Insegnamento di

Foundation of Computing

Pagina del corso: http://pages.di.unipi.it/montanari/FOC.html

Note di

Pi Calcolo

a cura di
Ugo Montanari
Dipartimento di Informatica
Università degli Studi di Pisa
ugo@di.unipi.it
$\text{CAR}(\text{talk}, \text{switch}) \overset{\text{def}}{=} \overline{\text{talk}}.\text{CAR}(\text{talk}, \text{switch}) + \text{switch}(xt, xs).\text{CAR}(xt, xs)$.

$\text{BASE}_i \overset{\text{def}}{=} \text{talk}_i.\text{BASE}_i + \text{give}_i(xt, xs).\overline{\text{switch}_i}(xt, xs).\text{IDLEBASE}_i$

$\text{IDLEBASE}_i \overset{\text{def}}{=} \text{alert}_i.\text{BASE}_i$.

$\text{CENTRE}_1 \overset{\text{def}}{=} \overline{\text{give}_1}\langle \text{talk}_2, \text{switch}_2 \rangle.\overline{\text{alert}_2}.\text{CENTRE}_2$

$\text{CENTRE}_2 \overset{\text{def}}{=} \overline{\text{give}_2}\langle \text{talk}_1, \text{switch}_1 \rangle.\overline{\text{alert}_1}.\text{CENTRE}_1$.

$\text{SYSTEM} \overset{\text{def}}{=} \text{CAR}(\text{talk}_1, \text{switch}_1) \mid \text{BASE}_1 \mid \text{IDLEBASE}_2 \mid \text{CENTRE}_1$. 
Operational semantics of the TT-calculus

\[
\begin{align*}
  p &::= \text{nil} \mid \lambda x \cdot p \mid [x = y] p \mid p + p \mid \text{pl} p \mid (\lambda) p \mid ! p \\
  x &::= x \mid y \mid \overline{x} \overline{y}
\end{align*}
\]

\[
\begin{align*}
  \frac{\tau p}{\Rightarrow p} &\quad \frac{x(y)p \gamma(w)}{\Rightarrow p[w/y]} \quad \frac{w \neq \text{fn}(x) p}{\overline{x} y p \Rightarrow p} \\
  \frac{p \Rightarrow p'}{\Rightarrow p'} &\quad \frac{p \Rightarrow p'}{\Rightarrow p'} \quad \frac{p \Rightarrow p'}{\Rightarrow p'} \quad \frac{w \neq \text{fn}(q)}{\Rightarrow p/q}
\end{align*}
\]

\[
\begin{align*}
  \frac{p \overline{x} q \Rightarrow p' q'}{\Rightarrow pl q \\ q' \overline{x} q' / q} &\quad \frac{p \overline{y} q' \Rightarrow p' q'}{\Rightarrow pl q \\ q' \overline{y} q' / q} \\
  \frac{p \overline{x} \overline{y} p' \Rightarrow p' q}{\Rightarrow \lambda x \overline{y} p \Rightarrow \lambda x \overline{y} q} &\quad \frac{p \overline{x} \overline{y} p' \Rightarrow p' q}{\Rightarrow \lambda x \overline{y} p \Rightarrow \lambda x \overline{y} q}
\end{align*}
\]

Example

\[
\begin{align*}
  \overline{x} y p \overline{x} y p &\Rightarrow p \\
  \frac{(5) \overline{x} y p \overline{x} y p}{\Rightarrow pl q \overline{y} q \overline{y} q / q} &\quad \frac{\Rightarrow \lambda x \overline{y} p}{\Rightarrow \lambda x \overline{y} q}
\end{align*}
\]
Example 13.3 (Scope extrusion). We conclude this section by showing an example of the use of the rule system. Let us consider the following system:

\[ (((y)\overline{x}y.p) \mid q) \mid x(z).r \]

where \( p, q, r \) are \( \pi \)-calculus processes. The process \( (y)\overline{x}y.p \) would like to set up a private channel with \( x(z).r \), which however should remain hidden to \( q \). By using the inference rules of the operational semantics we can proceed in a goal-oriented fashion to find a derivation for the corresponding transition:

