Constraint Logic Programming

Sylvain Soliman
Sylvain.Soliman@inria.fr

Project-Team LIFEWARE

MPRI 2.35.1 Course – September–November 2017

Part I: CLP - Introduction and Logical Background

Part II: Constraint Logic Programs

Part III: CLP - Operational and Fixpoint Semantics

Part IV: Logical Semantics

Part V: Constraint Solving

Part VI: Practical CLP Programming

Part VII: More Constraint Programming

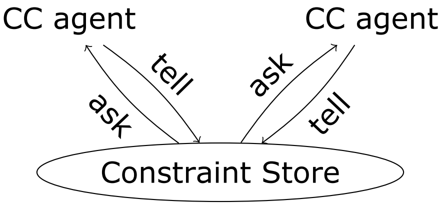
Part VIII: Programming Project

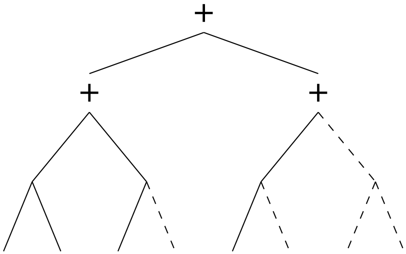
Part IX: Concurrent Constraint Programming

Concurrent Constraint Programs

Class of programming languages CC($\mathcal{X}$) introduced by Saraswat [Saraswat93mit] as a merge of Constraint and Concurrent Logic Programming.

Processes	$P ::= \mathcal{D}.A$
Declarations	$\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$
Agents	$A ::= tell(c) \mid \qquad\qquad \mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$

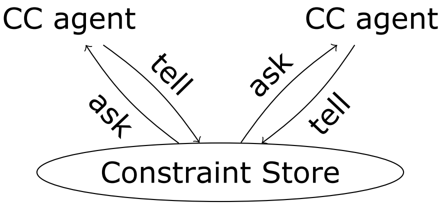


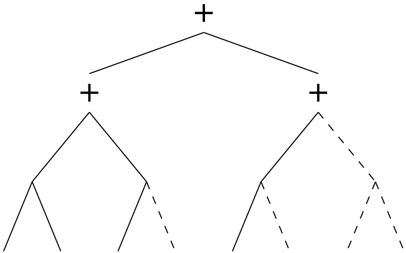


Concurrent Constraint Programs

Class of programming languages CC($\mathcal{X}$) introduced by Saraswat [Saraswat93mit] as a merge of Constraint and Concurrent Logic Programming.

Processes	$P ::= \mathcal{D}.A$
Declarations	$\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$
Agents	$A ::= tell(c) \mid \forall\vec{x}(c \to A) \mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$





CC($\mathcal{X}$) Transitions

Interleaving semantics

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

Tell
$$(\vec{x}; c; \mathit{tell}(d), \Gamma) \longrightarrow (\vec{x}; c \wedge d; \Gamma)$$

Ask

Blind choice (local/internal)
$$(\vec{x}; c; A + B, \Gamma) \longrightarrow (\vec{x}; c; A, \Gamma)$$
$$(\vec{x}; c; A + B, \Gamma) \longrightarrow (\vec{x}; c; B, \Gamma)$$

CC($\mathcal{X}$) Transitions

Interleaving semantics

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

Tell
$$(\vec{x}; c; tell(d), \Gamma) \longrightarrow (\vec{x}; c \wedge d; \Gamma)$$

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

Blind choice (local/internal)
$$(\vec{x}; c; A + B, \Gamma) \longrightarrow (\vec{x}; c; A, \Gamma)$$
$$(\vec{x}; c; A + B, \Gamma) \longrightarrow (\vec{x}; c; B, \Gamma)$$

CC($\mathcal{X}$) Operational Semanticssss

- observing the set of success stores,

$$\mathcal{O}_{ss}(\mathcal{D}.A; c) = \{\exists \vec{x} d \in \mathcal{X} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \epsilon)\}$$

- observing the set of terminal stores (successes and suspensions),

$$\mathcal{O}_{ts}(\mathcal{D}.A; c) = \{\exists \vec{x} d \in \mathcal{X} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \Gamma) \not\longrightarrow\}$$

