

Concurrent Constraint Programming

Part 2: Denotational Semantics

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

Determinate 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 $\uparrow c \subseteq \uparrow d$ where $\uparrow c = \{d \in \mathcal{C} \mid c \leq d\}$.

- $[[D.A]] : \mathcal{C} \rightarrow \mathcal{C}$ is
- i) Extensive: $\forall c \ c \leq [[D.A]]c$
 - ii) Monotone: $\forall c, d \ c \leq d \Rightarrow [[D.A]]c \leq [[D.A]]d$
 - iii) Idempotent: $\forall c \ [[D.A]]c = [[D.A]]([D.A]c)$

i.e. $[[D.A]]$ is a fixpoint operator 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 $\uparrow c \subseteq \uparrow d$ where $\uparrow c = \{d \in \mathcal{C} \mid c \leq d\}$.

- $[[D.A]] : \mathcal{C} \rightarrow \mathcal{C}$ is
- i) Extensive: $\forall c \ c \leq [[D.A]]c$
 - ii) Monotone: $\forall c, d \ c \leq d \Rightarrow [[D.A]]c \leq [[D.A]]d$
 - iii) Idempotent: $\forall c \ [[D.A]]c = [[D.A]]([D.A]c)$

i.e. $[[D.A]]$ 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, $y \in Fix(f) \cap \uparrow c$.

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

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

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

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

$$[[D.A \parallel B]] = [[D.A]] \cap [[D.B]] \quad (\simeq Y(\lambda s. [[D.A]][[D.B]]s))$$

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

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

Theorem 2 (SRP91) *For any deterministic process $D.A$*

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

Constraint Propagation and Closure Operators

An *environment* $E : \mathcal{V} \rightarrow 2^{\mathcal{D}}$ 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

- i) (extensivity) $E \sqsubseteq \bar{c}(E)$,
- ii) (monotonicity) if $E \sqsubseteq E'$ then $\bar{c}(E) \sqsubseteq \bar{c}(E')$
- iii) (idempotence) $\bar{c}(\bar{c}(E)) = \bar{c}(E)$.

Example in CC(FD)

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

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

we have

$$\bar{b}E(x) = [2, 10],$$

$$\bar{c}E(x) = [1, 9],$$

$$(\bar{b} \sqcup \bar{c})E(x) = [2, 9].$$

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 $\langle X^\delta : \delta \in \text{card}(L) \rangle$

$$X^0 = D,$$

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

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

Theorem 3 (Cousot77) *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 fixed point 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 .*

PROOF: 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 fixed point greater than E , i.e. $\bar{c}(E)$. □

Denotational Semantics of Non determinate CC

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

$$A = \text{ask}(x = a) \rightarrow \text{tell}(y = a) \\ + \text{ask}(\text{true}) \rightarrow \text{tell}(\text{false})$$

$$B = \text{tell}(x = a \wedge y = a)$$

A and B have the same set of terminal stores:

...

Denotational Semantics of Non determinate CC

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

$$A = \text{ask}(x = a) \rightarrow \text{tell}(y = a) \\ + \text{ask}(\text{true}) \rightarrow \text{tell}(\text{false})$$

$$B = \text{tell}(x = a \wedge y = a)$$

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 not $\exists x B$ and $\exists x A$...

Denotational Semantics of Non determinate CC

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

$$A = \text{ask}(x = a) \rightarrow \text{tell}(y = a) \\ + \text{ask}(\text{true}) \rightarrow \text{tell}(\text{false})$$

$$B = \text{tell}(x = a \wedge y = a)$$

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 not $\exists x B$ and $\exists x A$:

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

Non determinate $CC(\mathcal{X})$ with local choice

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

$$[[D.A + B]] = [[D.A]] \cup [[D.B]]$$

Theorem 5 (BGP96) $[[D.A]] = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(D.A; c)$

but the input-output relation cannot be recovered from $[[D.A]]$:

$$[[tell(true)]] =$$

$$[[tell(true) + tell(c)]] =$$

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

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

Idea:

Non determinate $CC(\mathcal{X})$ with local choice

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

$$[[D.A + B]] = [[D.A]] \cup [[D.B]]$$

Theorem 6 (BGP96) $[[D.A]] = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(D.A; c)$

but the input-output relation cannot be recovered from $[[D.A]]$:

$$[[tell(true)]] = \mathcal{C}$$

$$[[tell(true) + tell(c)]] =$$

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

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

Idea:

Non determinate $CC(\mathcal{X})$ with local choice

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

$$[[D.A + B]] = [[D.A]] \cup [[D.B]]$$

Theorem 7 (BGP96) $[[D.A]] = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(D.A; c)$

but the input-output relation cannot be recovered from $[[D.A]]$:

$$[[tell(true)]] = \mathcal{C}$$

$$[[tell(true) + tell(c)]] = \mathcal{C}$$

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

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

Idea:

Non determinate $\text{CC}(\mathcal{X})$ with local choice

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

$$[[D.A + B]] = [[D.A]] \cup [[D.B]]$$

Theorem 8 (BGP96) $[[D.A]] = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(D.A; c)$

but the input-output relation cannot be recovered from $[[D.A]]$:

$$[[\text{tell}(\text{true})]] = \mathcal{C}$$

$$[[\text{tell}(\text{true}) + \text{tell}(c)]] = \mathcal{C}$$

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

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

Idea: define $[[\square]] : D \times A \rightarrow \mathbf{P}(\mathbf{P}(\mathcal{C}))$ to distinguish between branches

Non determinate $\text{CC}(\mathcal{X})$ with local choice

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

$$[[D.A + B]] = [[D.A]] \cup [[D.B]]$$

Theorem 9 (BGP96) $[[D.A]] = \bigcup_{c \in \mathcal{C}} \mathcal{O}_{ts}(D.A; c)$

but the input-output relation cannot be recovered from $[[D.A]]$:

$$[[\text{tell}(\text{true})]] = \mathcal{C}$$

$$[[\text{tell}(\text{true}) + \text{tell}(c)]] = \mathcal{C}$$

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

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

Idea: define $[[\]]: D \times A \rightarrow \mathbf{P}(\mathbf{P}(\mathcal{C}))$ to distinguish between branches

Non determinate CC with local choice

Let $[[\cdot]] : D \times A \rightarrow \mathbf{P}(\mathbf{P}(\mathcal{C}))$ be the least fixed point (for \subseteq) of

$$[[D.c]] = \{\uparrow c\}$$

$$[[D.c \rightarrow A]] = \{\mathcal{C} \setminus \uparrow c\} \cup \{\uparrow c \cap X \mid X \in [[D.A]]\}$$

$$[[D.A \parallel B]] = \{X \cap Y \mid X \in [[D.A]], Y \in [[D.B]]\}$$

$$[[D.A + B]] = [[D.A]] \cup [[D.B]]$$

$$[[D.\exists x A]] = \{\{d \mid \exists xc = \exists xd, c \in X\} \mid X \in [[D.A]]\}$$

$$[[D.p(\vec{x})]] = [[D.A[\vec{x}/\vec{y}]]]$$

Theorem 10 (FGMP94) *For any process $D.A$,*

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