Concurrency: Languages, Programming and Theory

– CCS –

Session 4 – November 12, 2003

Uwe Nestmann

EPFL-LAMP

Plan

- Session 4
  - from λ-calculus to CCS: towards concurrency
  - Structural Operational Semantics (SOS)
- Session 5
  - examples ...
  - ... using the Scala-library
- Session 6
  - from CCS to π-calculus: pragmatics, syntax, semantics
  - more SOS
- Session 7
  - from π-calculus to Java ... and back again
  - ... using the Scala-library

Foundational Calculi ?

We are interested in the foundations of programming.
We use “foundational” mini-languages as vehicles that guide our intuition and style of expression. When does such a mini-language deserve to be called a “calculus”?

- few primitives
- mathematically tractable
  - *calculate* computational steps
  - notion of equivalence
- *computationally complete* (Turing, URM, GOTO, . . .)
- “*naturally complete*”: design of programming languages
  - easily “extensible” via *encodings*
  - *higher-order* principles

λ-Calculus

- **Syntax** (for example)
  a BNF-grammar generates the set of expressions ...

$$M, N \quad ::= \quad x \quad \Big| \quad \lambda x.N \quad \Big| \quad M N$$

- **Semantics** (for example)
  a set of inference rules generates (and controls) the possible reductions of terms

$$(\beta)\ \frac{}{(\lambda x.N)M \;\rightarrow\; [M/x]N}$$

$$(\mathrm{FUN})\ \frac{M \;\rightarrow\; M'}{M N \;\rightarrow\; M' N} \qquad\qquad (\mathrm{ARG})\ \frac{N \;\rightarrow\; N'}{M N \;\rightarrow\; M N'}$$

Concurrency?

- parallelism: ≥ 1 independent threads of control
- distribution:
  - logical concurrency
  - physical concurrency
  - failures
- synchronization / cooperation / coordination
  communication

$\Longrightarrow$ foundational calculus for (just) concurrency ?

Functional vs Concurrent

- functional / reduction systems:
  - reduce a term to value form
  - *only* the resulting value is interesting
  - observation after termination
- concurrent / reactive systems:
  - describe the possible interactions *during* evaluation
  - the resulting value is *not* (necessarily) interesting
  - observation through and during interaction

The notion of *interaction* (*communication*) is important !

Hoare (CSP) and Milner (CCS) proposed handshake-communication as the primitive form of interaction.

Functional vs Concurrent

	functional	concurrent
determinism	possible	?
confluence	wanted/needed	?
termination	?	?
foundation	λ	CCS, π, (Petri nets, ...)
ff-language	ML, Scala, ...	Pict, Join, Scala, ...

CCS

$\mathcal{I}$	process identifiers	$A, B \ldots$
$\mathcal{N}$	names	$a, b, c \ldots$
$\overline{\mathcal{N}}$	co-names	$\overline{a}, \overline{b}, \overline{c} \ldots$
$\mathcal{L}$	labels (buttons)	metavariables $\lambda \ldots \in \mathcal{L} := \mathcal{N} \cup \overline{\mathcal{N}}$
$\mathcal{A}$	actions	metavariables $\mu, \beta \ldots \in \mathcal{L} \cup \{\tau\}$

- visible/external actions: labels
- invisible/internal actions: τ
- **finite sequences** $\vec{a}$ for *names* $a_1 \ldots, a_n$ (*not co*-names!)
- **parametric processes** $A\langle\, a, c\,\rangle$ with *name* parameters (neither *co*-names, nor labels, . . .)

Sequential Process Expressions (I)

Definition: The sets $\mathcal{P}^{\text{seq}}$ and $\mathcal{M}^{\text{seq}}$ of sequential process expressions is defined (precisely) by the following BNF-syntax:

$$\begin{array}{lcl} P & ::= & A\langle\,\vec{a}\,\rangle \quad \big| \quad M \\ M & ::= & \mathbf{0} \quad \big| \quad \mu.P \quad \big| \quad M + M \end{array}$$

We use $P, Q, P_i \ldots$ to stand for *process expressions*, while M, M_i always stand for *choices* or *summations*. We also use the abbreviation

$$\sum_{i \in I} \mu_i.P_i := \mu_1.P_1 + \ldots + \mu_n.P_n$$

where I is the finite indexing set $\{1\ldots,n\}$. Note that then the order of summands is not fixed.

