

# Constraint Logic Programming

Sylvain Soliman

Sylvain.Soliman@inria.fr



Project-Team CONTRAINTEs

MPRI 2.35.1 Course – December 2011 - February 2012

# Part I: CLP - Introduction and Logical Background

- 1 The Constraint Programming paradigm
- 2 Examples and Applications
- 3 First Order Logic
- 4 Models
- 5 Logical Theories

# Part II: Constraint Logic Programs

6 Constraint Languages

7  $\text{CLP}(\mathcal{X})$

8  $\text{CLP}(\mathcal{H})$

9  $\text{CLP}(\mathcal{R}, \mathcal{FD}, \mathcal{B})$

# Part III: CLP - Operational and Fixpoint Semantics

10 Operational Semantics

11 Fixpoint Semantics

12 Program Analysis

# Part IV: Logical Semantics

13 Logical Semantics of CLP( $\mathcal{X}$ )

14 Automated Deduction

15 CLP( $\lambda$ )

16 Negation as Failure

# Part V: Constraint Solving

17 Solving by Rewriting

18 Solving by Domain Reduction

# Part VI: Practical CLP Programming

- 19 CLP implementation, the WAM
- 20 Optimizing CLP
- 21 Symmetries
- 22 Symmetry Breaking During Search
- 23 Detecting Symmetries

# Part VII: More Constraint Programming

24 Typing CLP

25 CHR

# Part VIII: Programming Project

26 check\_dice

27 dice

28 Optimizing

29 Theory

# Part IX: Concurrent Constraint Programming

30 Introduction

31 Operational Semantics

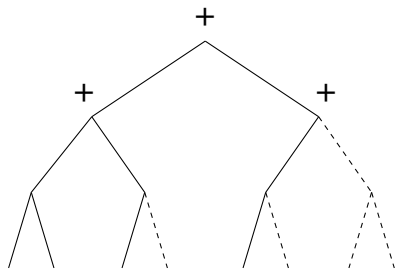
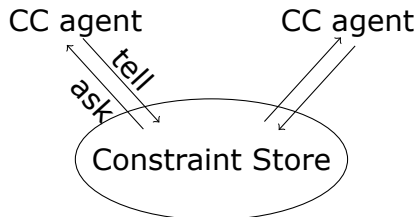
32 Examples

# Concurrent Constraint Programs

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

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

$\mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$



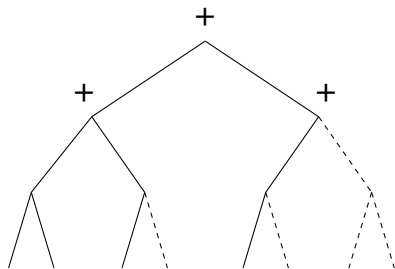
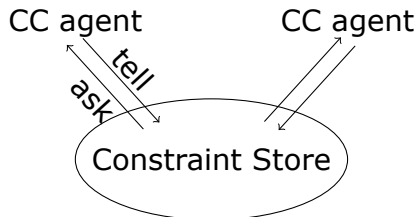
# Concurrent Constraint Programs

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

**Processes**  $P ::= D.A$

**Declarations**  $\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$

**Agents**  $A ::= tell(c) \mid \forall \vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists xA \mid p(\vec{x})$



# CC( $\mathcal{X}$ ) Transitions

Interleaving semantics

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

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

**Ask**

$$\begin{array}{l} \text{Blind choice} \quad (\vec{x}; c; A + B, \Gamma) \longrightarrow (\vec{x}; c; A, \Gamma) \\ \text{(local/internal)} \quad (\vec{x}; c; A + B, \Gamma) \longrightarrow (\vec{x}; c; B, \Gamma) \end{array}$$

# CC( $\mathcal{X}$ ) Transitions

Interleaving semantics

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

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

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

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

# 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),

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

- 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**?

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

# Part X: CC - Denotational Semantics

33 Deterministic Case

34 Constraint Propagation

35 Non-deterministic Case

36 Sequentiality

# 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  $\uparrow c = \{d \in \mathcal{C} \mid c \leq d\}$ .

$\llbracket \mathcal{D}.A \rrbracket : \mathcal{C} \rightarrow \mathcal{C}$  is

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

i.e.,  $\llbracket \mathcal{D}.A \rrbracket$  is a **closure operator** over  $(\mathcal{C}, \leq)$ .

