On Global Types and Multi-Party Sessions

Joint work with Giuseppe Castagna and Luca Padovani

Workshop on Behavioural Type Systems, Lisbon, 19 April 2011

UNIVERSITÀ DEGLI STUDI DI TORINO
ALMA UNIVERSITAS TAURINENSIS

Outline

Outline

Outline

Outline

Global types and session types

- Overview
- Global types
- Session types

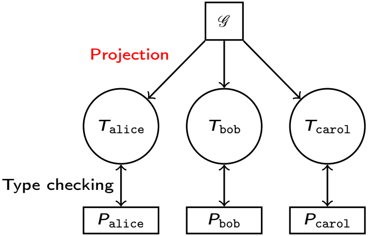
Projections

- Semantic projection
- Algorithmic projection
- Kleene star and recursion

Related approaches

- Sessions and Choreographies
- Automata
- Cryptographic protocols

Global types, session types and processes



Global Type $\mathscr{G} = \mathtt{alice} \xrightarrow{\textit{nat}} \mathtt{bob};\ \mathtt{bob} \xrightarrow{\textit{nat}} \mathtt{carol}$

Session Types $T_{\mathtt{bob}} = \mathtt{alice}?\textit{nat}.\ \mathtt{carol}!\textit{nat}.\ \mathsf{end}$

Processes $P_{\mathtt{bob}} =$

```
receive x from alice;
send x+42 to carol;
end
```

Informal descriptions, global types and session types

Seller sends buyer a price and a description of the product; then buyer initiate a loop of zero or more interactions in which buyer sends an offer and then seller sends a price; then buyer sends seller acceptance or it quits the conversation.

Informal descriptions, global types and session types

Seller sends buyer a price and a description of the product; then buyer initiate a loop of zero or more interactions in which buyer sends an offer and then seller sends a price; then buyer sends seller acceptance or it quits the conversation.

$$
\begin{array}{l}
(\mathtt{seller} \xrightarrow{\mathit{descr}} \mathtt{buyer} \wedge \mathtt{seller} \xrightarrow{\mathit{price}} \mathtt{buyer});\\
(\mathtt{buyer} \xrightarrow{\mathit{offer}} \mathtt{seller}; \mathtt{seller} \xrightarrow{\mathit{price}} \mathtt{buyer})^*;\\
(\mathtt{buyer} \xrightarrow{\mathit{accept}} \mathtt{seller} \vee \mathtt{buyer} \xrightarrow{\mathit{quit}} \mathtt{seller})
\end{array}
$$

Informal descriptions, global types and session types

$$(\texttt{seller} \xrightarrow{\textit{descr}} \texttt{buyer} \wedge \texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer});$$
$$(\texttt{buyer} \xrightarrow{\textit{offer}} \texttt{seller} ; \texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer})^{*};$$
$$(\texttt{buyer} \xrightarrow{\textit{accept}} \texttt{seller} \vee \texttt{buyer} \xrightarrow{\textit{quit}} \texttt{seller})$$

$$\begin{array}{lcl}
\texttt{seller} & \mapsto & \texttt{buyer}!\textit{descr}.\texttt{buyer}!\textit{price}.\mathtt{rec}\ X. \\
 & & (\texttt{buyer}?\textit{offer}.\texttt{buyer}!\textit{price}.X + \\
 & & \texttt{buyer}?\textit{accept} + \texttt{buyer}?\textit{quit}) \\
\texttt{buyer} & \mapsto & \texttt{seller}?\textit{descr}.\texttt{seller}?\textit{price}.\mathtt{rec}\ Y. \\
 & & (\texttt{seller}!\textit{offer}.\texttt{seller}?\textit{price}.Y \oplus \\
 & & \texttt{seller}!\textit{accept} \oplus \texttt{seller}!\textit{quit})
\end{array}$$

Informal descriptions, global types and session types

$$(\texttt{seller} \xrightarrow{\mathit{descr}} \texttt{buyer} \wedge \texttt{seller} \xrightarrow{\mathit{price}} \texttt{buyer});$$
$$(\texttt{buyer} \xrightarrow{\mathit{offer}} \texttt{seller}; \texttt{seller} \xrightarrow{\mathit{price}} \texttt{buyer})^{*};$$
$$(\texttt{buyer} \xrightarrow{\mathit{accept}} \texttt{seller} \vee \texttt{buyer} \xrightarrow{\mathit{quit}} \texttt{seller})$$

$$\begin{array}{lcl} \texttt{seller} & \mapsto & \texttt{buyer}!\mathit{price}.\texttt{buyer}!\mathit{descr}.\texttt{rec}\ X. \\ & & (\texttt{buyer}?\mathit{offer}.\texttt{buyer}!\mathit{price}.X + \\ & & \texttt{buyer}?\mathit{accept} + \texttt{buyer}?\mathit{quit}) \\ \texttt{buyer} & \mapsto & \texttt{seller}?\mathit{price}.\texttt{seller}?\mathit{descr}.\texttt{rec}\ Y. \\ & & (\texttt{seller}!\mathit{offer}.\texttt{seller}?\mathit{price}.Y \oplus \\ & & \texttt{seller}!\mathit{accept} \oplus \texttt{seller}!\mathit{quit}) \end{array}$$

Properties of projections

1. Sequentiality: an implementation in which `buyer` may send *accept* before receiving *price* violates the specification.

Properties of projections

1. Sequentiality: an implementation in which `buyer` may send *accept* before receiving *price* violates the specification.
2. Alternativeness: an implementation in which `buyer` emits both *accept* and *quit* (or none of them) in the same execution violates the specification.

Properties of projections

1. Sequentiality: an implementation in which `buyer` may send *accept* before receiving *price* violates the specification.
2. Alternativeness: an implementation in which `buyer` emits both *accept* and *quit* (or none of them) in the same execution violates the specification.
3. Shuffling: an implementation in which `seller` emits *price* without emitting *descr* violates the specification.

Properties of projections

1. Sequentiality: an implementation in which `buyer` may send *accept* before receiving *price* violates the specification.
2. Alternativeness: an implementation in which `buyer` emits both *accept* and *quit* (or none of them) in the same execution violates the specification.
3. Shuffling: an implementation in which `seller` emits *price* without emitting *descr* violates the specification.
4. Fitness: an implementation in which `seller` sends `buyer` any message other than *price* and *descr* violates the specification.

Properties of projections

1. Sequentiality: an implementation in which `buyer` may send *accept* before receiving *price* violates the specification.
2. Alternativeness: an implementation in which `buyer` emits both *accept* and *quit* (or none of them) in the same execution violates the specification.
3. Shuffling: an implementation in which `seller` emits *price* without emitting *descr* violates the specification.
4. Fitness: an implementation in which `seller` sends `buyer` any message other than *price* and *descr* violates the specification.
5. Exhaustivity: an implementation in which no execution of `buyer` emits *accept* violates the specification.

Flawed global types

no covert channel

Flawed global types

no covert channel

- No sequentiality: some sequentiality constraint between independent interactions
  $(\mathtt{p} \xrightarrow{a} \mathtt{q} ; \mathtt{r} \xrightarrow{b} \mathtt{s})$

Flawed global types

no covert channel

- No sequentiality: some sequentiality constraint between independent interactions
  $(\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s})$

- No knowledge for choice: some participant must behave in different ways in accordance with some choice it is unaware of
  $(\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{q} \xrightarrow{a} \mathtt{r}; \mathtt{r} \xrightarrow{a} \mathtt{p}) \quad \vee \quad (\mathtt{p} \xrightarrow{b} \mathtt{q}; \mathtt{q} \xrightarrow{a} \mathtt{r}; \mathtt{r} \xrightarrow{b} \mathtt{p})$

Flawed global types

no covert channel

- No sequentiality: some sequentiality constraint between independent interactions
  $(\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s})$
- No knowledge for choice: some participant must behave in different ways in accordance with some choice it is unaware of
  $(\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{q} \xrightarrow{a} \mathtt{r}; \mathtt{r} \xrightarrow{a} \mathtt{p}) \quad \vee \quad (\mathtt{p} \xrightarrow{b} \mathtt{q}; \mathtt{q} \xrightarrow{a} \mathtt{r}; \mathtt{r} \xrightarrow{b} \mathtt{p})$
- No knowledge, no choice: incompatible behaviours such as performing and input or an output in mutual exclusion
  $\mathtt{p} \xrightarrow{a} \mathtt{q} \vee \mathtt{q} \xrightarrow{b} \mathtt{p}$

Syntax of global types

$$\mathscr{G} \quad ::= \quad \textbf{Global Type}$$

Syntax of global types

$\mathscr{G}$	$::=$		**Global Type**
		skip	(skip)

Syntax of global types

$\mathscr{G}$	$::=$		**Global Type**	
		skip	(skip)	
	$\mid$	$\pi \xrightarrow{a} \mathrm{p}$	(interaction)	multiple senders

Syntax of global types

$\mathscr{G}$	$::=$		**Global Type**
		$\textsf{skip}$	(skip)
	$\mid$	$\pi \xrightarrow{a} \mathrm{p}$	(interaction)
	$\mid$	$\mathscr{G} ; \mathscr{G}$	(sequence)

Syntax of global types

$\mathscr{G}$	$::=$		**Global Type**
		skip	(skip)
	$\mid$	$\pi \xrightarrow{a} \mathtt{p}$	(interaction)
	$\mid$	$\mathscr{G} ; \mathscr{G}$	(sequence)
	$\mid$	$\mathscr{G} \wedge \mathscr{G}$	(both)

Syntax of global types

$\mathscr{G}$	$::=$		**Global Type**
		skip	(skip)
	$\mid$	$\pi \xrightarrow{a} \mathrm{p}$	(interaction)
	$\mid$	$\mathscr{G} ; \mathscr{G}$	(sequence)
	$\mid$	$\mathscr{G} \wedge \mathscr{G}$	(both)
	$\mid$	$\mathscr{G} \vee \mathscr{G}$	(either)