Sequential Process Expressions (II)

- each process identifier A is assumed
  to have a **defining equation** (note the brackets)

$$A(\,\vec{a}\,) \stackrel{\mathrm{def}}{=} M$$

  where M is a summation, $\vec{a}$ is (or: includes) $\mathrm{fn}(M)$.
  Note: $\vec{a}$ does only include *names* ($\in \mathcal{N}$), not co-names!

- $\mathrm{fn}(P)$: the set of all of the **(free) names** of P

- $A\langle\,\vec{b}\,\rangle$ means the same as $[^{\vec{b}}/_{\vec{a}}]M$

- **substitution** $[^{\vec{b}}/_{\vec{a}}]P$ (for matching $\vec{b}$ and $\vec{a}$)
  replaces *all* occurrences of a_i in P by b_i.

Free Names, Inductively

Definition: The set $\mathrm{fn}(P)$ is defined inductively by:

$$
\begin{array}{lcl}
\mathrm{fn}(\mu) & \stackrel{\mathrm{def}}{=} & \begin{cases} \{b\} & \text{if } \mu = b \\ \{b\} & \text{if } \mu = \bar{b} \\ \emptyset & \text{if } \mu = \tau \end{cases} \\
\hline
\mathrm{fn}(\mathbf{0}) & \stackrel{\mathrm{def}}{=} & \emptyset \\
\mathrm{fn}(\mu.P) & \stackrel{\mathrm{def}}{=} & \mathrm{fn}(\mu) \cup \mathrm{fn}(P) \\
\mathrm{fn}(M_1 + M_2) & \stackrel{\mathrm{def}}{=} & \mathrm{fn}(M_1) \cup \mathrm{fn}(M_2) \\
\hline
\mathrm{fn}(A\langle \vec{a} \rangle) & \stackrel{\mathrm{def}}{=} & \{\vec{a}\}
\end{array}
$$

Substitution, Inductively

Definition:

$$
\begin{array}{lcl}
[{}^{b}\!/\!_{c}]\mu & \stackrel{\text{def}}{=} & \begin{cases} b & \text{if } \mu = c \\ \overline{b} & \text{if } \mu = \overline{c} \\ \mu & \text{otherwise} \end{cases} \\
\hline
[{}^{b}\!/\!_{c}]\mathbf{0} & \stackrel{\text{def}}{=} & \mathbf{0} \\
[{}^{b}\!/\!_{c}](\mu.P) & \stackrel{\text{def}}{=} & [{}^{b}\!/\!_{c}]\mu.[{}^{b}\!/\!_{c}]P \\
[{}^{b}\!/\!_{c}](M_1 + M_2) & \stackrel{\text{def}}{=} & [{}^{b}\!/\!_{c}]M_1 + [{}^{b}\!/\!_{c}]M_2 \\
\hline
[{}^{b}\!/\!_{c}](A\langle\,\vec{a}\,\rangle) & \stackrel{\text{def}}{=} & A\langle\,[{}^{b}\!/\!_{c}]\vec{a}\,\rangle
\end{array}
$$

Simultaneous Substitution, Inductively

Definition:

Let $\vec{b} = b_1 \ldots, b_n$ and $\vec{c} = c_1 \ldots, c_n$.

$$[{}^{\vec{b}}/_{\vec{c}}]\mu \stackrel{\text{def}}{=} \begin{cases} b_i & \text{if } \exists 1 \leq i \leq n \text{ with } \mu = c_i \\ \overline{b_i} & \text{if } \exists 1 \leq i \leq n \text{ with } \mu = \overline{c_i} \\ \ldots & \text{otherwise} \end{cases}$$

$$[{}^{\vec{b}}/_{\vec{c}}]\mathbf{0} \stackrel{\text{def}}{=} \mathbf{0}$$

$$[{}^{\vec{b}}/_{\vec{c}}](\mu.P) \stackrel{\text{def}}{=} [{}^{\vec{b}}/_{\vec{c}}]\mu.[{}^{\vec{b}}/_{\vec{c}}]P$$

$$[{}^{\vec{b}}/_{\vec{c}}](M_1 + M_2) \stackrel{\text{def}}{=} [{}^{\vec{b}}/_{\vec{c}}]M_1 + [{}^{\vec{b}}/_{\vec{c}}]M_2$$

$$[{}^{\vec{b}}/_{\vec{c}}](A\langle\,\vec{a}\,\rangle) \stackrel{\text{def}}{=} A\langle\,[{}^{\vec{b}}/_{\vec{c}}]\vec{a}\,\rangle$$