- observing the set of accessible stores,

$$\mathcal{O}_{as}(\mathcal{D}.A; c) = \{\exists \vec{x} d \in \mathcal{X} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \Gamma)\}$$

- observing the set of limit stores?

$$\mathcal{O}_{\infty}(\mathcal{D}.A; c_0) = \{\sqcup_? \{\exists \vec{x}_i c_i\}_{i \geq 0} | (\emptyset; c_0; A) \longrightarrow (\vec{x}_1; c_1; \Gamma_1) \longrightarrow \ldots\}$$

Part X: CC - Denotational Semantics

Denotational semantics: input/output function

Input: initial store c_0
Output: terminal store c or *false* for infinite computations

Order the lattice of constraints $(\mathcal{C}, \leq)$ by the information ordering:
$\forall c, d \in \mathcal{C}\ c \leq d$ iff $d \vdash_{\mathcal{X}} c$ iff $\uparrow d \subset \uparrow c$ where $\uparrow c = \{d \in \mathcal{C} \mid c \leq d\}$.

$[\![\mathcal{D}.A]\!] : \mathcal{C} \to \mathcal{C}$ is

1. Extensive: $\forall c\ c \leq [\![\mathcal{D}.A]\!]c$
2. Monotone: $\forall c, d\ c \leq d \Rightarrow [\![\mathcal{D}.A]\!]c \leq [\![\mathcal{D}.A]\!]d$
3. Idempotent: $\forall c\ [\![\mathcal{D}.A]\!]c = [\![\mathcal{D}.A]\!]([\![\mathcal{D}.A]\!]c)$

i.e., $[\![\mathcal{D}.A]\!]$ is a closure operator over $(\mathcal{C}, \leq)$.

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:

$[\![\mathcal{D}.tell(c)]\!]$
$[\![\mathcal{D}.c \rightarrow A]\!]$

$[\![\mathcal{D}.A \parallel B]\!]$
$[\![\mathcal{D}.\exists x A]\!]$
$[\![\mathcal{D}.p(\vec{x})]\!]$ if $p(\vec{y}) = A \in \mathcal{D}$

Theorem 1 ([SRP91popl])

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

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

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:

$$[\![\mathcal{D}.tell(c)]\!] \quad = \uparrow c \qquad\qquad (\simeq \lambda s.s \wedge c)$$

$$[\![\mathcal{D}.c \rightarrow A]\!]$$

$$[\![\mathcal{D}.A \parallel B]\!]$$

$$[\![\mathcal{D}.\exists x A]\!]$$

$$[\![\mathcal{D}.p(\vec{x})]\!] \qquad \text{if } p(\vec{y}) = A \in \mathcal{D}$$

Theorem 1 ([SRP91popl])

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

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

Semantic Equations

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

$$
\begin{array}{ll}
[\![\mathcal{D}.tell(c)]\!] & = \uparrow c \qquad\qquad (\simeq \lambda s.s \wedge c) \\
[\![\mathcal{D}.c \to A]\!] & = (\mathcal{C} \setminus \uparrow c) \cup (\uparrow c \cap [\![\mathcal{D}.A]\!]) \\
 & \qquad (\simeq \lambda s.\text{ if } s \vdash_{\mathcal{C}} c \text{ then } [\![\mathcal{D}.A]\!]s \text{ else } s) \\
[\![\mathcal{D}.A \parallel B]\!] & \\
[\![\mathcal{D}.\exists x A]\!] & \\
[\![\mathcal{D}.p(\vec{x})]\!] & \qquad \text{if } p(\vec{y}) = A \in \mathcal{D}
\end{array}
$$

Theorem 1 ([SRP91popl])

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

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

Semantic Equations

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

$$
\begin{array}{lll}
[\![\mathcal{D}.tell(c)]\!] & = \uparrow c & (\simeq \lambda s.s \wedge c) \\
[\![\mathcal{D}.c \to A]\!] & = (\mathcal{C} \setminus \uparrow c) \cup (\uparrow c \cap [\![\mathcal{D}.A]\!]) & \\
 & & (\simeq \lambda s. \text{ if } s \vdash_{\mathcal{C}} c \text{ then } [\![\mathcal{D}.A]\!]s \text{ else } s) \\
[\![\mathcal{D}.A \parallel B]\!] & = [\![\mathcal{D}.A]\!] \cap [\![\mathcal{D}.B]\!] & (\simeq Y(\lambda s.[\![\mathcal{D}.A]\!][\![\mathcal{D}.B]\!]s)) \\
[\![\mathcal{D}.\exists x A]\!] & & \\
[\![\mathcal{D}.p(\vec{x})]\!] & \quad \text{if } p(\vec{y}) = A \in \mathcal{D} &
\end{array}
$$

