

Concurrent Constraint Programming

Part 3: Linear Logic Semantics

François Fages
INRIA Rocquencourt,
Francois.Fages@inria.fr

Logical Semantics of CC?

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

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

(reverses the arrow of CLP interpretation...)

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

$$lW \frac{\Gamma \vdash_{\mathcal{X}} c}{\Gamma, A^\dagger \vdash c}$$

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*

$$A \otimes A \not\vdash_{LL} A$$

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

$$A \otimes (A \multimap B) \not\vdash_{LL} A \otimes B$$

- $!A$ 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

Axiom - Cut

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

Constants

$$\frac{\Gamma \vdash A}{\Gamma, \mathbf{1} \vdash A} \quad \vdash \mathbf{1} \quad \perp \vdash \quad \frac{\Gamma \vdash}{\Gamma \vdash \perp} \quad \Gamma \vdash \top \quad \Gamma, \mathbf{0} \vdash A$$

Multiplicatives

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

Additives

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

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

Intuitionistic Linear Logic (cont.)

Bang

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

Quantifiers

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

Intuit. Linear Logic = the Logic of CC agents

$$\begin{array}{ll}
 \text{Translation: } (c \rightarrow A)^\dagger = c \multimap A^\dagger & (A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \\
 (A + B)^\dagger = A^\dagger \& B^\dagger & (\exists x A)^\dagger = \exists x A^\dagger \\
 \text{tell}(c)^\dagger = !c & p(\vec{x})^\dagger = p(\vec{x})
 \end{array}$$

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

Soundness and Completeness

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

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

If $A^\dagger \vdash_{LL(\mathcal{X}, \mathcal{D})} c \otimes \top$ alors *there exists an accessible store* d such that $(\text{true}; A) \longrightarrow_{CC} (d; \Gamma)$ and $d \vdash_{\mathcal{X}} c \otimes \top$.

Soundness

Theorem 1 (Soundness of transitions)

Let $(\vec{x}; c; \Gamma)$ and $(\vec{y}; d; \Delta)$ be LCC configurations.

If $(\vec{x}; c; \Gamma) \equiv (\vec{y}; d; \Delta)$ then $(\vec{x}; c; \Gamma)^\dagger \dashv\vdash_{ILL(\mathcal{C}, \mathcal{D})} (\vec{y}; d; \Delta)^\dagger$.

If $(\vec{x}; c; \Gamma) \longrightarrow (\vec{y}; d; \Delta)$ then $(\vec{x}; c; \Gamma)^\dagger \vdash_{ILL(\mathcal{C}, \mathcal{D})} (\vec{y}; d; \Delta)^\dagger$.

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 2 (Observation of successes)

Let A be an agent LCC and c be a linear constraint.

If $A^\dagger \vdash_{ILL(\mathcal{C}, \mathcal{D})} c$, then there exists a constraint d such that

$(\emptyset; 1; A) \longrightarrow (\vec{x}; d; stop)$ and $\exists \vec{x}d \vdash_{\mathcal{C}} c$.

Proof.

By induction on a sequent calculus proof π of $A_1^\dagger, \dots, A_n^\dagger \vdash_{ILL(\mathcal{C}, \mathcal{D})} \phi$,

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 3 (Observation of accessible stores)

Let A be an LCC agent and c be a linear 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

$(\emptyset; 1; A) \longrightarrow (\emptyset; d; \Gamma)$ and $d \vdash_{\mathcal{C}} c \otimes \top$.

