# Concurrency Semantics
# Week 3

Course Notes 2005
EPFL – I&C

Uwe Nestmann
Johannes Borgström

April 27, 2005

`$Id: notes-3.tex,v 1.8 2005/04/25 16:16:12 uwe Exp $`

# 1 Simulation & Bisimulation

**1.1 Definition (Labeled Transition System / LTS)**
Let $\mathcal{A}$ be an *action alphabet*.
An *LTS* over $\mathcal{A}$ is a pair $(\mathcal{Q}, \mathcal{T})$ with

- 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 $q \xrightarrow{\mu}_{\mathcal{T}} q'$.
If $q \xrightarrow{\mu_1}_{\mathcal{T}} q_1 \cdots \xrightarrow{\mu_n}_{\mathcal{T}} q_n$ we call $q_n$ a *derivative* of $q$.

Usually we omit the subscript $\mathcal{T}$ of arrows.

**1.2 Definition ((Strong) Simulation)**
Let $(\mathcal{Q}, \mathcal{T})$ be a LTS.

1. Let $S$ be a binary relation (on $\mathcal{Q}$).
   $S$ is a *(strong) simulation* on $(\mathcal{Q}, \mathcal{T})$ if,
   whenever $p \; S \; q$,
   if $p \xrightarrow{\mu} p'$ (for some $p' \in \mathcal{Q}$) then
   there is $q' \in \mathcal{Q}$ such that $q \xrightarrow{\mu} q'$ and $p' \; S \; q'$.
2. $q$ *(strongly) simulates* $p$, written $p \preceq q$,
   if there is a (strong) simulation $S$ (on $(\mathcal{Q}, \mathcal{T})$)
   such that $p \; S \; q$.

The relation $\preceq$ is sometimes called *similarity*.

In the following, if the underlying transition system is clear in the respective reasoning context, then we usually omit to specify "on $\mathcal{Q}$" (for binary relations) or "on $(\mathcal{Q}, \mathcal{T})$" (for simulations).

**1.3 Lemma** Let $(\mathcal{Q}, \mathcal{T})$ be a LTS.
If $S_1$ and $S_2$ are simulations, then

1. $S_1 \cup S_2$ is also a simulation.
2. $S_1 S_2$ is also a simulation.

Note that $S_1 \cap S_2$ is not necessarily a simulation.

**1.4 Proposition** Let $(\mathcal{Q}, \mathcal{T})$ be a LTS.

1. $\preceq \; = \bigcup \{ \; S \;\mid S \text{ is simulation on } (\mathcal{Q}, \mathcal{T}) \; \}$
2. $\preceq$ is the largest simulation on $(\mathcal{Q}, \mathcal{T})$.
3. $(\mathcal{Q}, \preceq)$ is a preorder.

**1.5 Definition (Mutual Simulation)**
Let $(\mathcal{Q}, \mathcal{T})$ be a LTS. Let $p, q \in \mathcal{Q}$.
$p$ and $q$ are *mutually similar*, written $p \gtrless q$,
if there is a pair $(S_1, S_2)$ of simulations $S_1$ and $S_2$
with $p\ S_1\ q\ S_2\ p$ (i.e., with $p\ S_1\ q$ and $q\ S_2\ p$).

**1.6 Proposition** $\gtrless$ is an equivalence relation.

**1.7 Definition ((Strong) Bisimulation)**
Let $(\mathcal{Q}, \mathcal{T})$ be a LTS.
A binary relation $B$ on $\mathcal{Q}$ is
*a (strong) bisimulation* on $(\mathcal{Q}, \mathcal{T})$
if both $B$ and $B^{-1}$ are (strong) simulations.
$p$ and $q$ are *(strongly) bisimilar*, written $p \sim q$,
if there is a (strong) bisimulation $B$ such that $p\ B\ q$.

**1.8 Proposition**

1. $\sim\ = \bigcup\{\ S \mid S$ is (strong) bisimulation on $(\mathcal{Q}, \mathcal{T})\ \}$
2. $\sim$ is the largest bisimulation on $(\mathcal{Q}, \mathcal{T})$.
3. $\sim$ is an equivalence relation.

# 2 Sequential Processes

**2.1 Notation** We use the following sets of entities with corresponding meta-variables:

| | | |
|---|---|---|
| $\mathcal{I}$ | *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* | $\lambda \ldots \in \mathcal{L} := \mathcal{N} \cup \overline{\mathcal{N}}$ |
| $\mathcal{A}$ | *actions* | $\mu, \beta \ldots \in \mathcal{L} \cup \{\tau\}$ |

Labels are often also called *visible/external* actions.
In contrast, $\tau$ is called *invisible/internal* action.
We use $\vec{a}$ to denote *finite sequences* $a_1 \ldots, a_n$ of *names*.
We will use *parameterized processes* $\mathsf{A}\langle\,\vec{a}\,\rangle$ with *name* parameters (neither *co*-names, nor labels, ...)