Theorem 1 ([SRP91popl])

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

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

Semantic Equations

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

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

Theorem 1 ([SRP91popl])

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

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

Semantic Equations

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

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

Theorem 1 ([SRP91popl])

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

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

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

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

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

Theorem 2 ([FGMP97tcs])

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

Part XI

CC and Linear Logic

Part XI: CC and Linear Logic

Logical Semantics of CC?

- CC calculus is **sound but not complete** w.r.t. CLP logical semantics interpreting *asks* as *tells*
- Interpreting $ask(c \to A)$ as logical implication leads to identify CC transitions with logical deductions:

$$\mathit{left} \to \quad \frac{c \vdash_{\mathcal{C}} d}{c \wedge (d \to A^{\dagger}) \vdash c \wedge A^{\dagger}} \qquad \frac{p(\vec{x}) \vdash_{\mathcal{D}} A^{\dagger}}{c \wedge p(\vec{x}) \vdash c \wedge A^{\dagger}}$$

(reverses the arrow of CLP interpretation…)

- To distinguish between successes and accessible stores agents shouldn't "disappear" by the rule:

Logical Semantics of CC?

- CC calculus is **sound but not complete** w.r.t. CLP logical semantics interpreting *asks* as *tells*
- Interpreting $ask(c \to A)$ as logical implication leads to identify CC transitions with logical deductions:

$$left \to \quad \frac{c \vdash_{\mathcal{C}} d}{c \wedge (d \to A^{\dagger}) \vdash c \wedge A^{\dagger}} \qquad \frac{p(\vec{x}) \vdash_{\mathcal{D}} A^{\dagger}}{c \wedge p(\vec{x}) \vdash c \wedge A^{\dagger}}$$

(reverses the arrow of CLP interpretation...)

- To distinguish between successes and accessible stores agents shouldn't "disappear" by the weakening rule:

$$leftW \ \frac{\Gamma \vdash c}{\Gamma, A^{\dagger} \vdash c}$$

Linear Logic

- Introduced by Jean-Yves Girard in 1986 as a new *constructive* logic without the asymmetry of intuitionistic logic (sequent calculus with symmetric left and right sides)
- Logic of resource consumption

$$A \otimes A \nvdash_{LL} A$$

$$A \otimes (A \multimap B) \vdash_{LL} B$$

$$A \otimes (A \multimap B) \nvdash_{LL} A \otimes B$$

- $!A$ provides arbitrary duplication (unbounded throwable resource)

$$!A \otimes (A \multimap B) \vdash_{LL} !A \otimes B \vdash_{LL} B$$

- Sequent calculus without weakening and contraction

Intuitionistic Linear Logic

Multiplicatives

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \qquad \frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Delta, \Gamma, A \multimap B \vdash C} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}$$

Additives

$$\frac{\Gamma, A \vdash C}{\Gamma, A \,\&\, B \vdash C} \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \,\&\, B \vdash C} \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}$$

$$\frac{\Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma, A \oplus B \vdash C} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}$$

Constants

$$\frac{\Gamma \vdash A}{\Gamma, \mathbf{1} \vdash A} \qquad \vdash \mathbf{1} \qquad \bot \vdash \qquad \frac{\Gamma \vdash}{\Gamma \vdash \bot} \qquad \Gamma \vdash \top \qquad \Gamma, \mathbf{0} \vdash A$$

Intuitionistic Linear Logic (cont.)

