

Constraint Logic Programming

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

INRIA – Project-Team CONTRAINTES

MPRI C-2-4-1 Course – September 2009 - February 2010

Part III: CLP - Operational and Fixpoint Semantics

- 1 Operational Semantics
 - CSLD resolution
 - Observables
- 2 Fixpoint Semantics
 - Fixpoint Preliminaries
 - Fixpoint Semantics of Successes
 - Fixpoint Semantics of Computed Answers
- 3 Program Analysis
 - Abstract Interpretation
 - Constraint-based Model Checking

Part IV: Logical Semantics

- 4 Logical Semantics of CLP(\mathcal{X})
 - Soundness
 - Completeness
- 5 Automated Deduction
 - Proofs in Group Theory
- 6 CLP(λ)
 - λ -calculus
 - Proofs in λ -calculus
- 7 Negation as Failure
 - Finite Failure
 - Clark's Completion
 - Soundness w.r.t. Clark's Completion
 - Completeness w.r.t. Clark's Completion

Part V: Practical CLP Programming

- 8 CLP implementation, the WAM
- 9 Optimizing CLP
- 10 Summing up

Part VI

Concurrent Constraint Programming

Part VI: Concurrent Constraint Programming

11 Introduction

- Syntax
- CC vs. CLP

12 Operational Semantics

- Transitions
- Properties
- Observables

13 Examples

- append
- merge
- $CC(\mathcal{FD})$

Concurrent Constraint Programs

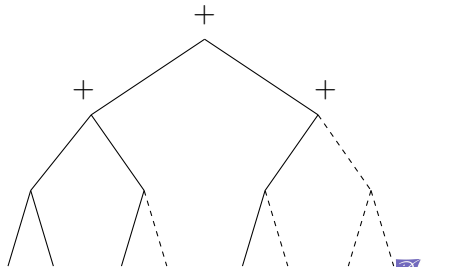
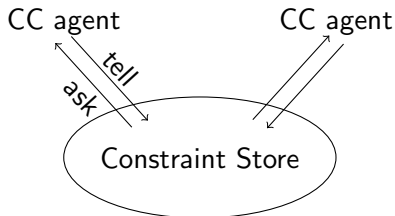
Class of programming languages $CC(\mathcal{X})$ introduced by Saraswat [Sar93] as a merge of Constraint and Concurrent Logic Programming.

Processes $P ::= D.A$

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

Agents $A ::= tell(c) \mid$

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



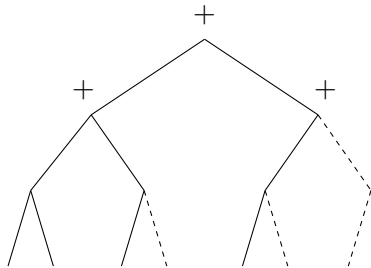
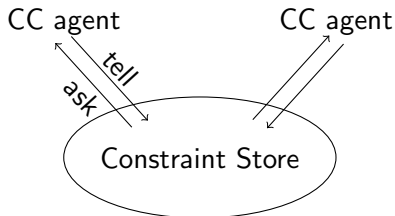
Concurrent Constraint Programs

Class of programming languages $CC(\mathcal{X})$ introduced by Saraswat [Sar93] 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 $A ::= tell(c) \mid \forall \vec{x}(c \rightarrow A) \mid A \parallel A \mid A + A \mid \exists x A \mid p(\vec{x})$



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

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

CC(\mathcal{X}) 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 Semantics

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

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

CC(\mathcal{X}) Operational Semantics

- 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),

- observing the set of **accessible stores**,

- observing the set of **limit stores**?

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

CC(\mathcal{X}) Operational Semantics

- 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),

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

- observing the set of **accessible stores**,

- observing the set of **limit stores**?

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

CC(\mathcal{X}) Operational Semantics

- 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**,

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

- observing the set of **limit stores**?

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

Part VII

CC - Denotational Semantics

Part VII: CC - Denotational Semantics

- 14 Deterministic Case
 - Syntax
 - I/O Function
 - Terminal Stores
- 15 Constraint Propagation
 - Closure Operators
 - Chaotic Iteration
- 16 Non-deterministic Case
 - Problems
 - Blind Choice
 - Example: merge
- 17 Sequentiality

Deterministic CC

Agents:

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

- No choice operator
- Deterministic ask.

Replace non-deterministic pattern matching

$$\forall \vec{x} (c \rightarrow A)$$

by deterministic ask and tell:

Deterministic CC

Agents:

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

- No choice operator
- Deterministic ask.

