Concurrency 3 = CCS (1/4)

CCS : syntax and transitions, coinduction

Pierre-Louis Curien (CNRS – Université Paris 7)

MPRI concurrency course 2004/2005 with :

Jean-Jacques Lévy (INRIA-Rocquencourt)
Eric Goubault (CEA)
James Leifer (INRIA - Rocq)
Catuscia Palamidessi (INRIA - Futurs)

———————

(http://pauillac.inria.fr/~leifer/teaching/mpri-concurrency-2004)

From automata to CCS (1/6)

Remove final (not interested in termination), and initial states (assimilate processes with states, hence any state is "initial" relative to the process it is identified with).

Such an automaton deprived from initial and final states is called a labelled transition system, or **LTS** for short.

From automata to CCS (2/6)

A LTS is given by

- a finite set of states, or $P, Q, \ldots$,
- a finite alphabet *Act* whose members are called actions, and
- transitions between them, written $P \xrightarrow{\mu} Q$.

From automata to CCS (3/6)

A LTS together with one of its states, that is, a process, can be described by the following syntax :

$$P ::= \Sigma_{i \in I} \mu_i \cdot P_i \mid let\ \vec{K} = \vec{P}\ in\ K_j \mid K$$

(empty sum denoted by 0)

From automata to CCS (4/6)

CCS $\quad P ::=$

$\Sigma_{i \in I} \mu_i \cdot P_i \mid let\ \vec{K} = \vec{P}\ in\ K_j \mid K \mid (P \mid Q) \mid (\nu a)P$

Synchronization Trees $\quad P ::=$

$\Sigma_{i \in I} \mu_i \cdot P_i$

Finitary CCS $\quad P ::=$

$\Sigma_{i \in I} \mu_i \cdot P_i \mid (P \mid Q) \mid (\nu a)P \quad (I$ finite)

From automata to CCS (5/6)

in CCS

$$Act = L \cup \overline{L} \cup \{\tau\}$$

(disjoint union), where L is the set of labels, also called names, or channels, and τ is a silent action that records a synchronisation. $\mu \in Act$, $\alpha \in L \cup \overline{L}$, $\overline{\overline{\alpha}} = \alpha$

From automata to CCS (6/6)

We write

$$\Sigma_{i \in I} a_i \cdot P_i = (\Sigma_{i \in I \setminus i_0} a_i \cdot P_i) + a_{i_0} \cdot P_{i_0}$$

(note that the notation implicitly views sums as associative and commutative – this will be made explicit later)

Labelled operational semantics (1/4)

$$\frac{}{\Sigma_{i\in I}\mu_i \cdot P_i \xrightarrow{\mu_i} P_i} \qquad \frac{P \xrightarrow{\mu} P' \quad (\mu \neq a, \overline{a})}{(\nu a)P \xrightarrow{\mu} (\nu a)P'}$$

$$\frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q} \quad \frac{Q \xrightarrow{\mu} Q'}{P \mid Q \xrightarrow{\mu} P \mid Q'} \quad \frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\overline{\alpha}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\frac{P_j[\vec{K} \leftarrow (let\ \vec{K} = \vec{P}\ in\ \vec{K})] \xrightarrow{\mu} P'}{let\ \vec{K} = \vec{P}\ in\ K_j \xrightarrow{\mu} P'}$$

Labelled operational semantics (2/4)

τ-transitions (resp. α-transitions) correspond to internal evolutions (resp. interactions with the environment).
Rule COMM involves **both**.

In λ-calculus, one considers only one (internal) reduction : β.

Labelled operational semantics (3/4)

Example :

$$P = (\nu c)(K_1 \mid K_2) \text{ where } \begin{cases} K_1 = a \cdot \overline{c} \cdot K_1 \\ K_2 = b \cdot c \cdot K_2 \end{cases}$$

Behaviour : do a and b independently, then τ, then loop.

Labelled operational semantics (4/4)

It is possible to formulate internal reduction in CCS without reference to the environment.

Price to pay : work modulo structural equivalence.

.

Structural equivalence

$\Sigma_{i \in I} \mu_i \cdot P_i \equiv \Sigma_{i \in I} \mu_{f(i)} \cdot P_{f(i)}$ (f permutation)

$P \mid Q \equiv Q \mid P$

$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$

$((\nu a)P) \mid Q \equiv (\nu a)\ (P \mid Q)$ (a not free in Q)

$let\ \vec{K} = \vec{P}\ in\ K_j \equiv P_j[\vec{K} \leftarrow (let\ \vec{K} = \vec{P}\ in\ \vec{K})]$

Reduction operational semantics (1/2)

$$\frac{}{P_1 + a \cdot P \mid \overline{a} \cdot Q + Q_1 \to P \mid Q} \qquad \frac{}{P_1 + \tau \cdot P \to P}$$

$$\frac{P_1 \to P_1'}{P_1 \mid P_2 \to P_1' \mid P_2} \qquad \frac{P \to P'}{(\nu a)P \to (\nu a)P'}$$

$$\frac{P_1 \equiv P_2 \to P_2' \equiv P_1'}{P_1 \to P_1'}$$

Reduction operational semantics (2/2)

The relations $\rightarrow$ and $\xrightarrow{\tau}\equiv$ coincide.