Syntax of global types

$\mathscr{G}$	$::=$		**Global Type**
		skip	(skip)
	$\mid$	$\pi \xrightarrow{a} \mathrm{p}$	(interaction)
	$\mid$	$\mathscr{G} ; \mathscr{G}$	(sequence)
	$\mid$	$\mathscr{G} \wedge \mathscr{G}$	(both)
	$\mid$	$\mathscr{G} \vee \mathscr{G}$	(either)
	$\mid$	$\mathscr{G}^*$	(star) fairness

Syntax of global types

$\mathscr{G}$	$::=$		**Global Type**
		skip	(skip)
	$\mid$	$\pi \xrightarrow{a} \mathrm{p}$	(interaction)
	$\mid$	$\mathscr{G} ; \mathscr{G}$	(sequence)
	$\mid$	$\mathscr{G} \wedge \mathscr{G}$	(both)
	$\mid$	$\mathscr{G} \vee \mathscr{G}$	(either)
	$\mid$	$\mathscr{G}^*$	(star)

$\pi \xrightarrow{a} \{\mathrm{p}_i\}_{i \in I}$ can be encoded as $\bigwedge_{i \in I}(\pi \xrightarrow{a} \mathrm{p}_i)$

Examples

join

$$(\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer1} \wedge \texttt{bank} \xrightarrow{\textit{mortgage}} \texttt{buyer2});$$
$$(\{\texttt{buyer1},\texttt{buyer2}\} \xrightarrow{\textit{accept}} \texttt{seller} \wedge \{\texttt{buyer1},\texttt{buyer2}\} \xrightarrow{\textit{accept}} \texttt{bank})$$

Examples

join

$$(\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer1} \wedge \texttt{bank} \xrightarrow{\textit{mortgage}} \texttt{buyer2});$$
$$(\{\texttt{buyer1},\texttt{buyer2}\} \xrightarrow{\textit{accept}} \texttt{seller} \wedge \{\texttt{buyer1},\texttt{buyer2}\} \xrightarrow{\textit{accept}} \texttt{bank})$$

fork

$$\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer1} \wedge \texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer2}$$

Examples

join

$$(\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer1} \wedge \texttt{bank} \xrightarrow{\textit{mortgage}} \texttt{buyer2});$$
$$(\{\texttt{buyer1},\texttt{buyer2}\} \xrightarrow{\textit{accept}} \texttt{seller} \wedge \{\texttt{buyer1},\texttt{buyer2}\} \xrightarrow{\textit{accept}} \texttt{bank})$$

fork

$$\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer1} \wedge \texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer2}$$

common participants in parallel actions

Examples

different receivers in a choice

$$\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer1}; \texttt{buyer1} \xrightarrow{\textit{price}} \texttt{buyer2} \vee$$
$$\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer2}; \texttt{buyer2} \xrightarrow{\textit{price}} \texttt{buyer1}$$

Examples

different receivers in a choice

$$\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer1}; \texttt{buyer1} \xrightarrow{\textit{price}} \texttt{buyer2} \vee$$
$$\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer2}; \texttt{buyer2} \xrightarrow{\textit{price}} \texttt{buyer1}$$

different sets of participants for alternatives

$$(\texttt{seller} \xrightarrow{\textit{agency}} \texttt{broker}; \texttt{broker} \xrightarrow{\textit{price}} \texttt{buyer} \vee \texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer});$$
$$\texttt{buyer} \xrightarrow{\textit{answer}} \texttt{broker}$$

Examples

different receivers in a choice

$$\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer1}; \texttt{buyer1} \xrightarrow{\textit{price}} \texttt{buyer2} \vee$$
$$\texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer2}; \texttt{buyer2} \xrightarrow{\textit{price}} \texttt{buyer1}$$

different sets of participants for alternatives

$$(\texttt{seller} \xrightarrow{\textit{agency}} \texttt{broker}; \texttt{broker} \xrightarrow{\textit{price}} \texttt{buyer} \vee \texttt{seller} \xrightarrow{\textit{price}} \texttt{buyer});$$
$$\texttt{buyer} \xrightarrow{\textit{answer}} \texttt{broker}$$

different sets of participants when choosing between repeating or exiting a loop

$$\texttt{seller} \xrightarrow{\textit{agency}} \texttt{broker}; (\texttt{broker} \xrightarrow{\textit{offer}} \texttt{buyer}; \texttt{buyer} \xrightarrow{\textit{counteroffer}} \texttt{broker})^*;$$
$$(\texttt{broker} \xrightarrow{\textit{result}} \texttt{seller} \wedge \texttt{broker} \xrightarrow{\textit{result}} \texttt{buyer})$$

Traces of global types

$$\mathrm{tr}(\mathsf{skip}) = \{\varepsilon\}$$

Traces of global types

$$\begin{aligned} \mathrm{tr}(\mathsf{skip}) &= \{\varepsilon\} \\ \mathrm{tr}(\pi \xrightarrow{a} \mathrm{p}) &= \{\pi \xrightarrow{a} \mathrm{p}\} \end{aligned}$$

Traces of global types

$$\begin{aligned}\mathrm{tr}(\mathsf{skip}) &= \{\varepsilon\}\\ \mathrm{tr}(\pi \xrightarrow{a} \mathrm{p}) &= \{\pi \xrightarrow{a} \mathrm{p}\}\\ \mathrm{tr}(\mathscr{G}_1;\mathscr{G}_2) &= \mathrm{tr}(\mathscr{G}_1)\mathrm{tr}(\mathscr{G}_2)\end{aligned}$$

Traces of global types

$$
\begin{aligned}
\mathrm{tr}(\mathsf{skip}) &= \{\varepsilon\} \\
\mathrm{tr}(\pi \xrightarrow{a} \mathtt{p}) &= \{\pi \xrightarrow{a} \mathtt{p}\} \\
\mathrm{tr}(\mathscr{G}_1 ; \mathscr{G}_2) &= \mathrm{tr}(\mathscr{G}_1)\mathrm{tr}(\mathscr{G}_2) \\
\mathrm{tr}(\mathscr{G}^*) &= (\mathrm{tr}(\mathscr{G}))^\star
\end{aligned}
$$

Traces of global types

$$
\begin{aligned}
\mathrm{tr}(\mathsf{skip}) &= \{\varepsilon\} \\
\mathrm{tr}(\pi \xrightarrow{a} \mathrm{p}) &= \{\pi \xrightarrow{a} \mathrm{p}\} \\
\mathrm{tr}(\mathscr{G}_1;\mathscr{G}_2) &= \mathrm{tr}(\mathscr{G}_1)\mathrm{tr}(\mathscr{G}_2) \\
\mathrm{tr}(\mathscr{G}^*) &= (\mathrm{tr}(\mathscr{G}))^\star \\
\mathrm{tr}(\mathscr{G}_1 \vee \mathscr{G}_2) &= \mathrm{tr}(\mathscr{G}_1) \cup \mathrm{tr}(\mathscr{G}_2)
\end{aligned}
$$

Traces of global types

$$
\begin{array}{rcl}
\mathrm{tr}(\mathsf{skip}) & = & \{\varepsilon\} \\
\mathrm{tr}(\pi \xrightarrow{a} \mathrm{p}) & = & \{\pi \xrightarrow{a} \mathrm{p}\} \\
\mathrm{tr}(\mathscr{G}_1 ; \mathscr{G}_2) & = & \mathrm{tr}(\mathscr{G}_1)\mathrm{tr}(\mathscr{G}_2) \\
\mathrm{tr}(\mathscr{G}^*) & = & (\mathrm{tr}(\mathscr{G}))^\star \\
\mathrm{tr}(\mathscr{G}_1 \vee \mathscr{G}_2) & = & \mathrm{tr}(\mathscr{G}_1) \cup \mathrm{tr}(\mathscr{G}_2) \\
\mathrm{tr}(\mathscr{G}_1 \wedge \mathscr{G}_2) & = & \mathrm{tr}(\mathscr{G}_1) ⧢ \mathrm{tr}(\mathscr{G}_2)
\end{array}
$$

$$
L_1 ⧢ L_2 \stackrel{\text{def}}{=} \{\varphi_1 \psi_1 \cdots \varphi_n \psi_n \mid \varphi_1 \cdots \varphi_n \in L_1 \wedge \psi_1 \cdots \psi_n \in L_2\}
$$

Traces of global types

$$
\begin{aligned}
\mathrm{tr}(\mathsf{skip}) &= \{\varepsilon\} \\
\mathrm{tr}(\pi \xrightarrow{a} \mathrm{p}) &= \{\pi \xrightarrow{a} \mathrm{p}\} \\
\mathrm{tr}(\mathscr{G}_1;\mathscr{G}_2) &= \mathrm{tr}(\mathscr{G}_1)\mathrm{tr}(\mathscr{G}_2) \\
\mathrm{tr}(\mathscr{G}^*) &= (\mathrm{tr}(\mathscr{G}))^\star \\
\mathrm{tr}(\mathscr{G}_1 \vee \mathscr{G}_2) &= \mathrm{tr}(\mathscr{G}_1) \cup \mathrm{tr}(\mathscr{G}_2) \\
\mathrm{tr}(\mathscr{G}_1 \wedge \mathscr{G}_2) &= \mathrm{tr}(\mathscr{G}_1) \sqcup\!\sqcup\, \mathrm{tr}(\mathscr{G}_2)
\end{aligned}
$$

$$L_1 \sqcup\!\sqcup L_2 \stackrel{\mathrm{def}}{=} \{\varphi_1\psi_1\cdots\varphi_n\psi_n \mid \varphi_1\cdots\varphi_n \in L_1 \wedge \psi_1\cdots\psi_n \in L_2\}$$

$$\mathscr{G} = (\mathrm{p} \xrightarrow{a} \mathrm{q} \wedge \mathrm{p} \xrightarrow{b} \mathrm{q});(\mathrm{q} \xrightarrow{c} \mathrm{p};\mathrm{p} \xrightarrow{b} \mathrm{q})^*;(\mathrm{q} \xrightarrow{d} \mathrm{p} \vee \mathrm{q} \xrightarrow{e} \mathrm{p})$$