\[
\begin{align*}
((y)\overline{x}y.p) \mid q & \xrightarrow{a} s \\
\wedge (\text{CloseL}), \alpha = \tau, s = (w)(s_1 \mid r_1) & \quad (((y)\overline{x}y.p) \mid q) \xrightarrow{\overline{x}(w)} s_1, \quad x(z).r \xrightarrow{x(w)} r_1 \\
\wedge (\text{ParL}), s_1 = p_1 \mid q, w \notin \text{fn}(q) & \quad (y)\overline{x}y.p \xrightarrow{\overline{x}(w)} p_1, \quad x(z).r \xrightarrow{x(w)} r_1 \\
\wedge (\text{Open}), p_1 = p_2[w/y], w \notin \text{fn}(y).p & \quad \overline{x}y.p \xrightarrow{\overline{x}y} p_2, \quad x(z).r \xrightarrow{x(w)} r_1 \\
\wedge^* (\text{Out}) + (\text{In}), r_1 = r[w/z], p_2 = p, w \notin \text{fn}(z).r &
\end{align*}
\]

so we have:

\[
\begin{align*}
p_2 &= p \\
p_1 &= p_2[w/y] = p[w/y] \\
r_1 &= r[w/z] \\
s_1 &= p_1 \mid q = p[w/y] \mid q \\
s &= (w)(s_1 \mid r_1) = (w) ( (p[w/y] \mid q) \mid (r[w/z]) ) \\
\alpha &= \tau
\end{align*}
\]

In conclusion:

\[
\begin{align*}
((y)\overline{x}y.p) \mid q & \xrightarrow{\tau} (w) ( (p[w/y] \mid q) \mid (r[w/z]) ) \\
\end{align*}
\]

under the condition that \( w \) is fresh, i.e., that \( w \notin \text{fn}(q) \cup \text{fn}((y)p) \cup \text{fn}((z)r) \).
Bisimulation

**Early** bisimilar processes

Formally, a binary relation $S$ on $\pi$-calculus agents is a *strong early ground bisimulation* if:

$$\forall p, q. \ p \ S \ q \Rightarrow \begin{cases} \forall \alpha, p'. \text{ if } p \xrightarrow{\alpha} p' \text{ with } \alpha \neq x(y) \land \text{bn}(\alpha) \cap \text{fn}(q) = \emptyset, \\
\text{then } \exists q'. q \xrightarrow{\alpha} q' \text{ and } p' \ S q' \\
\forall x, y, p'. \text{ if } p \xrightarrow{x(y)} p' \text{ with } y \notin \text{fn}(q), \\
\text{then } \forall w. \exists q'. q \xrightarrow{x(y)} q' \text{ and } p'[w/y] \ S q'[w/y] \\
\text{(and vice versa)} \end{cases}$$

**Late** bisimilar processes

$$\forall x, y, p'. \text{ if } p \xrightarrow{x(y)} p' \text{ with } y \notin \text{fn}(q), \\
\text{then } \exists q'. q \xrightarrow{x(y)} q' \text{ and } \forall w. \ p'[w/y] \ S q'[w/y]$$

Processes which are **early** but not **late** bisimilar

$$p \overset{\text{def}}{=} x(y).\tau.\text{nil} + x(y).\text{nil} \quad q \overset{\text{def}}{=} p + x(y).[y = z]\tau.\text{nil}$$

whose transitions are (for any fresh name $u$):

$$p \xrightarrow{x(u)} \tau.\text{nil} \quad q \xrightarrow{x(u)} \tau.\text{nil}$$

$$p \xrightarrow{x(u)} \text{nil} \quad q \xrightarrow{x(u)} \text{nil}$$

$$q \xrightarrow{x(u)} [u = z]\tau.\text{nil}$$
Example 13.6 (Ground bisimilarities are not congruences). Let us consider the following agents:

\[
p \overset{\text{def}}{=} \tilde{x}. \text{nil} \mid x'(y). \text{nil} \quad q \overset{\text{def}}{=} \tilde{x}.x'(y). \text{nil} + x'(y). \tilde{x}. \text{nil}
\]

We leave the reader to check that the agents \( p \) and \( q \) are bisimilar (according to both early and late bisimilarities). Now, in order to show that ground bisimulations are not congruences, we define the following context:

\[
C[\cdot] = z(x').[\cdot]
\]

by plugging \( p \) and \( q \) inside the hole of \( C[\cdot] \) we get:

\[
C[p] = z(x').(\tilde{x}. \text{nil} \mid x'(y). \text{nil}) \quad C[q] = z(x').(\tilde{x}.x'(y). \text{nil} + x'(y). \tilde{x}. \text{nil})
\]

\( C[p] \) and \( C[q] \) are not early bisimilar (and thus not late bisimilar). In fact, suppose the name \( x \) is received on \( z \): we need to compare the agents

\[
p' \overset{\text{def}}{=} \tilde{x}. \text{nil} \mid x(y). \text{nil} \quad q' \overset{\text{def}}{=} \tilde{x}.x(y). \text{nil} + x(y). \tilde{x}. \text{nil}
\]

Now \( p' \) can perform a \( \tau \)-transition, but \( q' \) cannot.