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

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

We use $P, P_i \ldots$ to denote *process expressions*,
while $M, M_i \ldots$ always denote *choices* or *summations*.

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

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

where $M$ is a summation, and $\vec{a}$ includes $\mathrm{fn}(M)$, which denotes the set of *(free) names* of $P$ (see Definition 2.3). Then, $\mathsf{A}\langle\,\vec{b}\,\rangle$ is supposed to mean the same as $\{\vec{b}/\vec{a}\}M$, which is defined as simultaneous substitution of all occurrences of $\vec{a}$ by $\vec{b}$ (see Definition 2.4).

Note that $\vec{a}$ does only include *names* ($\in \mathcal{N}$), not co-names, and neither $\tau$.

**2.3 Definition ((Free) Names)** $\mathrm{fn} : \mathcal{P}^{\mathrm{seq}} \to \mathcal{P}(\mathcal{N})$
The set $\mathrm{fn}(P)$ is defined inductively by:

$$\mathrm{fn}(\mu) \stackrel{\text{def}}{=} \begin{cases} \{b\} & \text{if } \mu = b \\ \{b\} & \text{if } \mu = \overline{b} \\ \emptyset & \text{if } \mu = \tau \end{cases}$$

$$\begin{array}{lcl} \mathrm{fn}(\mathbf{0}) & \stackrel{\text{def}}{=} & \emptyset \\ \mathrm{fn}(\mu.P) & \stackrel{\text{def}}{=} & \mathrm{fn}(\mu) \cup \mathrm{fn}(P) \\ \mathrm{fn}(M_1 + M_2) & \stackrel{\text{def}}{=} & \mathrm{fn}(M_1) \cup \mathrm{fn}(M_2) \end{array}$$

$$\mathrm{fn}(\mathsf{A}\langle\,\vec{a}\,\rangle) \stackrel{\text{def}}{=} \{\vec{a}\}$$

**2.4 Definition (Simultaneous Substitution)**
A substitution $\sigma$ is a total function $\sigma : \mathcal{N} \to \mathcal{N}$.
The set $\mathrm{sup}(\sigma) \stackrel{\text{def}}{=} \{\, n \in \mathcal{N} \mid \sigma(n) \neq n \,\}$
denotes the *support* of $\sigma$.

1. For $k \in \mathcal{N}$, we lift $\sigma$ to $\mathcal{N}^k \to \mathcal{N}^k$ to act on vectors of names by

$$\sigma(\,(n_1, \ldots, n_k)\,) \stackrel{\text{def}}{=} (\,(\sigma(n_1), \ldots, \sigma(n_k)\,)$$

2. We lift $\sigma$ to actions $\mathcal{A} \to \mathcal{A}$, as defined by:

$$\sigma(\mu) \stackrel{\text{def}}{=} \begin{cases} a' & \text{if } \mu = a \in \mathcal{N} \text{ and } \sigma(a) = a' \\ \overline{\sigma(a)} & \text{if } \mu = \overline{a} \\ \tau & \text{if } \mu = \tau \end{cases}$$

3. We lift $\sigma$ to processes $\mathcal{P}^{\mathrm{seq}} \to \mathcal{P}^{\mathrm{seq}}$, as inductively defined by:

$$\begin{array}{rcl} \sigma(\mathbf{0}) & \stackrel{\text{def}}{=} & \mathbf{0} \\ \sigma(\mu.P) & \stackrel{\text{def}}{=} & \sigma(\mu).\sigma(P) \\ \sigma(M_1 + M_2) & \stackrel{\text{def}}{=} & \sigma(M_1) + \sigma(M_2) \\ \sigma(\mathsf{A}\langle\,\vec{a}\,\rangle) & \stackrel{\text{def}}{=} & \mathsf{A}\langle\,\sigma(\vec{a})\,\rangle \end{array}$$

**2.5 Notation** Let $\sigma : \mathcal{N} \to \mathcal{N}$ be a substitution with finite support. Then, we often represent $\sigma$ more concretely as $\{^{\sigma(\vec{n})}/_{\vec{n}}\}$, where $\vec{n}$ denote an arbitrary vector enumerating the elements of $\sup(\sigma)$.

Dually, given any vectors $\vec{a}$ and $\vec{b}$ of equal length, then $\{^{\vec{a}}/_{\vec{b}}\}$ uniquely defines a substitution.

**2.6 Definition (Operational Semantics)**
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:

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

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

$$\text{DEF: } \frac{\{^{\vec{c}}/_{\vec{a}}\}M \xrightarrow{\mu} P'}{\mathsf{A}\langle\, \vec{c}\,\rangle \xrightarrow{\mu} P'} \quad \text{IF } \mathsf{A}(\,\vec{a}\,) \stackrel{\text{def}}{=} M$$

Note that "transition under prefix" is not allowed.

**2.7 Notation** 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.