Axiom - Cut

$$A \vdash A \qquad \frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Delta, \Gamma \vdash B}$$

Bang

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \qquad \frac{\Gamma \vdash B}{\Gamma, !A \vdash B} \qquad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}$$

Quantifiers

$$\frac{\Gamma, A[t/x] \vdash B}{\Gamma, \forall x A \vdash B} \qquad \frac{\Gamma \vdash A}{\Gamma \vdash \forall x A}\ x \notin fv(\Gamma)$$

$$\frac{\Gamma, A \vdash B}{\Gamma, \exists x A \vdash B}\ x \notin fv(\Gamma, B) \qquad \frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x A}$$

ILL = the Logic of CC agents

Translation:

$(A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \qquad (c \to A)^\dagger =$

ILL = the Logic of CC agents

Translation:

$(A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \qquad (c \rightarrow A)^\dagger = c \multimap A^\dagger \qquad \mathit{tell}(c)^\dagger =$

ILL = the Logic of CC agents

Translation:

$(A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \qquad (c \to A)^\dagger = c \multimap A^\dagger \qquad \mathit{tell}(c)^\dagger = !c$

$(A + B)^\dagger =$

ILL = the Logic of CC agents

Translation:

$$(A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \qquad (c \to A)^\dagger = c \multimap A^\dagger \qquad \mathit{tell}(c)^\dagger = !c$$

$$(A + B)^\dagger = A^\dagger \,\&\, B^\dagger \qquad (\exists x A)^\dagger = \exists x A^\dagger \qquad p(\vec{x})^\dagger = p(\vec{x})$$

$$(X; c; \Gamma)^\dagger = \exists X(!c \otimes \Gamma^\dagger)$$

Axioms: $!c \vdash !d$ for all $c \vdash_{\mathcal{C}} d$ $\qquad$ $p(\vec{x}) \vdash A^\dagger$ for all $p(\vec{x}) = A \in \mathcal{D}$

ILL = the Logic of CC agents

Translation:

$$(A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \qquad (c \to A)^\dagger = c \multimap A^\dagger \qquad \mathit{tell}(c)^\dagger = !c$$
$$(A + B)^\dagger = A^\dagger \,\&\, B^\dagger \qquad (\exists x A)^\dagger = \exists x A^\dagger \qquad p(\vec{x})^\dagger = p(\vec{x})$$
$$(X; c; \Gamma)^\dagger = \exists X(!c \otimes \Gamma^\dagger)$$

Axioms: $!c \vdash !d$ for all $c \vdash_{\mathcal{C}} d$ $\qquad$ $p(\vec{x}) \vdash A^\dagger$ for all $p(\vec{x}) = A \in \mathcal{D}$

Soundness and Completeness

If $(c; \Gamma) \longrightarrow_{CC} (d; \Delta)$ then $c^\dagger \otimes \Gamma^\dagger \vdash_{ILL(\mathcal{C},\mathcal{D})} d^\dagger \otimes \Delta^\dagger$

If $A^\dagger \vdash_{ILL(\mathcal{C},\mathcal{D})} c$ then *there exists a success store* d such that $(\mathit{true}; A) \longrightarrow_{CC} (d; \emptyset)$ and $d \vdash_{\mathcal{C}} c$

If $A^\dagger \vdash_{ILL(\mathcal{C},\mathcal{D})} c \otimes \top$ then *there exists an accessible store* d such that $(\mathit{true}; A) \longrightarrow_{CC} (d; \Gamma)$ and $d \vdash_{\mathcal{C}} c$

Soundness

Theorem 3 (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.

Soundness

Theorem 3 (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 case on $\equiv$, immediate.

Soundness

Theorem 3 (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 case on $\equiv$, immediate.
By case on $\longrightarrow$

Soundness

Theorem 3 (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 case on $\equiv$, immediate.
By case 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 4 (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.

Completeness I

Theorem 4 (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 5 (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 induction for the right introduction of the tensor connective in $c \otimes \top$. □

Observing "must" Properties

Properties true on all branches on the derivation tree.
Redefine the operational semantics by a rewriting relation on frontiers, i.e., multisets of configurations

Blind choice

$$\langle (X; c; A + B), \Phi \rangle \rightsquigarrow \langle (X; c; A), (X; c; B), \Phi \rangle$$

Tell

$$\langle (X; c; tell(d), \Gamma), \Phi \rangle \rightsquigarrow \langle (X; c \wedge d; \Gamma), \Phi \rangle$$

Ask

$$\frac{c \vdash_{\mathcal{C}} d}{\langle (X; c; d \rightarrow A, \Gamma), \Phi \rangle \rightsquigarrow \langle (X; c; A, \Gamma), \Phi \rangle}$$