Replace non-deterministic pattern matching

$$\forall \vec{x} (c \rightarrow A)$$

by deterministic ask and tell:

$$(\exists \vec{x} c) \rightarrow \exists \vec{x} (\text{tell}(c) \parallel A)$$

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 $\uparrow d \subseteq \uparrow c$ 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 over (\mathcal{C}, \leq) .

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 $\uparrow d \subseteq \uparrow c$ 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) .

Closure Operators

Proposition 1

A closure operator f is characterized by the set of its fixpoints $Fix(f)$.

Proof.

We show that $f = \lambda x. \min(Fix(f) \cap \uparrow x)$.

Let $y = f(x)$. By idempotence and extensivity, $y \in Fix(f) \cap \uparrow x$.

By monotonicity $y = f(x) \leq f(y')$ for any $y' \in \uparrow x$.

Hence, if $y' \in Fix(f) \cap \uparrow x$ then $y \leq y'$.



Semantic Equations

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

$$\llbracket \mathcal{D}.tell(c) \rrbracket$$

$$\llbracket \mathcal{D}.c \rightarrow A \rrbracket$$

$$\llbracket \mathcal{D}.A \parallel B \rrbracket$$

$$\llbracket \mathcal{D}.\exists x A \rrbracket$$

$$\llbracket \mathcal{D}.p(\vec{x}) \rrbracket$$

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

Theorem 2 ([SRP91])

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

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

Semantic Equations

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

$$\begin{aligned} \llbracket \mathcal{D}.tell(c) \rrbracket &= \uparrow c && (\simeq \lambda s.s \wedge c) \\ \llbracket \mathcal{D}.c \rightarrow A \rrbracket & && \end{aligned}$$

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

Theorem 2 ([SRP91])

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

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

Semantic Equations

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

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

$$\llbracket \mathcal{D}.c \rightarrow A \rrbracket = (\mathcal{C} \setminus \uparrow c) \cup (\uparrow c \cap \llbracket \mathcal{D}.A \rrbracket)$$

$(\simeq \lambda s. \text{if } s \vdash_{\mathcal{C}} c \text{ then } \llbracket \mathcal{D}.A \rrbracket s \text{ else } s)$

$$\llbracket \mathcal{D}.A \parallel B \rrbracket$$

$$\llbracket \mathcal{D}.\exists x A \rrbracket$$

$$\llbracket \mathcal{D}.p(\vec{x}) \rrbracket$$

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

Theorem 2 ([SRP91])

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

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

Semantic Equations

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

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

$$\llbracket \mathcal{D}.c \rightarrow A \rrbracket = (\mathcal{C} \setminus \uparrow c) \cup (\uparrow c \cap \llbracket \mathcal{D}.A \rrbracket)$$

$(\simeq \lambda s. \text{if } s \vdash_{\mathcal{C}} c \text{ then } \llbracket \mathcal{D}.A \rrbracket s \text{ else } s)$

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

$$\llbracket \mathcal{D}.\exists x A \rrbracket$$

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

Theorem 2 ([SRP91])

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

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

Semantic Equations

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

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

$$\llbracket \mathcal{D}.c \rightarrow A \rrbracket = (\mathcal{C} \setminus \uparrow c) \cup (\uparrow c \cap \llbracket \mathcal{D}.A \rrbracket)$$

$(\simeq \lambda s. \text{if } s \vdash_{\mathcal{C}} c \text{ then } \llbracket \mathcal{D}.A \rrbracket s \text{ else } s)$

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

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

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

Theorem 2 ([SRP91])

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

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

Semantic Equations

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

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

$$\llbracket \mathcal{D}.c \rightarrow A \rrbracket = (\mathcal{C} \setminus \uparrow c) \cup (\uparrow c \cap \llbracket \mathcal{D}.A \rrbracket)$$

$(\simeq \lambda s. \text{if } s \vdash_{\mathcal{C}} c \text{ then } \llbracket \mathcal{D}.A \rrbracket s \text{ else } s)$

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

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

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

Theorem 2 ([SRP91])

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

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

Constraint Propagation and Closure Operators

An **environment** $E : \mathcal{V} \rightarrow 2^D$ associates a domain of possible values to each variable.

Consider the lattice of environments $(\mathcal{E}, \sqsubseteq)$, for the **information ordering** defined by $E \sqsubseteq E'$ if and only if $\forall x \in \mathcal{V}, E(x) \supseteq E'(x)$.