Traces of global types

$$
\begin{array}{rcl}
\mathrm{tr}(\mathsf{skip}) & = & \{\varepsilon\} \\
\mathrm{tr}(\pi \xrightarrow{a} \mathrm{p}) & = & \{\pi \xrightarrow{a} \mathrm{p}\} \\
\mathrm{tr}(\mathscr{G}_1;\mathscr{G}_2) & = & \mathrm{tr}(\mathscr{G}_1)\mathrm{tr}(\mathscr{G}_2) \\
\mathrm{tr}(\mathscr{G}^*) & = & (\mathrm{tr}(\mathscr{G}))^\star \\
\mathrm{tr}(\mathscr{G}_1 \vee \mathscr{G}_2) & = & \mathrm{tr}(\mathscr{G}_1) \cup \mathrm{tr}(\mathscr{G}_2) \\
\mathrm{tr}(\mathscr{G}_1 \wedge \mathscr{G}_2) & = & \mathrm{tr}(\mathscr{G}_1) \sqcup\!\sqcup \mathrm{tr}(\mathscr{G}_2)
\end{array}
$$

$$L_1 \sqcup\!\sqcup L_2 \stackrel{\text{def}}{=} \{\varphi_1\psi_1\cdots\varphi_n\psi_n \mid \varphi_1\cdots\varphi_n \in L_1 \wedge \psi_1\cdots\psi_n \in L_2\}$$

$$\mathscr{G} = (\mathrm{p} \xrightarrow{a} \mathrm{q} \wedge \mathrm{p} \xrightarrow{b} \mathrm{q});(\mathrm{q} \xrightarrow{c} \mathrm{p};\mathrm{p} \xrightarrow{b} \mathrm{q})^*;(\mathrm{q} \xrightarrow{d} \mathrm{p} \vee \mathrm{q} \xrightarrow{e} \mathrm{p})$$

$$\mathrm{p} \xrightarrow{a} \mathrm{q};\mathrm{p} \xrightarrow{b} \mathrm{q};\mathrm{q} \xrightarrow{c} \mathrm{p};\mathrm{p} \xrightarrow{b} \mathrm{q};\cdots;\mathrm{q} \xrightarrow{d} \mathrm{p}$$

$$\mathrm{p} \xrightarrow{b} \mathrm{q};\mathrm{p} \xrightarrow{a} \mathrm{q};\mathrm{q} \xrightarrow{c} \mathrm{p};\mathrm{p} \xrightarrow{b} \mathrm{q};\cdots;\mathrm{q} \xrightarrow{e} \mathrm{p}$$

Syntax of session types

T ::= **Pre-Session Type**

Syntax of session types

$$T \quad ::= \qquad\qquad \textbf{Pre-Session Type}$$
$$\quad\quad \text{end} \qquad\qquad \text{(termination)}$$

Syntax of session types

$$
\begin{array}{llll}
T & ::= & & \textbf{Pre-Session Type} \\
 & & \textsf{end} & \text{(termination)} \\
 & | & X & \text{(variable)}
\end{array}
$$

Syntax of session types

T	$::=$		**Pre-Session Type**
		end	(termination)
	$\mid$	X	(variable)
	$\mid$	$\texttt{p}!a.T$	(output)

Syntax of session types

T	$::=$		**Pre-Session Type**
		end	(termination)
	$\mid$	X	(variable)
	$\mid$	$\mathtt{p}!a.T$	(output)
	$\mid$	$\pi?a.T$	(input)

Syntax of session types

T	$::=$		**Pre-Session Type**
		end	(termination)
	$\mid$	X	(variable)
	$\mid$	$\mathtt{p}!a.T$	(output)
	$\mid$	$\pi?a.T$	(input)
	$\mid$	$T \oplus T$	(internal choice)

Syntax of session types

T	$::=$		**Pre-Session Type**
		end	(termination)
	$\mid$	X	(variable)
	$\mid$	$\mathrm{p}!a.T$	(output)
	$\mid$	$\pi?a.T$	(input)
	$\mid$	$T \oplus T$	(internal choice)
	$\mid$	$T + T$	(external choice)

Syntax of session types

T	$::=$		**Pre-Session Type**
		end	(termination)
	$\mid$	X	(variable)
	$\mid$	$\mathtt{p}!a.T$	(output)
	$\mid$	$\pi?a.T$	(input)
	$\mid$	$T \oplus T$	(internal choice)
	$\mid$	$T + T$	(external choice)
	$\mid$	$\mathtt{rec}\ X.T$	(recursion)

Syntax of session types

T	$::=$		**Pre-Session Type**
		end	(termination)
	$\mid$	X	(variable)
	$\mid$	$\mathtt{p}!a.T$	(output)
	$\mid$	$\pi?a.T$	(input)
	$\mid$	$T \oplus T$	(internal choice)
	$\mid$	$T + T$	(external choice)
	$\mid$	$\mathtt{rec}\ X.T$	(recursion)

session type

Syntax of session types

T	$::=$		**Pre-Session Type**
		end	(termination)
	$\mid$	X	(variable)
	$\mid$	$\mathtt{p}!a.T$	(output)
	$\mid$	$\pi?a.T$	(input)
	$\mid$	$T \oplus T$	(internal choice)
	$\mid$	$T + T$	(external choice)
	$\mid$	$\mathtt{rec}\ X.T$	(recursion)

session type

- end

Syntax of session types

T	$::=$		**Pre-Session Type**
		end	(termination)
	$\mid$	X	(variable)
	$\mid$	$\mathtt{p}!a.T$	(output)
	$\mid$	$\pi?a.T$	(input)
	$\mid$	$T \oplus T$	(internal choice)
	$\mid$	$T + T$	(external choice)
	$\mid$	$\mathtt{rec}\ X.T$	(recursion)

session type

- end
- $\bigoplus_{i \in I} \mathtt{p}_i!a_i.T_i$ and $\forall i,j \in I$ we have that $\mathtt{p}_i!a_i = \mathtt{p}_j!a_j$ implies $i = j$ and each T_i is a session type

Syntax of session types

$$
\begin{array}{lllll}
T & ::= & & & \textbf{Pre-Session Type} \\
 & & & \mathsf{end} & \text{(termination)} \\
 & & | & X & \text{(variable)} \\
 & & | & \mathtt{p}!a.T & \text{(output)} \\
 & & | & \pi?a.T & \text{(input)} \\
 & & | & T \oplus T & \text{(internal choice)} \\
 & & | & T + T & \text{(external choice)} \\
 & & | & \mathtt{rec}\ X.T & \text{(recursion)}
\end{array}
$$

session type

- end
- $\bigoplus_{i\in I} \mathtt{p}_i!a_i.T_i$ and $\forall i,j \in I$ we have that $\mathtt{p}_i!a_i = \mathtt{p}_j!a_j$ implies $i = j$ and each T_i is a session type
- $\sum_{i\in I} \pi_i?a_i.T_i$ and $\forall i,j \in I$ we have that $\pi_i \subseteq \pi_j$ and $a_i = a_j$ imply $i = j$ and each T_i is a session type.

Session environments

$$\{\mathtt{p}_i : T_i\}_{i \in I}$$

Session environments

$$\{\mathtt{p}_i : T_i\}_{i \in I}$$

reduction of session environments

Session environments

$$\{\mathtt{p}_i : T_i\}_{i\in I}$$

reduction of session environments

$$\mathbb{B}\,⨟\,\{\mathtt{p} : \bigoplus_{i\in I} \mathtt{p}_i!a_i.T_i\} \uplus \Delta \quad \longrightarrow \quad (\mathtt{p} \xrightarrow{a_k} \mathtt{p}_k)::\mathbb{B}\,⨟\,\{\mathtt{p} : T_k\} \uplus \Delta \qquad (k\in I)$$

Session environments

$$\{\mathtt{p}_i : T_i\}_{i\in I}$$

reduction of session environments

$$\mathbb{B} \fatsemi \{\mathtt{p} : \textstyle\bigoplus_{i\in I} \mathtt{p}_i!a_i.T_i\} \uplus \Delta \quad \longrightarrow \quad (\mathtt{p} \xrightarrow{a_k} \mathtt{p}_k)::\mathbb{B} \fatsemi \{\mathtt{p} : T_k\} \uplus \Delta \qquad (k\in I)$$

$$\mathbb{B}::(\mathtt{p}_i \xrightarrow{a} \mathtt{p})_{i\in I} \fatsemi \{\mathtt{p} : \textstyle\sum_{j\in J} \pi_j?a_j.T_j\} \uplus \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} \mathtt{p}} \quad \mathbb{B} \fatsemi \{\mathtt{p} : T_k\} \uplus \Delta \quad \left(\begin{array}{ll} k\in J & a_k=a \\ \pi_k=\{\mathtt{p}_i \mid i\in I\} & \end{array}\right)$$

Session environments

$$\{\mathtt{p}_i : T_i\}_{i\in I}$$

reduction of session environments

$$\mathbb{B} \fatsemi \{\mathtt{p} : \textstyle\bigoplus_{i\in I} \mathtt{p}_i!a_i.T_i\} \uplus \Delta \quad \longrightarrow \quad (\mathtt{p} \xrightarrow{a_k} \mathtt{p}_k)::\mathbb{B} \fatsemi \{\mathtt{p} : T_k\} \uplus \Delta \qquad (k\in I)$$

$$\mathbb{B}::(\mathtt{p}_i \xrightarrow{a} \mathtt{p})_{i\in I} \fatsemi \{\mathtt{p} : \textstyle\sum_{j\in J} \pi_j?a_j.T_j\} \uplus \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} \mathtt{p}} \quad \mathbb{B} \fatsemi \{\mathtt{p} : T_k\} \uplus \Delta \quad \begin{pmatrix} k\in J & a_k=a \\ \pi_k=\{\mathtt{p}_i | i\in I\} & \end{pmatrix}$$

$$\Delta = \{\mathtt{p} : \mathtt{rec}\ X.(\mathtt{q}!a.X \oplus \mathtt{q}!b.\mathsf{end})\ ,\ \mathtt{q} : \mathtt{rec}\ Y.(\mathtt{p}?a.Y + \mathtt{p}?b.\mathsf{end})\}$$

$$\varepsilon \fatsemi \Delta \quad \longrightarrow \quad \mathtt{p} \xrightarrow{a} \mathtt{q} \fatsemi \Delta$$

