A Theoretical Basis of Communication-Centred Programming for *Web Service*

Nobuko Yoshida (Imperial)
Kohei Honda (Queen Mary)

TiC 2006, July 2006

In Collaboration with:

Marco Carbone (Queen Mary)

Gary Brown (pi4 technologies)

Steve Ross-Talbot (pi4 technologies)

Structure of Lectures

- **Part 1** Basic Theory (Processes and Types)
  - 1 Introduction to the π-Calculus
  - 2 Idioms for Interactions
  - 3 Session Types
- **Part 2** Web Services and the π-Calculus
  - 1 Web Services Choreography Description Language
  - 2 Global Language and the End-Point Calculus
  - 3 End-Point Projection and Correctness

Web Service (1)

Concurrency in theory and practice:

1. Understanding (CCS, CSP, ACP, Pi-Calculus...)
   - Message/event based: no shared memory.
   - Rich theories of communication behaviour.
   - Connection to other fields (games, Linear Logic, ...).
2. Building (Java, C/pthreads, OpenMP, ...)
   - Threads and shared variables.
   - Fits SMP (e.g. OpenMP).
   - Recent development (e.g. lock-free algorithms, software transaction, ...)

Web Service (2)

Key Characteristics of WS:

- Infrastructural basis: URI (naming), XML (message format), HTTP/TCP (message delivery).
- Applications purely based on communication (“business protocols”).
- Inter-organisational nature demanding standardisation and clear, rigorous understanding.

WS-CDL: Summary

- XML-based description language for business protocols.
- Developed by W3C's CDL WG (2003∼, chaired by Steve Ross-Talbot and Martin Chapman).
- Central idea: **choreography** (cf. orchestration).

  Dancers dance following a global scenario without a single point of control.
- Pi-calculus experts invited in 2004. Now CR, reaching a W3C standard soon.
- Offers general global programming unlike sequence diagrams etc.

Global Description (1)

Notations for cryptographic protocols.

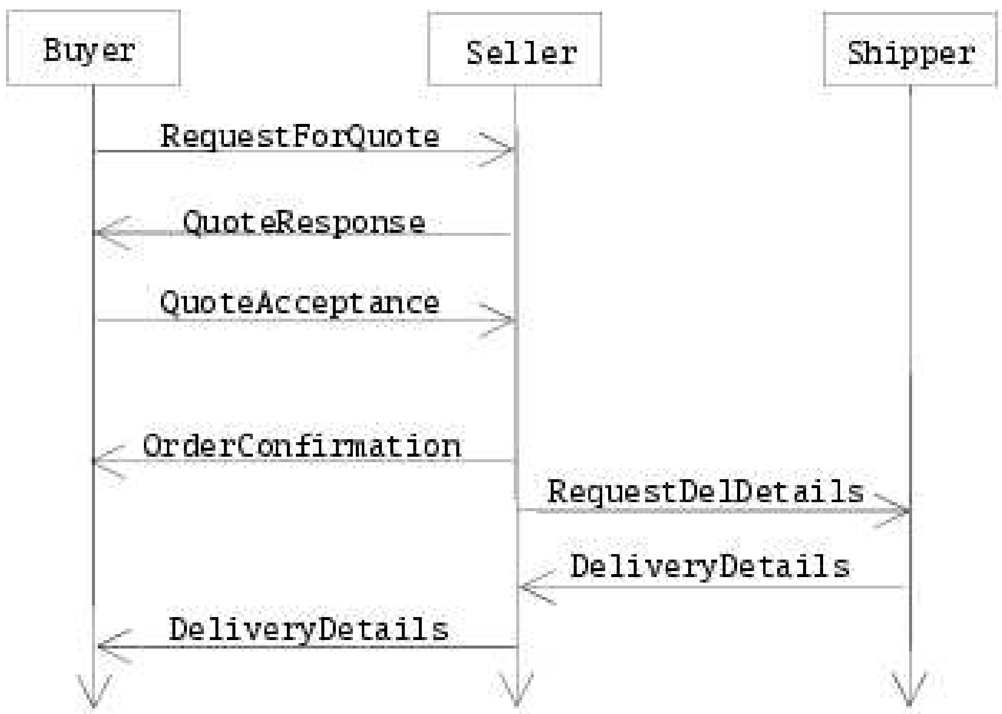
1. $\text{Alice} \longrightarrow \text{Trent} : \text{Alice},\ \text{Bob},\ N_A\ .$

2. $\text{Trent} \longrightarrow \text{Alice} : \{N_A,\ K_{AB}\}_{K_A},\ \{K_{AB}\}_{K_B}\ .$

3. $\text{Alice} \longrightarrow \text{Bob} : \{K_{AB}\}_{K_B}\ .$

Global Description (2)

UML sequence diagrams.



How To Use Types

Question:

We have two programs/specifications, P_1 and P_2.
Does $P_1|P_2$ behave all right?

Answer without Types.

...well let's run $P_1|P_2$ and see what happens.

Answer with Types. We put a small tag t_1 to P_1, t_2 to P_2.

Let's check $t_1|t_2$ makes sense. If it does, $P_1|P_2$ behaves well, and has a new tag $t_1|t_2$.

NB. *Checking consistency of tags is very efficient.*

Session Types

- A simple type abstraction of a structured collection of interactions, or a session.
- Studied for a variety of calculi and languages.
- Started from decomposition of communication idioms in the π-calculus (1994∼).
- Many real-world application-level communications are based on sessions (TCP sessions, cookies, ...).
- Gives a simple and robust basis for further analysis and verification.

End-Point Projection (EPP)

- A notion informally (introduced and) discussed in WS-CDL WG.

  How can we project a global description to endpoints so that their interactions precisely realise the original global description?

- Basis for execution, monitoring, validation, reuse, conformance, interoperability,...

- Demands formalisation of global and end-point descriptions.

Global Calculus: Syntax (1)

I	$::=$	$A \rightarrow B : \mathtt{ch}(\tilde{s}).I$	session initiation
		$A \rightarrow B : s\langle \mathsf{op}, e, y\rangle.I$	communication
		$x@A := e.I$	assignment
		$\mathsf{if}\ e@A\ \mathsf{then}\ I_1\ \mathsf{else}\ I_2$	conditional
		$I_1 + I_2$	sum
		$I_1 \mid I_2$	parallel
		$(\nu\, s)I$	hiding
		$\mathbf{rec}\ X^A.I$	recursion
		X^A	term variable
		$\mathbf{0}$	inaction

Global Calculus: Syntax (2)

We have *not* included, but can treat:

- New variable declaration.
- Channel/session passing, general sequencing (fork/join, parbegin/parend, async/finish).
- Mutex and/or ”atomic” constructs.