The semantics of a constraint propagator c can be defined as a closure operator over \mathcal{E} , noted \bar{c} , i.e. a mapping $\mathcal{E} \rightarrow \mathcal{E}$ satisfying

- ① (extensivity) $E \sqsubseteq \bar{c}(E)$,
- ② (monotonicity) if $E \sqsubseteq E'$ then $\bar{c}(E) \sqsubseteq \bar{c}(E')$
- ③ (idempotence) $\bar{c}(\bar{c}(E)) = \bar{c}(E)$.

Example in $CC(\mathcal{FD})$

Let $b = (x > y)$ and $c = (y > x)$.

Let $E(x) = [1, 10]$, $E(y) = [1, 10]$ be the initial environment
 we have

$$\begin{aligned}\bar{b}E(x) &= [2, 10] \\ \bar{c}E(x) &= [1, 9] \\ (\bar{b} \sqcup \bar{c})E(x) &= [2, 9]\end{aligned}$$

The closure operator $\overline{b, c}$ associated to the conjunction of
 constraints $b \wedge c$ gives the intended semantics:

$$\overline{b, c}E(x) = Y(\lambda s. \bar{b}(\bar{c}(s)))E(x) = \emptyset$$

Chaotic Iteration of Monotone Operators

Let $L(\sqsubseteq, \perp, \top, \sqcup, \sqcap)$ be a complete lattice, and $F : L^n \rightarrow L^n$ a monotone operator over L^n with $n > 0$.

The **chaotic iteration** of F from $D \in L^n$ for a fair transfinite choice sequence $\langle J^\delta : \delta \in \text{Ord} \rangle$ is the sequence $\langle X^\delta \rangle$:

$$X^0 = D,$$

$$X_i^{\delta+1} = F_i(X^\delta) \text{ if } i \in J^\delta, X_i^{\delta+1} = X_i^\delta \text{ otherwise,}$$

$$X_i^\delta = \bigsqcup_{\alpha < \delta} X_i^\alpha \text{ for any limit ordinal } \delta.$$

Theorem 3 ([CC77])

Let $D \in L^n$ be a pre fixpoint of F (i.e. $D \sqsubseteq F(D)$). Any chaotic iteration of F starting from D is increasing and has for limit the least fixpoint of F above D .

Constraint Propagation as Chaotic Iteration

Corollary 4 (Correctness of constraint propagation)

Let $c = a_1 \wedge \dots \wedge a_n$, and E be an environment. Then $\bar{c}(E)$ is the limit of any fair iteration of closure operators $\bar{a}_1, \dots, \bar{a}_n$ from E .

Let $F : L^{n+1} \rightarrow L^{n+1}$ be defined by its projections F_i 's:

$$\left\{ \begin{array}{l} E_1 = \bar{a}_1(E) = F_1(E_1, \dots, E_n, E) \\ E_2 = \bar{a}_2(E) = F_2(E_1, \dots, E_n, E) \\ \dots \\ E_n = \bar{a}_n(E) = F_n(E_1, \dots, E_n, E) \\ E = E_1 \cap \dots \cap E_n = F_{n+1}(E_1, \dots, E_n, E) \end{array} \right.$$

The functions F_i 's are obviously monotonic, any fair iteration of $\bar{a}_1, \dots, \bar{a}_n$ is thus a chaotic iteration of F_1, \dots, F_{n+1} therefore its limit is equal to the least fixpoint greater than E , i.e. $\bar{c}(E)$.

Denotational Semantics of Non-deterministic CC

Problem: the set of terminal stores of a CC process with **one step guarded choice** (i.e. *global choice*) is **not compositional**:

$$\begin{aligned}
 A &= \text{ask}(x = a) \rightarrow \text{tell}(y = a) \\
 &\quad + \text{ask}(\text{true}) \rightarrow \text{tell}(\text{false}) \\
 B &= \text{tell}(x = a \wedge y = a)
 \end{aligned}$$

A and B have the same set of terminal stores

but that is not the case for $\exists x B$ and $\exists x A$

Denotational Semantics of Non-deterministic CC

Problem: the set of terminal stores of a CC process with **one step guarded choice** (i.e. *global choice*) is **not compositional**:

$$\begin{aligned}
 A &= \text{ask}(x = a) \rightarrow \text{tell}(y = a) \\
 &\quad + \text{ask}(\text{true}) \rightarrow \text{tell}(\text{false}) \\
 B &= \text{tell}(x = a \wedge y = a)
 \end{aligned}$$

