<table>
<thead>
<tr>
<th>Global types and session types</th>
<th>Projections</th>
<th>Related approaches</th>
</tr>
</thead>
</table>

On Global Types and Multi-Party Sessions

Joint work with Giuseppe Castagna and Luca Padovani

Workshop on Behavioural Type Systems, Lisbon, 19 April 2011
Outline

Global types and session types

Overview
Global types
Session types
Outline

Global types and session types
- Overview
- Global types
- Session types

Projections
- Semantic projection
- Algorithmic projection
- Kleene star and recursion
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
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
\[ \mathcal{G} = \text{alice} \xrightarrow{\text{nat}} \text{bob}; \]
\[ \text{bob} \xrightarrow{\text{nat}} \text{carol} \]

Session Types
\[ T_{\text{bob}} = \text{alice}!\text{nat}. \]
\[ \text{carol}!!\text{nat}. \]
\[ \text{end} \]

Processes
\[ P_{\text{bob}} = \text{receive } x \text{ from alice; send } x+42 \text{ to carol; end} \]
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.

\[(\text{seller} \xrightarrow{\text{descr}} \text{buyer}) \land (\text{seller} \xrightarrow{\text{price}} \text{buyer})\);  
\[(\text{buyer} \xrightarrow{\text{offer}} \text{seller}; \text{seller} \xrightarrow{\text{price}} \text{buyer})^*\];  
\[(\text{buyer} \xrightarrow{\text{accept}} \text{seller}) \lor (\text{buyer} \xrightarrow{\text{quit}} \text{seller})\]
Informal descriptions, global types and session types

\[(\text{seller} \xrightarrow{\text{descr}} \text{buyer} \land \text{seller} \xrightarrow{\text{price}} \text{buyer});\
(\text{buyer} \xrightarrow{\text{offer}} \text{seller}; \text{seller} \xrightarrow{\text{price}} \text{buyer})^*;\
(\text{buyer} \xrightarrow{\text{accept}} \text{seller} \lor \text{buyer} \xrightarrow{\text{quit}} \text{seller})\]

\[
\text{seller} \xrightarrow{} \text{buyer!descr. buyer!price.rec X. (buyer?offer. buyer!price.X + buyer?accept + buyer?quit)} \\
\text{buyer} \xrightarrow{} \text{seller?descr. seller?price.rec Y. (seller!offer. seller?price.Y} \oplus \text{ seller!accept} \oplus \text{ seller!quit)}
\]
Informal descriptions, global types and session types

\[(\text{seller} \xrightarrow{\text{\textit{descr}}} \text{buyer} \land \text{seller} \xrightarrow{\text{\textit{price}}} \text{buyer});
\]
\[(\text{buyer} \xrightarrow{\text{\textit{offer}}} \text{seller}; \text{seller} \xrightarrow{\text{\textit{price}}} \text{buyer})^*;
\]
\[(\text{buyer} \xrightarrow{\text{\textit{accept}}} \text{seller} \lor \text{buyer} \xrightarrow{\text{\textit{quit}}} \text{seller})
\]

\[
\begin{align*}
\text{seller} & \mapsto \text{buyer!price.bbuyer!descr.rec } X. \\
& \quad (\text{buyer?offer.bbuyer!price.X} + \text{buyer?accept} + \text{buyer?quit})
\end{align*}
\]
\[
\begin{align*}
\text{buyer} & \mapsto \text{seller?price.seller?descr.rec } Y. \\
& \quad (\text{seller!offer.seller?price.Y} \oplus \text{seller!accept} \oplus \text{seller!quit})
\end{align*}
\]
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 \texttt{buyer} may send \texttt{accept} before receiving \texttt{price} violates the specification.

2. **Alternativeness**: an implementation in which \texttt{buyer} emits both \texttt{accept} and \texttt{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
Global types and session types

Overview

Flawed global types

- **no covert channel**

- **No sequentiality**: some sequentiality constraint between independent interactions

\[
(p \xrightarrow{a} q; r \xrightarrow{b} s)
\]
Flawed global types

no covert channel

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

- No knowledge for choice: some participant must behave in different ways in accordance with some choice it is unaware of
  \[(p \xrightarrow{a} q; q \xrightarrow{a} r; r \xrightarrow{a} p) \lor (p \xrightarrow{b} q; q \xrightarrow{a} r; r \xrightarrow{b} p)\]
Flawed global types

no covert channel

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

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

- No knowledge, no choice: incompatible behaviours such as performing and input or an output in mutual exclusion
  \[p \xrightarrow{a} q \lor q \xrightarrow{b} p\]
Syntax of global types

\[ G ::= \text{Global Type} \]
Syntax of global types

\[ G ::= \text{Global Type} \]

- \text{skip}
- (skip)

\[ \pi a \rightarrow \{ p_i \} \text{ for } i \in I \text{ can be encoded as } \bigcap_{i \in I} (\pi a \rightarrow p_i) \]
Syntax of global types

\[ G ::= \begin{align*}
\text{skip} & \quad \text{(skip)} \\
\pi \xrightarrow{a} p & \quad \text{(interaction)} \quad \text{multiple senders}
\end{align*} \]
Global types and session types

### Syntax of global types

\[\begin{align*}
G \quad ::= \\
\text{skip} & \quad \text{(skip)} \\
\pi \xrightarrow{a} p & \quad \text{(interaction)} \\
G ; G & \quad \text{(sequence)}
\end{align*}\]
Syntax of global types

\[ G ::= \]

\[ \text{skip} \quad \text{(skip)} \]
\[ \pi \xrightarrow{a} p \quad \text{(interaction)} \]
\[ G;G \quad \text{(sequence)} \]
\[ G \land G \quad \text{(both)} \]
Global types

Syntax of global types

\[ G ::= \]

- `skip` (skip)
- `π a → p` (interaction)
- `G ; G` (sequence)
- `G ∧ G` (both)
- `G ∨ G` (either)
Global types and session types

Syntax of global types

\[ G ::= \]

- \( \text{skip} \) (skip)
- \( \pi \xrightarrow{a} p \) (interaction)
- \( G;G \) (sequence)
- \( G \land G \) (both)
- \( G \lor G \) (either)
- \( G^* \) (star) fairness
Syntax of global types

\[ G ::= \]