Global Calculus: Reduction

Reduction is between configurations:

$$(I,\, \sigma) \quad \to \quad (I',\, \sigma')$$

where σ maps, for each participant A, variables *located* at A to their stored values. Samples of reduction rules:

$$\begin{array}{rcl}
(A \longrightarrow B : \mathtt{ch}(\tilde{s}).I,\, \sigma) & \to & ((\nu\, \tilde{s})I,\, \sigma) \\
(A \longrightarrow B : s\langle \mathsf{op}, V, x\rangle.I,\, \sigma) & \to & (I,\, \sigma[x@B \mapsto V]) \\
(x@A := V.I,\, \sigma) & \to & (I,\, \sigma[x@A \mapsto V]) \\
(I_1 + I_2,\, \sigma) & \to & (I_1,\, \sigma) \\
(I_1 | I_2,\, \sigma) & \to & (I_1' | I_2,\, \sigma') \quad \text{if } (I_1,\, \sigma) \longrightarrow (I_1',\, \sigma')
\end{array}$$

Global Calculus: Reduction

Reduction is between configurations:

$$(I,\ \sigma) \quad \rightarrow \quad (I',\ \sigma')$$

where σ maps, for each participant A, variables *located* at A to their stored values. Samples of reduction rules:

$$
\begin{array}{rcl}
(A \longrightarrow B : \mathtt{ch}(\tilde{s}).I,\ \sigma) & \rightarrow & ((\nu\, \tilde{s})I,\ \sigma) \\
(A \longrightarrow B : s\langle \mathsf{op}, V, x\rangle.I,\ \sigma) & \rightarrow & (I,\ \sigma[x@B \mapsto V]) \\
(x@A := V.I,\ \sigma) & \rightarrow & (I,\ \sigma[x@A \mapsto V]) \\
(I_1 + I_2,\ \sigma) & \rightarrow & (I_1,\ \sigma) \\
(I_1 | I_2,\ \sigma) & \rightarrow & (I_1' | I_2,\ \sigma') \quad \text{if } (I_1,\ \sigma) \longrightarrow (I_1',\ \sigma')
\end{array}
$$

Global Calculus: Reduction

Reduction is between configurations:

$$(I,\, \sigma) \quad \to \quad (I',\, \sigma')$$

where σ maps, for each participant A, variables *located* at A to their stored values. Samples of reduction rules:

$$
\begin{array}{rcl}
(A \longrightarrow B : \mathtt{ch}(\tilde{s}).I,\, \sigma) & \to & ((\nu\, \tilde{s})I,\, \sigma) \\
(A \longrightarrow B : s\langle \mathsf{op}, V, x\rangle.I,\, \sigma) & \to & (I,\, \sigma[x@B \mapsto V]) \\
(x@A := V.I,\, \sigma) & \to & (I,\, \sigma[x@A \mapsto V]) \\
(I_1 + I_2,\, \sigma) & \to & (I_1,\, \sigma) \\
(I_1 | I_2,\, \sigma) & \to & (I_1' | I_2,\, \sigma') \quad \text{if } (I_1,\, \sigma) \longrightarrow (I_1',\, \sigma')
\end{array}
$$

Global Calculus: Reduction

Reduction is between configurations:

$$(I,\ \sigma) \quad \rightarrow \quad (I',\ \sigma')$$

where σ maps, for each participant A, variables *located* at A to their stored values. Samples of reduction rules:

$$
\begin{array}{rcl}
(A \longrightarrow B : \mathtt{ch}(\tilde{s}).I,\ \sigma) & \rightarrow & ((\nu\, \tilde{s})I,\ \sigma) \\
(A \longrightarrow B : s\langle \mathsf{op}, V, x\rangle.I,\ \sigma) & \rightarrow & (I,\ \sigma[x@B \mapsto V]) \\
(x@A := V.I,\ \sigma) & \rightarrow & (I,\ \sigma[x@A \mapsto V]) \\
(I_1 + I_2,\ \sigma) & \rightarrow & (I_1,\ \sigma) \\
(I_1 | I_2,\ \sigma) & \rightarrow & (I_1' | I_2,\ \sigma') \quad \text{if } (I_1,\ \sigma) \longrightarrow (I_1',\ \sigma')
\end{array}
$$

Global Calculus: Reduction

Reduction is between configurations:

$$(I,\ \sigma) \quad \rightarrow \quad (I',\ \sigma')$$

where σ maps, for each participant A, variables *located* at A to their stored values. Samples of reduction rules:

$$
\begin{array}{rcl}
(A \longrightarrow B : \mathtt{ch}(\tilde{s}).I,\ \sigma) & \rightarrow & ((\nu\,\tilde{s})I,\ \sigma) \\
(A \longrightarrow B : s\langle \mathsf{op}, V, x\rangle.I,\ \sigma) & \rightarrow & (I,\ \sigma[x@B \mapsto V]) \\
(x@A := V.I,\ \sigma) & \rightarrow & (I,\ \sigma[x@A \mapsto V]) \\
(I_1 + I_2,\ \sigma) & \rightarrow & (I_1,\ \sigma) \\
(I_1 | I_2,\ \sigma) & \rightarrow & (I_1' | I_2,\ \sigma') \quad \text{if } (I_1,\ \sigma) \longrightarrow (I_1',\ \sigma')
\end{array}
$$

Global Calculus: Reduction

Reduction is between configurations:

$$(I,\, \sigma) \quad \rightarrow \quad (I',\, \sigma')$$

where σ maps, for each participant A, variables *located* at A to their stored values. Samples of reduction rules:

$$\begin{aligned}
(A \longrightarrow B : \mathtt{ch}(\tilde{s}).I,\, \sigma) &\rightarrow ((\nu\, \tilde{s})I,\, \sigma) \\
(A \longrightarrow B : s\langle \mathsf{op}, V, x\rangle.I,\, \sigma) &\rightarrow (I,\, \sigma[x@B \mapsto V]) \\
(x@A := V.I,\, \sigma) &\rightarrow (I,\, \sigma[x@A \mapsto V]) \\
(I_1 + I_2,\, \sigma) &\rightarrow (I_1,\, \sigma) \\
(I_1 | I_2,\, \sigma) &\rightarrow (I_1' | I_2,\, \sigma') \quad \text{if } (I_1,\, \sigma) \longrightarrow (I_1',\, \sigma')
\end{aligned}$$

Global Calculus: Reduction

Reduction is between configurations:

$$(I,\ \sigma) \quad \rightarrow \quad (I',\ \sigma')$$

where σ maps, for each participant A, variables *located* at A to their stored values. Samples of reduction rules:

$$
\begin{array}{rcl}
(A \rightarrow B : \mathtt{ch}(\tilde{s}).I,\ \sigma) & \rightarrow & ((\nu\, \tilde{s})I,\ \sigma) \\
(A \rightarrow B : s\langle \mathsf{op},\ V,\ x\rangle.I,\ \sigma) & \rightarrow & (I,\ \sigma[x@B \mapsto V]) \\
(x@A := V.I,\ \sigma) & \rightarrow & (I,\ \sigma[x@A \mapsto V]) \\
(I_1 + I_2,\ \sigma) & \rightarrow & (I_1,\ \sigma) \\
(I_1 | I_2,\ \sigma) & \rightarrow & (I_1' | I_2,\ \sigma') \quad \text{if } (I_1,\ \sigma) \rightarrow (I_1',\ \sigma')
\end{array}
$$