Example: 1-Place Binary Buffer

$$\mathcal{N} \quad := \quad \{\ \mathsf{in}_i, \mathsf{out}_i \mid i \in \{0,1\}\ \}$$

$$s \quad \in \quad \{\epsilon, 0, 1\}$$

$$\vec{a} \quad := \quad \mathsf{in}_0, \mathsf{in}_1, \mathsf{out}_0, \mathsf{out}_1$$

$$\mathsf{Buff}_s^{(1)} \quad : \quad \text{1-place buffer containing } s$$

$$\mathsf{Buff}^{(1)}(\,\vec{a}\,) \quad \stackrel{\text{def}}{=} \quad \sum_{i\in\{0,1\}} \mathsf{in}_i.\mathsf{Buff}_i^{(1)}\langle\,\vec{a}\,\rangle$$

$$\mathsf{Buff}_i^{(1)}(\,\vec{a}\,) \quad \stackrel{\text{def}}{=} \quad \overline{\mathsf{out}_i}.\mathsf{Buff}^{(1)}\langle\,\vec{a}\,\rangle$$

Example: 2-Place Binary Buffer

$$\begin{array}{ccl}
\mathcal{N} & := & \{\ \mathsf{in}_i, \mathsf{out}_i \mid i \in \{0,1\}\ \} \\
s & \in & \{\epsilon, 0, 1, 00, 01, 10, 11\} \\
\vec{a} & := & \mathsf{in}_0, \mathsf{in}_1, \mathsf{out}_0, \mathsf{out}_1 \\
\mathsf{Buff}_s^{(2)} & : & \text{2-place buffer containing } s \\
\hline
\mathsf{Buff}^{(2)}(\,\vec{a}\,) & \stackrel{\mathrm{def}}{=} & \sum_{i\in\{0,1\}} \mathsf{in}_i.\mathsf{Buff}_i^{(2)}\langle\,\vec{a}\,\rangle \\
\mathsf{Buff}_i^{(2)}(\,\vec{a}\,) & \stackrel{\mathrm{def}}{=} & \overline{\mathsf{out}_i}.\mathsf{Buff}^{(2)}\langle\,\vec{a}\,\rangle + \sum_{j\in\{0,1\}} \mathsf{in}_j.\mathsf{Buff}_{ji}^{(2)}\langle\,\vec{a}\,\rangle \\
\mathsf{Buff}_{ij}^{(2)}(\,\vec{a}\,) & \stackrel{\mathrm{def}}{=} & \overline{\mathsf{out}_j}.\mathsf{Buff}_i^{(2)}\langle\,\vec{a}\,\rangle \\
\hline
\end{array}$$

- modify $\mathsf{Buff}_s^{(2)}$ to release values in either order
- write an analogous definition for $\mathsf{Buff}_s^{(3)}$ ...

Labeled Transition Systems

Definition:
An **LTS** $(\mathcal{Q}, \mathcal{T})$ over an **action alphabet** $\mathcal{A}$:

- a set of **states** $\mathcal{Q} = \{q_0, q_1 \ldots\}$
- a ternary **transition relation** $\mathcal{T} \subseteq (\mathcal{Q} \times \mathcal{A} \times \mathcal{Q})$

A transition $(q, \mu, q') \in \mathcal{T}$ is also written $q \xrightarrow{\mu} q'$.

If $q \xrightarrow{\mu_1} q_1 \cdots \xrightarrow{\mu_n} q_n$ we call q_n a **derivative** of q.

LTSs are automata, but ignoring starting and accepting states.
Transition Graphs are useful . . .