Exercice 1 Prove it, via the following claims :

• If $P \xrightarrow{\mu} P'$ and $P \equiv Q$, then there exists Q' such that $Q \xrightarrow{\mu} Q'$ and $P' \equiv Q'$.

• If $P \xrightarrow{\alpha} P'$, then $P \equiv (\nu \vec{a})\,(\alpha \cdot Q + P_1 \mid P_2)$ and $P' \equiv (\nu \vec{a})\,(P_1 \mid P_2)$, for some $\vec{a}, P_1, P_2, Q$.

Semaphore in CCS

$$\mathit{Sem} = \mathsf{P} \cdot \mathsf{V} \cdot \mathit{Sem}$$

$$\begin{aligned}
&\mathit{Sem} \mid (\overline{\mathsf{P}} \cdot C_0; \overline{\mathsf{V}}) \mid (\overline{\mathsf{P}} \cdot C_1; \overline{\mathsf{V}}) \\
&\quad \rightarrow (\mathsf{V} \cdot \mathit{Sem}) \mid (\overline{\mathsf{P}} \cdot C_0; \overline{\mathsf{V}}) \mid (C_1; \overline{\mathsf{V}}) \\
&\quad \rightarrow^{\star} (\mathsf{V} \cdot \mathit{Sem}) \mid (\overline{\mathsf{P}} \cdot C_0; \overline{\mathsf{V}}) \mid \overline{\mathsf{V}} \\
&\quad \rightarrow \mathit{Sem} \mid (\overline{\mathsf{P}} \cdot C_0; \overline{\mathsf{V}})
\end{aligned}$$

Exercice 2 Encode $P;Q$ in CCS.

Value passing

$$\frac{}{P_1 + a(x) \cdot P \mid \overline{a}\langle v \rangle \cdot Q + Q_1 \to P[x \leftarrow v] \mid Q}$$

A memory cell :

$$\mathit{Reg}\langle x \rangle = \overline{\mathit{Get}}\langle x \rangle \cdot \mathit{Reg}\langle x \rangle + \mathit{Put}(y) \cdot \mathit{Reg}\langle y \rangle$$

One-shot : $\begin{cases} \mathit{Sem}\langle x \rangle = (\overline{\mathit{Get}}\langle x \rangle \cdot K) + K \\ K = \mathit{Put}(y) \cdot \mathit{Sem}\langle y \rangle \end{cases}$

(cf. Concurrency 2)

Bisimulation on a LTS (1/4)

A simulation is a relation $\mathcal{R}$ such that for all P, Q, if $P \,\mathcal{R}\, Q$ then

$$\forall\, \mu, P' \;\; (P \xrightarrow{\mu} P' \Rightarrow \exists\, Q' \;\; Q \xrightarrow{\mu} Q' \text{ and } P' \,\mathcal{R}\, Q')$$

Bisimulation on a LTS (2/4)

A bisimulation is a relation $\mathcal{R}$ such that $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.

P, Q are bisimilar (notation $P \sim Q$) if there exists a bisimulation $\mathcal{R}$ such that $P \mathcal{R} Q$.
$(\mathcal{R}^{-1} = \{(Q, P) \mid P \mathcal{R} Q\})$

Bisimulation on a LTS (3/4)

If $\mathcal{R}, \mathcal{S}$ are bisimulations, then so is their composition

$$RS = \{(P, R) \mid \exists Q \;\; P \,\mathcal{R}\, Q \text{ and } Q \,\mathcal{S} R\}$$

In particular, $\sim\sim \subseteq \sim$, i.e., bisimilarity is transitive.

.

Bisimulation on a LTS (4/4)

Two processes that simulate one another, yet are not bisimilar :

$$P_1 = a \cdot P_2 + a \cdot P_4 \qquad Q_1 = a \cdot Q_2$$

$$P_2 = b \cdot P_3 \qquad Q_2 = b \cdot Q_3$$

$$P_1 \,\mathcal{T}\, Q_1 \quad P_4 \,\mathcal{T}\, Q_2 \quad P_2 \,\mathcal{T}\, Q_2 \quad P_3 \,\mathcal{T}\, Q_3$$

$$Q_1 \,\mathcal{S}\, P_1 \quad Q_2 \,\mathcal{S}\, P_2 \quad Q_3 \,\mathcal{S}\, P_3 \ .$$

but for all simulation $\mathcal{R}$ containing (P_1, Q_1) we have :
$P_1 \,\mathcal{R}\, Q_1$ and $P_1 \xrightarrow{a} P_4 \Rightarrow P_4 \,\mathcal{R}\, Q_2$

Induction and coinduction (1/4)

A function $f : D \to E$, where D, E are partial orders, is monotonous if

$$\forall x, y \;\; x \leq y \Rightarrow f(x) \leq f(y)$$

Given (monotonous) $f : D \to D$, a prefixpoint (resp. a postfixpoint, a fixpoint) of f is a point x such that $f(x) \leq x$ (resp. $x \leq f(x)$, $x = f(x)$).