Examples of Reduction

$$\left(\begin{array}{l} \text{Buyer} \longrightarrow \text{Seller} : \text{QuoteCh}(\nu s)\,. \\ \quad \text{Seller} \longrightarrow \text{Buyer} : s\langle \text{Quote}, 300, x\rangle\,.\,I', \end{array} \qquad \sigma \right)$$

$$\downarrow$$

$$(\ (\nu s)\ \text{Seller} \longrightarrow \text{Buyer} : s\langle \text{Quote}, 300, x\rangle\,.\,I'), \qquad \sigma\ )$$

$$\downarrow$$

$$(\ (\nu s)\ I', \qquad \sigma[x@\text{Buyer} \mapsto 300]\ )$$

Global Calculus: Example (1)

$\mathbf{Buyer} \rightarrow \mathbf{Seller} : \mathbf{ch1}(\nu\, s, t).$

$\mathbf{Buyer} \rightarrow \mathbf{Seller} : s\langle \mathsf{QuoteReq}, prod@B, prod@S\rangle.$

$\mathbf{Seller} \rightarrow \mathbf{Buyer} : t\langle \mathsf{QuoteRes}, quote@S, quote@B\rangle\ .$

$\mathbf{Buyer} \rightarrow \mathbf{Seller} : s\langle \mathsf{QuoteAcc}, cred; adr@B, cred; adr@S\rangle.$

$\mathbf{Seller} \rightarrow \mathbf{Shipper} : \mathbf{ch2}(r).$

$\mathbf{Seller} \rightarrow \mathbf{Shipper} : r\langle \mathsf{ShipReq}, prod; adr@S, prod; adr@Sh\rangle.$

$\mathbf{Shipper} \rightarrow \mathbf{Seller} : r\langle \mathsf{ShipConf}\rangle.$

$\mathbf{Seller} \rightarrow \mathbf{Buyer} : t\langle \mathsf{OrderConf}\rangle.\ \mathbf{0}$

Global Calculus: Example (2)

$$\mathbf{Buyer} \to \mathbf{Seller} : \mathbf{ch1}(s,t).$$
$$\mathbf{Buyer} \to \mathbf{Seller} : s\langle \mathsf{QuoteReq}, prod@B, prod@S \rangle.$$
$$\mathbf{Seller} \to \mathbf{Buyer} : t\langle \mathsf{QuoteRes}, quote@S, quote@B \rangle\ .$$
choice
$$\mathbf{Buyer} \to \mathbf{Seller} : s\langle \mathsf{QuoteAcc}, cred;adr@B, cred;adr@S \rangle.$$
$$\mathbf{Seller} \to \mathbf{Shipper} : \mathbf{ch2}(r).$$
$$\cdots \text{(as before)} \cdots$$
$$+$$
$$\mathbf{Buyer} \to \mathbf{Seller} : s\langle \mathsf{QuoteNoGood} \rangle.\mathbf{0}$$
endchoice

Global Calculus: Example (3)

$\mathbf{Buyer} \rightarrow \mathbf{Seller} : \mathbf{ch1}(s,t).$

$\mathbf{Buyer} \rightarrow \mathbf{Seller} : s\langle \mathsf{QuoteReq}, prod@B, prod@S\rangle.$

$\mathbf{Seller} \rightarrow \mathbf{Buyer} : t\langle \mathsf{QuoteRes}, quote@S, quote@B\rangle \ .$

$\mathbf{if}\ \mathsf{reasonable}(quote)@Buyer\ \mathbf{then}$

$\quad \mathbf{Buyer} \rightarrow \mathbf{Seller} : s\langle \mathsf{QuoteAcc}, cred;adr@B, cred;adr@S\rangle.$

$\quad \mathbf{Seller} \rightarrow \mathbf{Shipper} : \mathbf{ch2}(r).$

$\quad \cdots$ (as before) $\cdots$

else

$\quad \mathbf{Buyer} \rightarrow \mathbf{Seller} : s\langle \mathsf{QuoteNoGood}\rangle.\mathbf{0}$

endif

Global Calculus: Example (4)

$$
\begin{array}{l}
\textbf{Buyer} \rightarrow \textbf{Seller} : \textbf{ch1}(s, t). \\
\textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathsf{QuoteReq}, \mathit{prod}, \mathit{prod} \rangle. \\
\textbf{rec X}. \\
\textbf{Seller} \rightarrow \textbf{Buyer} : t\langle \mathsf{QuoteRes}, \mathit{quote}, \mathit{quote} \rangle \ . \\
\textbf{if } \mathsf{reasonable}(\mathit{quote})@\mathit{Buyer} \textbf{ then} \\
\quad \textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathsf{QuoteAcc}, \mathit{cred}; \mathit{adr}@B, \mathit{cred}; \mathit{adr}@S \rangle. \\
\quad \textbf{Seller} \rightarrow \textbf{Shipper} : \textbf{ch2}(r). \\
\qquad \cdots (\text{as before}) \cdots \\
\textbf{else} \\
\qquad \textbf{Buyer} \rightarrow \textbf{Seller} : s\langle \mathsf{QuoteNoGood} \rangle.\textbf{X} \\
\textbf{endif}
\end{array}
$$

Global Calculus: Example (5)

$\mathbf{Buyer} \rightarrow \mathbf{Seller} : \mathbf{ch1}(s, t).$

$\mathbf{Buyer} \rightarrow \mathbf{Seller} : s\langle \mathsf{QuoteReq}, \mathit{prod}, \mathit{prod}\rangle.$

$\mathbf{Seller} \rightarrow \mathbf{Vendor} : \mathbf{ch2}(u).$

$\mathbf{rec\ X}.$

$\mathbf{Seller} \rightarrow \mathbf{Vendor} : u\langle \mathsf{QuoteReq}, \mathit{prod}, \mathit{prod}\rangle.$

$\mathbf{Vendor} \rightarrow \mathbf{Seller} : u\langle \mathsf{QuoteRes}, \mathit{orgq}, \mathit{orgq}\rangle.$

$\mathbf{Seller} \rightarrow \mathbf{Buyer} : t\langle \mathsf{QuoteRes}, \mathit{orgq} + 100, \mathit{quote}\rangle\ .$

$\mathbf{if}\ \mathsf{reasonable}(\mathit{quote})@\mathit{Buyer}\ \mathbf{then}$

$\quad \mathbf{Buyer} \rightarrow \mathbf{Seller} : s\langle \mathsf{QuoteAcc}, \mathit{cred}; \mathit{adr}@B, \mathit{cred}; \mathit{adr}@S\rangle.$

$\quad \mathbf{Seller} \rightarrow \mathbf{Vendor} : u\langle \mathsf{Done!}\rangle.$