# Semantic Equations

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

$$\llbracket \mathcal{D}.tell(c) \rrbracket = \uparrow c \quad (\simeq \lambda s. s \wedge c)$$

$$\begin{aligned} \llbracket \mathcal{D}.c \rightarrow \mathcal{A} \rrbracket &= (\mathcal{C} \setminus \uparrow c) \cup (\uparrow c \cap \llbracket \mathcal{D}.\mathcal{A} \rrbracket) \\ &(\simeq \lambda s. \text{if } s \vdash_{\mathcal{C}} c \text{ then } \llbracket \mathcal{D}.\mathcal{A} \rrbracket s \text{ else } s) \end{aligned}$$

$$\llbracket \mathcal{D}.\mathcal{A} \parallel \mathcal{B} \rrbracket = \llbracket \mathcal{D}.\mathcal{A} \rrbracket \cap \llbracket \mathcal{D}.\mathcal{B} \rrbracket \quad (\simeq Y(\lambda s. \llbracket \mathcal{D}.\mathcal{A} \rrbracket \llbracket \mathcal{D}.\mathcal{B} \rrbracket s))$$

$$\llbracket \mathcal{D}.\exists x \mathcal{A} \rrbracket = \{d \mid c \in \llbracket \mathcal{D}.\mathcal{A} \rrbracket, \exists x c = \exists x d\} \quad (\simeq \lambda s. \exists x \llbracket \mathcal{D}.\mathcal{A} \rrbracket \exists x s)$$

$$\llbracket \mathcal{D}.p(\vec{x}) \rrbracket = \llbracket \mathcal{D}.\mathcal{A}[\vec{x}/\vec{y}] \rrbracket \text{ if } p(\vec{y}) = \mathcal{A} \in \mathcal{D} \quad (\simeq \lambda s. \llbracket \mathcal{D}.\mathcal{A}[\vec{x}/\vec{y}] \rrbracket s)$$

## Theorem 1 ([SRP91popl])

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

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

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

Let  $\llbracket \cdot \rrbracket : \mathcal{D} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{P}(\mathcal{C}))$  be the least fixpoint (for  $\subset$ ) of

$$\begin{aligned}\llbracket \mathcal{D}.c \rrbracket &= \{\uparrow c\} \\ \llbracket \mathcal{D}.c \rightarrow \mathcal{A} \rrbracket &= \{\mathcal{C} \setminus \uparrow c\} \cup \{\uparrow c \cap X \mid X \in \llbracket \mathcal{D}.\mathcal{A} \rrbracket\} \\ \llbracket \mathcal{D}.\mathcal{A} \parallel \mathcal{B} \rrbracket &= \{X \cap Y \mid X \in \llbracket \mathcal{D}.\mathcal{A} \rrbracket, Y \in \llbracket \mathcal{D}.\mathcal{B} \rrbracket\} \\ \llbracket \mathcal{D}.\mathcal{A} + \mathcal{B} \rrbracket &= \llbracket \mathcal{D}.\mathcal{A} \rrbracket \cup \llbracket \mathcal{D}.\mathcal{B} \rrbracket \\ \llbracket \mathcal{D}.\exists x \mathcal{A} \rrbracket &= \{\{d \mid \exists xc = \exists xd, c \in X\} \mid X \in \llbracket \mathcal{D}.\mathcal{A} \rrbracket\} \\ \llbracket \mathcal{D}.p(\vec{x}) \rrbracket &= \llbracket \mathcal{D}.\mathcal{A}[\vec{x}/\vec{y}] \rrbracket\end{aligned}$$

### Theorem 2 ([FGMP97tcs])

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

$$\mathcal{O}_{ts}(\mathcal{D}.\mathcal{A}; c) = \{d \mid \text{there exists } X \in \llbracket \mathcal{D}.\mathcal{A} \rrbracket \text{ s.t. } d = \min(\uparrow c \cap X)\}.$$

# Sequentiality

Let us define a new operator,  $\bullet$ , as follows:

$$\frac{(X; c; A) \longrightarrow (Y; d; B)}{(X; c; A \bullet C, \Gamma) \longrightarrow (Y; d; B \bullet C, \Gamma)} \quad (X; c; \emptyset \bullet A) \longrightarrow (X; c; A)$$

We can characterize completely the observables of any  $CC_{seq}$  program,  $\mathcal{D}.A$ , by those of a new CC (without  $\bullet$ ) program,  $\mathcal{D}^\bullet.A^\bullet$ , in a new constraint system,  $\mathcal{C}^\bullet$ .

