Concurrency 3

CCS

Jean-Jacques Lévy

jeanjacqueslevy.net/dea
Minimal language for concurrency

The $\lambda$-calculus is a minimal language for functional languages. It can also be used as a basis for imperative languages (via continuations).

What is a minimal language for concurrent processes?

- CCS [Milner]
- $\pi$-calculus [Milner, Parrow, Walker, Sangiorgi]
- CSP [Hoare]
- Petri nets
- Mazurkiewicz traces
- Events structures $\Leftrightarrow$ True concurrency [Winskel]
- IO-automatas [Lynch, Tuttle]
Language

\[
\begin{align*}
  a, b, c & ::= \text{(channel) names} \\
  \overline{a}, \overline{b}, \overline{c} & ::= \text{co-names} \quad \overline{a} = a \\
  \alpha & ::= a \mid \overline{a} \mid \tau \\
  P, Q, R & ::= 0 \mid \alpha.P \mid P + Q \mid (P \mid Q) \mid (\nu \alpha)P \mid K \\
  K \overset{\text{def}}{=} P & ::= \text{constant definitions}
\end{align*}
\]

\[\mathcal{Act} = \{a, b, c, \ldots\} \cup \{\overline{a}, \overline{b}, \overline{c}, \ldots\} \cup \{\tau\}\]

Notation: \(\alpha\) for \(\alpha.0\)

- 0 null process
- \(\alpha.P\) sequential action
- \(P + Q\) non-deterministic (external) choice
- \(P \mid Q\) parallel composition
- \((\nu \alpha)P\) restriction on \(\alpha\)
- \(K\) (recursively defined) constant
Examples (coffee machine revisited)

\[ P_0 = A \]
\[ A = c.(k.d.A + t.d.A) \]
\[ P'_0 = B \]
\[ B = c.C \]
\[ C = (k.d.D + t.d.D) \]
\[ D = c.C \]

\[ P''_0 = E \]
\[ E = (c.k.d.E + c.t.d.E) \]
\[ P'''_0 = F \]
\[ F = c + (c.k.d.F + c.t.d.F') \]

Interaction with coffee machine

\[ P_0 | \overline{c.k.d} \quad P_0 | \overline{c.k.d} | \overline{c.t.d} \]
\[ P_0 | \text{Client1} \quad P'_0 | \text{Client1} \quad P''_0 | \text{Client1} \quad P'''_0 | \text{Client1} \]
\[ P_0 | \text{Client2} \quad P''_0 | \text{Client2} | \text{Client2} \quad P_0 | \text{Client1} | \text{Client2} \]

where

\[ \text{Client1} \overset{\text{def}}{=} \overline{c.k.d}.\text{Client1} \quad \text{Client2} \overset{\text{def}}{=} \overline{c.t.d}.\text{Client2} \]
Semantics (SOS)

[Act] $\alpha.P \xrightarrow{\alpha} P$

[Sum1] $\frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$

[Sum2] $\frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$

[Par1] $\frac{P \xrightarrow{\alpha} P'}{P \parallel Q \xrightarrow{\alpha} P' \parallel Q}$

[Par2] $\frac{Q \xrightarrow{\alpha} Q'}{P \parallel Q \xrightarrow{\alpha} P \parallel Q'}$

[Com] $\frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\overline{\alpha}} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'}$

[Res] $\frac{P \xrightarrow{\alpha} P' \quad \alpha \not\in \{a, \overline{a}\}}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'}$

[Rec] $\frac{P \xrightarrow{\alpha} P' \quad K \overset{\text{def}}{=} P}{K \xrightarrow{\alpha} P'}$

$\tau$ → internal move
$\xrightarrow{\alpha}$ (\(\alpha \neq \tau\)) interaction on external \(\alpha\)-channel

By convention, $\xrightarrow{a}$ input on \(a\)-channel and $\xrightarrow{\overline{a}}$ output on \(a\)-channel

Sum \(\neq\) internal choice $P + Q \xrightarrow{\tau} P$ or $P + Q \xrightarrow{\tau} Q$. 
At present time, no values passed on communication channels. (see later for value passing calculi)

No buffering in communications. Different from TCP sockets, from Kahn/MacQueen flow systems. ⇒ communication by rendez-vous. ≡ more basic calculus.

Rendez-vous exist in Occam, Ada, CML, Ocaml’s processes.
Theorem 1  Following relations hold.

\[
\begin{align*}
P + 0 & \sim P \\
P + Q & \sim Q + P \\
(P + Q) + R & \sim P + (Q + R) \\
P + P & \sim P
\end{align*}
\]

\[
\begin{align*}
P | 0 & \sim P \\
P | Q & \sim Q | P \\
(P | Q) | R & \sim P | (Q | R)
\end{align*}
\]

\[
\begin{align*}
(\nu a)(P | Q) & \sim ((\nu a)P) | Q & \text{if } a \text{ not free in } Q \\
(\nu a)(\nu b)P & \sim (\nu b)(\nu a)P \\
(\nu a)P & \sim (\nu b)P\{b/a\} & \text{if } b \text{ not bound in } Q \\
(\nu a)\alpha.P & \sim 0 & \text{if } \alpha = a \text{ or } \alpha = \overline{a} \\
(\nu a)\alpha.P & \sim \alpha.(\nu a).P & \text{otherwise}
\end{align*}
\]

\[
K \sim P & \text{ if } K \overset{\text{def}}{=} P
\]
CCS and strong bisimulation (2/4)

Proof of previous theorem

- $P + 0 \sim P$. Take $\mathcal{R} = \{(P + 0, P), (P, P + 0), (P, P)\}$ and show $\mathcal{R}$ is a bisimulation.
  Let $P + 0 \xrightarrow{\alpha} P'$. Then $P \xrightarrow{\alpha} P'$ by rule [Sum1] since $0 \xrightarrow{\alpha} P'$ is not possible. And $P' \mathcal{R} P'$.
  Conversely let $P \xrightarrow{\alpha} P'$. Then $P + 0 \xrightarrow{\alpha} P'$ by rule [Sum1]. And again $P' \mathcal{R} P'$.