$\qquad \cdots(\text{as before})\cdots$

$\mathbf{else}$

$\qquad \mathbf{Buyer} \rightarrow \mathbf{Seller} : s\langle \mathsf{QuoteNoGood}\rangle.\mathbf{X}$

$\mathbf{endif}$

Global Calculus: Example (6)

$\mathbf{Buyer} \to \mathbf{Seller} : \mathbf{ch1}(st);\ s\langle \mathsf{QuoteReq},\ prod,\ prod\rangle.$

$\mathbf{Seller} \to \mathbf{Buyer} : t\langle \mathsf{QuoteRes},\ quote,\ quote\rangle.$

$\mathbf{if}\ \mathsf{reasonable}(quote)@\mathbf{Buyer}\ \mathbf{then}$

$\quad \mathbf{Buyer} \to \mathbf{Seller} : s\langle \mathsf{QuoteAcc},\ cred;adr@B,\ cred;adr@S\rangle.$

$\quad \mathbf{Seller} \to \mathbf{CCA} : \mathbf{ch2}(u);\ u\langle \mathsf{plsCheck},\ cred,\ cred\rangle.$

$\quad \mathbf{if}\ \mathsf{cleared}(cred)@\mathbf{CCA}\ \mathbf{then}$

$\qquad \mathbf{CCA} \to \mathbf{Seller} : u\langle \mathsf{credOK}\rangle.$

$\qquad \mathbf{Seller} \to \mathbf{Shipper} : \mathbf{ch3}(r);\ r\langle \mathsf{ShipReq},\ prod;adr@S,\ prod;adr@Sh\rangle.$

$\quad \cdots(\text{as before})\cdots$

$\quad \mathbf{else}$

$\qquad \mathbf{CCA} \to \mathbf{Seller} : r\langle \mathsf{credNotOK}\rangle.$

$\qquad \mathbf{Seller} \to \mathbf{Buyer} : t\langle \mathsf{cannotProcess}\rangle.\mathbf{0}$

$\mathbf{else}$

$\quad \mathbf{Buyer} \to \mathbf{Seller} : s\langle \mathsf{QuoteNoGood}\rangle.\mathbf{0}$

Global Calculus: Typing (1)

- Session types ($\theta, \theta_i, \ldots$ are value types):

$$\begin{array}{lll} \alpha & ::= & s \blacktriangleright \Sigma_i \langle \mathsf{op_i}, \theta_i \rangle .\ \alpha_i \quad | \quad s \blacktriangleleft \Sigma_i \langle \mathsf{op_i}, \theta_i \rangle .\ \alpha_i \\ & | & \alpha_1 | \alpha_2 \quad | \quad \mathbf{rec}\ t.\alpha \quad | \quad t \quad | \quad \mathbf{end} \end{array}$$

- Typing: $\Gamma,\ \mathtt{ch}@A : (\tilde{s})\alpha \vdash I \triangleright \Delta,\ \tilde{s}[A,B] : \alpha.$
  - $\mathtt{ch}@A : (\tilde{s})\alpha$ says a service channel $\mathtt{ch}$ at A may be invoked with fresh $\tilde{s}$ followed by a session α.
  - $\tilde{s}[A,B] : \alpha$ says a session α from A to B occurs using $\tilde{s}$.
- Properties: *Subject Reduction*, *Minimal Typing*, ...

Global Calculus: Typing (2)

A term:

$$A \to B : \mathbf{b}(s_1 s_2).(\, A \to B : s_1\langle \mathsf{go}\rangle.B \to A : s_2\langle \mathsf{ok}\rangle.\mathbf{0} \;+$$
$$A \to B : s_1\langle \mathsf{stop}\rangle.B \to A : s_2\langle \mathsf{no}\rangle.\mathbf{0}\,)$$

can be given a typing (ε is the empty string):

$$\mathbf{b}@B \;:\; (s_1 s_2)\;\; s_1 \blacktriangleright \begin{bmatrix} \langle \mathsf{go}, \varepsilon\rangle.\;\; s_2 \blacktriangleleft \langle \mathsf{ok}, \varepsilon\rangle.\;\mathbf{end} \\ + \\ \langle \mathsf{stop}, \varepsilon\rangle.\; s_2 \blacktriangleleft \langle \mathsf{no}, \varepsilon\rangle.\;\mathbf{end} \\ + \\ \langle \mathsf{run}, \varepsilon\rangle.\; s_2 \blacktriangleleft \langle \mathsf{fine}, \varepsilon\rangle.\;\mathbf{end} \end{bmatrix}$$

Without the **red** part, the typing is minimal.

Global Calculus: Typing (3)

Types for the simple Buyer-Seller:

$$\mathbf{ch} \,@\, \mathbf{Buyer} : (st) \quad s \uparrow \mathsf{QuoteReq}(string) \ ; \ t \downarrow \mathsf{QuoteRes}(int) \ ; \ s \uparrow \mathsf{QuoteAcc}(string)$$

$$\mathbf{ch} \,@\, \mathbf{Seller} : (st) \quad s \downarrow \mathsf{QuoteReq}(string) \ ; \ t \uparrow \mathsf{QuoteRes}(int) \ ; \ s \downarrow \mathsf{QuoteAcc}(string)$$

Global Calculus: Typing (4)

Types for the Buyer-Seller with choice/conditional.

$$\mathbf{ch} @ \mathbf{Buyer} : (st) \quad s \uparrow \mathsf{QuoteReq}(\mathit{string})\ ;\ t \downarrow \mathsf{QuoteRes}(\mathit{int})\ ;\ s \uparrow \mathsf{QuoteAcc}(\mathit{string}) \oplus s \uparrow \mathsf{QuoteNG}(\mathit{void})$$

$$\mathbf{ch} @ \mathbf{Seller} : (st) \quad s \downarrow \mathsf{QuoteReq}(\mathit{string})\ ;\ t \uparrow \mathsf{QuoteRes}(\mathit{int})\ ;\ s \downarrow \mathsf{QuoteAcc}(\mathit{string})\ \&\ s \downarrow \mathsf{QuoteNG}(\mathit{void})$$

Global Calculus: Typing (5)

Types for the Buyer-Seller with loop.

$$\mathbf{ch} \,@\, \mathbf{Buyer} : (st) \quad s \uparrow \mathsf{QReq}(\mathit{string})\ ;\ \mu X.\ t \downarrow \mathsf{QRes}(\mathit{int})\ ;\ \begin{array}{l} s \uparrow \mathsf{QAcc}(\mathit{string})\ \oplus \\ s \uparrow \mathsf{QNoGood}(\mathit{void})\ ;\ X \end{array}$$

$$\mathbf{ch} \,@\, \mathbf{Seller} : (st) \quad s \downarrow \mathsf{QReq}(\mathit{string})\ ;\ \mu X.\ t \uparrow \mathsf{QRes}(\mathit{int})\ ;\ \begin{array}{l} s \downarrow \mathsf{QAcc}(\mathit{string})\ \& \\ s \downarrow \mathsf{QNoGood}(\mathit{void}) \end{array}$$