The problem illustrated by the previous example is due to aliasing, and it appears often in programming languages with both global variables and parameter passing to procedures. It can be solved by defining a finer relation between agents called strong early full bisimilarity and defined as follows:

\[
p \simeq_E q \iff p\sigma \sim_E q\sigma \quad \text{for every substitution } \sigma
\]

where a substitution \( \sigma \) is a function from names to names that is equal to the identity function almost everywhere (i.e., it differs from the identity function only on a finite number of elements of the domain).

Analogously, we can define strong late full bisimilarity \( \simeq_L \) by letting

\[
p \simeq_L q \iff p\sigma \sim_L q\sigma \quad \text{for every substitution } \sigma
\]
Axioms for structural equivalence

\[
\begin{align*}
p + \text{nil} & \equiv p \\
p \mid \text{nil} & \equiv p \\
(x)\text{nil} & \equiv \text{nil} \\
[x = y]\text{nil} & \equiv \text{nil} \\
\end{align*}
\]

\[
\begin{align*}
p + q & \equiv q + p \\
p \mid q & \equiv q \mid p \\
(y)(x)p & \equiv (x)(y)p \\
[x = x]p & \equiv p \\
\end{align*}
\]

\[
\begin{align*}
(p + q) + r & \equiv p + (q + r) \\
(p \mid q) \mid r & \equiv p \mid (q \mid r) \\
(x)(p \mid q) & \equiv p \mid (x)q \text{ if } x \not\in \text{fn}(p) \\
p \mid !p & \equiv !p
\end{align*}
\]
Categorical Semantics of $\Pi$-Calculus

\[ \text{CCS: algebra + coalgebra \Rightarrow bialgebra} \]

$\Pi$-calculus: names/name generation in addition

Some problems

\[ \frac{P \xrightarrow{a} P'}{P \upharpoonright q \xrightarrow{a} P' \upharpoonright q} \]

This does not correspond to a coalgebra

States with nulls

\[ \text{FSets}$\overline{\text{sets}}$ equipped with actions as \]

\[ \text{Func}_\leq \text{Kozen's coordinates, or natural numbers} \]

\[ \text{FSets}$\overline{\text{a}}$\text{lgbras with finite-Kozen permutations as operations} \]

\[ \text{Nat}_f = \frac{1}{2} \Pi : \omega \rightarrow \omega \]

\[ \frac{x \in \omega}{\exists y (x \neq y)} \]

\[ \text{kernel} \]

\[ \text{monic operation: } \text{id}(x) = x \]

\[ \text{with axioms: } \Pi_\leq (T_2(x)) = (\Pi_\leq \circ T_2)(x) \]

\[ \text{Permutation algebra: } \langle A \to A, \Pi \circ \Pi \text{ is A-Aut}_f \rangle \]

\[ \text{or } b\alpha = \{ \Pi_\alpha(x) \mid \Pi \in \text{A-Aut}_f \} \]
Symmetry of an element of a Peirce Algebra

$G_A(a) = \{ \pi \in \text{Aut} F | \pi_A(a) = a \}$

Identity group of $X \subseteq \mathcal{F}

$fix(x) = \{ \pi \in \text{Aut} F | \pi x = x \}$

$x$ supports $a \iff fix(x) \leq G_A(a)$

$supp_A(a)$ is the minimal such $x$

$	ext{Alg}_\mathcal{F} \xrightarrow{FSA_{alg}} \text{cat}(\text{ary algebra})$

There is a theory morphism (injective) $-^{*+1} : \text{Alg}_\mathcal{F} \xrightarrow{-^{*+1}}$

$\pi_{*+1}(i) = \left\{ \begin{array}{ll} 0 & i = 0 \\ \pi(i)+1 & \text{otherwise} \end{array} \right.$

