Process Algebra with Local Communication

Muck van Weerdenburg

FACS'07 September 21, 2007

Outline

- Introduction
  - Process Algebra
  - Parallelism
  - Global Communication
- Local communication
  - Extended Merge?
  - Communication Operator
  - Multiactions
  - Advantages
  - Example
- What's Next?

Introduction - Process Algebra

We use process algebra to model processes such that we can, for example, verify properties.

The focus is usually on interaction.

Introduction - Process Algebra

Processes $p, q, \ldots$ consist of:

- *actions* $a, b, \ldots$ and
- *inaction* (or *deadlock*) δ, combined with
- operators, such as
  - the *sequential composition* $\cdot$, and
  - the *alternative composition* $+$.

For example, $a \cdot (b + c)$ is a process that first does an a followed by either a b or a c.

Introduction - Parallelism

To put processes in parallel we have the *merge* $\parallel$.

The merge interleaves the actions of both parameters.

$$
\begin{aligned}
a \parallel b &= (a \cdot b) + (b \cdot a) \\
a \parallel (b \cdot c) &= (a \cdot b \cdot c) + (b \cdot ((a \cdot c) + (c \cdot a))) \\
a \parallel (b + c) &= (a \cdot (b + c)) + (b \cdot a) + (c \cdot a)
\end{aligned}
$$

Introduction - Parallelism

The merge can be axiomatised with the *left merge* $\mathbin{|\!\!\lfloor}$.

The left merge is similar to the merge, but ensures that the left argument performs the first action.

We have that: $p \parallel q \;=\; (p \mathbin{|\!\!\lfloor} q) + (q \mathbin{|\!\!\lfloor} p)$

Introduction - Global Communication

For communication we typically add the communication merge $|$.

$$p \parallel q \; = \; (p \mathbin{⌊\!\!⌊} q) + (q \mathbin{⌊\!\!⌊} p) + (p \mid q)$$

CCS-style: $(a \cdot p) \mid (\overline{a} \cdot q) \; = \; \tau \cdot (p \parallel q)$

ACP-style: $(a \cdot p) \mid (b \cdot q) \; = \; \gamma(a, b) \cdot (p \parallel q)$

γ is the communication function

Introduction - Global Communication (ACP)

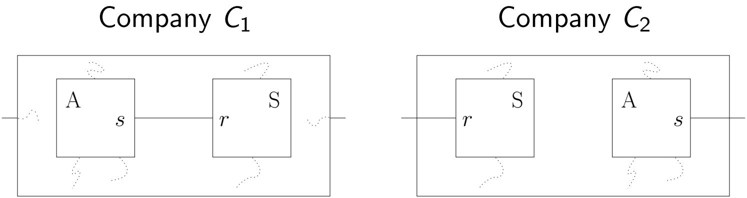
In ACP, a global communication function γ is defined.

Either a and b communicate (to an action c): $\gamma(a, b) = c$

Or they do *not* communicate: $\gamma(a, b) = \delta$

Introduction - Global Communication (ACP)

Assume two different companies C_1 and C_2 that develop components.

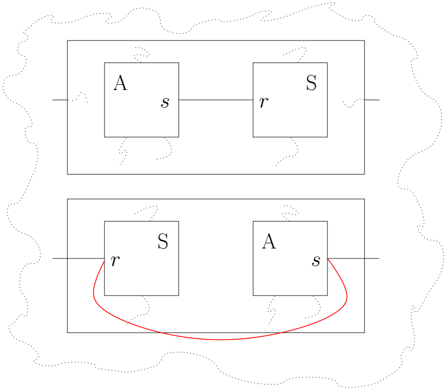


The component of C_1 requires r and s to communicate.

Introduction - Global Communication (ACP)

Simply putting
the components of
C_1 and C_2 together in a
systems possibly breaks
their functionality.

This can only
be solved by renaming
the internal actions
of the components!



Introduction - Global Communication (ACP)

Global communication breaks compositionality.

Conceptual oddity: actions can happen simultaniously, but must communicate to do so.

(As it is typically used, multi-way communication is elaborate.)

Outline

Local Communication

For a compositional language we need local communication.

Local communication only defines communication where it is used.

How to define local communication?