- `skip` (skip)
- `π \xrightarrow{a} p` (interaction)
- `G; G` (sequence)
- `G \land G` (both)
- `G \lor G` (either)
- `G^*` (star)

\[ \pi \xrightarrow{a} \{p_i\}_{i \in I} \] can be encoded as \( \land_{i \in I}(\pi \xrightarrow{a} p_i) \)
# Examples

join

\[(\text{seller} \xrightarrow{\text{price}} \text{buyer1} \land \text{bank} \xrightarrow{\text{mortgage}} \text{buyer2}) ; (\{\text{buyer1}, \text{buyer2}\} \xrightarrow{\text{accept}} \text{seller} \land \{\text{buyer1}, \text{buyer2}\} \xrightarrow{\text{accept}} \text{bank})\]
Examples

join

\[(\text{seller} \xrightarrow{\text{price}} \text{buyer1} \land \text{bank} \xrightarrow{\text{mortgage}} \text{buyer2});
(\{\text{buyer1}, \text{buyer2}\} \xrightarrow{\text{accept}} \text{seller} \land \{\text{buyer1}, \text{buyer2}\} \xrightarrow{\text{accept}} \text{bank})\]

fork

\[\text{seller} \xrightarrow{\text{price}} \text{buyer1} \land \text{seller} \xrightarrow{\text{price}} \text{buyer2}\]
Examples

join

\((\text{seller} \xrightarrow{\text{price}} \text{buyer1} \land \text{bank} \xrightarrow{\text{mortgage}} \text{buyer2});\)
\((\{\text{buyer1, buyer2}\} \xrightarrow{\text{accept}} \text{seller} \land \{\text{buyer1, buyer2}\} \xrightarrow{\text{accept}} \text{bank})\)

fork

\(\text{seller} \xrightarrow{\text{price}} \text{buyer1} \land \text{seller} \xrightarrow{\text{price}} \text{buyer2}\)

common participants in parallel actions
Examples

different receivers in a choice

seller $\xrightarrow{\text{price}}$ buyer1; buyer1 $\xrightarrow{\text{price}}$ buyer2 $\lor$

seller $\xrightarrow{\text{price}}$ buyer2; buyer2 $\xrightarrow{\text{price}}$ buyer1
Examples

different receivers in a choice

seller $\rightarrow$ buyer1; buyer1 $\rightarrow$ buyer2 $\lor$
seller $\rightarrow$ buyer2; buyer2 $\rightarrow$ buyer1

different sets of participants for alternatives

(seller $\rightarrow$ broker; broker $\rightarrow$ buyer $\lor$ seller $\rightarrow$ buyer);

buyer $\rightarrow$ broker
Global types and session types

Examples

different receivers in a choice

seller \rightarrow price buyer_1; buyer_1 \rightarrow price buyer_2 \vee
seller \rightarrow price buyer_2; buyer_2 \rightarrow price buyer_1

different sets of participants for alternatives

(seller \rightarrow agency broker; broker \rightarrow price buyer \vee seller \rightarrow price buyer);

buyer \rightarrow answer broker

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

seller \rightarrow agency broker;(broker \rightarrow offer buyer; buyer \rightarrow counteroffer broker)*;

(broker \rightarrow result seller \land broker \rightarrow result buyer)

On Global Types and Multi-Party Sessions
Global types and session types

Projections

Related approaches

Global types

Traces of global types

\[ \text{tr}(\text{skip}) = \{ \varepsilon \} \]
Traces of global types

\[
\begin{align*}
tr(skip) &= \{ \epsilon \} \\
tr(\pi \xrightarrow{a} p) &= \{ \pi \xrightarrow{a} p \}
\end{align*}
\]
Traces of global types

\[ \text{tr}(\text{skip}) = \{ \varepsilon \} \]
\[ \text{tr}(\pi \xrightarrow{a} p) = \{ \pi \xrightarrow{a} p \} \]
\[ \text{tr}(G_1 ; G_2) = \text{tr}(G_1) \text{tr}(G_2) \]
Traces of global types

\[
\begin{align*}
    \text{tr}(\text{skip}) &= \{ \varepsilon \} \\
    \text{tr}(\pi \xrightarrow{a} p) &= \{ \pi \xrightarrow{a} p \} \\
    \text{tr}(G_1; G_2) &= \text{tr}(G_1)\text{tr}(G_2) \\
    \text{tr}(G^*) &= (\text{tr}(G))^*
\end{align*}
\]
Traces of global types

$$\text{tr}(\text{skip}) = \{ \epsilon \}$$
$$\text{tr}(\pi \xrightarrow{a} p) = \{ \pi \xrightarrow{a} p \}$$
$$\text{tr}(G_1 ; G_2) = \text{tr}(G_1) \text{tr}(G_2)$$
$$\text{tr}(G^*) = (\text{tr}(G))^*$$
$$\text{tr}(G_1 \lor G_2) = \text{tr}(G_1) \cup \text{tr}(G_2)$$
Traces of global types

\[
\begin{align*}
\text{tr}(\text{skip}) &= \{ \varepsilon \} \\
\text{tr}(\pi \xrightarrow{a} p) &= \{ \pi \xrightarrow{a} p \} \\
\text{tr}(G_1; G_2) &= \text{tr}(G_1) \text{tr}(G_2) \\
\text{tr}(G^*) &= (\text{tr}(G))^* \\
\text{tr}(G_1 \lor G_2) &= \text{tr}(G_1) \cup \text{tr}(G_2) \\
\text{tr}(G_1 \land G_2) &= \text{tr}(G_1) \sqcup \text{tr}(G_2)
\end{align*}
\]

\[
L_1 \sqcup L_2 \overset{\text{def}}{=} \{ \varphi_1 \psi_1 \cdots \varphi_n \psi_n \mid \varphi_1 \cdots \varphi_n \in L_1 \land \psi_1 \cdots \psi_n \in L_2 \}
\]
Traces of global types

\[ \text{tr}(\text{skip}) = \{ \varepsilon \} \]
\[ \text{tr}(\pi \xrightarrow{a} p) = \{ \pi \xrightarrow{a} p \} \]
\[ \text{tr}(G_1; G_2) = \text{tr}(G_1) \text{tr}(G_2) \]
\[ \text{tr}(G^*) = (\text{tr}(G))^* \]
\[ \text{tr}(G_1 \lor G_2) = \text{tr}(G_1) \cup \text{tr}(G_2) \]
\[ \text{tr}(G_1 \land G_2) = \text{tr}(G_1) \sqcup \text{tr}(G_2) \]

\[ L_1 \sqcup L_2 \overset{\text{def}}{=} \{ \varphi_1 \psi_1 \cdots \varphi_n \psi_n \mid \varphi_1 \cdots \varphi_n \in L_1 \land \psi_1 \cdots \psi_n \in L_2 \} \]