Right-shift operator

$S : \text{Alg}_\mathcal{F} \to \text{Alg}^{\text{cat}}$ the forgetful functor

associated to the right-shift morphism

Syntax of a calculus: De Bruijn indexes

$L := \text{ax}. \text{ll} / \text{ll} / \text{x}$

$d_{x, y} : (x, y) \equiv d_{n}(10)$

$x \xrightarrow{T} \ \ \ T(x) = \delta(x) + x \times x + \omega$

$\text{Fixpoint} \ x = T(x) \text{ is the initial algebra}$
Similarly for the $\mathcal{H}$-algebra:

\[
\mathcal{H}X = \mathcal{K} + \omega \times \mathcal{H}X + \omega \times \delta X + \mathcal{H}X + \delta X + \cdots
\]

We seek a double $\mathcal{H}$-algebra to satisfy:

\[
\mathcal{H}G(\lambda) = \mathcal{H}(\mathcal{H}G(\lambda) - \mathcal{H}\lambda + \delta \lambda)
\]

This may allow for fresh inputs, if $\mathcal{H}\lambda$ can be lifted.

**First option:** codephrases in $\mathcal{H}$.  $\mathcal{H}\lambda$ is not a sequence $\Rightarrow$ no lifting possible.

**Second option:** codephrases in $\mathcal{H}$.

Fix #1: Eliminate prefix.

$\text{P} + \text{H}(H)$ is a constant, with second more anxious sometimes well with recursion.

Fix #2: Move from $\mathcal{H} \alpha$ to $\mathcal{H} \beta$.

Add additional restrictions $\overline{\mathcal{H}p, \mathcal{H}q} \text{ or equivalent: } \overline{\mathcal{H}p} \Rightarrow \{ \overline{\mathcal{H}q} \}$.
Bialgebraic semantics with axioms

We stay in the category \( \text{Alg}(\mathbb{E}) \)
and functor \( P_{\Delta} \) defined by the SOS rules.

\[
\begin{align*}
\text{Alg}(\mathbb{E}) & \quad h \\
A & \longrightarrow B \\
\eta & \downarrow \\
P_{\Delta}(A) & \downarrow \\
P_{\Delta}(B) & \\
\text{algebra } B \text{ gives the} & \\
\text{states of the transition systems} & \\
i.e. B = \mathbb{T}_{\mathbb{E}, \text{E}} \text{ with } h \text{ surjective} & \\
\text{where } E \text{ are the structural axioms} & \\
\end{align*}
\]

The actual transition systems is just defined in \( \text{Set} \)

\[
\begin{align*}
\text{Set} & \quad |B| \\
\eta & \downarrow \\
P_{\Delta}(B) & \downarrow \\
\text{where } g \text{ is defined by } h \times h & \\
\text{as feminine rules plus role STRUCT:} & \\
P_{\Delta}(|B|) & \\
\text{where } p \equiv p' \quad p' \quad g' \quad g & \\
\text{implies } p \quad g & \\
\text{Resume that its diagrams contains in } \text{Set} \]
\[
\begin{align*}
\text{Set} & \quad |A| \quad \overset{h}{\longrightarrow} |B| \\
\text{where } g & \quad \downarrow \quad = \quad \downarrow g \quad \text{it can be lifted!} \\
P_{\Delta}(A) & \quad \overset{\text{lifted}}{\longrightarrow} P_{\Delta}(B) \\
\qquad P_{\Delta}(A) & \quad \overset{P_{\Delta}(h)}{\longrightarrow} P_{\Delta}(B) \\
\end{align*}
\]

In fact, since \( h, \Delta \) and \( P_{\Delta}(h) \) are homomorphisms and \( h \) is surjective, then also \( g \) is a homomorphism!

\[
\begin{align*}
g(b) &= g(h(a)) = (P_{\Delta}(h)(f(a))) = f(a)(P_{\Delta}(h)) = f(a)(g(h(a))) = g(b) \\
b &= h(a) \\
\text{commutes require.}
\end{align*}
\]
To prove that the diagram commutes, we have:

\[ f \circ h = f \circ g \]

\[ P_L(h) \rightarrow P_L(g) \]

Corresponds to show that the kernel of \( h \)

(i.e., the equivalence \( u = v \iff h(u) = h(v) \)) is a bisimulation.