A and B have the same set of terminal stores

$$\uparrow \{x = a \wedge y = a\}$$

(with global choice $\mathcal{C} \setminus \uparrow (x = a)$ is not a terminal store for A)

but that is not the case for $\exists x B$ and $\exists x A$

Denotational Semantics of Non-deterministic CC

Problem: the set of terminal stores of a CC process with **one step guarded choice** (i.e. *global choice*) is **not compositional**:

$$\begin{aligned}
 A &= \text{ask}(x = a) \rightarrow \text{tell}(y = a) \\
 &\quad + \text{ask}(\text{true}) \rightarrow \text{tell}(\text{false}) \\
 B &= \text{tell}(x = a \wedge y = a)
 \end{aligned}$$

A and B have the same set of terminal stores

$$\uparrow \{x = a \wedge y = a\}$$

(with global choice $\mathcal{C} \setminus \uparrow (x = a)$ is not a terminal store for A)

but that is not the case for $\exists x B$ and $\exists x A$

$y = a$ is a terminal store for $\exists x B$ and not for $\exists x A$...

Non-deterministic $CC(\mathcal{X})$ with Local Choice (1)

The set of terminal stores of a CC process with **blind choice** can be characterized easily by adding the semantic equation:

$$\llbracket \mathcal{D}.A + B \rrbracket = \llbracket \mathcal{D}.A \rrbracket \cup \llbracket \mathcal{D}.B \rrbracket$$

Theorem 5 ([dBGP96])

$$\llbracket \mathcal{D}.A \rrbracket = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(\mathcal{D}.A; c)$$

but the input-output relation cannot be recovered from $\llbracket \mathcal{D}.A \rrbracket$:

$$\llbracket \text{tell}(\text{true}) \rrbracket =$$

$$\llbracket \text{tell}(\text{true}) + \text{tell}(c) \rrbracket =$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}); \text{true}) =$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}) + \text{tell}(c); \text{true}) =$$

Idea:

Non-deterministic $CC(\mathcal{X})$ with Local Choice (1)

The set of terminal stores of a CC process with **blind choice** can be characterized easily by adding the semantic equation:

$$\llbracket \mathcal{D}.A + B \rrbracket = \llbracket \mathcal{D}.A \rrbracket \cup \llbracket \mathcal{D}.B \rrbracket$$

Theorem 5 ([dBGP96])

$$\llbracket \mathcal{D}.A \rrbracket = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(\mathcal{D}.A; c)$$

but the input-output relation cannot be recovered from $\llbracket \mathcal{D}.A \rrbracket$:

$$\llbracket \text{tell}(\text{true}) \rrbracket = \mathcal{C}$$

$$\llbracket \text{tell}(\text{true}) + \text{tell}(c) \rrbracket =$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}); \text{true}) =$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}) + \text{tell}(c); \text{true}) =$$

Idea:

Non-deterministic $CC(\mathcal{X})$ with Local Choice (1)

The set of terminal stores of a CC process with **blind choice** can be characterized easily by adding the semantic equation:

$$\llbracket \mathcal{D}.A + B \rrbracket = \llbracket \mathcal{D}.A \rrbracket \cup \llbracket \mathcal{D}.B \rrbracket$$

Theorem 5 ([dBGP96])

$$\llbracket \mathcal{D}.A \rrbracket = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(\mathcal{D}.A; c)$$

but the input-output relation cannot be recovered from $\llbracket \mathcal{D}.A \rrbracket$:

$$\llbracket \text{tell}(\text{true}) \rrbracket = \mathcal{C}$$

$$\llbracket \text{tell}(\text{true}) + \text{tell}(c) \rrbracket = \mathcal{C}$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}); \text{true}) =$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}) + \text{tell}(c); \text{true}) =$$

Idea:

Non-deterministic $CC(\mathcal{X})$ with Local Choice (1)

The set of terminal stores of a CC process with **blind choice** can be characterized easily by adding the semantic equation:

$$\llbracket \mathcal{D}.A + B \rrbracket = \llbracket \mathcal{D}.A \rrbracket \cup \llbracket \mathcal{D}.B \rrbracket$$

Theorem 5 ([dBGP96])

$$\llbracket \mathcal{D}.A \rrbracket = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(\mathcal{D}.A; c)$$

but the input-output relation cannot be recovered from $\llbracket \mathcal{D}.A \rrbracket$:

$$\llbracket \text{tell}(\text{true}) \rrbracket = \mathcal{C}$$

$$\llbracket \text{tell}(\text{true}) + \text{tell}(c) \rrbracket = \mathcal{C}$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}); \text{true}) = \{\text{true}\}$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}) + \text{tell}(c); \text{true}) =$$

Idea:

Non-deterministic $CC(\mathcal{X})$ with Local Choice (1)