Session environments

$$\{\mathtt{p}_i : T_i\}_{i\in I}$$

reduction of session environments

$$\mathbb{B} \fatsemi \{\mathtt{p} : \bigoplus_{i\in I} \mathtt{p}_i!a_i.T_i\} \uplus \Delta \quad \longrightarrow \quad (\mathtt{p} \xrightarrow{a_k} \mathtt{p}_k)::\mathbb{B} \fatsemi \{\mathtt{p} : T_k\} \uplus \Delta \qquad (k\in I)$$

$$\mathbb{B}::(\mathtt{p}_i \xrightarrow{a} \mathtt{p})_{i\in I} \fatsemi \{\mathtt{p} : \sum_{j\in J} \pi_j?a_j.T_j\} \uplus \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} \mathtt{p}} \quad \mathbb{B} \fatsemi \{\mathtt{p} : T_k\} \uplus \Delta \quad \left(\begin{array}{cc} k\in J & a_k=a \\ \pi_k=\{\mathtt{p}_i | i\in I\} & \end{array}\right)$$

$$\Delta = \{\mathtt{p} : \mathtt{rec}\ X.(\mathtt{q}!a.X \oplus \mathtt{q}!b.\mathsf{end})\ ,\ \mathtt{q} : \mathtt{rec}\ Y.(\mathtt{p}?a.Y + \mathtt{p}?b.\mathsf{end})\}$$

$$\varepsilon \fatsemi \Delta \quad \longrightarrow \quad \mathtt{p} \xrightarrow{a} \mathtt{q} \fatsemi \Delta \quad \xrightarrow{\mathtt{p} \xrightarrow{a} \mathtt{q}} \quad \varepsilon \fatsemi \Delta$$

Session environments

$$\{\mathrm{p}_i : T_i\}_{i\in I}$$

reduction of session environments

$$\mathbb{B} \,⨟\, \{\mathrm{p} : \textstyle\bigoplus_{i\in I} \mathrm{p}_i!a_i.T_i\} \uplus \Delta \quad \longrightarrow \quad (\mathrm{p} \xrightarrow{a_k} \mathrm{p}_k)::\mathbb{B} \,⨟\, \{\mathrm{p} : T_k\} \uplus \Delta \qquad (k\in I)$$

$$\mathbb{B}::(\mathrm{p}_i \xrightarrow{a} \mathrm{p})_{i\in I} \,⨟\, \{\mathrm{p} : \textstyle\sum_{j\in J} \pi_j?a_j.T_j\} \uplus \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} \mathrm{p}} \quad \mathbb{B} \,⨟\, \{\mathrm{p} : T_k\} \uplus \Delta \qquad \begin{pmatrix} k\in J & a_k=a \\ \pi_k=\{\mathrm{p}_i \mid i\in I\} & \end{pmatrix}$$

$$\Delta = \{\mathrm{p} : \mathtt{rec}\ X.(\mathrm{q}!a.X \oplus \mathrm{q}!b.\mathsf{end})\ ,\ \mathrm{q} : \mathtt{rec}\ Y.(\mathrm{p}?a.Y + \mathrm{p}?b.\mathsf{end})\}$$
$$\Delta' = \{\mathrm{p} : \mathsf{end}\ ,\ \mathrm{q} : \mathtt{rec}\ Y.(\mathrm{p}?a.Y + \mathrm{p}?b.\mathsf{end})\}$$

$$\varepsilon \,⨟\, \Delta \quad \longrightarrow \quad \mathrm{p} \xrightarrow{a} \mathrm{q} \,⨟\, \Delta \quad \xrightarrow{\mathrm{p} \xrightarrow{a} \mathrm{q}} \quad \varepsilon \,⨟\, \Delta \quad \longrightarrow$$

$$\mathrm{p} \xrightarrow{b} \mathrm{q} \,⨟\, \Delta'$$

Session environments

$$\{\mathtt{p}_i : T_i\}_{i\in I}$$

reduction of session environments

$$\mathbb{B} \mathbin{;} \{\mathtt{p} : \textstyle\bigoplus_{i\in I} \mathtt{p}_i!a_i.T_i\} \uplus \Delta \quad \longrightarrow \quad (\mathtt{p} \xrightarrow{a_k} \mathtt{p}_k)::\mathbb{B} \mathbin{;} \{\mathtt{p} : T_k\} \uplus \Delta \qquad (k\in I)$$

$$\mathbb{B}::(\mathtt{p}_i \xrightarrow{a} \mathtt{p})_{i\in I} \mathbin{;} \{\mathtt{p} : \textstyle\sum_{j\in J} \pi_j?a_j.T_j\} \uplus \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} \mathtt{p}} \quad \mathbb{B} \mathbin{;} \{\mathtt{p} : T_k\} \uplus \Delta \quad \left(\begin{array}{cc} k\in J & a_k=a \\ \pi_k=\{\mathtt{p}_i | i\in I\} & \end{array}\right)$$

$$\Delta = \{\mathtt{p} : \mathtt{rec}\ X.(\mathtt{q}!a.X \oplus \mathtt{q}!b.\mathsf{end})\ ,\ \mathtt{q} : \mathtt{rec}\ Y.(\mathtt{p}?a.Y + \mathtt{p}?b.\mathsf{end})\}$$
$$\Delta' = \{\mathtt{p} : \mathsf{end}\ ,\ \mathtt{q} : \mathtt{rec}\ Y.(\mathtt{p}?a.Y + \mathtt{p}?b.\mathsf{end})\}$$

$$\varepsilon \mathbin{;} \Delta \quad \longrightarrow \quad \mathtt{p} \xrightarrow{a} \mathtt{q} \mathbin{;} \Delta \quad \xrightarrow{\mathtt{p} \xrightarrow{a} \mathtt{q}} \quad \varepsilon \mathbin{;} \Delta \quad \longrightarrow$$
$$\mathtt{p} \xrightarrow{b} \mathtt{q} \mathbin{;} \Delta' \quad \xrightarrow{\mathtt{p} \xrightarrow{b} \mathtt{q}} \quad \varepsilon \mathbin{;} \{\mathtt{p} : \mathsf{end}, \mathtt{q} : \mathsf{end}\}$$

Traces of session environments

Δ is a live session if $\varepsilon \,;\, \Delta \stackrel{\varphi}{\Longrightarrow} \mathbb{B} \,;\, \Delta'$ implies $\mathbb{B} \,;\, \Delta' \stackrel{\psi}{\Longrightarrow} \varepsilon \,;\, \{\mathrm{p}_i : \mathsf{end}\}_{i \in I}$ for some ψ

stronger than progress

Traces of session environments

Δ is a live session if $\varepsilon \,;\, \Delta \stackrel{\varphi}{\Longrightarrow} \mathbb{B} \,;\, \Delta'$ implies $\mathbb{B} \,;\, \Delta' \stackrel{\psi}{\Longrightarrow} \varepsilon \,;\, \{\mathsf{p}_i : \mathsf{end}\}_{i \in I}$ for some ψ

$$\mathrm{tr}(\Delta) \stackrel{\text{def}}{=} \begin{cases} \{\varphi \mid \varepsilon \,;\, \Delta \stackrel{\varphi}{\Longrightarrow} \varepsilon \,;\, \{\mathsf{p}_i : \mathsf{end}\}_{i \in I}\} & \text{if } \Delta \text{ is a live session} \\ \emptyset & \text{otherwise} \end{cases}$$

Traces of session environments

Δ is a live session if $\varepsilon \,;\, \Delta \stackrel{\varphi}{\Longrightarrow} \mathbb{B} \,;\, \Delta'$ implies $\mathbb{B} \,;\, \Delta' \stackrel{\psi}{\Longrightarrow} \varepsilon \,;\, \{\mathtt{p}_i : \mathsf{end}\}_{i \in I}$ for some ψ

$$\mathrm{tr}(\Delta) \stackrel{\text{def}}{=} \begin{cases} \{\varphi \mid \varepsilon \,;\, \Delta \stackrel{\varphi}{\Longrightarrow} \varepsilon \,;\, \{\mathtt{p}_i : \mathsf{end}\}_{i \in I}\} & \text{if } \Delta \text{ is a live session} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathrm{tr}(\{\mathtt{p} : \mathtt{rec}\ X.(\mathtt{q}!a.X \oplus \mathtt{q}!b.\mathsf{end})\ ,\ \mathtt{q} : \mathtt{rec}\ Y.(\mathtt{p}?a.Y + \mathtt{p}?b.\mathsf{end})\}) = \mathrm{tr}((\mathtt{p} \xrightarrow{a} \mathtt{q})^* ; \mathtt{p} \xrightarrow{b} \mathtt{q})$$