To do this, it is enough to prove that \( P_L \), and with every D.S. proof, the axioms in \( P_L \) bisimulates, namely

\[ L = R \cdot E \]

implies that for every D.S. proof

\[ \text{prec} \quad \frac{\text{another D.S. proof}}{L \rightarrow q_1} \]

\[ \frac{\text{prec} \quad \frac{\text{with } \varphi_1 \equiv \varphi_2}{R \rightarrow q_2}}{\text{starting from the sequent of premises } \text{prec}} \]

and vice versa.

(i) The kernel of \( h \) is a bisimulation: \( p \equiv q \implies p \equiv q \) purely logical

(ii) The diagram commutes in \( k \)

(iii) The diagram can be lifted to \( k \)
Satisfaction via Right Kan Extension

Construction by Fiore et al.

Instead of \( \text{Alg}^\mathbb{I} \), we use \( \text{Set}^\mathbb{I} \): the presheaf where the exponent is \( \mathbb{I} \), the category of finite sets with injective mappings.

\( \text{Set}^\mathbb{I} \) is equivalent to \( \text{Alg}^\mathbb{I} \), since it has only monadic operations.

However, \( \text{Set}^\mathbb{I} \) has its own problems:

- No lifting to \( \text{Alg}^\mathbb{I} \). We could consider \( \text{Set}^F \) (all mappings), but in this case the LTS of \( \lambda \)-calculus could not be lifted to a coalgebra in \( \text{Set}^F \) for the usual reason: bisimulation is not a congruence for \( \lambda \)-calculus.

The forgetful functor \( \text{Set}^\mathbb{I} \rightarrow \text{Set}^\mathbb{I} \) \( \text{L}A \) \( \text{L}A \) \( \text{L}A \) \( \text{L}A \)

has not only a left but also a right adjoint (\( \text{Kan Extension} \) ) \( \text{K} \) \( \text{K} \) \( \text{K} \) \( \text{K} \)

\[ \text{Set}^F \xrightarrow{\text{L}A} \text{Set}^\mathbb{I} \]

Now a behavioral functor \( B \) in \( \text{Set}^\mathbb{I} \) can be lifted to \( \hat{B}(P) = [B(LP)] \) and a coalgebra \( (LP, \rho) \) for \( B \) can be lifted to a coalgebra \( (P, [\rho] \circ \eta_P) \) for \( \hat{B} \)

\[ \rho : [\rho] \rightarrow [B(LP)] = \hat{B}(P) \quad \eta_P : P \rightarrow [LP] \quad \text{the unity} \]

\[ \text{The adjunction}. \]
The lifted coalgebra has richer labels $(\sigma, \delta)$ where $\sigma \in \mathcal{F}[N_1 \to N_2]$ (possibly a fusion) and $\delta$ is a label of $B$, we have

$$N_1 \vdash p \xrightarrow{(\sigma, \delta)} N' \vdash p' \iff N_2 \vdash p[\delta] \xrightarrow{(\text{id}, \delta)} N' \vdash p'$$

where the second transition has a corresponding transition $N_2 \vdash p[\delta] \xrightarrow{\text{id}} N' \vdash p'$ in $(\mathcal{LPI}, P)$. Thus lifting adds transitions (saturation) which correspond to observe the effects of all the arrows in the index category. Thus similar states must behave corresponding lift for all such substitutions.
History: Dependent Automata

1. $P \xrightarrow{\alpha} q$
2. $(3) P \xrightarrow{(\alpha, \beta)} q [\beta]$

Infinite branching.

Choose one for finite state.

Different than all the others.

$p \xrightarrow{\bar{\alpha} (\beta)} q$

Choose the same as $\beta$.

$p \xrightarrow{\overline{\alpha} (\beta)} q$

OK, essentially, doesn't work for minimization.

HD automata: local names

\[ \begin{array}{c}
q \quad p \\
\uparrow \quad \uparrow \\
\phi \quad \phi
\end{array} \]

Injection: $n_q \to n_p \cup \{ \phi \}$

Bisimulace: set of states with partial bijection.

Group of symmetries: similar to itself with a p. bijection different from id.

Categorical theory based on named sets

\[ N = \langle Q_N, S_N \rangle \]

Set $\to \Sigma N \to \text{Symmet} \times \text{set group}$

Named sets are compact representations of objects.

Categorical structures in named set with Alg but smaller objects.