- $P + Q \sim Q + P$. Show following $\mathcal{R}$ is a bisimulation. Take $\mathcal{R} = \{P + Q, Q + P, (P, P)\}$.
  Let $P + Q \xrightarrow{\alpha} S$.
    - Case 1: let $P + Q \xrightarrow{\alpha} S$ using [Sum1]. Then $P \xrightarrow{\alpha} S$.
      But $Q + P \xrightarrow{\alpha} S$ using [Sum2].
      QED since $S \mathcal{R} S$.
    - Case 2: let $P + Q \xrightarrow{\alpha} S$ using [Sum2]. Then $Q \xrightarrow{\alpha} S$.
      But $Q + P \xrightarrow{\alpha} S$ using [Sum1].
      QED since $S \mathcal{R} S$.
  Conversely let $Q + P \xrightarrow{\alpha} S$. QED by symmetry.
CCS and strong bisimulation (3/4)

Proof of theorem (continued)

- \((P + Q) + R \sim P + (Q + R)\). Show following \(R\) is a bisimulation.
  Take \(R = \{(P + Q) + R, P + (Q + R), (P, P)\}\).

Let \((P + Q) + R \xrightarrow{\alpha} S\).

  - Case 1: let \((P + Q) \xrightarrow{\alpha} S\) using [Sum1].
    - Case 1.1: let \(P \xrightarrow{\alpha} S\) using [Sum1].
      Then \(P + (Q + R) \xrightarrow{\alpha} S\) by [Sum1].
      QED since \(S \not\sim S\).
    - Case 1.2: Let \(Q \xrightarrow{\alpha} S\). Then \((Q + R) \xrightarrow{\alpha} S\) by [Sum1], and \(P + (Q + R) \xrightarrow{\alpha} S\) by [Sum2].
      QED since \(S \not\sim S\).

  - Case 2: Let \(R \xrightarrow{\alpha} S\) by [Sum2]. Then \((Q + R) \xrightarrow{\alpha} S\) by [Sum2], and \(P + (Q + R) \xrightarrow{\alpha} S\) by [Sum2].
    QED since \(S \not\sim S\).

By symmetry when \(P + (Q + R) \xrightarrow{\alpha} S\).

- other equations . . .

Exercice 1 Give full proof of theorem.
Theorem 2 [Expansion]
\[
a.P \parallel b.Q \sim a.(P \parallel b.Q) + b.(a.P \parallel Q) \\
\]
\[
a.P \parallel \bar{a}.Q \sim a.(P \parallel \bar{a}.Q) + \bar{a}.(a.P \parallel Q) + \tau.(P \parallel Q)
\]

Exercice 2 Prove it.

Concurrency in CCS relies on interleaving. Never two actions occur at same time. Different from “true concurrency”.

Exercice 3 Draw LTS for following processes:
\[
P = (\nu a)((a + b) \parallel \bar{a})
\]
\[
K_1 \overset{\text{def}}{=} a.(\tau.K_1 + b) + \tau.a.K_1 \\
K_2 \overset{\text{def}}{=} \tau.(\nu a)(a \parallel (\bar{a} + b)) + c.K_3
\]
\[
K_3 \overset{\text{def}}{=} d.K_3
\]

Exercice 4 Draw LTS for \((\nu c)(K_1 \parallel K_2)\) where
\[
K_1 \overset{\text{def}}{=} a.\bar{c}.K_1 \\
K_2 \overset{\text{def}}{=} b.c.K_2
\]

Exercice 5 Give a CCS term for boolean semaphores.
Exercice 6 Give a CCS term for \(n\)-ary semaphores.
Strong bisimulation and congruence

**Theorem 3**  Strong bisimulation $\sim$ is a congruence. Namely:

$$P \sim Q \Rightarrow C[P] \sim C[Q] \text{ for any context } C[\cdot].$$

**Exercice 7**  Prove it.

This means that $\sim$ can be used as standard equations.

**Exercice 8**  Prove by using equations of Theorems 1 and 2 that:

$$(\nu b)(a.(b \mid c) + \tau.(b \mid \overline{b}.c)) \sim a.c + \tau.\tau.c$$

**Exercice 9**  Show $K \mid K \sim K$ when $K \overset{\text{def}}{=} a.K$.

**Exercice 10**  Show $K \sim K'$ when $K \overset{\text{def}}{=} a.K$ and $K' \overset{\text{def}}{=} a.a.K'$.

**Exercice 11**  Show $K \sim a.K'$ when $K \overset{\text{def}}{=} a.b.K$ and $K' \overset{\text{def}}{=} b.a.K'$.

**Exercice 12**  Show that $a.(b + c) \not\sim ab + ac$. 

11