## Idea

Let  $ok$  be a **new** relation symbol of arity one.  $\mathcal{C}^\bullet$  is the constraint system  $\mathcal{C}$  to which  $ok$  is added, without any non-logical axiom. The program  $\mathcal{D}^\bullet.A^\bullet$  is defined inductively as follows:

$$(p(\vec{y}) = A)^\bullet = p^\bullet(x, \vec{y}) = A_x^\bullet$$

$$A^\bullet = \exists x A_x^\bullet$$

$$tell(c)_x^\bullet = tell(c \wedge ok(x))$$

$$p(\vec{y})_x^\bullet = p^\bullet(x, \vec{y})$$

$$(A \parallel B)_x^\bullet = \exists y, z (A_y^\bullet \parallel B_z^\bullet \parallel (ok(y) \wedge ok(z)) \rightarrow ok(x))$$

$$(A + B)_x^\bullet = A_x^\bullet + B_x^\bullet$$

$$(\forall \vec{y} (c \rightarrow A))_x^\bullet = \forall \vec{z} (c[\vec{z}/\vec{y}] \rightarrow A[\vec{z}/\vec{y}]_x^\bullet) \text{ with } x \notin \vec{z}$$

$$(\exists y A)_x^\bullet = \exists z A[z/y]_x^\bullet \text{ with } z \neq x$$

$$(A \bullet B)_x^\bullet =$$

## Idea

Let  $ok$  be a **new** relation symbol of arity one.  $\mathcal{C}^\bullet$  is the constraint system  $\mathcal{C}$  to which  $ok$  is added, without any non-logical axiom. The program  $\mathcal{D}^\bullet.A^\bullet$  is defined inductively as follows:

$$(p(\vec{y}) = A)^\bullet = p^\bullet(x, \vec{y}) = A_x^\bullet$$

$$A^\bullet = \exists x A_x^\bullet$$

$$tell(c)_x^\bullet = tell(c \wedge ok(x))$$

$$p(\vec{y})_x^\bullet = p^\bullet(x, \vec{y})$$

$$(A \parallel B)_x^\bullet = \exists y, z (A_y^\bullet \parallel B_z^\bullet \parallel (ok(y) \wedge ok(z)) \rightarrow ok(x))$$

$$(A + B)_x^\bullet = A_x^\bullet + B_x^\bullet$$

$$(\forall \vec{y}(c \rightarrow A))_x^\bullet = \forall \vec{z}(c[\vec{z}/\vec{y}] \rightarrow A[\vec{z}/\vec{y}]_x^\bullet) \text{ with } x \notin \vec{z}$$

$$(\exists y A)_x^\bullet = \exists z A[z/y]_x^\bullet \text{ with } z \neq x$$

$$(A \bullet B)_x^\bullet = \exists y (A_y^\bullet \parallel ok(y) \rightarrow B_x^\bullet)$$

# Part XI

## CC and Linear Logic

# Part XI: CC and Linear Logic

37 CC - Logical Semantics

38 Must Properties

39 Program Analysis

## 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**:

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

(reverses the arrow of CLP interpretation. . . )

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

## Logical Semantics of CC?

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

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

(reverses the arrow of CLP interpretation. . . )

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

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

# 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**

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

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

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

- $!A$  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

## 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

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

## Constants

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

# Intuitionistic Linear Logic (cont.)

## Axiom - Cut

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

## 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

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

# ILL = the Logic of CC agents

Translation:

$$(A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \quad (c \rightarrow A)^\dagger =$$

# ILL = the Logic of CC agents

Translation:

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

# ILL = the Logic of CC agents

Translation:

$$(A \parallel B)^\dagger = A^\dagger \otimes B^\dagger \quad (c \rightarrow A)^\dagger = c \multimap A^\dagger \quad \textit{tell}(c)^\dagger = !c$$

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

# ILL = the Logic of CC agents

Translation:

$$\begin{aligned}(A \parallel B)^\dagger &= A^\dagger \otimes B^\dagger & (c \rightarrow A)^\dagger &= c \multimap A^\dagger & \text{tell}(c)^\dagger &= !c \\(A + B)^\dagger &= A^\dagger \& B^\dagger & (\exists xA)^\dagger &= \exists xA^\dagger & p(\vec{x})^\dagger &= p(\vec{x}) \\(X; c; \Gamma)^\dagger &= \exists X(!c \otimes \Gamma^\dagger)\end{aligned}$$

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

## Soundness and Completeness

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

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

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

# 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  $(X; c; \Gamma)^\dagger \dashv\vdash_{ILL(c, \mathcal{D})} (Y; d; \Delta)^\dagger$ .*

*If  $(X; c; \Gamma) \longrightarrow (Y; d; \Delta)$  then  $(X; c; \Gamma)^\dagger \vdash_{ILL(c, \mathcal{D})} (Y; d; \Delta)^\dagger$ .*

**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  $(X; c; \Gamma)^\dagger \dashv\vdash_{ILL(c, \mathcal{D})} (Y; d; \Delta)^\dagger$ .*

