Constraint system $\mathcal{D}$ generates a set of constraints (a lattice) $|\mathcal{D}|$. The notion of constraint system has been defined.

Road-map 2.
What observation can we make of a process?

A la Hoare:

A determinate cc-model Semantics
\[ \{ X \leq 1 \} \] is not a Resting point for process \( p \).

\[ \{ 10 > X \} \] is a Resting point for process \( p \).

Observation to make of a process: its set of Resting points.

- A Resting point for process \( p \) is a constraint on input of which \( p \) halts without adding any information.
3. If then \( \forall \delta \) more information, more information out.

\[
(\forall x \exists y \forall \delta) \delta f \subseteq (\forall \delta) \delta f
\]

2. Returns a resting point \( \beta \) such that:

\[
\beta \oplus \beta = (\forall \delta) \delta f
\]

1. Only adds information:

\[
\beta \oplus \beta
\]

Each process is associated to a function \( \beta \) such that:

**Closure Operator**
Closure Operator over complete lattices

▷ Function $f_P$ is a closure operator.

▷ A closure operator over complete lattices (like $(|D|, \subseteq)$) can be represented by its range (i.e. its set of fixed points).

▷ To give the meaning of $P$ by $f_P$ we only need to specify the range of $f_P$.

قسمة علامة تبديل

▷ Coming up: Meaning of Processes and Revisited Operational Semantics.
Denotational Semantics: Tells

\[
\langle T \text{ell} \rangle (\tau, \tau') \leftarrow \text{true}
\]

Optionally:

\[
\{ \sigma \models \exists \, \tau \mid \tau \models \exists \, \sigma \} = \langle \text{tell} \rangle ^\| = \mathfrak{v} [\langle \text{tell} \rangle]
\]

Optionally, in terms of fixed points:

\[
\mathfrak{v} \left[ \langle \text{tell} \rangle \right] \models \exists \, \tau \mid \tau \models \exists \, \sigma \, \models \sigma = \langle \text{tell} \rangle (\sigma, \tau) \models \text{asserts input with } \sigma', \text{ i.e., } \text{asserts input with } \sigma
\]
\{ \forall [A] \in \mathcal{L} \leftarrow \mathcal{O} \preceq \mathcal{L} \mid |P| \in \mathcal{L} \} = \forall [A] \leftarrow \mathcal{O} \preceq \mathcal{L} \]

In terms of fixed points:

\[ \mathcal{O} (x) \forall f \text{ else } \mathcal{O} \preceq x \text{ if } \chi x = (\forall \mathcal{O} \leftarrow \mathcal{L}) f \]

Information as \( \mathcal{O} \), i.e.:

Agent \( \mathcal{O} \leftarrow \mathcal{L} \) blocks \( A \) until the store contains at least as much

Denotational Semantics: \( \mathcal{O} \leftarrow A \)
\[ \varphi \vdash \bot \quad \forall \varphi \left( \forall \psi \varphi \right) \]

Optionally:

\[ \forall \varphi \left( \land \psi \varphi \right) = \quad \forall \varphi \left( \exists \varphi \varphi \right) = \quad \forall \varphi \left( \forall \varphi \varphi \right) = \quad \forall \varphi \left( \exists \psi \varphi \right) = \quad \forall \varphi \left( \exists \psi \varphi \right) = \quad \forall \varphi \left( \exists \psi \varphi \right) = \]

It follows that:

**Denotational Semantics: ASKS**
The set of resting points of $A \parallel B$ is just $f_A \cup f_B$.

By monotonicity of $f_A$ and $f_B$ and $f$ is just $\exists o, \forall A \in f_A$, $\forall B \in f_B$.

Consider $A \parallel B$. Suppose $o$ and $\nu$ are resting points of $A$ and $B$, i.e. $o \in f_A$ and $\nu \in f_B$.