Traces of session environments

Δ is a live session if $\varepsilon\,;\,\Delta \stackrel{\varphi}{\Longrightarrow} \mathbb{B}\,;\,\Delta'$ implies $\mathbb{B}\,;\,\Delta' \stackrel{\psi}{\Longrightarrow} \varepsilon\,;\,\{\mathtt{p}_i : \mathsf{end}\}_{i\in I}$ for some ψ

$$\mathrm{tr}(\Delta) \stackrel{\text{def}}{=} \begin{cases} \{\varphi \mid \varepsilon\,;\,\Delta \stackrel{\varphi}{\Longrightarrow} \varepsilon\,;\,\{\mathtt{p}_i : \mathsf{end}\}_{i\in I}\} & \text{if } \Delta \text{ is a live session} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathrm{tr}(\{\mathtt{p} : \mathtt{rec}\ X.(\mathtt{q}!a.X \oplus \mathtt{q}!b.\mathsf{end})\ ,\ \mathtt{q} : \mathtt{rec}\ Y.(\mathtt{p}?a.Y + \mathtt{p}?b.\mathsf{end})\}) = \mathrm{tr}((\mathtt{p} \stackrel{a}{\longrightarrow} \mathtt{q})^*; \mathtt{p} \stackrel{b}{\longrightarrow} \mathtt{q})$$

$$\mathrm{tr}(\{\mathtt{p} : \mathtt{rec}\ X.\mathtt{q}!a.X\ ,\ \mathtt{q} : \mathtt{rec}\ Y.\mathtt{p}?a.Y\}) = \emptyset$$

Outline

Traces of global types and session environments

first try (too strong condition):

$\mathrm{tr}(\mathscr{G}) = \mathrm{tr}(\Delta)$ does not allow to project $\mathscr{G}_1 \wedge \mathscr{G}_2$

Traces of global types and session environments

first try (too strong condition):

$\mathrm{tr}(\mathscr{G}) = \mathrm{tr}(\Delta)$ does not allow to project $\mathscr{G}_1 \wedge \mathscr{G}_2$

second try (too weak condition):

$\mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)$ looses the exhaustivity property

$\{\mathrm{p} : \mathrm{q}!a.\mathsf{end}\ ,\ \mathrm{q} : \mathrm{p}?a.\mathsf{end}\}$ would implement $\mathrm{p} \xrightarrow{a} \mathrm{q} \vee \mathrm{p} \xrightarrow{b} \mathrm{q}$

Traces of global types and session environments

first try (too strong condition):

$\mathrm{tr}(\mathscr{G}) = \mathrm{tr}(\Delta)$ does not allow to project $\mathscr{G}_1 \wedge \mathscr{G}_2$

second try (too weak condition):

$\mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)$ looses the exhaustivity property

$\{\mathsf{p} : \mathsf{q}!a.\mathsf{end}\ ,\ \mathsf{q} : \mathsf{p}?a.\mathsf{end}\}$ would implement $\mathsf{p} \xrightarrow{a} \mathsf{q} \vee \mathsf{p} \xrightarrow{b} \mathsf{q}$

$$\mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)^{\circ}$$
$$\Delta \leqslant \mathscr{G}$$

$L^{\circ} \stackrel{\text{def}}{=} \{\alpha_1 \cdots \alpha_n \mid \text{there exists a permutation } \sigma \text{ such that } \alpha_{\sigma(1)} \cdots \alpha_{\sigma(n)} \in L\}$

Traces of global types and session environments

first try (too strong condition):

$\mathrm{tr}(\mathscr{G}) = \mathrm{tr}(\Delta)$ does not allow to project $\mathscr{G}_1 \wedge \mathscr{G}_2$

second try (too weak condition):

$\mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)$ looses the exhaustivity property

$\{\mathtt{p} : \mathtt{q}!a.\mathsf{end}\ , \ \mathtt{q} : \mathtt{p}?a.\mathsf{end}\}$ would implement $\mathtt{p} \xrightarrow{a} \mathtt{q} \vee \mathtt{p} \xrightarrow{b} \mathtt{q}$

$$\mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)^{\circ}$$
$$\Delta \leqslant \mathscr{G}$$

$L^{\circ} \stackrel{\text{def}}{=} \{\alpha_1 \cdots \alpha_n \mid \text{there exists a permutation } \sigma \text{ such that } \alpha_{\sigma(1)} \cdots \alpha_{\sigma(n)} \in L\}$

$\mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G})$: every trace of Δ is a trace of $\mathscr{G}$ (soundness)

Traces of global types and session environments

first try (too strong condition):

$\mathrm{tr}(\mathscr{G}) = \mathrm{tr}(\Delta)$ does not allow to project $\mathscr{G}_1 \wedge \mathscr{G}_2$

second try (too weak condition):

$\mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)$ looses the exhaustivity property

$\{\mathtt{p} : \mathtt{q}!a.\mathsf{end}\ ,\ \mathtt{q} : \mathtt{p}?a.\mathsf{end}\}$ would implement $\mathtt{p} \xrightarrow{a} \mathtt{q} \vee \mathtt{p} \xrightarrow{b} \mathtt{q}$

$$\mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)^\circ$$
$$\Delta \leqslant \mathscr{G}$$

$L^\circ \stackrel{\text{def}}{=} \{\alpha_1 \cdots \alpha_n \mid \text{there exists a permutation } \sigma \text{ such that } \alpha_{\sigma(1)} \cdots \alpha_{\sigma(n)} \in L\}$

$\mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G})$: every trace of Δ is a trace of $\mathscr{G}$ (soundness)

$\mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)^\circ$: every trace of $\mathscr{G}$ is the permutation of a trace of Δ (completeness)

Projection rules I

$$\Delta \vdash \mathscr{G} \triangleright \Delta'$$

Projection rules I

$$\Delta \vdash \mathscr{G} \ \triangleright\ \Delta'$$

(SP-Skip)

$\Delta \vdash \mathsf{skip} \ \triangleright\ \Delta$

Projection rules I

$$\Delta \vdash \mathscr{G} \triangleright \Delta'$$

(SP-Skip)

$\Delta \vdash \text{skip} \triangleright \Delta$

(SP-Action)

$\{\mathrm{p}_i : T_i\}_{i\in I} \uplus \{\mathrm{p} : T\} \uplus \Delta \vdash \{\mathrm{p}_i\}_{i\in I} \xrightarrow{a} \mathrm{p} \triangleright \{\mathrm{p}_i : \mathrm{p}!a.T_i\}_{i\in I} \uplus \{\mathrm{p} : \{\mathrm{p}_i\}_{i\in I}?a.T\} \uplus \Delta$

Projection rules I

$$\Delta \vdash \mathscr{G} \triangleright \Delta'$$

(SP-Skip)

$\Delta \vdash \mathsf{skip} \triangleright \Delta$

(SP-Action)

$\{\mathrm{p}_i : T_i\}_{i\in I} \uplus \{\mathrm{p} : T\} \uplus \Delta \vdash \{\mathrm{p}_i\}_{i\in I} \xrightarrow{a} \mathrm{p} \triangleright \{\mathrm{p}_i : \mathrm{p}!a.T_i\}_{i\in I} \uplus \{\mathrm{p} : \{\mathrm{p}_i\}_{i\in I}?a.T\} \uplus \Delta$

$\{\mathrm{p} : \mathsf{end}, \mathrm{q} : \mathsf{end}\} \vdash \mathrm{p} \xrightarrow{a} \mathrm{q} \triangleright \{\mathrm{p} : \mathrm{q}!a.\mathsf{end}, \quad \mathrm{q} : \mathrm{p}?a.\mathsf{end}\}$

Projection rules I

$$\Delta \vdash \mathscr{G} \ \triangleright \ \Delta'$$

(SP-Skip)

$$\Delta \vdash \text{skip} \ \triangleright \ \Delta$$

(SP-Action)

$$\{\mathrm{p}_i : T_i\}_{i\in I} \uplus \{\mathrm{p} : T\} \uplus \Delta \vdash \{\mathrm{p}_i\}_{i\in I} \xrightarrow{a} \mathrm{p} \ \triangleright \ \{\mathrm{p}_i : \mathrm{p}!a.T_i\}_{i\in I} \uplus \{\mathrm{p} : \{\mathrm{p}_i\}_{i\in I}?a.T\} \uplus \Delta$$

$$\{\mathrm{p} : \text{end}, \mathrm{q} : \text{end}\} \vdash \mathrm{p} \xrightarrow{a} \mathrm{q} \ \triangleright \ \{\mathrm{p} : \mathrm{q}!a.\text{end}, \quad \mathrm{q} : \mathrm{p}?a.\text{end}\}$$

(SP-Sequence)

$$\frac{\Delta \vdash \mathscr{G}_2 \ \triangleright \ \Delta' \qquad \Delta' \vdash \mathscr{G}_1 \ \triangleright \ \Delta''}{\Delta \vdash \mathscr{G}_1 ; \mathscr{G}_2 \ \triangleright \ \Delta''}$$

Projection rules II

(SP-Alternative)

$$\frac{\Delta \vdash \mathscr{G}_1 \rhd \{\mathtt{p} : T_1\} \uplus \Delta' \qquad \Delta \vdash \mathscr{G}_2 \rhd \{\mathtt{p} : T_2\} \uplus \Delta'}{\Delta \vdash \mathscr{G}_1 \vee \mathscr{G}_2 \rhd \{\mathtt{p} : T_1 \oplus T_2\} \uplus \Delta'}$$

Projection rules II

(SP-Alternative)

$$\frac{\Delta \vdash \mathscr{G}_1 \triangleright \{\mathtt{p} : T_1\} \uplus \Delta' \qquad \Delta \vdash \mathscr{G}_2 \triangleright \{\mathtt{p} : T_2\} \uplus \Delta'}{\Delta \vdash \mathscr{G}_1 \vee \mathscr{G}_2 \triangleright \{\mathtt{p} : T_1 \oplus T_2\} \uplus \Delta'}$$

$$\frac{\Delta_0 \vdash \mathtt{p} \xrightarrow{a} \mathtt{q} \triangleright \{\mathtt{p} : \mathtt{q}!a.\mathsf{end}\ ,\ \mathtt{q} : T\} \quad \Delta_0 \vdash \mathtt{p} \xrightarrow{b} \mathtt{q} \triangleright \{\mathtt{p} : \mathtt{q}!b.\mathsf{end}\ ,\ \mathtt{q} : T\}}{\Delta_0 \vdash \mathtt{p} \xrightarrow{a} \mathtt{q} \vee \mathtt{p} \xrightarrow{b} \mathtt{q} \triangleright \{\mathtt{p} : \mathtt{q}!a.\mathsf{end} \oplus \mathtt{q}!b.\mathsf{end}\ ,\ \ \mathtt{q} : T\}}$$

$$\Delta_0 = \{\mathtt{p} : \mathsf{end}, \mathtt{q} : \mathsf{end}\} \qquad T = \mathtt{p}?a.\mathsf{end} + \mathtt{p}?b.\mathsf{end}$$