Procedure calls

$$\frac{(p(\vec{y}) = A) \in \mathcal{D}}{\langle (X; c; p(\vec{y}), \Gamma), \Phi \rangle \rightsquigarrow \langle (X; c; A, \Gamma), \Phi \rangle}$$

Translating the Frontier Calculus in LL with

Translate

$$(A + B)^{\ddagger} =$$

$$\langle (X; c; A), \Phi \rangle^{\ddagger} =$$

same translation for the other operations

Theorem 6 (Soundness of transitions)

Let Φ and Ψ be two frontiers.
If $\Phi \equiv \Psi$ then $(\Phi)^{\ddagger} \dashv\vdash_{ILL(\mathcal{C},\mathcal{D})} (\Psi)^{\ddagger}$.
If $\Phi \rightsquigarrow \Psi$ then $\Phi^{\ddagger} \vdash_{ILL(\mathcal{C},\mathcal{D})} \Psi^{\ddagger}$.

Translating the Frontier Calculus in LL with $\oplus$

Translate

$$(A + B)^{\ddagger} = A^{\ddagger} \oplus B^{\ddagger}$$

$$\langle (X; c; A), \Phi \rangle^{\ddagger} = \exists X (c^{\ddagger} \otimes A^{\ddagger}) \oplus \Phi^{\ddagger}$$

same translation for the other operations

Theorem 6 (Soundness of transitions)

Let Φ and Ψ be two frontiers.
If $\Phi \equiv \Psi$ then $(\Phi)^{\ddagger} \dashv\vdash_{ILL(\mathcal{C},\mathcal{D})} (\Psi)^{\ddagger}$.
If $\Phi \rightsquigarrow \Psi$ then $\Phi^{\ddagger} \vdash_{ILL(\mathcal{C},\mathcal{D})} \Psi^{\ddagger}$.

Completeness III for “must” Properties

Theorem 7 (Observation of frontiers’ accessible stores)

Let A be a CC agent and c be a constraint.
If $A^{\ddagger} \vdash_{ILL(\mathcal{C},\mathcal{D})} c \otimes \top$
then $\langle(\emptyset; 1; A)\rangle \rightsquigarrow \langle(X_1; d_1; \Gamma_1), \ldots, (X_n; d_n; \Gamma_n)\rangle$ with $\forall j\ \exists X_j d_j \vdash_{\mathcal{C}} c$

Theorem 8 (Observation of frontiers’ success stores)

Let A be an CC agent and c be a constraint.
If $A^{\ddagger} \vdash_{ILL(\mathcal{C},\mathcal{D})} c$
then $\langle(\emptyset; 1; A)\rangle \rightsquigarrow \langle(X_1; d_1; \emptyset), \ldots, (X_n; d_n; \emptyset)\rangle$ with $\forall j\ \exists X_j d_j \vdash_{\mathcal{C}} c$

Logical Equivalence of CC programs

Let P and P' be two CC($\mathcal{C}$) processes

Corollary 9

If $P^{\dagger} \dashv\vdash_{ILL(\mathcal{C},\mathcal{D},\mathcal{D}')} P'^{\dagger}$
then $\downarrow \mathcal{O}_{ss}(P) = \downarrow \mathcal{O}_{ss}(P')$ *(same set of success stores)*
and $\downarrow \mathcal{O}_{as}(P) = \downarrow \mathcal{O}_{as}(P')$ *(same set of accessible stores).*

Corollary 10

If $P^{\ddagger} \dashv\vdash_{ILL(\mathcal{C},\mathcal{D},\mathcal{D}')} P'^{\ddagger}$
then P *and* P' *have the same set of accessible stores on all branches*
and the same success frontiers.

Proving Properties of CC Programs

- Proving *logical equivalence* of CC programs with the sequent calculus of LL:
  - focusing proofs (deterministic rules for the additives first)
  - lazy splitting (input/output contexts for the multiplicatives)

- Proving *safety properties* of CC programs with the *phase semantics* of LL [FRS98lics]
  Soundness gives $\Gamma \vdash_{ILL} A$ implies $\forall \mathbf{P} \forall \eta\ \mathbf{P}, \eta \models (\Gamma \vdash A)$.
  $\exists \mathbf{P}, \eta,$ s.t. $\mathbf{P}, \eta \not\models (\Gamma \vdash A)$ implies $\Gamma \not\vdash_{ILL_{\mathcal{C},\mathcal{D}}} A$.