\[ G = (p \xrightarrow{a} q \land p \xrightarrow{b} q); (q \xrightarrow{c} p; p \xrightarrow{b} q)^*; (q \xrightarrow{d} p \lor q \xrightarrow{e} p) \]
Traces of global types

\[
\begin{align*}
\text{tr}(\text{skip}) &= \{ \varepsilon \} \\
\text{tr}(\pi \rightarrow^a p) &= \{ \pi \rightarrow^a p \} \\
\text{tr}(G_1 ; G_2) &= \text{tr}(G_1) \text{tr}(G_2) \\
\text{tr}(G^*) &= (\text{tr}(G))^* \\
\text{tr}(G_1 \lor G_2) &= \text{tr}(G_1) \cup \text{tr}(G_2) \\
\text{tr}(G_1 \land G_2) &= \text{tr}(G_1) \sqcup \text{tr}(G_2)
\end{align*}
\]

Let \( L_1 \sqcup L_2 \defeq \{ \varphi_1 \psi_1 \cdots \varphi_n \psi_n \mid \varphi_1 \cdots \varphi_n \in L_1 \land \psi_1 \cdots \psi_n \in L_2 \} \)

\[
G = (p \rightarrow^a q \land p \rightarrow^b q); (q \rightarrow^c p; p \rightarrow^b q)^*; (q \rightarrow^d p \lor q \rightarrow^e p)
\]

\[
p \rightarrow^a q; p \rightarrow^b q; q \rightarrow^c p; p \rightarrow^b q; \cdots; q \rightarrow^d p
\]

\[
p \rightarrow^b q; p \rightarrow^a q; q \rightarrow^c p; p \rightarrow^b q; \cdots; q \rightarrow^e p
\]
Syntax of session types

\[ T ::= \text{Pre-Session Type} \]
Syntax of session types

\[ T ::= \text{Pre-Session Type} \]

end (termination)
Syntax of session types

\[
T ::= \text{Pre-Session Type}
\]

- `end` (termination)
- `X` (variable)
Syntax of session types

\[ T ::= \]

- \textbf{Pre-Session Type}
  - \textbf{end} (termination)
  - \textbf{X} (variable)
  - \textbf{p!a.} \textbf{T} (output)
Session types

Syntax of session types

\[ T ::= \]

- `end` (termination)
- `X` (variable)
- `p!a.T` (output)
- `π?a.T` (input)
Session types

Syntax of session types

\[ T ::= \begin{align*}
& \text{end} \quad \text{(termination)} \\
| & X \quad \text{(variable)} \\
| & p!a.T \quad \text{(output)} \\
| & \pi?a.T \quad \text{(input)} \\
| & T \oplus T \quad \text{(internal choice)}
\end{align*} \]
Syntax of session types

\[ T ::= \]

- `end` (termination)
- `X` (variable)
- `p!a.T` (output)
- `\pi?a.T` (input)
- `T \oplus T` (internal choice)
- `T + T` (external choice)
Syntax of session types

\[
T ::= \begin{aligned}
\text{end} & \quad \text{(termination)} \\
X & \quad \text{(variable)} \\
p!a.T & \quad \text{(output)} \\
\pi?a.T & \quad \text{(input)} \\
T \oplus T & \quad \text{(internal choice)} \\
T + T & \quad \text{(external choice)} \\
\text{rec } X.T & \quad \text{(recursion)}
\end{aligned}
\]
Syntax of session types

\[ T ::= \]

- `end` (termination)
- `X` (variable)
- `p!a.T` (output)
- `π?a.T` (input)
- `T ⊕ T` (internal choice)
- `T + T` (external choice)
- `rec X.T` (recursion)

session type
Syntax of session types

\[ T ::= \]

- **Pre-Session Type**
  - `end` (termination)
  - `X` (variable)
  - `p!a.T` (output)
  - `π?a.T` (input)
  - `T ⊕ T` (internal choice)
  - `T + T` (external choice)
  - `rec X.T` (recursion)

**session type**

- `end`
Session types

Syntax of session types

\[ T ::= \begin{align*}
\text{Pre-Session Type} & \quad \text{(termination)} \\
\text{end} & \\
\mathcal{X} & \quad \text{(variable)} \\
p!a.T & \quad \text{(output)} \\
\pi?a.T & \quad \text{(input)} \\
T \oplus T & \quad \text{(internal choice)} \\
T + T & \quad \text{(external choice)} \\
\text{rec } \mathcal{X}.T & \quad \text{(recursion)}
\end{align*} \]

session type

\[ \begin{align*}
\text{end} & \\
\bigoplus_{i \in I} p_i!a_i. T_i & \quad \text{and } \forall i, j \in I \text{ we have that } p_i!a_i = p_j!a_j \text{ implies } i = j \text{ and each } T_i \text{ is a session type}
\end{align*} \]
Session types

Syntax of session types

\[ T ::= \]

- `end` (termination)
- `X` (variable)
- `p!a.T` (output)
- `π?a.T` (input)
- `T ⊕ T` (internal choice)
- `T + T` (external choice)
- `rec X.T` (recursion)

Session type

- `end`
- `\bigoplus_{i \in I} p_i!a_i.T_i` and \( \forall i, j \in I \) we have that \( p_i!a_i = p_j!a_j \) implies \( i = j \) and each \( T_i \) is a session type
- `\bigoplus_{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

\( \{ p_i : T_i \}_{i \in I} \)
Session environments

\{ p_i : T_i \}_{i \in I}

reduction of session environments
Session environments

$$\{p_i : T_i\}_{i \in I}$$

reduction of session environments

$$\mathbb{B}; \{p : \bigoplus_{i \in I} p_i! a_i. T_i\} \cup \Delta \rightarrow (p \xrightarrow{a_k} p_k) :: \mathbb{B}; \{p : T_k\} \cup \Delta \quad (k \in I)$$
Session environments

\[ \{ p_i : T_i \}_i \in I \]

reduction of session environments

\[ \mathbb{B} ; \{ p : \bigoplus_{i \in I} p_i ! a_i . T_i \} \cup \Delta \quad \rightarrow \quad (p \xrightarrow{a_k} p_k) :: \mathbb{B} ; \{ p : T_k \} \cup \Delta \quad (k \in I) \]