End-Point Calculus: syntax & reduction

$$
\begin{array}{rcl}
P & ::= & !\mathtt{ch}(\tilde{s}).P \mid \overline{\mathtt{ch}}(\nu\,\tilde{s}).P \mid s \triangleright \Sigma_i \mathsf{op}_i\langle y_i\rangle.P_i \mid s \triangleleft \mathsf{op}\langle e\rangle.P \\
& \mid & x := e.P \quad \mid \quad \text{if } e \text{ then } P_1 \text{ else } P_2 \quad \mid \quad P \oplus Q \\
& \mid & P|Q \quad \mid \quad (\nu\, s)P \quad \mid \quad X \quad \mid \quad \mathbf{rec}\, X.P \quad \mid \quad \mathbf{0} \\
M & ::= & A[P]_\sigma \quad \mid \quad M_1 \,|\, M_2 \quad \mid \quad (\nu\, s)M \quad \mid \quad \mathbf{0}
\end{array}
$$

Reduction rules (part):

$$
\begin{array}{l}
A[!\mathtt{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \mid B[\overline{\mathtt{ch}}(\nu\,\tilde{s}).Q \,|\, S]_{\sigma_B} \\
\qquad\qquad \rightarrow \; (\nu\,\tilde{s})(A[P \,|\, !\mathtt{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \mid B[Q \,|\, S]_{\sigma_B}) \\
A[s \triangleright \Sigma_i \mathsf{op}_i\langle y_i\rangle.P_i \,|\, R]_{\sigma_A} \mid B[s \triangleleft \mathsf{op}_j\langle V\rangle Q \,|\, S]_{\sigma_B} \\
\qquad\qquad \rightarrow \; A[P_i \,|\, R]_{\sigma_A[y \mapsto V]} \mid B[Q \,|\, S]_{\sigma_B}
\end{array}
$$

End-Point Calculus: syntax & reduction

$$
\begin{array}{rcl}
P & ::= & !\mathtt{ch}(\tilde{s}).P \mid \overline{\mathtt{ch}}(\nu\,\tilde{s}).P \mid s \rhd \Sigma_i \mathsf{op}_i\langle y_i \rangle.P_i \mid s \lhd \mathsf{op}\langle e \rangle.P \\
 & \mid & x := e.P \quad \mid \quad \text{if } e \text{ then } P_1 \text{ else } P_2 \quad \mid \quad P \oplus Q \\
 & \mid & P|Q \quad \mid \quad (\nu\, s)P \quad \mid \quad X \quad \mid \quad \mathbf{rec}\, X.P \quad \mid \quad \mathbf{0} \\
M & ::= & A[P]_\sigma \quad \mid \quad M_1 \,|\, M_2 \quad \mid \quad (\nu\, s)M \quad \mid \quad \mathbf{0}
\end{array}
$$

Reduction rules (part):

$$
A[\,!\mathtt{ch}(\tilde{s}).P \mid R]_{\sigma_A} \mid B[\overline{\mathtt{ch}}(\nu\,\tilde{s}).Q \mid S]_{\sigma_B}
$$

$$
\rightarrow \; (\nu\,\tilde{s})(A[P \mid !\mathtt{ch}(\tilde{s}).P \mid R]_{\sigma_A} \mid B[Q \mid S]_{\sigma_B})
$$

$$
A[s \rhd \Sigma_i \mathsf{op}_i\langle y_i \rangle.P_i \mid R]_{\sigma_A} \mid B[s \lhd \mathsf{op_j}\langle V \rangle Q \mid S]_{\sigma_B}
$$

$$
\rightarrow \; A[P_i \mid R]_{\sigma_A[y \mapsto V]} \mid B[Q \mid S]_{\sigma_B}
$$

End-Point Calculus: syntax & reduction

$$
\begin{array}{lcl}
P & ::= & !\mathtt{ch}(\tilde{s}).P \mid \overline{\mathtt{ch}}(\nu\,\tilde{s}).P \mid s \rhd \Sigma_i \mathsf{op}_i\langle y_i\rangle.P_i \mid s \lhd \mathsf{op}\langle e\rangle.P \\
& \mid & x := e.P \quad \mid \quad \text{if } e \text{ then } P_1 \text{ else } P_2 \quad \mid \quad P \oplus Q \\
& \mid & P|Q \quad \mid \quad (\nu\, s)P \quad \mid \quad X \quad \mid \quad \mathbf{rec}\, X.P \quad \mid \quad \mathbf{0} \\
M & ::= & A[P]_\sigma \quad \mid \quad M_1 \,|\, M_2 \quad \mid \quad (\nu\, s)M \quad \mid \quad \mathbf{0}
\end{array}
$$

Reduction rules (part):

$$
\begin{array}{l}
A[!\mathtt{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \mid B[\overline{\mathtt{ch}}(\nu\,\tilde{s}).Q \,|\, S]_{\sigma_B} \\
\qquad\qquad \to \ (\nu\,\tilde{s})(A[P \,|\, !\mathtt{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \mid B[Q \,|\, S]_{\sigma_B}) \\
A[s \rhd \Sigma_i \mathsf{op}_i\langle y_i\rangle.P_i \,|\, R]_{\sigma_A} \mid B[s \lhd \mathsf{op_j}\langle V\rangle Q \,|\, S]_{\sigma_B} \\
\qquad\qquad \to \ A[P_i \,|\, R]_{\sigma_A[y \mapsto V]} \mid B[Q \,|\, S]_{\sigma_B}
\end{array}
$$

End-Point Calculus: syntax & reduction

$$
\begin{array}{rcl}
P & ::= & !\mathtt{ch}(\tilde{s}).P \mid \overline{\mathtt{ch}}(\nu\,\tilde{s}).P \mid s \rhd \Sigma_i \mathsf{op}_i\langle y_i\rangle.P_i \mid s \lhd \mathsf{op}\langle e\rangle.P \\
& \mid & x := e.P \quad \mid \quad \text{if } e \text{ then } P_1 \text{ else } P_2 \quad \mid \quad P \oplus Q \\
& \mid & P|Q \quad \mid \quad (\nu\, s)P \quad \mid \quad X \quad \mid \quad \mathbf{rec}\, X.P \quad \mid \quad \mathbf{0} \\
M & ::= & A[P]_\sigma \quad \mid \quad M_1 \,|\, M_2 \quad \mid \quad (\nu\, s)M \quad \mid \quad \mathbf{0}
\end{array}
$$