Projection rules III

(SP-Iteration)

$$\frac{\{\mathtt{p} : T_1 \oplus T_2\} \uplus \Delta \vdash \mathscr{G} \ \triangleright\ \{\mathtt{p} : T_1\} \uplus \Delta}{\{\mathtt{p} : T_2\} \uplus \Delta \vdash \mathscr{G}^* \ \triangleright\ \{\mathtt{p} : T_1 \oplus T_2\} \uplus \Delta}$$

Projection rules III

(SP-Iteration)

$$\frac{\{\mathtt{p} : T_1 \oplus T_2\} \uplus \Delta \vdash \mathscr{G} \triangleright \{\mathtt{p} : T_1\} \uplus \Delta}{\{\mathtt{p} : T_2\} \uplus \Delta \vdash \mathscr{G}^* \triangleright \{\mathtt{p} : T_1 \oplus T_2\} \uplus \Delta}$$

$$\frac{\{\mathtt{p} : T_1 \oplus T_2, \mathtt{q} : S\} \vdash \mathtt{p} \xrightarrow{a} \mathtt{q} \triangleright \{\mathtt{p} : T_1, \mathtt{q} : S\}}{\{\mathtt{p} : T_2, \mathtt{q} : S\} \vdash (\mathtt{p} \xrightarrow{a} \mathtt{q})^* \triangleright \{\mathtt{p} : T_1 \oplus T_2, \ \ \mathtt{q} : S\}}$$

$T_1 = \mathtt{q}!a.\mathtt{rec}\ X.(\mathtt{q}!a.X \oplus \mathtt{q}!b.\mathsf{end})$
$T_2 = \mathtt{q}!b.\mathsf{end}$
$S = \mathtt{rec}\ Y.(\mathtt{p}?a.Y + \mathtt{p}?b.\mathsf{end})$

Projection rules IV

(SP-Subsumption)

$$\frac{\Delta \vdash \mathscr{G}' \triangleright \Delta' \qquad \mathscr{G}' \leqslant \mathscr{G} \qquad \Delta'' \leqslant \Delta'}{\Delta \vdash \mathscr{G} \triangleright \Delta''}$$

Projection rules IV

(SP-Subsumption)

$$\frac{\Delta \vdash \mathscr{G}' \triangleright \Delta' \qquad \mathscr{G}' \leqslant \mathscr{G} \qquad \Delta'' \leqslant \Delta'}{\Delta \vdash \mathscr{G} \triangleright \Delta''}$$

subsumption on global types

Projection rules IV

(SP-Subsumption)

$$\frac{\Delta \vdash \mathscr{G}' \rhd \Delta' \qquad \color{red}{\mathscr{G}' \leqslant \mathscr{G}} \qquad \Delta'' \leqslant \Delta'}{\Delta \vdash \mathscr{G} \rhd \Delta''}$$

$$\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s} \leqslant \mathtt{p} \xrightarrow{a} \mathtt{q} \wedge \mathtt{r} \xrightarrow{b} \mathtt{s}$$

Projection rules IV

(SP-Subsumption)

$$\frac{\Delta \vdash \mathscr{G}' \triangleright \Delta' \qquad \mathscr{G}' \leqslant \mathscr{G} \qquad \Delta'' \leqslant \Delta'}{\Delta \vdash \mathscr{G} \triangleright \Delta''}$$

$$\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s} \leqslant \mathtt{p} \xrightarrow{a} \mathtt{q} \wedge \mathtt{r} \xrightarrow{b} \mathtt{s}$$

$$\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s} \leqslant (\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s}) \vee (\mathtt{r} \xrightarrow{b} \mathtt{s}; \mathtt{p} \xrightarrow{a} \mathtt{q})$$

Projection rules IV

(SP-Subsumption)

$$\frac{\Delta \vdash \mathscr{G}' \triangleright \Delta' \qquad \mathscr{G}' \leqslant \mathscr{G} \qquad \Delta'' \leqslant \Delta'}{\Delta \vdash \mathscr{G} \triangleright \Delta''}$$

$$\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s} \leqslant \mathtt{p} \xrightarrow{a} \mathtt{q} \wedge \mathtt{r} \xrightarrow{b} \mathtt{s}$$

$$\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s} \leqslant (\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s}) \vee (\mathtt{r} \xrightarrow{b} \mathtt{s}; \mathtt{p} \xrightarrow{a} \mathtt{q})$$

$$\mathtt{r} \xrightarrow{b} \mathtt{p}; (\mathtt{p} \xrightarrow{a} \mathtt{q} \vee \mathtt{p} \xrightarrow{b} \mathtt{q}) \leqslant (\mathtt{r} \xrightarrow{b} \mathtt{p}; \mathtt{p} \xrightarrow{a} \mathtt{q}) \vee (\mathtt{r} \xrightarrow{b} \mathtt{p}; \mathtt{p} \xrightarrow{b} \mathtt{q})$$

Projection rules IV

(SP-Subsumption)

$$\frac{\Delta \vdash \mathscr{G}' \rhd \Delta' \qquad \mathscr{G}' \leqslant \mathscr{G} \qquad \Delta'' \leqslant \Delta'}{\Delta \vdash \mathscr{G} \rhd \Delta''}$$

$$\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s} \leqslant \mathtt{p} \xrightarrow{a} \mathtt{q} \wedge \mathtt{r} \xrightarrow{b} \mathtt{s}$$

$$\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s} \leqslant (\mathtt{p} \xrightarrow{a} \mathtt{q}; \mathtt{r} \xrightarrow{b} \mathtt{s}) \vee (\mathtt{r} \xrightarrow{b} \mathtt{s}; \mathtt{p} \xrightarrow{a} \mathtt{q})$$

$$\mathtt{r} \xrightarrow{b} \mathtt{p}; (\mathtt{p} \xrightarrow{a} \mathtt{q} \vee \mathtt{p} \xrightarrow{b} \mathtt{q}) \leqslant (\mathtt{r} \xrightarrow{b} \mathtt{p}; \mathtt{p} \xrightarrow{a} \mathtt{q}) \vee (\mathtt{r} \xrightarrow{b} \mathtt{p}; \mathtt{p} \xrightarrow{b} \mathtt{q})$$

Projection rules IV

(SP-Subsumption)

$$\frac{\Delta \vdash \mathscr{G}' \ \triangleright \ \Delta' \qquad \mathscr{G}' \leqslant \mathscr{G} \qquad \Delta'' \leqslant \Delta'}{\Delta \vdash \mathscr{G} \ \triangleright \ \Delta''}$$

subsumption on session environments

Main results

$\mathscr{G}$ is well formed if $\varphi;\pi \xrightarrow{a} \mathrm{p};\pi' \xrightarrow{b} \mathrm{p}';\psi \in \mathrm{tr}(\mathscr{G})$ implies either $\mathrm{p} \in \pi' \cup \{\mathrm{p}'\}$ or $\varphi;\pi' \xrightarrow{b} \mathrm{p}';\pi \xrightarrow{a} \mathrm{p};\psi \in \mathrm{tr}(\mathscr{G})$

Main results

$\mathscr{G}$ is well formed if $\varphi;\pi \xrightarrow{a} \mathrm{p};\pi' \xrightarrow{b} \mathrm{p}';\psi \in \mathrm{tr}(\mathscr{G})$ implies either $\mathrm{p} \in \pi' \cup \{\mathrm{p}'\}$ or $\varphi;\pi' \xrightarrow{b} \mathrm{p}';\pi \xrightarrow{a} \mathrm{p};\psi \in \mathrm{tr}(\mathscr{G})$

If $\mathscr{G}$ is well formed and $\vdash \mathscr{G} \triangleright \Delta$, then $\Delta \leqslant \mathscr{G}$

Main results

$\mathscr{G}$ is well formed if $\varphi;\pi \xrightarrow{a} \mathrm{p};\pi' \xrightarrow{b} \mathrm{p}';\psi \in \mathrm{tr}(\mathscr{G})$ implies either $\mathrm{p} \in \pi' \cup \{\mathrm{p}'\}$ or $\varphi;\pi' \xrightarrow{b} \mathrm{p}';\pi \xrightarrow{a} \mathrm{p};\psi \in \mathrm{tr}(\mathscr{G})$

If $\mathscr{G}$ is well formed and $\vdash \mathscr{G} \triangleright \Delta$, then $\Delta \leqslant \mathscr{G}$

- No sequentiality: $\nexists\Delta : \Delta \leqslant \mathscr{G}$ and $\exists\Delta : \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G})^{\#}$
  $L^{\#}$ is the smallest well-formed set such that $L \subseteq L^{\#}$

Main results

$\mathscr{G}$ is well formed if $\varphi; \pi \xrightarrow{a} \mathrm{p}; \pi' \xrightarrow{b} \mathrm{p}'; \psi \in \mathrm{tr}(\mathscr{G})$ implies either $\mathrm{p} \in \pi' \cup \{\mathrm{p}'\}$ or $\varphi; \pi' \xrightarrow{b} \mathrm{p}'; \pi \xrightarrow{a} \mathrm{p}; \psi \in \mathrm{tr}(\mathscr{G})$

If $\mathscr{G}$ is well formed and $\vdash \mathscr{G} \triangleright \Delta$, then $\Delta \leqslant \mathscr{G}$

- No sequentiality: $\nexists \Delta : \Delta \leqslant \mathscr{G}$ and $\exists \Delta : \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G})^{\#}$
  $L^{\#}$ is the smallest well-formed set such that $L \subseteq L^{\#}$
- No knowledge for choice:
  $\nexists \Delta : \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G})^{\#}$ and $\exists \Delta : \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)$

