A structural hybrid logic for CSP and other process algebras

Till Mossakowski, Lutz Schröder
DFKI Bremen and University of Bremen

IFIP WG 1.3, Winchester — 2011, Sep 4th

Motivation

- project SHIP: semantic integration of heterogeneous processes
- formal development of concurrent systems from requirement to design
- requirements: more abstract formalism than "CSP + refinement"

Example: mutual exclusion

System in CSP:

$$
\begin{aligned}
P_1 &= try_1 \to enter_1 \to exit_1 \to P_1 \\
P_2 &= try_2 \to enter_2 \to exit_2 \to P_2 \\
System &= P_1 \mid\mid_{\emptyset} P_2
\end{aligned}
$$

Example: mutual exclusion

System in CSP:

$$P_1 = try_1 \rightarrow enter_1 \rightarrow exit_1 \rightarrow P_1$$
$$P_2 = try_2 \rightarrow enter_2 \rightarrow exit_2 \rightarrow P_2$$
$$System = P_1 \, ||_{\emptyset} \, P_2$$

System requirement in CSP:

$$Req = (enter_1 \rightarrow exit_1 \rightarrow Req) \Box (enter_2 \rightarrow exit_2 \rightarrow Req)$$
$Req \sqsubseteq_T System \setminus \{try_1, try_2\}$ does not hold here...

Example: mutual exclusion

System in CSP:

$$P_1 = try_1 \to enter_1 \to exit_1 \to P_1$$
$$P_2 = try_2 \to enter_2 \to exit_2 \to P_2$$
$$System = P_1 \mid\mid_\emptyset P_2$$

System requirement in CSP:

$$Req = (enter_1 \to exit_1 \to Req) \Box (enter_2 \to exit_2 \to Req)$$
$$Req \sqsubseteq_T System \setminus \{try_1, try_2\}$$ does not hold here...

System requirement in μCSP:

$P_i \Rightarrow \neg c_i$

$\varphi_i = P_i \wedge (\neg c_i U (c_i U P_i))$

$(\varphi_1 \mid\mid_X \varphi_2) \wedge G \neg (c_1 \wedge c_2)$

System design in μCSP:

$c_i \Rightarrow [A \setminus exit_i] c_i$

$[exit_i] \neg c_i$ $\qquad$ $[enter_i] c_i$

$\neg c_i \Rightarrow [A \setminus enter_i] \neg c_i$

Example: mutual exclusion

System in CSP:

$$P_1 = try_1 \to enter_1 \to exit_1 \to P_1$$
$$P_2 = try_2 \to enter_2 \to exit_2 \to P_2$$
$$System = P_1 \mid\mid_\emptyset P_2$$

System requirement in CSP:

$$Req = (enter_1 \to exit_1 \to Req) \Box (enter_2 \to exit_2 \to Req)$$

$Req \sqsubseteq_T System \setminus \{try_1, try_2\}$ does not hold here...

System requirement in μCSP:

$P_i \Rightarrow \neg c_i$

$\varphi_i = P_i \wedge (\neg c_i U (c_i U P_i))$

$(\varphi_1 \mid\mid_X \varphi_2) \wedge G \neg (c_1 \wedge c_2)$

System design in μCSP:

$c_i \Rightarrow [A \setminus exit_i] c_i$

$[exit_i] \neg c_i$ $[enter_i] c_i$

$\neg c_i \Rightarrow [A \setminus enter_i] \neg c_i$

Example: mutual exclusion

System in CSP:

$$P_1 = try_1 \to enter_1 \to exit_1 \to P_1$$
$$P_2 = try_2 \to enter_2 \to exit_2 \to P_2$$
$$System = P_1 \;||_\emptyset\; P_2$$

System requirement in CSP:

$$Req = (enter_1 \to exit_1 \to Req) \Box (enter_2 \to exit_2 \to Req)$$
$$Req \sqsubseteq_T System \setminus \{try_1, try_2\}$$ does not hold here. . .

System requirement in μCSP:

$$P_i \Rightarrow \neg c_i$$
$$\varphi_i = P_i \wedge (\neg c_i U (c_i U P_i))$$
$$(\varphi_1 \;||_X\; \varphi_2) \wedge G \neg (c_1 \wedge c_2)$$

System design in μCSP:

$$c_i \Rightarrow [A \setminus exit_i] c_i$$
$$[exit_i] \neg c_i \qquad [enter_i] c_i$$
$$\neg c_i \Rightarrow [A \setminus enter_i] \neg c_i$$

Related work

- A. Roscoe: Theory and Practice of Concurrency, Prentice Hall 1997
- Lus Caires, Luca Cardelli: A spatial logic for concurrency. Part I: Inf. Comput. 186(2): 194-235 (2003), Part II: Theor. Comput. Sci. 322(3): 517-565 (2004)
- Martin Berger, Kohei Honda and Nobuko Yoshida: Completeness and Logical Full Abstraction in Modal Logics for Typed Mobile Processes, ICALP (2), 2008, p. 99-111

contextual equivalence	$\subseteq$	bisimilarity	$\subseteq$	trace equivalence
	spatial logics		CSP	

LTS semantics of CSP

$$\frac{}{Skip \xrightarrow{\checkmark} \Omega}$$

$$\frac{}{(a \to P) \xrightarrow{a} P}$$

$$\frac{P \xrightarrow{a} P'}{P \parallel_X Q \xrightarrow{a} P' \parallel_X Q'} (a \in (A \setminus X) \cup \{\tau\})$$

$$\frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{a} Q'}{P \parallel_X Q \xrightarrow{a} P' \parallel_X Q'} (a \in X)$$

...