Proposition 11

To prove a safety property $(c, A) \not\longrightarrow (d, B)$, it is enough to show that $\exists$ a phase space **P**, *a valuation η , and an element $a \in \eta((c, A)^\dagger)$ such that $a \notin \eta((d, B)^\dagger)$.*

Implementations of LL Sequent Calculi

- Forum [Miller&al.] specification languages based on LL
- LO [Andreoli] Property of "focusing proofs" in LL
- Lolli [Cervesato Hodas Pfenning] Search for "Uniform proofs"
- Lygon [Harland Winikoff] Linear Logic Programming language

Problem of lazy splitting:

$$\frac{\vdash A, \Gamma \quad \vdash B, \Delta}{\vdash A \otimes B, \Gamma, \Delta}(\otimes)$$

First idea:

$$\frac{\vdash A - (\Gamma, \Delta); \Delta \quad \vdash B, \Delta}{\vdash A \otimes B, \Gamma, \Delta}(\otimes)$$

- problems with the rules for ! and for $\top$...
- stacks are necessary

Part XII

LCC

Part XII: LCC

Linear Constraint Systems $(\mathcal{C}, \vdash_\mathcal{C})$

$\mathcal{C}$ is a set of formulas built from V, Σ with logical operators: 1, $\otimes$, $\exists$ and !;

$\Vdash_\mathcal{C} \subset \mathcal{C} \times \mathcal{C}$ defines the non-logical axioms of the constraint system.

$\vdash_\mathcal{C}$ is the least subset of $\mathcal{C}^\star \times \mathcal{C}$ containing $\Vdash_\mathcal{C}$ and closed by:

$$c \vdash c \qquad \frac{\Gamma, c \vdash d \quad \Delta \vdash c}{\Gamma, \Delta \vdash d} \qquad \vdash 1 \qquad \frac{\Gamma \vdash c}{\Gamma, 1 \vdash c}$$

$$\frac{\Gamma \vdash c_1 \quad \Delta \vdash c_2}{\Gamma, \Delta \vdash c_1 \otimes c_2} \qquad \frac{\Gamma, c_1, c_2 \vdash c}{\Gamma, c_1 \otimes c_2 \vdash c} \qquad \frac{\Gamma \vdash c[t/x]}{\Gamma \vdash \exists x\, c} \qquad \frac{\Gamma, c \vdash d}{\Gamma, \exists x\, c \vdash d}\ x \notin fv(\Gamma, d)$$

$$\frac{\Gamma, c \vdash d}{\Gamma, !c \vdash d} \qquad \frac{!\Gamma \vdash d}{!\Gamma \vdash !d} \qquad \frac{\Gamma \vdash d}{\Gamma, !c \vdash d} \qquad \frac{\Gamma, !c, !c \vdash d}{\Gamma, !c \vdash d}$$

A synchronization constraint is a constraint not appearing in $\Vdash_\mathcal{C}$

Same agents and observables as CC

Processes	$P ::= \mathcal{D}.A$
Declarations	$\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$
Agents	$A ::= tell(c) \mid \forall\vec{x}(c \to A) \mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$

- observing the set of success stores,
- observing the set of terminal stores (successes and suspensions),
- observing the set of accessible stores,

Same agents and observables as CC

Processes	$P ::= \mathcal{D}.A$
Declarations	$\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$
Agents	$A ::= tell(c) \mid \forall \vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$

- observing the set of success stores,

  $$\mathcal{O}_{ss}(\mathcal{D}.A; c) = \{\exists \vec{x} d \in \mathcal{C} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \epsilon)\}$$

- observing the set of terminal stores (successes and suspensions),

- observing the set of accessible stores,

Same agents and observables as CC

Processes	$P ::= \mathcal{D}.A$
Declarations	$\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$
Agents	$A ::= tell(c) \mid \forall\vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$

- observing the set of success stores,

  $$\mathcal{O}_{ss}(\mathcal{D}.A; c) = \{\exists\vec{x}d \in \mathcal{C} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \epsilon)\}$$