Main results

$\mathscr{G}$ is well formed if $\varphi;\pi \xrightarrow{a} \mathrm{p};\pi' \xrightarrow{b} \mathrm{p}';\psi \in \mathrm{tr}(\mathscr{G})$ implies either $\mathrm{p} \in \pi' \cup \{\mathrm{p}'\}$ or $\varphi;\pi' \xrightarrow{b} \mathrm{p}';\pi \xrightarrow{a} \mathrm{p};\psi \in \mathrm{tr}(\mathscr{G})$

If $\mathscr{G}$ is well formed and $\vdash \mathscr{G} \triangleright \Delta$, then $\Delta \leqslant \mathscr{G}$

- No sequentiality: $\nexists\Delta : \Delta \leqslant \mathscr{G}$ and $\exists\Delta : \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G})^{\#}$
  $L^{\#}$ is the smallest well-formed set such that $L \subseteq L^{\#}$
- No knowledge for choice:
  $\nexists\Delta : \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta) \subseteq \mathrm{tr}(\mathscr{G})^{\#}$ and $\exists\Delta : \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)$
- No knowledge, no choice: $\nexists\Delta : \mathrm{tr}(\mathscr{G}) \subseteq \mathrm{tr}(\Delta)$

Projection rules

no subsumption on session environments

Projection rules

no subsumption on session environments

(AP-Alternative)

$$\frac{\Delta \vdash_{\mathsf{a}} \mathscr{G}_1 \triangleright \{\mathtt{p} : T_1\} \uplus \Delta_1 \qquad \Delta \vdash_{\mathsf{a}} \mathscr{G}_2 \triangleright \{\mathtt{p} : T_2\} \uplus \Delta_2}{\Delta \vdash_{\mathsf{a}} \mathscr{G}_1 \vee \mathscr{G}_2 \triangleright \{\mathtt{p} : T_1 \oplus T_2\} \uplus (\Delta_1 \curlywedge \Delta_2)}$$

Projection rules

no subsumption on session environments

(AP-Alternative)

$$\frac{\Delta \vdash_{\mathsf{a}} \mathscr{G}_1 \triangleright \{\mathtt{p} : T_1\} \uplus \Delta_1 \qquad \Delta \vdash_{\mathsf{a}} \mathscr{G}_2 \triangleright \{\mathtt{p} : T_2\} \uplus \Delta_2}{\Delta \vdash_{\mathsf{a}} \mathscr{G}_1 \vee \mathscr{G}_2 \triangleright \{\mathtt{p} : T_1 \oplus T_2\} \uplus (\Delta_1 \curlywedge \Delta_2)}$$

(AP-Iteration)

$$\frac{\{\mathtt{p} : X\} \uplus \{\mathtt{p}_i : X_i\}_{i \in I} \vdash_{\mathsf{a}} \mathscr{G} \triangleright \{\mathtt{p} : S\} \uplus \{\mathtt{p}_i : S_i\}_{i \in I}}{\{\mathtt{p} : T\} \uplus \{\mathtt{p}_i : T_i\}_{i \in I} \uplus \Delta \vdash_{\mathsf{a}} \mathscr{G}^* \triangleright \{\mathtt{p} : \mathtt{rec}\ X.(T \oplus S)\} \uplus \{\mathtt{p}_i : \mathtt{rec}\ X_i.(T_i \curlywedge S_i)\}_{i \in I} \uplus \Delta}$$

Projection rules

no subsumption on session environments

(AP-Alternative)

$$\frac{\Delta \vdash_{\mathsf{a}} \mathscr{G}_1 \rhd \{\mathtt{p} : T_1\} \uplus \Delta_1 \qquad \Delta \vdash_{\mathsf{a}} \mathscr{G}_2 \rhd \{\mathtt{p} : T_2\} \uplus \Delta_2}{\Delta \vdash_{\mathsf{a}} \mathscr{G}_1 \vee \mathscr{G}_2 \rhd \{\mathtt{p} : T_1 \oplus T_2\} \uplus (\Delta_1 \curlywedge \Delta_2)}$$

(AP-Iteration)

$$\frac{\{\mathtt{p} : X\} \uplus \{\mathtt{p}_i : X_i\}_{i \in I} \vdash_{\mathsf{a}} \mathscr{G} \rhd \{\mathtt{p} : S\} \uplus \{\mathtt{p}_i : S_i\}_{i \in I}}{\{\mathtt{p} : T\} \uplus \{\mathtt{p}_i : T_i\}_{i \in I} \uplus \Delta \vdash_{\mathsf{a}} \mathscr{G}^* \rhd \{\mathtt{p} : \mathtt{rec}\ X.(T \oplus S)\} \uplus \{\mathtt{p}_i : \mathtt{rec}\ X_i.(T_i \curlywedge S_i)\}_{i \in I} \uplus \Delta}$$

no subsumption on global types: $\wedge$-types must be eliminated

Projection rules

no subsumption on session environments

(AP-Alternative)

$$\frac{\Delta \vdash_{\mathsf{a}} \mathscr{G}_1 \rhd \{\mathtt{p} : T_1\} \uplus \Delta_1 \qquad \Delta \vdash_{\mathsf{a}} \mathscr{G}_2 \rhd \{\mathtt{p} : T_2\} \uplus \Delta_2}{\Delta \vdash_{\mathsf{a}} \mathscr{G}_1 \vee \mathscr{G}_2 \rhd \{\mathtt{p} : T_1 \oplus T_2\} \uplus (\Delta_1 \bigwedge\!\!\!\bigwedge \Delta_2)}$$

(AP-Iteration)

$$\frac{\{\mathtt{p} : X\} \uplus \{\mathtt{p}_i : X_i\}_{i \in I} \vdash_{\mathsf{a}} \mathscr{G} \rhd \{\mathtt{p} : S\} \uplus \{\mathtt{p}_i : S_i\}_{i \in I}}{\{\mathtt{p} : T\} \uplus \{\mathtt{p}_i : T_i\}_{i \in I} \uplus \Delta \vdash_{\mathsf{a}} \mathscr{G}^* \rhd \{\mathtt{p} : \mathtt{rec}\ X.(T \oplus S)\} \uplus \{\mathtt{p}_i : \mathtt{rec}\ X_i.(T_i \bigwedge\!\!\!\bigwedge S_i)\}_{i \in I} \uplus \Delta}$$

no subsumption on global types: $\wedge$-types must be eliminated

$\mathscr{G} \leqslant \mathscr{G}'$ is decidable by the decidability of the Parikh equivalence on regular languages

k-Exit Iterations

$$(\mathtt{p} \xrightarrow{\mathit{handover}} \mathtt{q}; \mathtt{q} \xrightarrow{\mathit{handover}} \mathtt{p})^*; (\mathtt{p} \xrightarrow{\mathit{bailout}} \mathtt{q} \vee \mathtt{p} \xrightarrow{\mathit{handover}} \mathtt{q}; \mathtt{q} \xrightarrow{\mathit{bailout}} \mathtt{p})$$

k-Exit Iterations

$$(\mathtt{p} \xrightarrow{\mathit{handover}} \mathtt{q}; \mathtt{q} \xrightarrow{\mathit{handover}} \mathtt{p})^*; (\mathtt{p} \xrightarrow{\mathit{bailout}} \mathtt{q} \vee \mathtt{p} \xrightarrow{\mathit{handover}} \mathtt{q}; \mathtt{q} \xrightarrow{\mathit{bailout}} \mathtt{p})$$

$$(\mathscr{G}_1, \ldots, \mathscr{G}_k)^{k*}(\mathscr{G}'_1, \ldots, \mathscr{G}'_k)$$

k-Exit Iterations

$$(\mathtt{p} \xrightarrow{\textit{handover}} \mathtt{q}; \mathtt{q} \xrightarrow{\textit{handover}} \mathtt{p})^*; (\mathtt{p} \xrightarrow{\textit{bailout}} \mathtt{q} \vee \mathtt{p} \xrightarrow{\textit{handover}} \mathtt{q}; \mathtt{q} \xrightarrow{\textit{bailout}} \mathtt{p})$$

$$(\mathscr{G}_1, \ldots, \mathscr{G}_k)^{k*}(\mathscr{G}'_1, \ldots, \mathscr{G}'_k)$$

$$(\mathtt{p} \xrightarrow{\textit{handover}} \mathtt{q}, \mathtt{q} \xrightarrow{\textit{handover}} \mathtt{p})^{2*}(\mathtt{p} \xrightarrow{\textit{bailout}} \mathtt{q}, \mathtt{q} \xrightarrow{\textit{bailout}} \mathtt{p})$$

k-Exit Iterations

$$(\mathtt{p} \xrightarrow{\mathit{handover}} \mathtt{q}; \mathtt{q} \xrightarrow{\mathit{handover}} \mathtt{p})^*; (\mathtt{p} \xrightarrow{\mathit{bailout}} \mathtt{q} \vee \mathtt{p} \xrightarrow{\mathit{handover}} \mathtt{q}; \mathtt{q} \xrightarrow{\mathit{bailout}} \mathtt{p})$$

$$(\mathscr{G}_1, \ldots, \mathscr{G}_k)^{k*} (\mathscr{G}'_1, \ldots, \mathscr{G}'_k)$$

$$(\mathtt{p} \xrightarrow{\mathit{handover}} \mathtt{q}, \mathtt{q} \xrightarrow{\mathit{handover}} \mathtt{p})^{2*} (\mathtt{p} \xrightarrow{\mathit{bailout}} \mathtt{q}, \mathtt{q} \xrightarrow{\mathit{bailout}} \mathtt{p})$$

$$\mathtt{p} : \mathtt{rec}\ X.(\mathtt{q}!\mathit{handover}.(\mathtt{q}?\mathit{handover}.X + \mathtt{q}?\mathit{bailout}.\mathsf{end}) \oplus \mathtt{q}!\mathit{bailout}.\mathsf{end})$$
$$\mathtt{q} : \mathtt{rec}\ Y.(\mathtt{p}?\mathit{handover}.(\mathtt{p}!\mathit{handover}.Y \oplus \mathtt{p}!\mathit{bailout}.\mathsf{end}) + \mathtt{p}?\mathit{bailout}.\mathsf{end})$$

Outline

Sessions and Choreographies

Honda Yoshida Carbone Bravetti Lanese Zavattaro ...