LTS - Sequential Expressions

Definition: The LTS $(\mathcal{P}^{\text{seq}}, \mathcal{T})$ of sequential process expressions over $\mathcal{A}$ has $\mathcal{P}^{\text{seq}}$ as states, and its transitions $\mathcal{T}$ are precisely generated by the following rules:

$$\text{PRE: } \mu.P \xrightarrow{\mu} P$$

$$\text{SUM}_1\text{: } \frac{M_1 \xrightarrow{\mu} M_1'}{M_1 + M_2 \xrightarrow{\mu} M_1'} \qquad \text{SUM}_2\text{: } \frac{M_2 \xrightarrow{\mu} M_2'}{M_1 + M_2 \xrightarrow{\mu} M_2'}$$

$$\text{DEF: } \frac{[{}^{\vec{b}}/_{\vec{a}}]M_A \xrightarrow{\mu} P'}{A\langle \vec{b} \rangle \xrightarrow{\mu} P'} \quad \text{IF } A(\vec{a}) \stackrel{\text{def}}{=} M_A$$

Note that transition under prefix is not allowed/included.

Concurrent Process Expressions (I)

<u>**Definition:**</u> The set $\mathcal{P}$ of concurrent process expressions is defined (precisely) by the following BNF-syntax:

$$
\begin{array}{rcl}
P & ::= & A\langle\,\vec{a}\,\rangle \quad \Big| \quad M \quad \Big| \quad P|P \quad \Big| \quad (\boldsymbol{\nu}a)\,P \\
M & ::= & \mathbf{0} \quad \Big| \quad \alpha.P \quad \Big| \quad M+M
\end{array}
$$

We use P, Q, P_i to stand for process expressions.

- ☐ $(\boldsymbol{\nu}a)\,P$ restricts the scope of a to P
- ☐ $(\boldsymbol{\nu}ab)\,P$ abbreviates $(\boldsymbol{\nu}a)\,(\boldsymbol{\nu}b)\,P$

Concurrent Process Expressions (II)

- precedence: unary binds tighter than binary

$$
\begin{aligned}
(\boldsymbol{\nu} a)\, P \mid Q &= ((\boldsymbol{\nu} a)\, P) \mid Q \\
a.P + M &= (a.P) + M
\end{aligned}
$$

$$
[{}^{a}\!/\!{}_{b}]M_1 + M_2 = ([{}^{a}\!/\!{}_{b}]M_1) + M_2
$$

- what about:

$$
\begin{aligned}
P \,|\, Q + R &\stackrel{?}{=} (P \,|\, Q) + R \\
P \,|\, Q + R &\stackrel{?}{=} P \,|\, (Q + R)
\end{aligned}
$$

Bound and Free Names

- $(\boldsymbol{\nu} a)\, P$ **binds** a in P
- a occurs **bound** in P,
  if it occurs in a subterm $(\boldsymbol{\nu} a)\, Q$ of P
- a occurs **free** in P,
  if it occurs without enclosing $(\boldsymbol{\nu} a)\, Q$ in P
- Define $\mathrm{fn}(P)$ and $\mathrm{bn}(P)$ inductively on $\mathcal{P}$
  (sets of free/bound names of P):

$$\begin{aligned}
\mathrm{fn}(P_1|P_2) &\stackrel{\mathrm{def}}{=} \mathrm{fn}(P_1) \cup \mathrm{fn}(P_2) \\
\mathrm{bn}(P_1|P_2) &\stackrel{\mathrm{def}}{=} \mathrm{bn}(P_1) \cup \mathrm{bn}(P_2) \\
\dots \\
\mathrm{fn}((\boldsymbol{\nu} a)\, P) &\stackrel{\mathrm{def}}{=} \mathrm{fn}(P) \setminus \{a\} \\
\mathrm{bn}((\boldsymbol{\nu} a)\, P) &\stackrel{\mathrm{def}}{=} \mathrm{bn}(P) \cup \{a\}
\end{aligned}$$

α-Conversion & Substitution

- **substitution** $[^{\vec{b}}/_{\vec{a}}]P$ (for matching $\vec{b}$ and $\vec{a}$) replaces *all* **free** occurrences of a_i in P by b_i.