Denotational Semantics: Parallel Composition.
\[
\begin{array}{c}
\frac{\mathcal{B} \parallel \mathcal{A} \mathrel{\lll} \mathcal{B} \mathrel{\lll} \mathcal{A}}{\mathcal{B} \mathrel{\lll} \mathcal{A} \mathrel{\lll} \mathcal{B}} \quad \frac{\mathcal{B} \parallel \mathcal{A} \mathrel{\lll} \mathcal{B} \mathrel{\lll} \mathcal{A}}{\mathcal{A} \mathrel{\lll} \mathcal{B} \mathrel{\lll} \mathcal{A}}
\end{array}
\]

Operationally:

\[
\forall [\mathcal{B}] \cup \forall [\mathcal{A}] = \forall [\mathcal{B} \parallel \mathcal{A}]
\]

Denotationally:

\[
\forall [\mathcal{B}] \cup \forall [\mathcal{A}] = \forall [\mathcal{B} \parallel \mathcal{A}]
\]
\begin{align*}
\land \subseteq \land \subseteq \top \iff (\land) \text{tell} & \leftarrow (\land) \text{ask} \quad \forall \quad \text{ask} \quad (\land) \text{tell} \iff (\land) \text{ask} \\
(B \leftarrow (\land) \text{ask}) \| (A \leftarrow (\land) \text{ask}) &= \forall \quad \text{true} \\
\top \| (B \| A) &= \forall \quad (C \| B) \| A \\
\forall \| B &= \forall \quad B \| A
\end{align*}

From the denotational definition several equivalences follow, e.g.

Denotational Semantics: Parallel Composition.
\{0 < X\} X E X \notin \{0 < X\} X E X \notin \{0 > X\} X E X \notin \{0 \leq X\} X E X \notin \{0 \geq X\} X E X \notin

\begin{equation}
\begin{aligned}
\{0 > X\} \forall a (X_a) \quad \| \quad (0 < X) \forall a (X_a) = E \\
\end{aligned}
\end{equation}

Should we see from \( A \) on input \( a \) is e.g. the interactions on \( E \) from its environment \( F \). What Agent \( A \) hides the interactions on \( X \) from its environment \( E \). What

Denotational Semantics: Hiding.
\[(\varnothing X \in) BF X \in \cap \varnothing = BF (X^n)\]

In general:

\[\{0 < X\} X \in \{g > X\} \cap \{0 < X\} \quad \text{Alice sees:} \quad B \quad \text{\begin{Cases}
        \{0 < X\} X \in \{g > X\} = (\{0 < X\} X \in) BF
    \end{Cases}\]

Alice sees what Bob sees on input \(X\), i.e.:

\[\{0 < X\} X \in \{g > X\} \quad \text{\begin{Cases}
        Alice sees: \quad B of \text{Bob's sees from} \quad \text{Alice's, so we'd better hide it from her.}
    \end{Cases}\]
\[
\{ \forall X \in \exists [B] \in \forall \exists \text{ there exists } 0 | 0 \in V \} = \forall [B(X^n)]
\]

\[
\begin{array}{c}
\frac{B(X^n)}{(\exists X \in B(X^n))} \\
\frac{B(X^n)}{(\exists X \in B(X^n))} \\
\frac{B(X^n)}{(\exists X \in B(X^n))}
\end{array}
\]

Operations:

Denotation:

Semantic Semantics: Hiding.
Then any other output sequence is terminal with final store \( y \).

**Theorem (Confluence)** If \( D \) has a terminal \( o \) sequence, then any sequence of transitions from \( A \) is a sequence of transitions from \( D \).