Multiparty Asynchronous Session Types

Kohei Honda
Queen Mary, University of London
kohei@dcs.qmul.ac.uk

Nobuko Yoshida
Imperial College London
yoshida@doc.ic.ac.uk

Marco Carbone
Queen Mary, University of London
carbonem@dcs.qmul.ac.uk

Abstract

Communication is becoming one of the central elements in software development. As a potential typed foundation for structured communication-centred programming, session types have been studied over the last decade for a wide range of process calculi and programming languages, focussing on binary (two-party) sessions. This work extends the foregoing theories of binary session types to multiparty, asynchronous sessions, which often arise in practical communication-centred applications. Presented as a typed calculus for mobile processes, the theory introduces a new notion of types in which interactions involving multiple peers are directly abstracted as a global scenario. Global types retain a friendly type syntax of binary session types while capturing complex causal chains of multiparty asynchronous interactions. A global type plays the role of a shared agreement among communication peers, and is used as a basis of efficient type checking through its projection onto individual peers. The fundamental properties of the session type discipline such as communication safety, progress and session fidelity are established for general *n*-party asynchronous interactions.

Categories and Subject Descriptors D.3.1 [*Programming Languages*]: Formal Definitions and Theory; F.3.2 [*Semantics of Programming Languages*]: Process models

General Terms Theory, Types, Design

Keywords communications, multiparty, structured programming, session types, mobile processes, causality, choreography

1. Introduction

Backgrounds Communication is becoming one of the central elements in software development, ranging from web services to business protocols to parallel scientific computing to multicore programming. As a potential typed foundation for structured communication-centred programming, session types have been studied in many contexts over the last decade, including calculi of mobile processes (Takeuchi et al. 1994; Gay and Hole 2005; Honda et al. 1998; Bonelli and Compagnoni 2008), higher-order processes (Mostrous and Yoshida 2007), Ambients (Garralda et al. 2006), multi-threaded ML (Vasconcelos et al. 2006), Haskell (Neubauer and Thiemann 2004b), F# (Corin et al. 2007), operating systems (Fähndrich et al. 2006), Java (Dezani-Ciancaglini et al. 2006; Coppo et al. 2007; Hu et al. 2007), and Web Services (Carbone et al. 2006, 2007; WS-CDL; Sparkes 2006; Honda et al. 2007a). A basic observation underlying session types is that a communication-centred application often exhibits a highly structured sequence of interactions involving, for example, branching and recursion, which as a whole form a natural unit of conversation, or *session*. The structure of a conversation is abstracted as a type through an intuitive syntax, which is then used as a basis of validating programs through an associated type discipline.

As an example, the following session type describes a simple business protocol between Buyer and Seller from Buyer's viewpoint: Buyer sends the title of a book (a string), Seller sends a quote (an integer). If Buyer is satisfied by the quote, then sends his address (a string) and Seller sends back the delivery date (a date); otherwise it quits the conversation.

$$!\mathsf{string}; ?\mathsf{int}; \oplus\{\mathsf{ok} : !\mathsf{string}; ?\mathsf{date}; \mathsf{end}, \quad \mathsf{quit} : \mathsf{end}\} \qquad (1)$$

Above !t denotes an output of a value of type t, dually for ?t; $\oplus$ denotes a choice of the options; and end represents the termination of the conversation.

Such explicit representation of conversation structures helps us deal with one of the most common bugs in programming with communication, the synchronisation bugs. A programmer expects that communicating programs should together realise a consistent conversation, but they easily fail to handle a specific incoming message or to send a message at the correct timing, with no way to detect such errors before runtime. An explicit specification as in (1) guides principled programming of communication behaviour and enables automatic protocol validation (WS-CDL; UNIFI; Hu et al. 2007). In addition, a clean separation between abstraction and implementation given by type-based abstraction and associated primitives leads to intelligible programs and flexible implementations (Hu et al. 2007). Underlying these merits are the following central properties guaranteed by session types.

1. Interactions within a session never incur a communication error (communication safety).
2. Channels for a session are used linearly (linearity) and are deadlock-free in a single session (progress).
3. The communication sequence in a session follows the scenario declared in the session type (session fidelity, predictability).

Multiparty Asynchronous Sessions The foregoing studies on session types have focussed on binary (two-party) sessions. While many conversation patterns can be captured through a composition of binary sessions, there are cases where binary session types are not powerful enough for describing and validating interactions which involve more than two parties.

As an example, let us consider a simple refinement of the above Buyer-Seller protocol: consider two buyers, Buyer1 and Buyer2, wish to buy an expensive book from Seller by combining their money. Buyer1 sends the title of the book to Seller, Seller sends to both Buyer1 and Buyer2 its quote, Buyer1 tells Buyer2 how

POPL'08, January 7–12, 2008, San Francisco, California, USA.

Contract-Driven Implementation of Choreographies*

Mario Bravetti, Ivan Lanese, and Gianluigi Zavattaro

Department of Computer Science, University of Bologna, Italy
{bravetti,lanese,zavattar}@cs.unibo.it

Abstract. Choreographies and Contracts are important concepts in Service Oriented Computing. Choreographies are the description of the behaviour of a service system from a global point of view, while contracts are the description of the externally observable message-passing behaviour of a given service. Exploiting some of our previous results about choreography projection and contract refinement, we show how to solve the problem of implementing a choreography via the composition of already available services that are retrieved according to their contracts.

1 Introduction

SENSORIA (Software Engineering for Service-Oriented Overlay Computers) [6] is a European project funded under the 6th Framework Programme as part of the Global Computing Initiative. The aim of SENSORIA is to develop a novel comprehensive approach to the engineering of software systems for service oriented computing where foundational theories, techniques and methods are fully integrated in a pragmatic software engineering approach.

Service Oriented Computing (SOC) is a paradigm for distributed computing based on services intended as autonomous and heterogeneous components that can be published and discovered via standard interface languages and publish/discovery protocols. Web Services is the most prominent service oriented technology: Web Services publish their interface expressed in WSDL, they are discovered through the UDDI protocol, and they are invoked using SOAP.

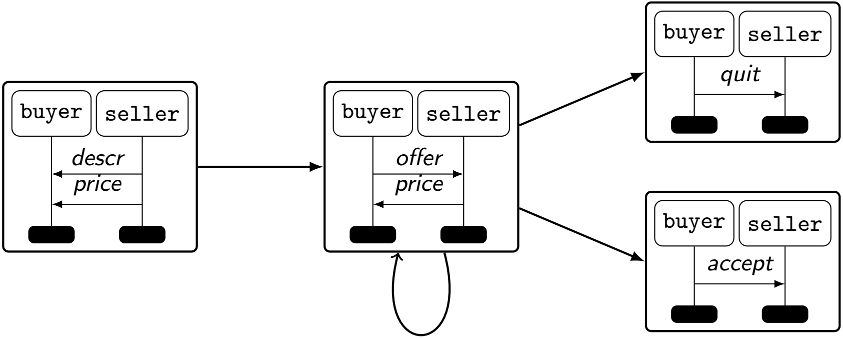
This paper addresses the problem of implementing service oriented systems, specified by means of high level languages called *choreography* languages in the SOC literature, by assembling already available services that can be automatically retrieved. The approach proposed in this paper in order to solve this problem is based on the assumption that services expose their behavioural interface expressed in terms of a *contract*, i.e. "the externally observable message-passing behaviour" [7].

More precisely, choreography languages are intended as notations for representing multi-party service compositions, that is, descriptions of the global behavior of service-based applications in which several services reciprocally communicate

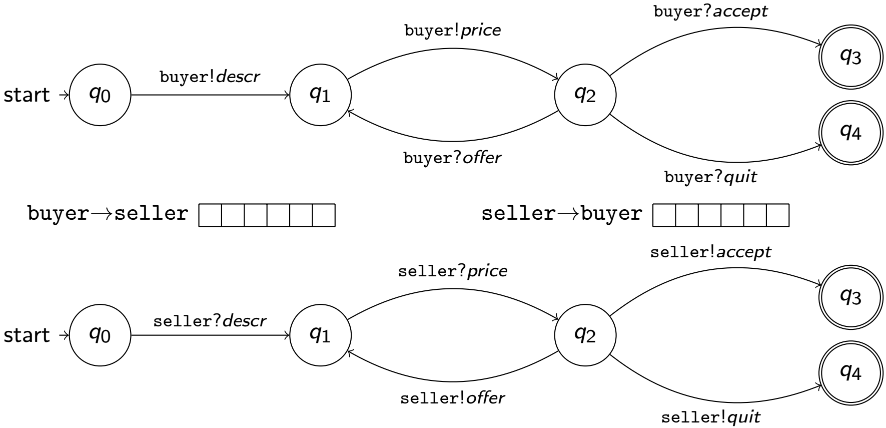
* Research partially funded by EU Integrated Project Sensoria, contract n. 016004.

C. Kaklamanis and F. Nielson (Eds.): TGC 2009, LNCS 5474, pp. 1–18, 2009.

MSG of the seller-buyer protocol



CFMs implementing the seller-buyer protocol



Kao Chow protocol in WPPL

```
(spec ([a (a b s kas) (kab)]
       [b (b s kbs) (kab)] [s (a b s kas kbs) ()])
[a -> s : a, b, na:nonce]
[s -> b : |a, b, na, kab| kas, |a, b, na, kab| kbs]
[b -> a : |a, b, na, kab| kas, |na| kab, nb:nonce]
[a -> b : |nb| kab] .)
```