Process Algebra (2IMF10)
Basic Communicating Processes

Bas Luttik
MF 6.072
s.p.luttik@tue.nl
http://www.win.tue.nl/~luttik

Lecture 10–11

Parallel composition

We extend BSP(A) with a binary operator \( \parallel \) for parallel composition:

The process \( p \parallel q \) executes \( p \) and \( q \) in parallel.

But what does this mean?
Recall that we are assigning a transition-system semantics to processes, which treats actions as (observationally) atomic.

Exercise
Consider the following process terms, and try to draw a transition system for each of them (on the basis of your intuition):
- \( a.b.1 \parallel c.1 \)
- \( (a.1 + b.1) \parallel c.1 \)

Operational rules for parallel composition

\[
\begin{align*}
x \xrightarrow{a} x' & \quad \frac{x \parallel y \xrightarrow{a} x' \parallel y}{x \parallel y \xrightarrow{a} x' \parallel y'}
\end{align*}
\]

Exercise
Compute the transition systems for
- \( a.b.1 \parallel c.1 \); and
- \( (a.1 + b.1) \parallel c.1 \).
Synchronisation and communication

The main idea (so far) is that, from an observer’s point of view, the executions of two processes running in parallel are interleaved.

We also want to model interaction between parallel components!

This will be achieved by declaring that components may synchronise on certain actions, using a communication function:

\[ A \rightarrow A \]

that satisfies the following two conditions:

1. \( \gamma(a, b) = \gamma(b, a) \) for all \( a, b \in A \) (commutativity);
2. \( \gamma(\gamma(a, b), c) = \gamma(a, \gamma(b, c)) \) for all \( a, b \in A \) (associativity).

Example

We could, e.g., assume that there are actions \( c?k \), \( c!k \), \( c\oplus k \) \( (k \in \{0, 1, 2, \ldots, n\}) \), and that \( \gamma(c?k, c!k) = \gamma(c!k, c?k) = c?k \), while \( \gamma \) is undefined on all other pairs of actions (i.e., \( \gamma(c?1, c!2) \) is undefined).

Operational rules for interaction

\[ x \xrightarrow{a} x', y \xrightarrow{b} y' \quad \gamma(a, b) = c \quad x \parallel y \xrightarrow{c} x' \parallel y' \]

Exercise

Assume that \( \gamma(a, b) = \gamma(b, a) = c \) and \( \gamma \) is undefined otherwise, and compute the transition system for

\[ \rightarrow a.1 \parallel (b.1 + d.0) \]

Enforcing interaction

To enforce interaction between parallel components in a parallel composition, we use the encapsulation operator \( \partial_H \).

\[ H \subseteq A \] is a set of actions that are not allowed to occur (typically, it contains send- and receive actions).

\[ x \downarrow \quad x \xrightarrow{a} x' \quad a \notin H \quad x \parallel y \xrightarrow{c} x' \parallel y' \]

Interaction: an illustration

Consider processes

\[ A = runA.give.1 \quad \text{and} \quad B = take.runB.1 \]

Let \( \gamma(give, take) = \gamma(take, give) = pass \) (and undefined for all other combinations of actions), and let \( H = \{ give, take \} \).

Compute the transition systems for \( A \parallel B \) and \( \partial_H(A \parallel B) \).
To summarize: the operational semantics of $\text{BCP}(A, \gamma)$ consists of the operational rules for $\text{BSP}(A)$ extended with rules for $\parallel$ and $\partial_H$:

\[
\begin{align*}
    x \xrightarrow{a} x' & \quad \text{and} \quad y \xrightarrow{a} y' \\
    x \parallel y \xrightarrow{a} x' \parallel y & \quad \text{and} \quad x \parallel y \xrightarrow{a} x \parallel y'
\end{align*}
\]

To facilitate a finite ground-complete axiomatisation the following auxiliary operators have been introduced:

- **left-merge:** $p \parallel q$ executes $p$ and $q$ in parallel, but the first execution step must come from $p$.
- **communication-merge:** $p \mid q$ executes $p$ and $q$ in parallel, but the first execution step must be a synchronisation step from $p$ and $q$.

Then $x \parallel y$ can be eliminated with the axiom

\[
x \parallel y = x \parallel y + y \parallel x + x \mid y \quad M,
\]

so it remains to provide axioms allowing the elimination of $\parallel$ and $\mid$.
BCP: encapsulation (axioms)

\[
\begin{align*}
\partial_H(1) &= 1 & \text{D1} \\
\partial_H(0) &= 0 & \text{D2} \\
\partial_H(a.x) &= 0 & \text{D3} \quad \text{if } a \in H \\
\partial_H(a.x) &= a.H(x) & \text{D4} \quad \text{if } a \notin H \\
\partial_H(x + y) &= \partial_H(x) + \partial_H(y) & \text{D5}
\end{align*}
\]

BCP: standard concurrency

The following axioms are often added for convenience:

\[
\begin{align*}
x \parallel 1 &= x & \text{SC2} \\
1 \parallel x + 1 &= 1 & \text{SC3} \\
(x \parallel y) \parallel z &= x \parallel (y \parallel z) & \text{SC4} \\
(x \parallel y) \mid z &= x \mid (y \parallel z) & \text{SC5} \\
(x \parallel y) \mid z &= x \mid (y \parallel z) & \text{SC6} \\
(x \parallel y) \mid z &= x \mid (y \parallel z) & \text{SC7}
\end{align*}
\]

BCP: results

Let \( P(BCP(A, \gamma)) = (C(BCP(A, \gamma))) \cup \parallel, \mid, \bowtie, \land, \land, (a.I)_{a \in A}, ( \partial_H )_{H \subseteq A}, 0, 1 \).