**Definition.** A sequence for a program \( D \) is a sequence of transitions that leads to a terminal output.
Theorem. The function $\phi(p)$ is a closure operator.

\[
\begin{align*}
\text{Define operational observation function:} \\
\phi(p) &= \begin{cases} 
\text{false} & \text{otherwise} \\
\text{true} & \text{if } p \text{ has a terminal } \sigma -\text{sequence with final store } y
\end{cases}
\end{align*}
\]
For \( ([\emptyset]c) \cdot \emptyset = ([p]c) \cdot \emptyset \) and \( d[p] = d[p] \), which is the Full Abstraction theorem.

\[ d[p] = (p) \cdot \emptyset \] is the Strong Adequacy theorem.

Correspondence.
Conclusions.
Research on CP includes:

- Logical view of CP
- Linear CP
- Mobility on CP
- Concurrent Objects qua CP
- True Concurrency in CP
Constraint Solvers.

Functional Programming with Constraints: Sccamer.

Domain Specific Languages: Patchwork.

Linear Janus, CORVIAL.

CCP (Visual Object Oriented Programming Languages: OZ, Pictorial Lanes).

Research on CP.
\[ o \vdash (\{ \lambda x p \} \cap o) X \in \{ \lambda x p \} \]

3. If \( X \neq Z \) then \( \{ \lambda z p, \lambda x p \} Z \in \{ \lambda x p \} \)

1. \( \forall X p \vdash \emptyset \)

Set required to contain for every \( \lambda p \) a token of \( \lambda p \) satisfying:

\[ \lambda = X \]

\( \phi \land (\lambda = X) X \in \text{Formula is just } [X/\lambda] \phi \)

Diagonal Elements (skip).

Towards Modeling Passing a la Tarski.

Concurrent Constraint Programming
No explicit representation of $C_1$ in denotation of $A$ is needed.

The effect of $C_1$ on $Y$ can be obtained by running the original $A$ on $Y$.

What is its subsequent behavior on input $Y$?

quiescing leaving a "residual agent $C_1$"

Restarts ability: Suppose agent $A$ is initiated in $o$ and produces $o'$ before (skip).

Denotational Semantics: Parallel Composition
The operational rule for calls is:

$$((d) \forall x \forall p)(x \forall)$$

$$\xrightarrow{(\exists D)} (X) D$$

$$((d) \forall X \forall p) \forall E = \forall [(X) D]$$

$$[(\forall \forall [V] \forall X \forall p) \forall E \xleftarrow{d} \forall \[d]] = \forall \[D \forall \forall : (X) D]$$

Using Tarski's diagonal elements for modeling parameter passing:

Using Tarski's diagonal elements for modeling parameter passing:

Procedure calls handled by looking up their name in the environment:

Denotational Semantics: Definition Call (skip)
\[ \underbrace{\varphi[D] \times \varphi[A]}_{\text{Recursion is handled as usual}} = \varphi[A \cdot D] \]
Theorem. If \( A \) and \( B \) have the same normal form,

\[
\mathcal{A}[\mathcal{E}] = \mathcal{A}[\mathcal{E}] \iff \mathcal{B} = \mathcal{B}
\]

Lemma. Any agent \( A \) containing no \( X \), \( X \) has a normal form.

\[
\begin{align*}

\forall \, \forall \, A \subseteq B & \iff \forall \, \forall \, B \subseteq A \\
\forall \, \forall \, A \subseteq B & \iff \forall \, \forall \, B \subseteq A \\
\forall \, \forall \, A \subseteq B & \iff \forall \, \forall \, B \subseteq A \\
\end{align*}
\]

Definition. An agent \( A \) is normal form iff \( A \) is true or \( A \) is true or \( A \) is true or \( A \) is true.

Completeness Results (skip).
2. Vice versa with the roles of $p$ and $q$ exchanged.

Definition: Two process-store pairs $(p, q)$ and $(p', q')$ are bisimilar if

$$\mathcal{O}(\cdot, q') \approx_{\mathcal{O}(\cdot) \cap q} \mathcal{O}(\cdot, q) \cap p \approx_{\mathcal{O}(\cdot) \cap p} \mathcal{O}(\cdot, q') \cap q$$