Drawback: one big syntactic LTS, not compositional

Compositional LTS semantics of parallel composition

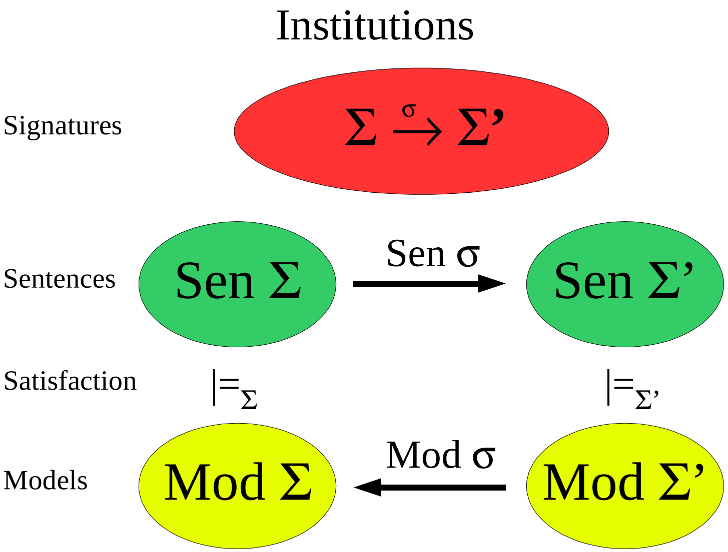
$$(\|\textsc{Sync}) \text{ is read as } \qquad \frac{x \xrightarrow{a} x' \ \boxed{\text{in } \mathcal{S}} \qquad y \xrightarrow{\overline{a}} y' \ \boxed{\text{in } \mathcal{T}}}{x \parallel y \xrightarrow{\tau} x' \parallel y' \quad \boxed{\text{in } \mathcal{S} \parallel \mathcal{T}}} \ (\|\textsc{Sync})$$

$\mathcal{S} \parallel \mathcal{T}$ is defined over the product of $\mathcal{S}$-states and $\mathcal{T}$-states.

Works also for other process algebra operators.

Ichiro Hasuo: The Microcosm Principle and Compositionality of GSOS-Based Component Calculi, CALCO 2011, p. 222-236

Institutions



The CSP institution – signatures

A *signature* a pair (A, N) where

- A is an alphabet of communications and
- N is a set of process names;

A *signature morphism* $\sigma = (\alpha, \nu) : (A, N) \to (A', N')$ consists of two maps

- $\alpha : A \to A'$, an injective translation of communications, and
- $\nu : N \to N'$, a translation of process names.

The CSP institution – sentences

Process over (A, N):

$$
\begin{array}{rlll}
P, Q ::= & n & & \%\%\ \text{process name } n \in N \\
| & \mathit{Skip} & & \%\%\ \text{successfully terminating process} \\
| & \mathit{Stop} & & \%\%\ \text{deadlock process} \\
| & a \to P & & \%\%\ \text{action prefix with a communication } a \in A \\
| & P \,\Box\, Q & & \%\%\ \text{external choice} \\
| & P \sqcap Q & & \%\%\ \text{internal choice} \\
| & \mathit{if}\ \varphi\ \mathit{then}\ P\ \mathit{else}\ Q & & \%\%\ \text{conditional} \\
| & P \mid\mid_X Q & & \%\%\ \text{generalized parallel} \\
| & P \setminus X & & \%\%\ \text{hiding} \\
| & P[[r]] & & \%\%\ \text{relational renaming} \\
| & P \fatsemi Q & & \%\%\ \text{sequential composition}
\end{array}
$$

Sentences over (A, N) are process definitions:

$$n = P$$

Sentence translation along $\sigma = (\alpha, \nu)$ is substitution.

The CSP institution – LTS models

An (A, N)-model $(L, init)$ consists of

- a labeled transition system $L = (S, \rightarrow \subseteq S \times A \times S)$ with labels in A
- an assignment $init : N \rightarrow S$ of states to the names in N

The model reduct of $(L_2, init_2)$ along $\sigma = (\alpha, \nu) : (A_1, N_1) \rightarrow (A_2, N_2)$ is $(L_1, init_1)$ with

- $s_1 \stackrel{a}{\rightarrow} s_2$ in L_1 if $s_1 \stackrel{\alpha(a)}{\rightarrow} s_2$ in L_2
- $init_1 = init_2 \circ \nu$