*If  $(X; c; \Gamma) \longrightarrow (Y; d; \Delta)$  then  $(X; c; \Gamma)^\dagger \vdash_{ILL(c, \mathcal{D})} (Y; d; \Delta)^\dagger$ .*

## Proof.

By induction 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  $(X; c; \Gamma)^\dagger \dashv\vdash_{ILL(c, \mathcal{D})} (Y; d; \Delta)^\dagger$ .*

*If  $(X; c; \Gamma) \longrightarrow (Y; d; \Delta)$  then  $(X; c; \Gamma)^\dagger \vdash_{ILL(c, \mathcal{D})} (Y; d; \Delta)^\dagger$ .*

## Proof.

By induction on  $\equiv$ , immediate.

By induction 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  $(X; c; \Gamma)^\dagger \Vdash_{ILL(c, \mathcal{D})} (Y; d; \Delta)^\dagger$ .

If  $(X; c; \Gamma) \longrightarrow (Y; d; \Delta)$  then  $(X; c; \Gamma)^\dagger \vdash_{ILL(c, \mathcal{D})} (Y; d; \Delta)^\dagger$ .

## 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 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  $\exists Xd \vdash_c c$ .*

**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  $\exists X d \vdash_c c$ .

## Proof.

By induction on a sequent calculus proof  $\pi$  of

$$A_1^\dagger, \dots, A_n^\dagger \vdash_{ILL(\mathcal{C}, \mathcal{D})} \phi$$

where the  $A_i$ 's are agents and  $\phi$  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  $\Gamma$  of agents such that  $(\emptyset; 1; A) \longrightarrow (X; d; \Gamma)$  and  $\exists X d \vdash_c c$ .*

### 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

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

### Ask

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

### 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 $\oplus$

Translate

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

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

same translation for the other operations

## Theorem 6 (Soundness of transitions)

*Let  $\Phi$  and  $\Psi$  be two frontiers.*

*If  $\Phi \equiv \Psi$  then  $(\Phi)^\dagger \Vdash_{ILL(\mathcal{C}, \mathcal{D})} (\Psi)^\dagger$ .*

*If  $\Phi \rightsquigarrow \Psi$  then  $\Phi^\dagger \vdash_{ILL(\mathcal{C}, \mathcal{D})} \Psi^\dagger$ .*

## 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  $\langle (\emptyset; 1; A) \rangle \rightsquigarrow \langle (X_1; d_1; \Gamma_1), \dots, (X_n; d_n; \Gamma_n) \rangle$  with  $\forall j \exists X_j d_j \vdash_c c$*

### 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  $\langle (\emptyset; 1; A) \rangle \rightsquigarrow \langle (X_1; d_1; \emptyset), \dots, (X_n; d_n; \emptyset) \rangle$  with  $\forall j \exists X_j d_j \vdash_c c$*

# 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  $\Gamma \not\vdash_{ILL_{c,D}} A$ .

## Proposition 11

*To prove a safety property  $(c, A) \dashrightarrow (d, B)$ , it is enough to show that  $\exists$  a phase space  $\mathbf{P}$ , a valuation  $\eta$ , 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

40 LCC

41 Examples

42 Logical Semantics

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

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

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

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

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

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

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

A **synchronization constraint** is a constraint not appearing in  $\Vdash_c$

## Same agents and observables as CC

Processes  $P ::= \mathcal{D}.A$

Declarations  $\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$

Agents  $A ::= \text{tell}(c) \mid \forall \vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$

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

## Same agents and observables as CC

Processes  $P ::= D.A$

Declarations  $\mathcal{D} ::= p(\vec{x}) = A, \mathcal{D} \mid \epsilon$

Agents  $A ::= \text{tell}(c) \mid \forall \vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists xA \mid p(\vec{x})$

- 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  $A ::= \text{tell}(c) \mid \forall \vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists xA \mid p(\vec{x})$

- 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),

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

- 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  $A ::= \text{tell}(c) \mid \forall \vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists xA \mid p(\vec{x})$

- 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),

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

- 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**  $(X; c; \text{tell}(d), \Gamma) \longrightarrow (X; c \otimes d; \Gamma)$

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

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

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

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

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

RecPhil(M,P) =

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

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

Philo(I,N) =

# An LCC( $\mathcal{FD}$ ) 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) =

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

# An LCC( $\mathcal{FD}$ ) 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) =

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

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

# An LCC( $\mathcal{FD}$ ) 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.

## 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 $\pi$ -calculus in LCC( $\mathcal{H}$ )

- Direct encoding of the asynchronous  $\pi$ -calculus:

$$\begin{aligned} [0] &= 1 \\ [(y)P] &= \exists y[P] \\ [\bar{x}y.0] &= \\ [x(y).P] &= \\ [P|Q] &= [P] \parallel [Q] \\ [[x = y]P] &= (x = y) \rightarrow [P] \\ [P + Q] &= [P] + [Q] \end{aligned}$$

- The usual (synchronous)  $\pi$ -calculus can be simulated with a synchronous communication protocol.