\[ \mathbb{B} :: (p_i \xrightarrow{a} p)_{i \in I} ; \{ p : \sum_{j \in J} \pi_j ? a_j . T_j \} \cup \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} p} \quad \mathbb{B} ; \{ p : T_k \} \cup \Delta \]

\[ \begin{cases} k \in J & a_k = a \\ \pi_k = \{ p_i | i \in I \} \end{cases} \]
Session environments

\[{p_i : T_i}_{i \in I}\]

reduction of session environments

\[\Box ; \{p : \bigoplus_{i \in I} p ! a_i . T_i\} \cup \Delta \quad \rightarrow \quad (p \xrightarrow{a_k} p_k) :: \Box ; \{p : T_k\} \cup \Delta \quad (k \in I)\]

\[\Box :: (\langle p_i \xrightarrow{a} p \rangle)_{i \in I} ; \{p : \sum_{j \in J} \pi_j ? a_j . T_j\} \cup \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} p} \quad \Box ; \{p : T_k\} \cup \Delta\]

\[\Delta = \{p : \text{rec } X. (q! a . X \oplus q! b . \text{end}), q : \text{rec } Y. (p? a . Y + p? b . \text{end})\}\]

\[\varepsilon ; \Delta \quad \rightarrow \quad p \xrightarrow{a} q ; \Delta\]
Session types

Session environments

\{p_i : T_i\}_{i \in I}

reduction of session environments

\begin{align*}
\mathbb{B} \triangleright \{p : \bigoplus_{i \in I} p!a_i \cdot T_i\} \cup \Delta \quad \longrightarrow \quad (p \xrightarrow{a_k} p_k) :: \mathbb{B} \triangleright \{p : T_k\} \cup \Delta \quad (k \in I) \\
\mathbb{B} :: (p_i \xrightarrow{a} p)_{i \in I} \triangleright \{p : \sum_{j \in J} \pi_j ? a_j \cdot T_j\} \cup \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} p} \quad \mathbb{B} \triangleright \{p : T_k\} \cup \Delta
\end{align*}

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

\epsilon \triangleright \Delta \quad \longrightarrow \quad p \xrightarrow{a} q \triangleright \Delta \quad \xrightarrow{p \xrightarrow{a} q} \quad \epsilon \triangleright \Delta

On Global Types and Multi-Party Sessions
Session environments

\{ p_i : T_i \}_{i \in I}

reduction of session environments

\begin{align*}
\mathbb{B} \cup \{ p : \bigoplus_{i \in I} p_i! a_i . T_i \} & \cup \Delta \quad \longrightarrow \quad (p \xrightarrow{a_k} p_k) :: \mathbb{B} \cup \{ p : T_k \} \cup \Delta & (k \in I) \\
\mathbb{B} :: (p_i \xrightarrow{a} p)_{i \in I} \cup \{ p : \sum_{j \in J} \pi_j \? a_j . T_j \} & \cup \Delta \quad \xrightarrow{\pi_k \xrightarrow{a} p} \quad \mathbb{B} \cup \{ p : T_k \} \cup \Delta \\
\Delta & = \{ p : \text{rec } X . (q! a . X \oplus q! b . \text{end}) , q : \text{rec } Y . (p? a . Y + p? b . \text{end}) \} \\
\Delta' & = \{ p : \text{end} , q : \text{rec } Y . (p? a . Y + p? b . \text{end}) \} \\
\epsilon \cup \Delta & \longrightarrow \quad p \xrightarrow{a} q \cup \Delta \quad \xrightarrow{p \xrightarrow{a} q} \quad \epsilon \cup \Delta \quad \longrightarrow \\
\quad p \xrightarrow{b} q \cup \Delta'
\end{align*}
Session environments

\(\{p_i : T_i\}_{i \in I}\)

reduction of session environments

\[
\begin{align*}
B \vdash \{p : \bigoplus_{i \in I} p_i ! a_i. T_i\} \cup \Delta & \quad \longrightarrow \quad (p \xrightarrow{a_k} p_k) :: B \vdash \{p : T_k\} \cup \Delta \quad (k \in I) \\
B :: (p_i \xrightarrow{a} p)_{i \in I}, \{p : \sum_{j \in J} p_j ? a_j. T_j\} \cup \Delta & \quad \longrightarrow \quad \left(\begin{array}{c}
k \in J \\
a_k = a \\
\pi_k = \{p_i | i \in I\}
\end{array}\right) \\
\Delta & = \{p : \text{rec } X.(q!a.X \oplus q!b.\text{end}) , q : \text{rec } Y.(p?a.Y + p?b.\text{end})\} \\
\Delta' & = \{p : \text{end} , q : \text{rec } Y.(p?a.Y + p?b.\text{end})\} \\
\varepsilon \vdash \Delta & \quad \longrightarrow \quad p \xrightarrow{a} q \vdash \Delta \quad p \xrightarrow{a} q \quad \varepsilon \vdash \Delta \quad \longrightarrow \\
p \xrightarrow{b} q \vdash \Delta' & \quad \longrightarrow \quad p \xrightarrow{b} q \quad \varepsilon \vdash \{p : \text{end}, q : \text{end}\}
\end{align*}
\]
Traces of session environments

\[ \Delta \text{ is a live session if } \varepsilon \varnothing \Delta \xrightarrow{\varphi} \mathbb{B} \varnothing \Delta' \text{ implies } \mathbb{B} \varnothing \Delta' \xrightarrow{\psi} \varepsilon \varnothing \{p_i : \text{end}\}_{i \in I} \text{ for some } \psi \]

stronger than progress
Traces of session environments

\( \Delta \) is a \textit{live session} if \( \varepsilon \circ \Delta \xrightarrow{\varphi} B \circ \Delta' \) implies

\[
B \circ \Delta' \xrightarrow{\psi} \varepsilon \circ \{ p_i : \text{end} \}_{i \in I} \quad \text{for some } \psi
\]

\[
\text{tr}(\Delta) \overset{\text{def}}{=} \begin{cases} 
\{ \varphi \mid \varepsilon \circ \Delta \xrightarrow{\varphi} \varepsilon \circ \{ p_i : \text{end} \}_{i \in I} \} & \text{if } \Delta \text{ is a live session} \\
\emptyset & \text{otherwise}
\end{cases}
\]
Traces of session environments

$\Delta$ is a live session if $\epsilon \varphi \Delta \xrightarrow{\varphi} B \varphi \Delta'$ implies