Proof. The proof uses theorem 2, and proceeds by an easy 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 $\langle (\vec{x}; c; A + B), \Phi \rangle \Longrightarrow \langle (\vec{x}; c; A), (\vec{x}; c; B), \Phi \rangle$

Tell $\langle (\vec{x}; c; \text{tell}(d), \Gamma), \Phi \rangle \Longrightarrow_{LCC} \langle (\vec{x}; c \wedge d; \Gamma), \Phi \rangle$

Ask
$$\frac{c \vdash_c d \otimes e}{\langle (\vec{x}; c; e \rightarrow A, \Gamma), \Phi \rangle \Longrightarrow_{LCC} \langle (\vec{x}; d; A, \Gamma), \Phi \rangle}$$

Procedure calls
$$\frac{(p(\vec{y}) = A) \in D}{\langle (\vec{x}; c; p(\vec{y}), \Gamma), \Phi \rangle \Longrightarrow_{LCC} \langle (\vec{x}; c; A, \Gamma), \Phi \rangle}$$

Translating the Frontier Calculus in LL with \oplus

Translate $(A + B)^\dagger = A^\dagger \oplus B^\dagger$

$$\langle (\vec{x}; c; A), \Phi \rangle^\dagger = \exists \vec{x} (c^\dagger \otimes A^\dagger) \oplus \Phi^\dagger$$

same translation for the other operations

Theorem 4 (Soundness of transitions)

Let Φ and Ψ be two frontiers.

If $\Phi \equiv \Psi$ then $(\Phi)^\dagger \dashv\vdash_{ILL(c, \mathcal{D})} (\Psi)^\dagger$.

If $\Phi \Longrightarrow \Psi$ then $\Phi^\dagger \vdash_{ILL(c, \mathcal{D})} \Psi^\dagger$.

Completeness III for “must” properties

Theorem 5 (Observation of frontiers’ accessible stores)

Let A be an LCC agent and c be a linear constraint.

If $A^\ddagger \vdash_{ILL(\mathcal{C}, \mathcal{D})} c \otimes \top$

then $\langle (\emptyset; 1; A) \rangle \Longrightarrow \langle (\vec{x}_1; d_1; \Gamma_1), \dots, (\vec{x}_n; d_n; \Gamma_n) \rangle$ with $\forall j \exists \vec{x}_j d_j \vdash_{\mathcal{C}} c \otimes \top$

Theorem 6 (Observation of frontiers’ success stores)

Let A be an LCC agent and c be a linear constraint.

If $A^\ddagger \vdash_{ILL(\mathcal{C}, \mathcal{D})} c$

then $\langle (\emptyset; 1; A) \rangle \Longrightarrow \langle (\vec{x}_1; d_1; \emptyset), \dots, (\vec{x}_n; d_n; \emptyset) \rangle$ with $\forall j \exists \vec{x}_j d_j \vdash_{\mathcal{C}} c$

Logical Equivalence of CC programs

Let $P = D.A$ be a $\text{CC}(\mathcal{X})$ process.

Corollary 7

If $P^\dagger \dashv\vdash_{ILL(\mathcal{X}, \mathcal{D})} P'^\dagger$

then $\mathcal{O}(P) = \mathcal{O}(P')$ (same set of success stores)

and $\mathcal{O}_{as}(P) = \mathcal{O}_{as}(P')$ (same set of accessible stores).

Corollary 8

If $P^\ddagger \dashv\vdash_{ILL(\mathcal{X}, \mathcal{D})} P'^\ddagger$

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 [FRS,LICS'98]

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

$\exists P, \eta, s.t. P, \eta \not\models (\Gamma \vdash A)$ implies $\Gamma \not\vdash_{ILL_{c,D}} A$.

Corollary 9 *To prove a safety property $(c, A) \dashrightarrow (d, B)$, it is enough to show that \exists a phase space \mathbf{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 [Andreoli91] Property of “focusing proofs” in LL

Lolli [Cervesato Hodas Pfenning] Search for “Uniform proofs”

Lygon [Harland Winikoff 91] 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 (&)
- stacks are necessary