- observing the set of terminal stores (successes and suspensions),

  $$\mathcal{O}_{ts}(\mathcal{D}.A; c) = \{\exists\vec{x}d \in \mathcal{C} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \Gamma) \not\longrightarrow\}$$

- observing the set of accessible stores,

Same agents and observables as CC

Processes	$P ::= \mathcal{D}.A$
Declarations	$\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$
Agents	$A ::= tell(c) \mid \forall \vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$

- observing the set of success stores,

  $$\mathcal{O}_{ss}(\mathcal{D}.A; c) = \{\exists \vec{x} d \in \mathcal{C} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \epsilon)\}$$

- observing the set of terminal stores (successes and suspensions),

  $$\mathcal{O}_{ts}(\mathcal{D}.A; c) = \{\exists \vec{x} d \in \mathcal{C} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \Gamma) \not\longrightarrow\}$$

- observing the set of accessible stores,

  $$\mathcal{O}_{as}(\mathcal{D}.A; c) = \{\exists \vec{x} d \in \mathcal{C} \mid (\emptyset; c; A) \longrightarrow^* (\vec{x}; d; \Gamma)\}$$

Linear-CC($\mathcal{C}$) Transitions

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

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

```
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) =
```

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)) →
```

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) ∥
```

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

$$
\begin{array}{l}
\texttt{Goal(N) = RecPhil(1,N).} \\
\texttt{RecPhil(M,P) =} \\
\quad \texttt{M} \neq \texttt{P} \rightarrow \texttt{(Philo(M,P)} \parallel \texttt{fork(M)} \parallel \texttt{RecPhil(M+1,P))} \\
\parallel \\
\quad \texttt{M = P} \rightarrow \texttt{(Philo(M,P)} \parallel \texttt{fork(M)).} \\
\texttt{Philo(I,N) =} \\
\quad \texttt{(fork(I)} \otimes \texttt{fork(I+1 mod N))} \rightarrow \\
\qquad \texttt{(eat(I)} \parallel \\
\qquad \texttt{eat(I)} \rightarrow \\
\qquad\quad \texttt{(fork(I)} \parallel \texttt{fork(I+1 mod N)} \parallel \texttt{Philo(I,N))).}
\end{array}
$$

Safety properties: deadlock freeness, two neighbors don't eat at the same time, etc.

Encoding Linda in LCC($\mathcal{H}$)

- Shared tuple space
- Asynchronous communication (through tuple space)
- *input* consumes the tuple, *read* doesn't
- One-step guarded choice
- Conditional with else case (check the absence of tuple) not encodable in LCC

Encoding Linda in LCC($\mathcal{H}$)

- Shared tuple space
- Asynchronous communication (through tuple space)
- *input* consumes the tuple, *read* doesn't
- One-step guarded choice
- Conditional with else case (check the absence of tuple) not encodable in LCC transitions are still monotonic!

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

- Direct encoding of the asynchronous π-calculus:

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

- 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:

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

- 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:

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

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

Producer Consumer Protocol in LCC

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

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

Number of units consumed always $<$ number of units produced:

$\mathtt{P} = \mathtt{dem} \rightarrow (\mathtt{pro} \parallel \mathtt{P} \parallel$
$\quad \forall \mathtt{X}\ (\mathtt{count(np,X)} \rightarrow \mathtt{count(np,X+1)}))$
$\mathtt{C} = \mathtt{pro} \rightarrow (\mathtt{dem} \parallel \mathtt{C} \parallel$
$\quad \forall \mathtt{X}\ (\mathtt{count(nc,X)} \rightarrow \mathtt{count(nc,X+1)}))$
$\mathtt{init} = \mathtt{dem}^{n} \parallel \mathtt{P}^{m} \parallel \mathtt{C}^{k} \parallel \mathtt{np=0} \parallel \mathtt{nc=0}$
$\mathtt{init} \not\longrightarrow_{LCC} \mathtt{dem}^{n'} \parallel \mathtt{pro}^{l'} \parallel \mathtt{P}^{m} \parallel \mathtt{C}^{k} \parallel \mathtt{np=np_0} \parallel \mathtt{nc=nc_0}$
with $\mathtt{nc_0} > \mathtt{np_0}$