The Csp institution – satisfaction

$$(L, init) \models n = P \text{ iff } (L, init(n)) \sim [\![P]\!]_{L,init}$$

$$[\![Skip]\!]_{L,init} = O \xrightarrow{\checkmark} \Omega$$

$$[\![Stop]\!]_{L,init} = O$$

$$[\![a \rightarrow P]\!]_{L,init} = a \rightarrow [\![P]\!]_{L,init}$$

$$[\![P \,\Box\, Q]\!]_{L,init} = [\![P]\!]_{L,init} \,\Box\, [\![Q]\!]_{L,init}$$

$$[\![P \sqcap Q]\!]_{L,init} = [\![P]\!]_{L,init} \sqcap [\![Q]\!]_{L,init}$$

$$[\![\mathit{if}\ \varphi\ \mathit{then}\ P\ \mathit{else}\ Q]\!]_{L,init} = \mathit{if}\ [\![\varphi]\!]_{L,init}\ \mathit{then}\ [\![P]\!]_{L,init}\ \mathit{else}\ [\![Q]\!]_{L,init}$$

$$[\![P \,||_X\, Q]\!]_{L,init} = [\![P]\!]_{L,init} \,||_X\, [\![Q]\!]_{L,init}$$

$$[\![P \setminus X]\!]_{L,init} = [\![P]\!]_{L,init} \setminus X$$

$$[\![P[[r]]]\!]_{L,init} = [\![P]\!]_{L,init}[[r]]$$

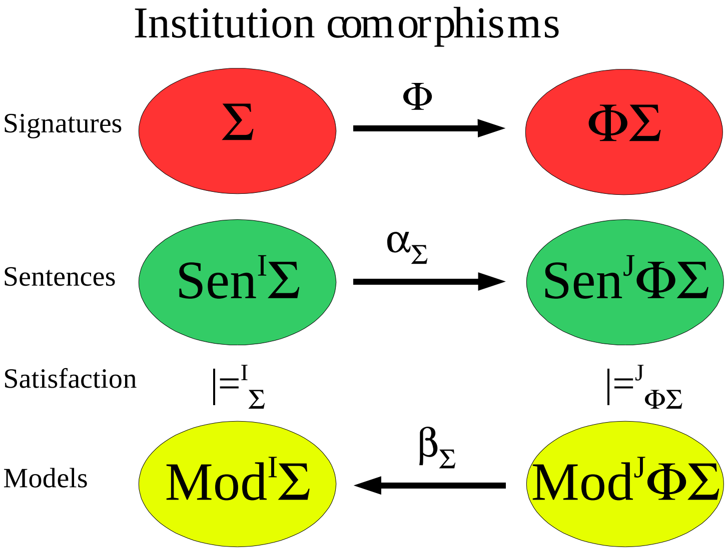
$$[\![P \,⨾\, Q]\!]_{L,init} = [\![P]\!]_{L,init} \,⨾\, [\![Q]\!]_{L,init}$$

Modal logic μ CSP

- Signatures (A, P, O) where
  - A is an alphabet of labels,
  - P is a set of propositions, and
  - O is a set of nominals
- models are Kripke frames (=LTS) (L, Val^P, Val^O), where $L = (S, \rightarrow \subseteq S \times A \times S)$, $Val^P : P \rightarrow \mathcal{P}(S)$, $Val^O : O \rightarrow S$
- Sentences:

$$\begin{array}{rl} \varphi ::= & \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \\ & < a > \varphi \mid i \mid @_i\varphi \mid \mu X.\varphi \mid \\ & Skip \mid Stop \mid \varphi \,\square\, \varphi \mid \varphi \,||_X\, \varphi \mid \ldots \end{array}$$

- Satisfaction: as in hybrid μ-calculus, and: $(L, Val^P, Val^O) \models_s \varphi_1 \,||_X\, \varphi_2$ if $(L, Val^P, Val^O, s) \sim (L_1, Val_1^P, Val_1^O, s_1) \,||_X\, (L_2, Val_2^P, Val_2^O, s_2)$ and $(L_i, Val_i^P, Val_i^O) \models_{s_i} \varphi_i$ for $i = 1, 2$, etc.


Institution comorphisms
Signatures
Σ
Φ
ΦΣ
Sentences
Sen^IΣ
α_Σ
Sen^JΦΣ
Satisfaction
|=^I_Σ
|=^J_ΦΣ
Models
Mod^IΣ
β_Σ
Mod^JΦΣ

Institution comorphism $\mathrm{CSP} \to \mu\,\mathrm{CSP}$

- signatures:
  communications $\to$ labels, process names $\to$ nominals
  i.e. $(A, N) \mapsto (A, \emptyset, N)$
- sentences: $n = P \mapsto @_n P$
- models: $(L, Val^P, Val^O) \mapsto (L, Val^O)$

Development methodology

- Start with a μ Csp specification
- Refine it, also using the structural operators...
- until a process decomposition has been reached
  - note that propositions have to be hidden
- This then "is" a Csp process

Future work

- proof system, model checking
- Can we express everthing that can be expressed as "CSP process plus (trace, stable failures, ...) refinement"?
- Generalisation to CSP-CASL

Generalisation to CSP-CASL

- alphabet = disjoint union of CASL carrier sets
- process names as well as nominals have parameter sorts
- alphabet letters are replaced with terms
  - in particular: term modalities