Reduction rules (part):

$$
\begin{array}{l}
A[!\mathtt{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \mid B[\overline{\mathtt{ch}}(\nu\,\tilde{s}).Q \,|\, S]_{\sigma_B} \\
\qquad\qquad \to \ (\nu\,\tilde{s})(A[P \,|\, !\mathtt{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \mid B[Q \,|\, S]_{\sigma_B}) \\
A[s \rhd \Sigma_i \mathsf{op}_i\langle y_i\rangle.P_i \,|\, R]_{\sigma_A} \mid B[s \lhd \mathsf{op_j}\langle V\rangle Q \,|\, S]_{\sigma_B} \\
\qquad\qquad \to \ A[P_i \,|\, R]_{\sigma_A[y \mapsto V]} \mid B[Q \,|\, S]_{\sigma_B}
\end{array}
$$

End-Point Calculus: syntax & reduction

$$
\begin{array}{rcl}
P & ::= & !\mathtt{ch}(\tilde{s}).P \mid \overline{\mathtt{ch}}(\nu\,\tilde{s}).P \mid s \triangleright \Sigma_i \mathsf{op}_i\langle y_i\rangle.P_i \mid s \triangleleft \mathsf{op}\langle e\rangle.P \\
& \mid & x := e.P \quad \mid \quad \text{if } e \text{ then } P_1 \text{ else } P_2 \quad \mid \quad P \oplus Q \\
& \mid & P|Q \quad \mid \quad (\nu\, s)P \quad \mid \quad X \quad \mid \quad \mathbf{rec}\, X.P \quad \mid \quad \mathbf{0} \\
M & ::= & A[P]_\sigma \quad \mid \quad M_1 \,|\, M_2 \quad \mid \quad (\nu\, s)M \quad \mid \quad \mathbf{0}
\end{array}
$$

Reduction rules (part):

$$
\begin{array}{l}
A[!\mathtt{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \mid B[\overline{\mathtt{ch}}(\nu\,\tilde{s}).Q \,|\, S]_{\sigma_B} \\
\qquad\qquad \rightarrow \; (\nu\,\tilde{s})(A[P \,|\, !\mathtt{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \mid B[Q \,|\, S]_{\sigma_B}) \\
A[s \triangleright \Sigma_i \mathsf{op}_i\langle y_i\rangle.P_i \,|\, R]_{\sigma_A} \mid B[s \triangleleft \mathsf{op}_\mathsf{j}\langle V\rangle Q \,|\, S]_{\sigma_B} \\
\qquad\qquad \rightarrow \; A[P_i \,|\, R]_{\sigma_A[y \mapsto V]} \mid B[Q \,|\, S]_{\sigma_B}
\end{array}
$$

End-Point Calculus: typing (1)

- Session types: as before.

$$\alpha \quad ::= \quad s \blacktriangleright \Sigma_i \langle \mathsf{op_i}, \theta_i \rangle.\ \alpha_i \quad | \quad s \blacktriangleleft \Sigma_i \langle \mathsf{op_i}, \theta_i \rangle.\ \alpha_i$$
$$| \quad \alpha_1 | \alpha_2 \quad | \quad \mathbf{rec}\ t.\alpha \quad | \quad t \quad | \quad \mathbf{end}$$

- Typing: standard session typing with multiple session channels, with the initialisation rule:

$$(\textsc{Init})\ \frac{\Gamma \vdash_A P \triangleright \tilde{s}:\alpha}{\Gamma, ch@A:(\tilde{s})\alpha \vdash_A\ !ch(\tilde{s}).P \triangleright \emptyset}$$

- Properties: *Subject Reduction*, *Communication Error Freedom*, *Minimal Typing w.r.t. session subtyping*, ...

End-Point Calculus: Typing (2)

A term:

$$B[\ \ !\mathbf{b}(s_1 s_2).s_1 \rhd [\ \mathsf{go}.\ s_2 \lhd \mathsf{ok}.\mathbf{0} + \mathsf{stop}.\ s_2 \lhd \mathsf{no}.\mathbf{0}\ ]\ \ ]$$

can be given a typing (ε is the empty string):

$$\mathbf{b}@B\ :\ (s_1 s_2)\ \ s_1 \blacktriangleright \begin{bmatrix} \langle \mathsf{go}, \varepsilon \rangle.\ \ s_2 \blacktriangleleft \langle \mathsf{ok}, \varepsilon \rangle.\ \mathbf{end} \\ + \\ \langle \mathsf{stop}, \varepsilon \rangle.\ s_2 \blacktriangleleft \langle \mathsf{no}, \varepsilon \rangle.\ \mathbf{end} \end{bmatrix}$$

This is the minimal typing (the **green** part can be taken off by subtyping).

Formalising EPP

EPP projects a global description to multiple end-points:

$$(I, \sigma) \quad \mapsto \quad A[P]_{\sigma@A} \mid B[Q]_{\sigma@B} \mid C[R]_{\sigma@C} \mid \cdots$$

Desirable properties:

- *Type preservation: the typing is preserved through EPP.*
- *Soundness: nothing but behaviours in I are in its EPP.*
- *Completeness: all behaviours in I are in its EPP.*

Three descriptive principles guarantee them.

Three Principles

Well-structuredness conditions for global descriptions.

1. Connectedness: Basic local causality principle.
2. Well-threadedness: Stronger local causality principle based on session types (by which we can consistently extract a thread of local actions from a global code).
3. Coherence: Consistency among descriptions for a single code scattered over a global description.

All three properties can be efficiently (in)validated.

Descriptive Principle (1): Connectedness

First observed by Ross-Talbot (2004).

Bad...

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{C} \longrightarrow \mathbf{D} : t\langle \mathsf{hello} \rangle.$

$\ldots$

Good...

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{C} : \mathbf{c}(t).$

$\ldots$

Descriptive Principle (1): Connectedness

First observed by Ross-Talbot (2004).

Bad...

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{C} \longrightarrow \mathbf{D} : t\langle \mathsf{hello} \rangle.$

$\ldots$

Good...

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{C} : \mathbf{c}(t).$

$\ldots$

Desc. Principle (2): Well-Threadedness

Bad...

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{C} : \mathbf{c}(t).$

$\mathbf{C} \longrightarrow \mathbf{A} : \mathbf{a}(r).$

$\mathbf{A} \longrightarrow \mathbf{C} : r\langle \mathsf{hi} \rangle.$

$\mathbf{C} \longrightarrow \mathbf{A} : r\langle \mathsf{hi} \rangle.$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{hello} \rangle$

$\ldots$

Good...

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{C} : \mathbf{c}(t).$

$\mathbf{C} \longrightarrow \mathbf{A} : \mathbf{a}(r).$

$\mathbf{A} \longrightarrow \mathbf{C} : r\langle \mathsf{hi} \rangle.$

$\mathbf{C} \longrightarrow \mathbf{B} : t\langle \mathsf{acc} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s\langle \mathsf{ok} \rangle.$

$\ldots$

Desc. Principle (2): Well-Threadedness

Bad...

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{C} : \mathbf{c}(t).$

$\mathbf{C} \longrightarrow \mathbf{A} : \mathbf{a}(r).$

$\mathbf{A} \longrightarrow \mathbf{C} : r\langle \mathsf{hi} \rangle.$

$\mathbf{C} \longrightarrow \mathbf{A} : r\langle \mathsf{hi} \rangle.$

$\mathbf{A}\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{hello} \rangle.$

$\ldots$

Good...