$B \varphi \Delta' \xrightarrow{\psi} \epsilon \varphi \{p_i : \text{end}\}_{i \in I}$ for some $\psi$

$$\text{tr}(\Delta) \overset{\text{def}}{=} \begin{cases} \{ \varphi \mid \epsilon \varphi \Delta \xrightarrow{\varphi} \epsilon \varphi \{p_i : \text{end}\}_{i \in I} \} & \text{if } \Delta \text{ is a live session} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\text{tr}(\{p : \text{rec } X.(q!a.X \oplus q!b.\text{end}) \), q : \text{rec } Y.(p?a.Y + p?b.\text{end})\}) = \text{tr}((p \xrightarrow{a} q)^* ; p \xrightarrow{b} q)$$
Traces of session environments

\( \Delta \) is a **live session** if \( \varepsilon \circ \Delta \xrightarrow{\varphi} \mathbb{B} \circ \Delta' \) implies

\[ \mathbb{B} \circ \Delta' \xrightarrow{\psi} \varepsilon \circ \{ p_i : \text{end} \}_{i \in I} \text{ for some } \psi \]

\[
\begin{align*}
\text{tr}(\Delta) & \overset{\text{def}}{=} \begin{cases} 
\{ \varphi \mid \varepsilon \circ \Delta \xrightarrow{\varphi} \varepsilon \circ \{ p_i : \text{end} \}_{i \in I} \} & \text{if } \Delta \text{ is a live session} \\
\emptyset & \text{otherwise}
\end{cases} \\
\text{tr}(\{ p : \text{rec } X.(q!a.X \oplus q!b.\text{end}) , q : \text{rec } Y.(p?a.Y + p?b.\text{end}) \}) & = \\
\text{tr}((p \xrightarrow{a} q)^*; p \xrightarrow{b} q) \\
\text{tr}(\{ p : \text{rec } X.q!a.X, q : \text{rec } Y.p?a.Y \}) & = \emptyset
\end{align*}
\]
On Global Types and Multi-Party Sessions

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
Traces of global types and session environments

first try (too strong condition):
\( \text{tr}(\mathcal{G}) = \text{tr}(\Delta) \) does not allow to project \( \mathcal{G}_1 \land \mathcal{G}_2 \)
Traces of global types and session environments

first try (too strong condition):
\( \text{tr}(G) = \text{tr}(\Delta) \) does not allow to project \( G_1 \land G_2 \)

second try (too weak condition):
\( \text{tr}(G) \subseteq \text{tr}(\Delta) \) looses the exhaustivity property
\( \{p : q!a.end, q : p?a.end\} \) would implement \( p \xrightarrow{a} q \lor p \xrightarrow{b} q \)
Traces of global types and session environments

first try (too strong condition):
\[ \text{tr}(\mathcal{G}) = \text{tr}(\Delta) \] does not allow to project \( \mathcal{G}_1 \land \mathcal{G}_2 \)

second try (too weak condition):
\[ \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta) \] looses the exhaustivity property

\{\text{p} : \text{q}!a.\text{end}, \text{q} : \text{p}?a.\text{end}\} would implement \( \text{p} \xrightarrow{a} \text{q} \lor \text{p} \xrightarrow{b} \text{q} \)

\[ \text{tr}(\Delta) \subseteq \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta)^\circ \]

\[ \Delta \sqsubseteq \mathcal{G} \]

\( L^\circ \overset{\text{def}}{=} \{\alpha_1 \cdots \alpha_n \mid \text{there exists a permutation } \sigma \text{ such that } \alpha_{\sigma(1)} \cdots \alpha_{\sigma(n)} \in L\} \)
Traces of global types and session environments

first try (too strong condition):
\( \text{tr}(\mathcal{G}) = \text{tr}(\Delta) \) does not allow to project \( \mathcal{G}_1 \land \mathcal{G}_2 \)

second try (too weak condition):
\( \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta) \) looses the exhaustivity property
\{p : q!a.end, q : p?a.end\} would implement \( p \xrightarrow{a} q \lor p \xrightarrow{b} q \)

\( \text{tr}(\Delta) \subseteq \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta)^{\circ} \)
\( \Delta \preceq G \)

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

\( \text{tr}(\Delta) \subseteq \text{tr}(\mathcal{G}) \): every trace of \( \Delta \) is a trace of \( \mathcal{G} \) (soundness)
Traces of global types and session environments

first try (**too strong condition**):
\[ \text{tr}(\mathcal{G}) = \text{tr}(\Delta) \] does not allow to project \( \mathcal{G}_1 \land \mathcal{G}_2 \)

second try (**too weak condition**):
\[ \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta) \] looses the exhaustivity property
\{p : q!a.end, q : p?a.end\} would implement \( p \xrightarrow{a} q \lor p \xrightarrow{b} q \)

\[
\text{tr}(\Delta) \subseteq \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta)^{\circ}
\]
\( \Delta \preceq \mathcal{G} \)

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

\( \text{tr}(\Delta) \subseteq \text{tr}(\mathcal{G}) \): every trace of \( \Delta \) is a trace of \( \mathcal{G} \) (**soundness**)

\( \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta)^{\circ} \): every trace of \( \mathcal{G} \) is the permutation of a trace of \( \Delta \) (**completeness**)

On Global Types and Multi-Party Sessions
On Global Types and Multi-Party Sessions

<table>
<thead>
<tr>
<th>Global types and session types</th>
<th>Projections</th>
<th>Related approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic projection</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Projection rules I**

\[ \Delta \vdash G \Rightarrow \Delta' \]
Semantic projection

Projection rules I

\[ \Delta \vdash \mathcal{G} \triangleright \Delta' \]

(SP-Skip)
\[ \Delta \vdash \text{skip} \triangleright \Delta \]
### Semantic projection

**Projection rules I**

\[ \Delta \vdash \mathcal{G} \triangleright \Delta' \]

(SP-Skip)
\[ \Delta \vdash \text{skip} \triangleright \Delta \]

(SP-Action)
\[ \{p_i : T_i\}_{i \in I} \cup \{p : T\} \cup \Delta \vdash \{p_i\}_{i \in I} \xrightarrow{a} p \triangleright \{p_i : p! a. T\}_{i \in I} \cup \{p : \{p_i\}_{i \in I} ? a. T\} \cup \Delta \]
Semantic projection

Projection rules I

\[ \Delta \vdash G \triangleright \Delta' \]

(SP-Skip)
\[ \Delta \vdash \text{skip} \triangleright \Delta \]