Local Communication - Extended Merge?

As parameter to the *merge* : $p \parallel_{\{a|b \to c\}} q$?

Very similar to ACP, but *every* parallel operator must contain communication.

Nesting is tricky:

$$(p \parallel_{\{a|b \to c\}} q) \parallel_{\{d|e \to f\}} r \text{ vs. } p \parallel_{\{a|b \to c\}} (q \parallel_{\{d|e \to f\}} r).$$

This is more a theoretical solution.

Local Communication - Communication Operator

We separate the concepts of parallelism and communication!

The merge only takes care of "interleaving".

A new *communication operator* Γ_C takes care of communication.

Local Communication - Communication Operator

We want a and b to communicate.

$$\Gamma_{\{a|b \to c\}}(a \parallel b) \;=\; \Gamma_{\{a|b \to c\}}((a \cdot b) + (b \cdot a)) \;=\, ??\; c\; ??$$

The merge no longer takes care of communication, but now it has to facilitate communication.

Local Communication - Multiactions

We need true concurrency; the merge should not just interleave processes.

Actions must be able to occur simultaniously: multiactions.

A multiaction is a bag/multiset of actions. E.g. $\langle a, b, b \rangle$.

(Instead of action a we now write the singleton multiaction $\langle a \rangle$.)

Local Communication - Multiactions

Instead of adding a communication merge we add a synchronisation operator $|$.

$$p \parallel q \;=\; (p \mathbin{⫝̸} q) + (q \mathbin{⫝̸} p) + (p \mid q)$$

With $(\langle a, b\rangle \cdot p) \mid (\langle b, c\rangle \cdot q) \;=\; \langle a, b, b, c\rangle \cdot (p \parallel q)$.

$$\begin{array}{rcl}\Gamma_{\{a|b\to c\}}(\langle a\rangle \parallel \langle b\rangle) & = & \Gamma_{\{a|b\to c\}}((\langle a\rangle \cdot \langle b\rangle) + (\langle b\rangle \cdot \langle a\rangle) + \langle a, b\rangle) \\ & = & (\langle a\rangle \cdot \langle b\rangle) + (\langle b\rangle \cdot \langle a\rangle) + \langle c\rangle\end{array}$$

Local Communication - Advantages

Our process algebra is compositional and has true concurrency.

Multi-way communication is much easier than before:

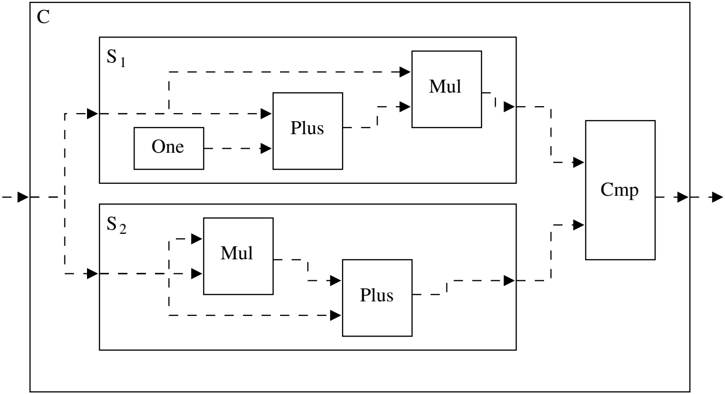
$$\nabla_{\{a|b|c|d\}}(\Gamma_{\{a|b|c|d \to e\}}(a \parallel b \parallel c \parallel d)) = e$$

The empty multiaction $\langle\rangle$ is the silent step τ!

$$\langle a, b, b\rangle \mid \langle\rangle = \langle a, b, b\rangle$$

$$\tau_{\{a\}}(\tau_{\{b\}}(\langle a, b, b\rangle)) = \tau_{\{a\}}(\langle a\rangle) = \langle\rangle$$

Local Communication - Example



Outline

What's Next? mCRL2!

mCRL2 is LoCo with:

- slightly different syntax ($a|b|c$ vs. $\langle a, b, c \rangle$)
- higher-order data language (incl. predefined parts)
- time ($a^{@5}$)
- a cross-platform toolset

Info and downloads at http://www.mcrl2.org/.

Thank you for your attention!