The set of terminal stores of a CC process with **blind choice** can be characterized easily by adding the semantic equation:

$$\llbracket \mathcal{D}.A + B \rrbracket = \llbracket \mathcal{D}.A \rrbracket \cup \llbracket \mathcal{D}.B \rrbracket$$

Theorem 5 ([dBGP96])

$$\llbracket \mathcal{D}.A \rrbracket = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(\mathcal{D}.A; c)$$

but the input-output relation cannot be recovered from $\llbracket \mathcal{D}.A \rrbracket$:

$$\llbracket \text{tell}(\text{true}) \rrbracket = \mathcal{C}$$

$$\llbracket \text{tell}(\text{true}) + \text{tell}(c) \rrbracket = \mathcal{C}$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}); \text{true}) = \{\text{true}\}$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}) + \text{tell}(c); \text{true}) = \{\text{true}, c\}$$

Idea:

Non-deterministic $CC(\mathcal{X})$ with Local Choice (1)

The set of terminal stores of a CC process with **blind choice** can be characterized easily by adding the semantic equation:

$$\llbracket \mathcal{D}.A + B \rrbracket = \llbracket \mathcal{D}.A \rrbracket \cup \llbracket \mathcal{D}.B \rrbracket$$

Theorem 5 ([dBGP96])

$$\llbracket \mathcal{D}.A \rrbracket = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(\mathcal{D}.A; c)$$

but the input-output relation cannot be recovered from $\llbracket \mathcal{D}.A \rrbracket$:

$$\llbracket \text{tell}(\text{true}) \rrbracket = \mathcal{C}$$

$$\llbracket \text{tell}(\text{true}) + \text{tell}(c) \rrbracket = \mathcal{C}$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}); \text{true}) = \{\text{true}\}$$

$$\mathcal{O}_{ts}(\text{tell}(\text{true}) + \text{tell}(c); \text{true}) = \{\text{true}, c\}$$

Idea: define $\llbracket \cdot \rrbracket : \mathcal{D} \times A \rightarrow \mathcal{P}(\mathcal{P}(\mathcal{C}))$ to distinguish between branches.

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

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

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

Theorem 6 ([MFP97])

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

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

'merge' Example Revisited

Merging streams

$$\begin{aligned}
 \text{merge}(A, B, C) = & \\
 & (A = [] \rightarrow \text{tell}(C = B)) \parallel \\
 & (B = [] \rightarrow \text{tell}(C = A)) \parallel \\
 & (\forall X, L(A = [X|L] \rightarrow \text{tell}(C = [X|R]) \parallel \text{merge}(L, B, R)) + \\
 & \forall X, L(B = [X|L] \rightarrow \text{tell}(C = [X|R]) \parallel \text{merge}(A, L, R)))
 \end{aligned}$$

Do we have the expected terminal stores?

'merge' Example Revisited

Merging streams

$$\begin{aligned}
 \text{merge}(A, B, C) = & \\
 & (A = [] \rightarrow \text{tell}(C = B)) \parallel \\
 & (B = [] \rightarrow \text{tell}(C = A)) \parallel \\
 & (\forall X, L(A = [X|L] \rightarrow \text{tell}(C = [X|R]) \parallel \text{merge}(L, B, R)) + \\
 & \forall X, L(B = [X|L] \rightarrow \text{tell}(C = [X|R]) \parallel \text{merge}(A, L, R)))
 \end{aligned}$$

Do we have the expected terminal stores?

No!

for $\text{merge}(X, [1|Y], Z)$ we don't get 1 in Z , the merging is not *greedy*...

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:

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

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

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

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

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

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

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

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

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

Bibliography I



Patrick Cousot and Radhia Cousot.

Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints.

In *POPL'77: Proceedings of the 6th ACM Symposium on Principles of Programming Languages*, pages 238–252, New York, 1977. ACM Press.
Los Angeles.



Frank S. de Boer, Maurizio Gabbrielli, and Catuscia Palamidessi.

Proving correctness of constraint logic programming with dynamic scheduling.
In *Proceedings of SAS'96*, LNCS 1145. Springer-Verlag, 1996.



Kim Marriott Moreno Falaschi, Maurizio Gabbrielli and Catuscia Palamidessi.

Confluence in concurrent constraint programming.
Theoretical Computer Science, 183(2):281–315, 1997.



Vijay A. Saraswat.

Concurrent constraint programming.
ACM Doctoral Dissertation Awards. MIT Press, 1993.



Vijay A. Saraswat, Martin C. Rinard, and Prakash Panangaden.

Semantic foundations of concurrent constraint programming.
In *POPL'91: Proceedings of the 18th ACM Symposium on Principles of Programming Languages*, 1991.