(SP-Action)
\[ \{ p_i : T_i \}_{i \in I} \cup \{ p : T \} \cup \Delta \vdash \{ p_i \}_{i \in I} \xrightarrow{a} p \triangleright \{ p_i : p!a.T_i \}_{i \in I} \cup \{ p : \{ p_i \}_{i \in I}?a.T \} \cup \Delta \]
\[ \{ p : \text{end}, q : \text{end} \} \vdash p \xrightarrow{a} q \triangleright \{ p : q!a.end, q : p?a.end \} \]
## Projection rules I

\( \Delta \vdash G \triangleright \Delta' \)

**SP-Skip**
\[
\Delta \vdash \text{skip} \triangleright \Delta
\]

**SP-Action**
\[
\{ p_i : T_i \}_{i \in I} \cup \{ p : T \} \cup \Delta \vdash \{ p_i \}_{i \in I} \rightarrow^a p \triangleright \{ p_i : p!a.T_i \}_{i \in I} \cup \{ p : \{ p_i \}_{i \in I}?a.T \} \cup \Delta
\]
\[
\{ p : \text{end}, q : \text{end} \} \vdash p \rightarrow^a q \triangleright \{ p : q!a.end, \quad q : p?a.end \}
\]

**SP-Sequence**
\[
\Delta \vdash G_2 \triangleright \Delta' \quad \Delta' \vdash G_1 \triangleright \Delta''
\]
\[
\Delta \vdash G_1 ; G_2 \triangleright \Delta''
\]
Global types and session types

Projections

Related approaches

Semantic projection

Projection rules II

(SP-Alternative)

\[ \Delta \vdash \mathcal{G}_1 \triangleright \{p : T_1\} \cup \Delta' \]
\[ \Delta \vdash \mathcal{G}_2 \triangleright \{p : T_2\} \cup \Delta' \]
\[ \Delta \vdash \mathcal{G}_1 \lor \mathcal{G}_2 \triangleright \{p : T_1 \oplus T_2\} \cup \Delta' \]
Projection rules II

(SP-Alternative)

\[
\Delta \vdash G_1 \triangleright \{p : T_1\} \uplus \Delta' \quad \Delta \vdash G_2 \triangleright \{p : T_2\} \uplus \Delta'
\]

\[
\Delta \vdash G_1 \lor G_2 \triangleright \{p : T_1 \oplus T_2\} \uplus \Delta'
\]

\[
\Delta_0 \vdash p \xrightarrow{a} q \triangleright \{p : q!a.end, q : T\} \quad \Delta_0 \vdash p \xrightarrow{b} q \triangleright \{p : q!b.end, q : T\}
\]

\[
\Delta_0 \vdash p \xrightarrow{a} q \lor p \xrightarrow{b} q \triangleright \{p : q!a.end \oplus q!b.end, q : T\}
\]

\[
\Delta_0 = \{p : end, q : end\} \quad T = p?a.end + p?b.end
\]
Semantic projection

**Projection rules III**

(SP-Iteration)

\[
\{ p : T_1 \oplus T_2 \} \cup \Delta \vdash \mathcal{G} \triangleright \{ p : T_1 \} \cup \Delta
\]

\[
\{ p : T_2 \} \cup \Delta \vdash \mathcal{G}^* \triangleright \{ p : T_1 \oplus T_2 \} \cup \Delta
\]
Global types and session types

Semantic projection

Projection rules III

(SP-Iteration)

\[
\begin{align*}
\{p : T_1 \oplus T_2\} \cup \Delta &\vdash G \triangleright \{p : T_1\} \cup \Delta \\
\{p : T_2\} \cup \Delta &\vdash G^* \triangleright \{p : T_1 \oplus T_2\} \cup \Delta
\end{align*}
\]

\[
\begin{align*}
\{p : T_1 \oplus T_2, q : S\} &\vdash p \xrightarrow{a} q \triangleright \{p : T_1, q : S\} \\
\{p : T_2, q : S\} &\vdash (p \xrightarrow{a} q)^* \triangleright \{p : T_1 \oplus T_2, q : S\}
\end{align*}
\]

\[T_1 = q!a.\text{rec }X.(q!a.X \oplus q!b.\text{end})\]

\[T_2 = q!b.\text{end}\]

\[S = \text{rec }Y.(p?a.Y + p?b.\text{end})\]
Projection rules IV

( )

\[ \Delta \vdash G' \triangleright \Delta' \quad G' \subseteq G \quad \Delta'' \subseteq \Delta' \]

\[ \Delta \vdash G \triangleright \Delta'' \]
Projection rules IV

(SP-Subsumption)

\[ \Gamma \vdash \Delta' \triangleright \Delta'' \quad \Delta' \preceq \Delta \quad \Delta'' \preceq \Delta' \]

subsumption on global types
Global types and session types

Projections

Related approaches

Semantic projection

Projection rules IV

(\text{SP-Subsumption})

\[ \Delta \vdash \mathcal{G}' \triangleright \Delta' \quad \mathcal{G}' \leq \mathcal{G} \quad \Delta'' \leq \Delta' \]

\[ \Delta \vdash \mathcal{G} \triangleright \Delta'' \]

\[ p \xrightarrow{a} q; r \xrightarrow{b} s \leq p \xrightarrow{a} q \land r \xrightarrow{b} s \]
Projection rules IV

(SP-Subsumption)

\[
\Delta \vdash \Delta' \triangleright \Delta'' \quad \Delta' \leq \Delta \\
\Delta \vdash \Delta''
\]

\[
p \xrightarrow{a} q; r \xrightarrow{b} s \leq p \xrightarrow{a} q \land r \xrightarrow{b} s
\]

\[
p \xrightarrow{a} q; r \xrightarrow{b} s \leq (p \xrightarrow{a} q; r \xrightarrow{b} s) \lor (r \xrightarrow{b} s; p \xrightarrow{a} q)
\]
Projection rules IV

(SP-Subsumption)

$$\Delta \vdash \mathcal{G}' \triangleright \Delta' \quad \mathcal{G}' \leq \mathcal{G} \quad \Delta'' \leq \Delta'$$

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

$$p \overset{a}{\rightarrow} q; r \overset{b}{\rightarrow} s \leq p \overset{a}{\rightarrow} q \land r \overset{b}{\rightarrow} s$$

$$p \overset{a}{\rightarrow} q; r \overset{b}{\rightarrow} s \leq (p \overset{a}{\rightarrow} q; r \overset{b}{\rightarrow} s) \lor (r \overset{b}{\rightarrow} s; p \overset{a}{\rightarrow} q)$$

$$r \overset{b}{\rightarrow} p; (p \overset{a}{\rightarrow} q \lor p \overset{b}{\rightarrow} q) \leq (r \overset{b}{\rightarrow} p; p \overset{a}{\rightarrow} q) \lor (r \overset{b}{\rightarrow} p; p \overset{b}{\rightarrow} q)$$
Semantic projection