Theorems

1. Bisimilarity is a congruence on \( P(BCP(A, \gamma)) \).
2. BCP\((A, \gamma)\) is sound for \( P(BCP(A, \gamma))/\bowtie \).
3. Elimination: for every closed BCP\((A, \gamma)\)-term \( p \) there exists a closed BSP\((A)\)-term \( q \) such that BCP\((A, \gamma) \vdash p \equiv q \).
4. BCP\((A, \gamma)\) is ground-complete for \( P(BCP(A, \gamma))/\bowtie \).

Theorem (Expansion)

\[
BCP(A, \gamma) \vdash \parallel \land x_i = \sum_{i \neq j \subseteq I} (\parallel \land x_j) \land (\parallel \land x_i) + \mid x_i
\]

No communication

If \( \gamma = \emptyset \) (i.e., \( \gamma(a, b) \) is undefined for all \( a, b \in A \)), then we may add the Free Merge Axiom:

\[
x \parallel y + 1 = 1 \quad \text{FMA}.
\]

The Expansion Theorem then reduces to:

\[
BCP(A, \gamma) + \text{FMA} \vdash \parallel \land x_i = \sum_{i \in I} x_i \land (\parallel \land x_j) + \mid x_i
\]
Handshaking communication

If $\gamma(\gamma(a, b), c)$ is undefined for all $a, b, c \in A$, then we may add the Handshaking Axiom:

$$x \mid y \mid z + 1 = 1 \quad \text{HA}.$$ 

The Expansion Theorem then reduces to:

$$\text{BCP}(A, \gamma) \vdash \sum_{i \in I} x_i = \sum_{i \in I} x_i \sum_{j \in \{i\}} + \sum_{i \neq j} \left( \sum_{k \in \{i,j\}} x_k \right) + \sum_{i \in I} x_i.$$ 

Buffers

One-place buffer

$$Buf_1 = 1 + \sum_{d \in D} i? d. old. Buf_1$$

Two-place buffer

$$Buf_2 = 1 + \sum_{d \in D} i? d. Buf \_d$$

$$B_d = old. Buf_2 + \sum_{e \in D} i? e. old. B_e$$

Towards building a two-place buffer with two one-place buffers:

$$Buf_{1_{il}} = 1 + \sum_{d \in D} i? d. l! d. Buf_{1_{il}}$$

$$Buf_{1_{lo}} = 1 + \sum_{d \in D} i? d. o! d. Buf_{1_{lo}}$$

Let $\gamma(l! d, l? d) = \gamma(l? d, l! d) = l! d$ and undefined otherwise, and let $H = \{l? d, l! d \mid d \in D\}$.

Then $\partial H(Buf_{1_{il}} \parallel Buf_{1_{lo}})$ is a solution of the following finite guarded recursive specification over $\text{BCP}(A, \gamma)$ (see book for a proof):

$$X = 1 + \sum_{d \in D} i? d. l? d. X_d$$

$$X_d = \sum_{e \in D} i? e. old. l! e. X_e + old. X.$$ 

Skip

The behaviour of $\varepsilon_l(x)$ ($l \subseteq A$) is characterised by the following rules:

$$\begin{align*}
\varepsilon_l(x) & \downarrow x \\
\varepsilon_l(x') & \downarrow x \\
\varepsilon_l(x) & \downarrow x \\
\varepsilon_l(x) & \downarrow x
\end{align*}$$
Recall that $\partial_H(\text{Buf}^1_{IL} \parallel \text{Buf}^1_{IO})$ is a solution of the following finite guarded recursive specification over $\text{BCP}(A, \gamma)$:

$$X = 1 + \sum_{d \in D} l?d.\text{ld}.X_d \quad \text{and}$$

$$X_d = \sum_{e \in D} l?e.\text{le}.X_e + \text{old}.X_d.$$ 

**Exercise**

Let $\gamma(l?d, l?d) = \gamma(l?d, l?d) = \text{ld}$ and undefined otherwise.

Let $H = \{l?d, l?d \mid d \in D\}$, and let $I = \{\text{ld} \mid d \in D\}$.

Prove that $\varepsilon_I(X), \varepsilon_I(X_d)$ is a solution for $\text{Buf}^2, B_d$ in the recursive specification of $\text{Buf}^2$, and conclude that $\varepsilon_I(\partial_H(\text{Buf}^1_{IL} \parallel \text{Buf}^1_{IO})) = \text{Buf}^2$. 