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

- Direct encoding of the asynchronous  $\pi$ -calculus:

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

- The usual (synchronous)  $\pi$ -calculus can be simulated with a synchronous communication protocol.

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

- Direct encoding of the asynchronous  $\pi$ -calculus:

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

- The usual (synchronous)  $\pi$ -calculus can be simulated with a synchronous communication protocol.

# 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:**

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

$\forall X (\text{count}(\text{np}, X) \rightarrow \text{count}(\text{np}, X+1))$ )

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

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

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

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

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

fd(x) =

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

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

`fd(X) = tell(min(X,min_integer)  $\otimes$  max(X,max_integer))`

`' $x \geq_1 y + c$ ' (X,Y,C) =`

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

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

$\text{fd}(X) = \text{tell}(\min(X, \text{min\_integer}) \otimes \max(X, \text{max\_integer}))$

$'x \geq_1 y + c'(X, Y, C) =$   
 $\min(X, \text{MinX}) \otimes \min(Y, \text{MinY}) \otimes (\text{MinX} < \text{MinY} + C)$   
 $\rightarrow (\text{tell}(\min(X, \text{MinY} + C) \otimes \min(Y, \text{MinY}))$   
 $\quad \parallel 'x \geq_1 y + c'(X, Y, C))$

$'x \geq y + c'(X, Y, C) =$

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

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

$\text{fd}(X) = \text{tell}(\text{min}(X, \text{min\_integer}) \otimes \text{max}(X, \text{max\_integer}))$

$'x \geq_1 y + c'(X, Y, C) =$   
 $\text{min}(X, \text{Min}X) \otimes \text{min}(Y, \text{Min}Y) \otimes (\text{Min}X < \text{Min}Y + C)$   
 $\rightarrow (\text{tell}(\text{min}(X, \text{Min}Y + C) \otimes \text{min}(Y, \text{Min}Y))$   
 $\parallel 'x \geq_1 y + c'(X, Y, C))$

$'x \geq y + c'(X, Y, C) = 'x \geq_1 y + c'(X, Y, C) \parallel 'x \geq_2 y + c'(X, Y, C)$

$'\text{ask}(x \geq y) \rightarrow a'(X, Y, A) =$

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

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

$\text{fd}(X) = \text{tell}(\text{min}(X, \text{min\_integer}) \otimes \text{max}(X, \text{max\_integer}))$

$'x \geq_1 y + c'(X, Y, C) =$   
 $\text{min}(X, \text{MinX}) \otimes \text{min}(Y, \text{MinY}) \otimes (\text{MinX} < \text{MinY} + C)$   
 $\rightarrow (\text{tell}(\text{min}(X, \text{MinY} + C) \otimes \text{min}(Y, \text{MinY}))$   
 $\parallel 'x \geq_1 y + c'(X, Y, C))$

$'x \geq y + c'(X, Y, C) = 'x \geq_1 y + c'(X, Y, C) \parallel 'x \geq_2 y + c'(X, Y, C)$

$'\text{ask}(x \geq y) \rightarrow a'(X, Y, A) =$   
 $\text{min}(X, \text{MinX}) \otimes \text{max}(Y, \text{MaxY}) \otimes (\text{MinX} \geq \text{MaxY})$   
 $\rightarrow A \parallel \text{tell}(\text{min}(X, \text{MinX}) \otimes \text{max}(Y, \text{MaxY}))$

Imperative variables allow a finer control, which is necessary for certain constraint solvers, e.g. the implementation of a Simplex solver in LCC [Schachter99these]

# Logical Semantics

Simple translation of LCC into ILL:

$$\mathit{tell}(c)^\dagger =$$

# Logical Semantics

Simple translation of LCC into ILL:

$$\begin{aligned} \mathit{tell}(c)^\dagger &= c \\ \forall \vec{y} (c \rightarrow A)^\dagger &= \end{aligned}$$

$$(A \parallel B)^\dagger = A^\dagger \otimes B^\dagger$$

# Logical Semantics

Simple translation of LCC into ILL:

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

$\text{ILL}(\mathcal{C}, \mathcal{D})$  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 results as for CC.