Projection rules IV

(SP-Subsumption)

\[ \Delta \vdash \mathcal{G}' \supset \Delta' \quad \mathcal{G}' \leq \mathcal{G} \quad \Delta'' \leq \Delta' \]

\[ \Delta \vdash \mathcal{G} \supset \Delta'' \]

\[ p \xrightarrow{a} q; r \xrightarrow{b} s \leq p \xrightarrow{a} q \land r \xrightarrow{b} s \]

\[ p \xrightarrow{a} q; r \xrightarrow{b} s \leq (p \xrightarrow{a} q; r \xrightarrow{b} s) \lor (r \xrightarrow{b} s; p \xrightarrow{a} q) \]

\[ r \xrightarrow{b} p; (p \xrightarrow{a} q \lor p \xrightarrow{b} q) \leq (r \xrightarrow{b} p; p \xrightarrow{a} q) \lor (r \xrightarrow{b} p; p \xrightarrow{b} q) \]
Projection rules IV

(SP-Subsumption)

\[ \Delta \vdash \mathcal{G}' \triangleright \Delta' \quad \mathcal{G}' \leq \mathcal{G} \quad \Delta'' \leq \Delta' \]

\[ \Delta \vdash \mathcal{G} \triangleright \Delta'' \]

subsumption on session environments
Main results

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

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

If $G$ is well formed and $\vdash G \triangleright \Delta$, then $\Delta \sqsubseteq G$
Main results

\[ \mathcal{G} \text{ is well formed if } \varphi; \pi \xrightarrow{a} p; \pi' \xrightarrow{b} p'; \psi \in \text{tr}(\mathcal{G}) \text{ implies either} \]
\[ p \in \pi' \cup \{p'\} \text{ or } \varphi; \pi' \xrightarrow{b} p'; \pi \xrightarrow{a} p; \psi \in \text{tr}(\mathcal{G}) \]

If \( \mathcal{G} \) is well formed and \( \vdash \mathcal{G} \triangleright \Delta \), then \( \Delta \leq \mathcal{G} \)

> No sequentiality: \( \nexists \Delta : \Delta \leq \mathcal{G} \) and \( \exists \Delta : \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta) \subseteq \text{tr}(\mathcal{G})^\# \)

\( L^\# \) is the smallest well-formed set such that \( L \subseteq L^\# \)
Main results

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

If \( G \) is well formed and \( \vdash G \triangleright \Delta \), then \( \Delta \leq G \)

- **No sequentiality:** \( \not\exists \Delta : \Delta \leq G \) and \( \exists \Delta : \text{tr}(G) \subseteq \text{tr}(\Delta) \subseteq \text{tr}(G)^\# \)
  
  \( L^\# \) is the smallest well-formed set such that \( L \subseteq L^\# \)

- **No knowledge for choice:**
  
  \( \not\exists \Delta : \text{tr}(G) \subseteq \text{tr}(\Delta) \subseteq \text{tr}(G)^\# \) and \( \exists \Delta : \text{tr}(G) \subseteq \text{tr}(\Delta) \)
Main results

\[ \mathcal{G} \text{ is well formed if } \varphi; \pi \xrightarrow{a} p; \pi' \xrightarrow{b} p'; \psi \in \text{tr}(\mathcal{G}) \implies \text{either } p \in \pi' \cup \{p'\} \text{ or } \varphi; \pi' \xrightarrow{b} p'; \pi \xrightarrow{a} p; \psi \in \text{tr}(\mathcal{G}) \]

If \( \mathcal{G} \) is well formed and \( \vdash \mathcal{G} \triangleright \Delta \), then \( \Delta \leq \mathcal{G} \)

- **No sequentiality:** \( \# \Delta : \Delta \leq \mathcal{G} \) and \( \exists \Delta : \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta) \subseteq \text{tr}(\mathcal{G})\# \)

  \( \mathcal{L}\# \) is the smallest well-formed set such that \( \mathcal{L} \subseteq \mathcal{L}\# \)

- **No knowledge for choice:**

  \( \# \Delta : \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta) \subseteq \text{tr}(\mathcal{G})\# \) and \( \exists \Delta : \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta) \)

- **No knowledge, no choice:** \( \# \Delta : \text{tr}(\mathcal{G}) \subseteq \text{tr}(\Delta) \)
Projection rules

no subsumption on session environments
Algorithmic projection

Projection rules

no subsumption on session environments

(AP-Alternative)

\[
\Delta \vdash_a \mathcal{G}_1 \triangleright \{p : T_1\} \cup \Delta_1 \quad \Delta \vdash_a \mathcal{G}_2 \triangleright \{p : T_2\} \cup \Delta_2
\]

\[
\Delta \vdash_a \mathcal{G}_1 \lor \mathcal{G}_2 \triangleright \{p : T_1 \oplus T_2\} \cup (\Delta_1 \land \Delta_2)
\]
Algorithmic projection

Projection rules

no subsumption on session environments

(AP-Alternative)

\[
\Delta \vdash_a G_1 \triangleright \{p : T_1\} \uplus \Delta_1 \quad \Delta \vdash_a G_2 \triangleright \{p : T_2\} \uplus \Delta_2
\]

\[
\Delta \vdash_a G_1 \lor G_2 \triangleright \{p : T_1 \oplus T_2\} \uplus (\Delta_1 \land \Delta_2)
\]

(AP-Iteration)

\[
\{p : X\} \uplus \{p_i : X_i\}_{i \in I} \vdash_a G \triangleright \{p : S\} \uplus \{p_i : S_i\}_{i \in I}
\]

\[
\{p : T\} \uplus \{p_i : T_i\}_{i \in I} \uplus \Delta \vdash_a G^* \triangleright \{p : \text{rec } X.(T \oplus S)\} \uplus \{p_i : \text{rec } X_i.(T_i \land S_i)\}_{i \in I} \uplus \Delta
\]
Projection rules

no subsumption on session environments

(AP-Alternative)

\[
\Delta \vdash_a G_1 \triangleright \{ p : T_1 \} \uplus \Delta_1 \quad \Delta \vdash_a G_2 \triangleright \{ p : T_2 \} \uplus \Delta_2 \\
\Delta \vdash_a G_1 \lor G_2 \triangleright \{ p : T_1 \oplus T_2 \} \uplus (\Delta_1 \land \Delta_2)
\]