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{C} : \mathbf{c}(t).$

$\mathbf{C} \longrightarrow \mathbf{A} : \mathbf{a}(r).$

$\mathbf{A} \longrightarrow \mathbf{C} : r\langle \mathsf{hi} \rangle.$

$\mathbf{C} \longrightarrow \mathbf{B} : t\langle \mathsf{acc} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s\langle \mathsf{ok} \rangle.$

$\ldots$

Projecting Threads (1)

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{C} : \mathbf{c}(t).$

$\mathbf{C} \longrightarrow \mathbf{A} : \mathbf{a}(r).$

$\mathbf{A} \longrightarrow \mathbf{C} : r\langle \mathsf{hi} \rangle.$

$\mathbf{C} \longrightarrow \mathbf{A} : r\langle \mathsf{hi} \rangle.$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{hello} \rangle.$

$\ldots$

$$\mathbf{A} \left[\begin{array}{l} \overline{\mathbf{b}}(s).\, s \lhd \mathsf{go}.\, \underline{s \lhd \mathsf{hello}}.P \\ !\mathbf{a}(r).\, r \lhd \mathsf{hi}.\, r \rhd \mathsf{hi}.P' \end{array} \right]$$

$$\mathbf{B} \left[\begin{array}{l} !\mathbf{b}(s).\; s \rhd \mathsf{go}. \\ \qquad\qquad \overline{\mathbf{c}}(t).\; \underline{s \rhd \mathsf{hello}}.Q \end{array} \right]$$

$$\mathbf{C} \;\left[\; !\mathbf{c}(t).\overline{\mathbf{a}}(r).\, r \rhd \mathsf{hi}.\, r \lhd \mathsf{hi}.S \,\right]$$

Projecting Threads (2)

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s).$

$\mathbf{A} \longrightarrow \mathbf{B} : s\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{C} : \mathbf{c}(t).$

$\mathbf{C} \longrightarrow \mathbf{A} : \mathbf{a}(r).$

$\mathbf{A} \longrightarrow \mathbf{C} : r\langle \mathsf{hi} \rangle.$

$\mathbf{C} \longrightarrow \mathbf{B} : t\langle \mathsf{acc} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s\langle \mathsf{ok} \rangle.$

$\ldots$

$$\mathbf{A} \left[\begin{array}{l} \overline{\mathbf{b}}(s).\, s \lhd \mathsf{go}.\, s \rhd \mathsf{ok}.P \\ !\mathbf{a}(r).\, r \lhd \mathsf{hi}.\, P' \end{array} \right]$$

$$\mathbf{B} \left[\begin{array}{l} !\mathbf{b}(s).\ s \rhd \mathsf{go}. \\ \qquad \overline{\mathbf{c}}(t).\ t \rhd \mathsf{acc}.s \lhd \mathsf{ok}.Q \end{array} \right]$$

$$\mathbf{C}\ [\ !\mathbf{c}(t).\overline{\mathbf{a}}(r).\, r \rhd \mathsf{hi}.\, t \lhd \mathsf{acc}.\ S\ ]$$

Descriptive Principle (3): Coherence

Bad...

if $x@\mathbf{A}$ **then**

$\mathbf{A} \to \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \to \mathbf{B} : s_1 \langle \mathsf{go} \rangle.$

$\mathbf{B} \to \mathbf{A} : s_2 \langle \mathsf{ok} \rangle.\ \mathbf{0}$

else

$\mathbf{A} \to \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \to \mathbf{B} : s_1 \langle \mathsf{go} \rangle.$

$\mathbf{B} \to \mathbf{A} : s_2 \langle \mathsf{no} \rangle.\ \mathbf{0}$

end

Good...

if $x@\mathbf{A}$ **then**

$\mathbf{A} \to \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \to \mathbf{B} : s_1 \langle \mathsf{go} \rangle.$

$\mathbf{B} \to \mathbf{A} : s_2 \langle \mathsf{ok} \rangle.\ \mathbf{0}$

else

$\mathbf{A} \to \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \to \mathbf{B} : s_1 \langle \mathsf{stop} \rangle.$

$\mathbf{B} \to \mathbf{A} : s_2 \langle \mathsf{no} \rangle.\ \mathbf{0}$

end

Descriptive Principle (3): Coherence

Bad...

if x@**A then**

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \longrightarrow \mathbf{B} : s_1\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s_2\langle \mathsf{ok} \rangle.\ \mathbf{0}$

else

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \longrightarrow \mathbf{B} : s_1\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s_2\langle \mathsf{no} \rangle.\ \mathbf{0}$

end

Good...

if x@**A then**

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \longrightarrow \mathbf{B} : s_1\langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s_2\langle \mathsf{ok} \rangle.\ \mathbf{0}$

else

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \longrightarrow \mathbf{B} : s_1\langle \mathsf{stop} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s_2\langle \mathsf{no} \rangle.\ \mathbf{0}$

end

Descriptive Principle (3): Coherence

Bad...

if x@**A** **then**

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \longrightarrow \mathbf{B} : s_1 \langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s_2 \langle \mathsf{ok} \rangle.\ \mathbf{0}$

else

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \longrightarrow \mathbf{B} : s_1 \langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s_2 \langle \mathsf{no} \rangle.\ \mathbf{0}$

end

Good...

if x@**A** **then**

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \longrightarrow \mathbf{B} : s_1 \langle \mathsf{go} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s_2 \langle \mathsf{ok} \rangle.\ \mathbf{0}$

else

$\mathbf{A} \longrightarrow \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \longrightarrow \mathbf{B} : s_1 \langle \mathsf{stop} \rangle.$

$\mathbf{B} \longrightarrow \mathbf{A} : s_2 \langle \mathsf{no} \rangle.\ \mathbf{0}$

end

Merging Coherent Threads

We project each branch onto **B** and merge the results.

if x@**A then**

$\mathbf{A} \to \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \to \mathbf{B} : s_1 \langle \mathsf{go} \rangle.$

$\mathbf{B} \to \mathbf{A} : s_2 \langle \mathsf{ok} \rangle . \mathbf{0}$

else

$\mathbf{A} \to \mathbf{B} : \mathbf{b}(s_1 s_2).$

$\mathbf{A} \to \mathbf{B} : s_1 \langle \mathsf{stop} \rangle.$

$\mathbf{B} \to \mathbf{A} : s_2 \langle \mathsf{no} \rangle . \ \mathbf{0}$

end

if-branch.

$!\mathbf{b}(s_1 s_2).\ s_1 \triangleright \mathsf{go}.\ s_2 \triangleleft \mathsf{ok}.\ \mathbf{0}$

else-branch.

