Concurrency: Theory, Languages and Programming

– CCS Example –

Session 5 – November 20, 2002

Uwe Nestmann

EPFL-LAMP

$\mathcal{N}$	channels	$a, b, c \ldots$
$\mathcal{V}$	values	v, w
$\mathcal{X}$	variables	x, y, z
$\mathcal{A}$	actions	$\mu \ ::= \ \overline{a}\langle v \rangle \ \mid \ a(x) \ \mid \ \tau$

"negative" actions $\overline{a}\langle v \rangle$: *send name* v *over channel* a.

"positive" actions $a(x)$: *receive any value, say* v*, over channel* a *and "bind the result" to variable* x.

Binding results in *substitution* $[^v/_x]$
of the formal parameter x by the actual parameter v.

polyadic communication $\overline{a}\langle \vec{v} \rangle$ and $a(\vec{x})$ (with $\vec{x}$ pairwise different) *transmit many values at a time.*

Value-Passing: Semantics I

directly: via LTS

$\ldots$ TAU: $\tau.P \xrightarrow{\tau} P$ OUT: $\overline{a}\langle v \rangle.P \xrightarrow{\overline{a}\langle v \rangle} P$

$$\text{INP:} \ \frac{v \in \mathcal{V}}{a(x).P \xrightarrow{av} [^v/_x]P}$$

$$\text{COMM:} \ \frac{P \xrightarrow{\overline{a}\langle v \rangle} P' \qquad Q \xrightarrow{av} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

Value-Passing: Semantics II

indirectly: via translation

$$\begin{array}{lcl}
[\![\ ]\!] : \mathcal{P}^{\mathsf{VP}} & \to & \mathcal{P} \\
\hline
[\![\, \overline{a}\langle v \rangle.P \,]\!] & \stackrel{\text{def}}{=} & \overline{a_v}.[\![\, P \,]\!] \\
[\![\, a(x).P \,]\!] & \stackrel{\text{def}}{=} & \sum\limits_{v \in \mathcal{V}} a_v.[\![\, [^v/_x]P \,]\!] \\
\vdots & & \\
[\![\, P_1 \mid P_2 \,]\!] & \stackrel{\text{def}}{=} & [\![\, P_1 \,]\!] \mid [\![\, P_2 \,]\!] \\
\vdots & & \\
[\![\, A\langle \vec{v} \rangle \,]\!] & \stackrel{\text{def}}{=} & A\langle \vec{v} \rangle
\end{array}$$

Buffers in New Clothes . . .

$$\begin{array}{lcl}
\mathcal{N} & := & \{ \text{ in, out } \} \\
\mathcal{V} & := & \{ \, 0, 1 \, \} \\
s & \in & \{\epsilon\} \cup \mathcal{V} \\
\vec{a} & := & \text{in, out} \\
\text{Buff}_s^{(1)} & : & \text{1-place buffer containing } s \\
\hline
\text{Buff}_\epsilon^{(1)}(\,\vec{a}\,) & \stackrel{\text{def}}{=} & \text{in}(x).\text{Buff}_x^{(1)}\langle\,\vec{a}\,\rangle \\
\text{Buff}_v^{(1)}(\,\vec{a}\,) & \stackrel{\text{def}}{=} & \overline{\text{out}}\langle v \rangle.\text{Buff}_\epsilon^{(1)}\langle\,\vec{a}\,\rangle
\end{array}$$

- Observe how much nicer name/value-passing is :-)

Bound and Free Names

- $(\boldsymbol{\nu} x)\, P$ and $a(x).P$ **bind** x in P
- x occurs **bound** in P, if it occurs in a subterm $(\boldsymbol{\nu} x)\, Q$ or $a(x).P$ of P
- x occurs **free** in P, if it occurs without enclosing $(\boldsymbol{\nu} x)\, Q$ or $a(x).P$ in P
- Note the use of parentheses (round brackets).
- Define $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$ inductively on $\mathcal{P}$ (sets of free/bound names of P) . . .

Scheduler, Informally [Mil99, § 3.6]

- a set of processes $P_i, 1 \leq i \leq n$ is to be scheduled
- P_i starts by signalling $\overline{a_i}$ to the scheduler
- P_i completes by signalling $\overline{b_i}$ to the scheduler
- each P_i *must not* run two tasks at a time
- tasks of different P_i *may* run at the same time
- a_i are required to occur cyclically (initially, 1 starts)
- for each i, a_i and b_i must occur cyclically
- maximal "progress":
the scheduling must permit any of the buttons to be pressed

Formal "Implementation" [§ 7.3]

$$\begin{array}{lcl}
A(\,a, b, c, d\,) & \stackrel{\text{def}}{=} & a.c.b.\overline{d}.A \\
\hline
A(\,a, b, c, d\,) & \stackrel{\text{def}}{=} & a.C\langle\,a, b, c, d\,\rangle \\
C(\,a, b, c, d\,) & \stackrel{\text{def}}{=} & c.B\langle\,a, b, c, d\,\rangle \\
B(\,a, b, c, d\,) & \stackrel{\text{def}}{=} & b.D\langle\,a, b, c, d\,\rangle \\
D(\,a, b, c, d\,) & \stackrel{\text{def}}{=} & \overline{d}.A\langle\,a, b, c, d\,\rangle \\
\hline
\multicolumn{3}{l}{\vec{a} := a_1 \ldots, a_n, \ \ \vec{b} := b_1 \ldots, b_n \ \ \vec{c} := c_1 \ldots, c_n} \\
A_i(\,\vec{a}, \vec{b}, \vec{c}\,) & \stackrel{\text{def}}{=} & A\langle\,a_i, b_i, c_i, c_{i\ominus_n 1}\,\rangle \\
B_i(\,\vec{a}, \vec{b}, \vec{c}\,) & \stackrel{\text{def}}{=} & B\langle\,a_i, b_i, c_i, c_{i\ominus_n 1}\,\rangle \\
C_i(\,\vec{a}, \vec{b}, \vec{c}\,) & \stackrel{\text{def}}{=} & C\langle\,a_i, b_i, c_i, c_{i\ominus_n 1}\,\rangle
\end{array}$$