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

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

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

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

$$\begin{array}{c}
 c \vdash c \quad \frac{\Gamma, c \vdash d \quad \Delta \vdash c}{\Gamma, \Delta \vdash d} \quad \vdash 1 \quad \frac{\Gamma \vdash c}{\Gamma, 1 \vdash c} \\
 \\
 \frac{\Gamma, c_1, c_2 \vdash c}{\Gamma, c_1 \otimes c_2 \vdash c} \quad \frac{\Gamma \vdash c_1 \quad \Delta \vdash c_2}{\Gamma, \Delta \vdash c_1 \otimes c_2} \quad \frac{\Gamma \vdash c}{\Gamma \vdash \exists x c} \quad \frac{\Gamma, c \vdash d}{\Gamma, \exists x c \vdash d} \quad x \notin fv(\Gamma, d) \\
 \\
 \frac{\Gamma, c \vdash d}{\Gamma, !c \vdash d} \quad \frac{!\Gamma \vdash d}{!\Gamma \vdash !d} \quad \frac{\Gamma \vdash d}{\Gamma, !c \vdash d} \quad \frac{\Gamma, !c, !c \vdash d}{\Gamma, !c \vdash d}
 \end{array}$$

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

Linear-CC(\mathcal{X}) Operational Semantics

$$\text{Congruence} \quad \frac{(\vec{x}; c; \Gamma) \equiv (\vec{x}'; c'; \Gamma') \longrightarrow (\vec{y}'; d'; \Delta') \equiv (\vec{y}; d; \Delta)}{(\vec{x}; c; \Gamma) \longrightarrow (\vec{y}; d; \Delta)}$$

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

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

$$\text{Ask} \quad \frac{c \vdash_c d \otimes e}{(\vec{x}; c; e \rightarrow A, \Gamma) \longrightarrow (\vec{x}; d; A, \Gamma)}$$

$$\text{Blind choice} \quad \begin{aligned} &(\vec{x}; c; A + B, \Gamma) \longrightarrow (\vec{x}; c; A, \Gamma) \\ &(\vec{x}; c; A + B, \Gamma) \longrightarrow (\vec{x}; c; B, \Gamma) \end{aligned}$$

An LCC(N) program for the dining philosophers

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

RecPhil(M,P) =

$M \neq P \rightarrow (\text{Philo}(M,P) \parallel \text{fork}(M) \parallel \text{RecPhil}(M+1,P)) \parallel$

$M = P \rightarrow (\text{Philo}(M,P) \parallel \text{fork}(M)).$

Philo(I,N) =

$(\text{fork}(I) \otimes \text{fork}(I+1 \bmod N)) \rightarrow$

$(\text{eat}(I) \parallel$

$\text{eat}(I) \rightarrow (\text{fork}(I) \parallel \text{fork}(I+1 \bmod N) \parallel \text{Philo}(I,N))).$

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

Producer Consumer Protocol in LCC

$P = \text{dem} \rightarrow (\text{pro} \parallel P)$

$C = \text{pro} \rightarrow (\text{dem} \parallel C)$

$\text{init} = \text{dem}^n \parallel P^m \parallel C^k$

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

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

$P = \text{dem} \rightarrow (\text{pro} \parallel P \parallel \forall X (\text{np}=X \rightarrow \text{np}=X+1))$

$C = \text{pro} \rightarrow (\text{dem} \parallel C \parallel \forall X (\text{nc}=X \rightarrow \text{nc}=X+1))$

$\text{init} = \text{dem}^n \parallel P^m \parallel C^k \parallel \text{np}=0 \parallel \text{nc}=0$

$\text{init} \not\rightarrow_{LCC} \text{dem}^{n'} \parallel \text{pro}^{l'} \parallel P^m \parallel C^k \parallel \text{np} = \text{np}_0 \parallel \text{nc} = \text{nc}_0$

with $\text{nc}_0 > \text{np}_0$

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 the π -calculus in LCC(\mathcal{H})

- Direct encoding of the asynchronous π -calculus:

$$\begin{aligned} [0] &= 1 \\ [(y)P] &= \exists y[P] \\ [\bar{x}y.0] &= \text{tell}(c(x, y)) \\ [x(y).P] &= \forall yc(x, y) \rightarrow [P] \\ [P|Q] &= [P]||[Q] \\ [[x = y]P] &= (x = y) \rightarrow [P] \\ [P + Q] &= [P] + [Q] \end{aligned}$$

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