$!\mathbf{b}(s_1 s_2).\ s_1 \triangleright \mathsf{stop}.\ s_2 \triangleleft \mathsf{no}.\ \mathbf{0}$

merging the two.

$$!\mathbf{b}(s_1 s_2).\ s_1 \triangleright \left[\begin{array}{l} \mathsf{go}.\ s_2 \triangleleft \mathsf{ok}.\ \mathbf{0} + \\ \mathsf{stop}.\ s_2 \triangleleft \mathsf{no}.\ \mathbf{0} \end{array} \right]$$

Three Principles: Summary

Definition. A term I in the global calculus is:

1. *Connected* if, in each prefix, either we have (1) e.g. $A \to B; A \to B$; or (2) e.g. $A \to B; B \to C$.

2. *Well-threaded* if it has a consistent colouring.

3. *Coherent* if its two threads for the same subject are always compatible.

A typable I is *well-formed* if it satisfies all three conditions.
A well-formed I is *annotated* if it is appropriately coloured.

EPP: Definition

Definition. Let I be well-formed, be annotated and does not contain hiding. Let $((\nu\,\tilde{s})I,\sigma)$ be well-typed. We set:

$$
\begin{aligned}
\mathtt{Epp}((\nu\,\tilde{s})I,\sigma) &\stackrel{\text{def}}{=} (\nu\,\tilde{s})\,\Pi_{A\in\mathsf{part}(I)}A[\,\mathtt{epp}(I,A)\,]_{\sigma@A}\\
\mathtt{epp}(I,A) &\stackrel{\text{def}}{=} \Pi_{g\in\mathtt{tg}(I,A)}\mathtt{merge}(\{\mathtt{tp}(I,i)\mid i\in g\})
\end{aligned}
$$

where $\mathtt{tg}(I,A)$ are thread groups for A: and $\mathtt{tp}(I,i)$ projects I onto the i-th thread.

Pruning

Definition. (pruning)

$P \ll Q|!R$ when (1) $\Gamma \vdash_{\mathtt{min}} P$, $\Gamma \vdash Q$ and $\mathsf{fn}(Q) \cap \mathsf{sbj}(!R) = \emptyset$;

(2) Cutting off Q's input branches not in Γ coincides with P.

Pruning

Definition. (pruning)
$P \ll Q|!R$ when (1) $\Gamma \vdash_{\mathtt{min}} P$, $\Gamma \vdash Q$ and $\mathsf{fn}(Q) \cap \mathsf{sbj}(!R) = \emptyset$;
(2) Cutting off Q's input branches not in Γ coincides with P.

E.g. $!\mathbf{b}(s).s \rhd \mathsf{go}.P_1 \mid \overline{\mathbf{b}}(s).s \lhd \mathsf{go}.Q \quad \ll$

$$!\mathbf{b}(s).s \rhd [\,\mathsf{go}.P_1 + \mathsf{stop}.P_2\,] \mid \overline{\mathbf{b}}(s).s \lhd \mathsf{go}.Q \mid !R$$

Pruning

Definition. (pruning)

$P \ll Q|!R$ when (1) $\Gamma \vdash_{\mathtt{min}} P$, $\Gamma \vdash Q$ and $\mathsf{fn}(Q) \cap \mathsf{sbj}(!R) = \emptyset$; (2) Cutting off Q's input branches not in Γ coincides with P.

E.g. $!\mathbf{b}(s).s \rhd \mathsf{go}.P_1 \mid \overline{\mathbf{b}}(s).s \lhd \mathsf{go}.Q \quad \ll$

$$!\mathbf{b}(s).s \rhd [\, \mathsf{go}.P_1 + \mathsf{stop}.P_2 \,] \mid \overline{\mathbf{b}}(s).s \lhd \mathsf{go}.Q \mid !R$$

Lemma. (pruning lemma)

(1) The relation $\ll$ is a strong bisimulation if we do not count visible actions at pruned inputs. (2) $\ll$ is transitive.

EPP Theorem

Theorem. Assume I is well-formed and $\Gamma \vdash \sigma$. Then:

1. $\Gamma \vdash_{\mathtt{min}} I$ implies $\Gamma \vdash_{\mathtt{min}} \mathtt{Epp}(I, \sigma)$.
2. $(I, \sigma) \to (I', \sigma')$ implies $\mathtt{Epp}(I, \sigma) \to M \gg \mathtt{Epp}(I', \sigma')$.
3. $\mathtt{Epp}(I, \sigma) \to M$ implies $(I, \sigma) \to (I', \sigma')$ s.t. $\mathtt{Epp}(I', \sigma') \ll M$.

Corollary. If $\Gamma \vdash I$ is well-formed and $\Gamma \vdash \sigma$:

1. $\mathtt{Epp}(I, \sigma)$ never goes wrong.
2. $(I, \sigma) \to^n (I', \sigma')$ implies $\mathtt{Epp}(I, \sigma) \to^n M \gg \mathtt{Epp}(I', \sigma')$.
   Conversely $\mathtt{Epp}(I, \sigma) \to^n M$ implies $(I, \sigma) \to^n (I', \sigma')$ such that $\mathtt{Epp}(I', \sigma') \ll M$.

So what does EPP theorem means?

We first write down a global description and ...

- Produce a prototype code (only communication behaviour).
- Produce a full application.
- Produce a run-time monitor.
- Do a debugging (how is everybody behaving?)
- Do a conformance checking (can we really use that web service for this protocol?)
- Do a conformance checking for team programming (is my code conformant to a global specification?).

Conformance (for safety)

1. We say *P conforms to Q* when Q weakly simulates P, i.e. all visible typed communication behaviours of P can be simulated by those of Q.

2. We say *P conforms to I at A* when P conforms to $\mathsf{epp}(I, A)$ (up to the minimum type of the latter).

3. We can use model checking, syntactic approximation, hand calculation (coinduction), ... to validate or invalidate conformance.

CDL Requirements and EPP

C-CSF-007: To be successful a CDL description MUST be verifiable at runtime.

$\Longrightarrow$ Use LTS and process equivalence via EPP.

C-CSF-008: To be successful a CDL description MUST enable static verification of correctness properties.

$\Longrightarrow$ Use types and logics for processes via EPP.

Current Status

- The current implementation of EPP in the pi4soa tool is independently designed by Gary Brown, and closely follows the presented framework.
- The implementation of a formally-based EPP is currently underway for the pi4soa code base.
- A W3C working note presenting an EPP theory is planned, augmenting the WS-CDL 1.0 specification.

References

- [Bhargavan/Fournet/Gordon 2006] Verification of web service secruity.
- [Laneve/Padovani 2006] Analysis of orchastration.
- [Busi et al. 2006] Conformance check in web service (between choreographies and orchestration).

The technical paper on the presented material:

http://www.doc.ic.ac.uk/∼yoshida/tic

Conclusion

Part 1 Basic Theory (Processes and Types)

- The π-Calulus
- Idioms for Interactions
- Session Types

Part 2 Web Services and the π-Calculus

- Web Services Choreography Description Language
- Global Language and the End-Point Calculus
- Correctness