Induction and coinduction (2/4)

Any monotonous function $G : \mathcal{P}(X) \to \mathcal{P}(X)$ has a least prefixpoint, which is moreover a fixpoint, and a greatest postfixpoint, which is moreover a fixpoint. They are respectively :

$$\mathit{lfp}(G) = \bigcap\{X \mid G(X) \subseteq X\}$$
$$\mathit{gfp}(G) = \bigcup\{X \mid X \subseteq G(X)\}$$

Induction and coinduction (3/4)

Induction principle : To show *lfp*$(\mu) \subseteq R$ is is enough to show $\mu(R) \subseteq R$.

In practice, the induction principle is often used for a subset of *lfp*(μ), and then serves to show that $R =$ *lfp*(μ).

Induction and coinduction (4/4)

Coinduction principle : To show $R \subseteq gfp(\mu)$ it is enough to show $R \subseteq \mu(R)$.

In practice, the principle of coinduction is used to show that some element x is in $gfp(\mu)$, and for this it is enough to find a postfixpoint R such that $x \in R$.

Operators defined by rules (1/4)

Monotonous operators G_K on $\mathcal{P}(X)$ defined via a set K of rules, each of the form (Y, x), with $Y \subseteq X$ and $x \in X$, or, graphically (for $Y = \{x_1, \ldots, x_n\}$ finite) :

$$\frac{\{x_1, \ldots, x_n\}}{x}$$

Set $G_K(R) = \{x \in X \mid \exists\, (Y, x) \in K \;\; Y \subseteq R\}$.

Operators defined by rules (2/4)

Prefixpoints of G_K =
subsets R closed forwards by the rules :

$$\forall (Y, x) \in K \;\; (Y \subseteq R \Rightarrow x \in R)$$

Postfixpoints of G_K =
subsets R closed backwards by the rules :

$$\forall x \in R \; \exists (Y, x) \in K \quad Y \subseteq R$$

.

Operators defined by rules (3/4)

Bisimulation is defined by a set of rules : take K to be the set of all

$$\frac{\{(P', f(\mu, P')) \mid P \xrightarrow{\mu} P'\} \cup \{(g(\mu, Q'), Q') \mid Q \xrightarrow{\mu} Q'\}}{(P, Q)}$$

where f is any function mapping each pair μ, P' such that $P \xrightarrow{\mu} P'$ to a process $f(\mu, P')$ such that $Q \xrightarrow{\mu} f(\mu, P')$ (resp. g ...).

Operators defined by rules (4/4)

What do we gain by knowing that $\sim$, first defined as the union of all bisimulations, is actually the largest fixpoint of some operator ?

First, that $\sim$ itself is a bisimulation, second that it is a prefixpoint, not only a post-fixpoint.

Continuity (1/3)

$G : \mathcal{P}(X) \to \mathcal{P}(X)$ is continuous if it preserves $\bigcup$ of increasing chains, i.e. $G(\bigcup_{n\in\omega} X_n) = \bigcup_{n\in\omega} G(X_n)$. G is called anti-continuous if it preserves $\bigcap$ of decreasing chains.

$$\begin{array}{rcl} G \text{ continuous} & \Rightarrow & \mathit{lfp}(G) = \bigcup_{n\in\omega} G^n(\emptyset) \\ G \text{ anti-continuous} & \Rightarrow & \mathit{gfp}(G) = \bigcap_{n\in\omega} G^n(X) \end{array}$$

Continuity (2/3)

If all the Y's in the rules of K are finite, then G_K is continuous.
If, for all x, $\{(Y \mid (Y, x) \in K\}$ is finite, then G_K is anti-continuous.

In finitary CCS the bisimulation operator G_K is both continuous and anti-continuous.

NB : finite sum assumption is not enough : take $let\ K = (a \cdot 0 \mid K)\ in\ K$.

Continuity (3/3)

Consider the following K :

$$\frac{}{\textit{nil}} \qquad \frac{l}{\textit{cons}(a,l)}$$

The *lfp* of G_K is the set of lists. The *gfp* of G_K is the set of finite and infinite lists.

Exercice 3 How to obtain infinite lists ?