  $[^{b}/_{a}](\boldsymbol{\nu} b)\, b.a =?$

- α**-conversion**, written $=_\alpha$:
  conflict-free **renaming of bound names**
  (no new name-bindings shall be generated)

- **substitution** $[^{\vec{b}}/_{\vec{a}}]P$ (for matching $\vec{b}$ and $\vec{a}$, where $\vec{a}$ p.w.d.) replaces *all* **free** occurrences of a_i in P by b_i,
  possibly enforcing α-conversion.

Examples

$$
\begin{aligned}
(\boldsymbol{\nu} a)\,(\overline{a}.\mathbf{0}|b.\mathbf{0}) \quad &=_\alpha \quad (\boldsymbol{\nu} c)\,(\overline{c}.\mathbf{0}|b.\mathbf{0}) \\
&=_\alpha \quad (\boldsymbol{\nu} b)\,(\overline{b}.\mathbf{0}|b.\mathbf{0})
\end{aligned}
$$

$$
\begin{aligned}
[{}^{a}\!/\!_{b}](\ (\boldsymbol{\nu} b)\,\overline{b}.\mathbf{0}\ |\ b.\mathbf{0}\ ) \quad &=_\alpha \quad (\ (\boldsymbol{\nu} b)\,\overline{a}.\mathbf{0}\ |\ a.\mathbf{0}\ ) \\
&=_\alpha \quad (\ (\boldsymbol{\nu} b)\,\overline{b}.\mathbf{0}\ |\ a.\mathbf{0}\ )
\end{aligned}
$$

$$
\begin{aligned}
[{}^{a}\!/\!_{b}](\ (\boldsymbol{\nu} a)\,\overline{b}.a.\mathbf{0}\ |\ b.\mathbf{0}\ ) \quad &=_\alpha \quad (\ (\boldsymbol{\nu} a)\,\overline{a}.a.\mathbf{0}\ |\ a.\mathbf{0}\ ) \\
&=_\alpha \quad (\ (\boldsymbol{\nu} c)\,\overline{a}.c.\mathbf{0}\ |\ a.\mathbf{0}\ )
\end{aligned}
$$

LTS — Concurrent Expressions

$\ldots$

PAR$_1$: $$\frac{P_1 \xrightarrow{\mu} P_1'}{P_1|P_2 \xrightarrow{\mu} P_1'|P_2}$$

PAR$_2$: $$\frac{P_2 \xrightarrow{\mu} P_2'}{P_1|P_2 \xrightarrow{\mu} P_1|P_2'}$$

REACT: $$\frac{P \xrightarrow{\lambda} P' \qquad Q \xrightarrow{\overline{\lambda}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$$

RES: $$\frac{P \xrightarrow{\mu} P'}{(\boldsymbol{\nu} a)\, P \xrightarrow{\mu} (\boldsymbol{\nu} a)\, P'}$$ IF $\mu \notin \{a, \overline{a}\}$

ALPHA: $$\frac{Q \xrightarrow{\mu} Q'}{P \xrightarrow{\mu} P'}$$ IF $P =_\alpha Q$ AND $P' =_\alpha Q'$

Buffers, revisited . . .

$$\begin{array}{lcl} \mathcal{N} & := & \{\ \mathsf{in}_i, \mathsf{out}_i, \mathsf{x}_i \mid i \in \{0,1\}\ \} \\ \vec{a} & := & \mathsf{in}_0, \mathsf{in}_1, \mathsf{out}_0, \mathsf{out}_1 \\ \mathsf{Bluff}^{(2)}(\,\vec{a}\,) & \stackrel{\text{def}}{=} & (\boldsymbol{\nu}\mathsf{x}_0, \mathsf{x}_1)\ \ (\quad \mathsf{Buff}^{(1)}\langle\, \mathsf{in}_0, \mathsf{in}_1, \mathsf{x}_0, \mathsf{x}_1\,\rangle \\ & & \qquad\qquad\quad \mid \quad \mathsf{Buff}^{(1)}\langle\, \mathsf{x}_0, \mathsf{x}_1, \mathsf{out}_0, \mathsf{out}_1\,\rangle\ ) \end{array}$$

- compare the behavior (= LTSs) of $\mathsf{Buff}^{(2)}$ and $\mathsf{Bluff}^{(2)}$
- regard both as black boxes with “buttons” $\vec{a}$ . . .