(AP-Iteration)

\[
\{ p : X \} \uplus \{ p_i : X_i \}_{i \in I} \vdash_a G \triangleright \{ p : S \} \uplus \{ p_i : S_i \}_{i \in I} \\
\{ p : T \} \uplus \{ p_i : T_i \}_{i \in I} \uplus \Delta \vdash_a G^* \triangleright \{ p : \text{rec } X.(T \oplus S) \} \uplus \{ p_i : \text{rec } X_i.(T_i \land S_i) \}_{i \in I} \uplus \Delta
\]

no subsumption on global types: \(\land\)-types must be eliminated
Algorithmic projection

Projection rules

no subsumption on session environments

(AP-Alternative)

\[
\Delta \vdash_a \mathcal{G}_1 \triangleright \{p : T_1\} \uplus \Delta_1 \\
\Delta \vdash_a \mathcal{G}_2 \triangleright \{p : T_2\} \uplus \Delta_2 \\
\Delta \vdash_a \mathcal{G}_1 \lor \mathcal{G}_2 \triangleright \{p : T_1 \oplus T_2\} \uplus (\Delta_1 \land \Delta_2)
\]

(AP-Iteration)

\[
\{p : X\} \uplus \{p_i : X_i\}_{i \in I} \vdash_a \mathcal{G} \triangleright \{p : S\} \uplus \{p_i : S_i\}_{i \in I} \\
\{p : T\} \uplus \{p_i : T_i\}_{i \in I} \uplus \Delta \vdash_a \mathcal{G}^* \triangleright \{p : \text{rec } X.(T \oplus S)\} \uplus \{p_i : \text{rec } X_i.(T_i \land S_i)\}_{i \in I} \uplus \Delta
\]

no subsumption on global types: \(\land\)-types must be eliminated

\(\mathcal{G} \preceq \mathcal{G}'\) is decidable by the decidability of the Parikh equivalence on regular languages
Global types and session types

Kleene star and recursion

$k$-Exit Iterations

\[
\left( p \xrightarrow{\text{handover}} q; q \xrightarrow{\text{handover}} p \right)^*; \left( p \xrightarrow{\text{bailout}} q \lor p \xrightarrow{\text{handover}} q; q \xrightarrow{\text{bailout}} p \right)
\]
**k-Exit Iterations**

\[
(p \xrightarrow{\text{handover}} q; q \xrightarrow{\text{handover}} p)^*; (p \xrightarrow{\text{bailout}} q \lor p \xrightarrow{\text{handover}} q; q \xrightarrow{\text{bailout}} p)
\]

\[
(G_1, \ldots, G_k)^k \times (G'_1, \ldots, G'_k)
\]
**k-Exit Iterations**

\[(p \xrightarrow{\text{handover}} q; q \xrightarrow{\text{handover}} p)^*; (p \xrightarrow{\text{bailout}} q \lor p \xrightarrow{\text{handover}} q; q \xrightarrow{\text{bailout}} p)\]

\[(G_1, \ldots, G_k)^k (G'_1, \ldots, G'_k)\]

\[(p \xrightarrow{\text{handover}} q, q \xrightarrow{\text{handover}} p)^2* (p \xrightarrow{\text{bailout}} q, q \xrightarrow{\text{bailout}} p)\]
Global types and session types

Projections

Related approaches

Kleene star and recursion

**k-Exit Iterations**

\[
(p \xrightarrow{\text{handover}} q; q \xrightarrow{\text{handover}} p)^*; (p \xrightarrow{\text{bailout}} q \lor p \xrightarrow{\text{handover}} q; q \xrightarrow{\text{bailout}} p)
\]

\[
(G_1, \ldots, G_k)^k (G'_1, \ldots, G'_k)
\]

\[
(p \xrightarrow{\text{handover}} q, q \xrightarrow{\text{handover}} p)^2 (p \xrightarrow{\text{bailout}} q, q \xrightarrow{\text{bailout}} p)
\]

\[
p : \text{rec } X. (q!\text{handover}. (q?\text{handover}. X + q?\text{bailout}.\text{end}) \oplus q!\text{bailout}.\text{end})
\]

\[
q : \text{rec } Y. (p?\text{handover}. (p!\text{handover}. Y \oplus p!\text{bailout}.\text{end}) + p?\text{bailout}.\text{end})
\]
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
Abstract

Communication is becoming one of the central elements in software development. As a potential (and foundational) for structured communication-oriented programming, session types have been studied over the last decade (for a wide range of process calculi and programming language theories) in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. Presently a typical calculus for multi-party sessions, the theory identifies a new range of types in a multi-party context. Based on this foundation, the paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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

General Terms Theory, 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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications.

This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.

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 multiparty programming. As a potential typed foundational for structured communication-oriented programming, session types have been studied over the last decade for a wide range of process calculi and programming language theories, in session or binary (two-party) sessions. This work extends the fragmenting theories of binary session types to multi-party, asynchronous sessions, which often arise in practical communication-oriented applications. This paper presents a formalization of session types as a global semantic structure. Global types retain a linear type system from binary session types while capturing complex causal states of multi-party asynchronous interactions. A global type plays the role of a shared agreement among communication parties, and is used as a basis of efficient type checking through its projection onto individual pairs. The fundamental properties of the session type discipline such as communication safety, progress and soundness are established for general multi-party asynchronous interactions.
MSG of the seller-buyer protocol

 Automata

On Global Types and Multi-Party Sessions
CFMs implementing the seller-buyer protocol
Kao Chow protocol in WPPL

1 \( (\text{spec} \ ( [a \ (a \ b \ s \ kas) \ (kab)] \)) \)
2 \( [b \ (b \ s \ kbs) \ (kab)] \ [s \ (a \ b \ s \ kas \ kbs) \ ()] \)) \)
3 \( [a \ -> \ s : \ a, \ b, \ na:nonce] \)
4 \( [s \ -> \ b : \ |a, \ b, \ na, \ kab| \ kas, \ |a, \ b, \ na, \ kab| \ kbs] \)
5 \( [b \ -> \ a : \ |a, \ b, \ na, \ kab| \ kas, \ |na| \ kab, \ nb:nonce] \)
6 \( [a \ -> \ b : \ |nb| \ kab] \). \)