Global Types for Asynchronous Multiparty Sessions

Paola Giannini

DiSSTE, Università del Piemonte Orientale

joint work with
Ilaria Castellani & Francesco Dagnino & Mariangiola Dezani

Kickoff T-Ladies, Pisa, 6-7 July, 2022

From the project's description

T3.1: Behavioral types of entities

We will develop type theories to specify and verify properties of dynamic systems, as in IoT, characterized by a high number of heterogeneous entities with possibly both synchronous (e.g., clock synchronization protocols for real-time monitoring) and asynchronous interactions (e.g., publish/subscribe models in the context of IoT event-driven architectures).
....

T4.3: Global types

In this task we will investigate a top-down methodology for the development of IoT applications based on global types to ensure that the interactions among "things" satisfy a given property by design. ....

UPO
UNIVERSITÀ DEL PIEMONTE ORIENTALE

From the project's description

T3.1: Behavioral types of entities

We will develop type theories to specify and verify properties of dynamic systems, as in IoT, characterized by a high number of heterogeneous entities with possibly both synchronous (e.g., clock synchronization protocols for real-time monitoring) and asynchronous interactions (e.g., publish/subscribe models in the context of IoT event-driven architectures).
....

T4.3: Global types

In this task we will investigate a top-down methodology for the development of IoT applications based on global types to ensure that the interactions among "things" satisfy a given property by design. ....

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Index

Multiparty Sessions Methodology

- A multiparty session[1] is an interaction between participants exchanging messages according to a predefined protocol.
- The communication protocol is described by a global type, which specifies the overall behaviour of the system of interacting processes.
- The local behaviour for each participant, called session type, is algorithmically obtained as the projection of the global type.
- Session types can be used to
  - type-check the processes associated to participants (statically)
  - generate monitors to ensure that the processes behave according the the protocol specification (dynamically)

[1] K. Honda, N. Yoshida, M. Carbone: Multiparty asynchronous session types, POPL, 2008.

UNIVERSITÀ DEL PIEMONTE ORIENTALE

Multiparty Sessions Methodology

- A multiparty session[1] is an interaction between participants exchanging messages according to a predefined protocol.
- The communication protocol is described by a global type, which specifies the overall behaviour of the system of interacting processes.
- The local behaviour for each participant, called session type, is algorithmically obtained as the projection of the global type.
- Session types can be used to
  - type-check the processes associated to participants (statically)
  - generate monitors to ensure that the processes behave according the the protocol specification (dynamically)

[1] K. Honda, N. Yoshida, M. Carbone: Multiparty asynchronous session types, POPL, 2008.

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Multiparty Sessions Methodology

- A multiparty session[1] is an interaction between participants exchanging messages according to a predefined protocol.
- The communication protocol is described by a global type, which specifies the overall behaviour of the system of interacting processes.
- The local behaviour for each participant, called session type, is algorithmically obtained as the projection of the global type.
- Session types can be used to
  - type-check the processes associated to participants (statically)
  - generate monitors to ensure that the processes behave according the the protocol specification (dynamically)

[1] K. Honda, N. Yoshida, M. Carbone: Multiparty asynchronous session types, POPL, 2008.

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Multiparty Sessions Methodology

- A multiparty session[1] is an interaction between participants exchanging messages according to a predefined protocol.
- The communication protocol is described by a global type, which specifies the overall behaviour of the system of interacting processes.
- The local behaviour for each participant, called session type, is algorithmically obtained as the projection of the global type.
- Session types can be used to
  - type-check the processes associated to participants (statically)
  - generate monitors to ensure that the processes behave according the the protocol specification (dynamically)

[1] K. Honda, N. Yoshida, M. Carbone: Multiparty asynchronous session types, POPL, 2008.

Multiparty Sessions Methodology

- A multiparty session[1] is an interaction between participants exchanging messages according to a predefined protocol.
- The communication protocol is described by a global type, which specifies the overall behaviour of the system of interacting processes.
- The local behaviour for each participant, called session type, is algorithmically obtained as the projection of the global type.
- Session types can be used to
  - type-check the processes associated to participants (statically)
  - generate monitors to ensure that the processes behave according the the protocol specification (dynamically)

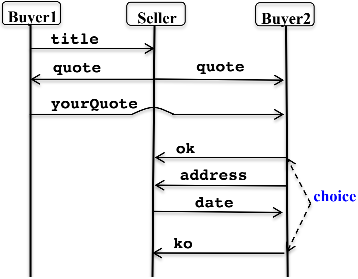
[1] K. Honda, N. Yoshida, M. Carbone: Multiparty asynchronous session types, POPL, 2008.

Multiparty Sessions Methodology

- A multiparty session[1] is an interaction between participants exchanging messages according to a predefined protocol.
- The communication protocol is described by a global type, which specifies the overall behaviour of the system of interacting processes.
- The local behaviour for each participant, called session type, is algorithmically obtained as the projection of the global type.
- Session types can be used to
  - type-check the processes associated to participants (statically)
  - generate monitors to ensure that the processes behave according the the protocol specification (dynamically)

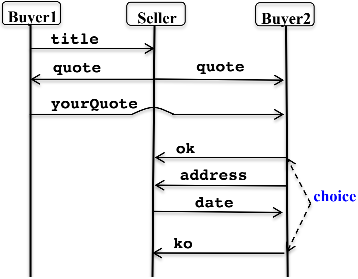
[1] K. Honda, N. Yoshida, M. Carbone: Multiparty asynchronous session types, POPL, 2008.

A multiparty session



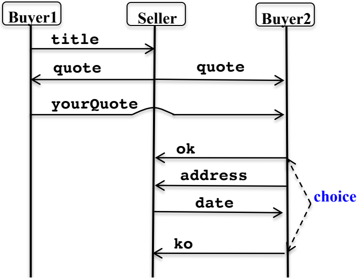
- Buyer1 sends a message to Seller with the `title` of the book she wants to buy
- Seller after receiving a `title` sends to both buyers a `quote` of the price
- Buyer1 computes how much she wants to pay and sends to Buyer2 the amount she should contribute, `yourQuote`
- Buyer2 using this information may decide
  - either to send an ok message to the Seller followed by the address the book should be sent to, and then she waits for a date from the Seller
  - or to give up and send a ko message

A multiparty session



- Buyer1 sends a message to Seller with the `title` of the book she wants to buy
- Seller after receiving a `title` sends to both buyers a `quote` of the price
- Buyer1 computes how much she wants to pay and sends to Buyer2 the amount she should contribute, `yourQuote`
- Buyer2 using this information may decide
  - either to send an `ok` message to the Seller followed by the `address` the book should be sent to, and then she waits for a `date` from the Seller,
  - or to give up and send a `ko` message.

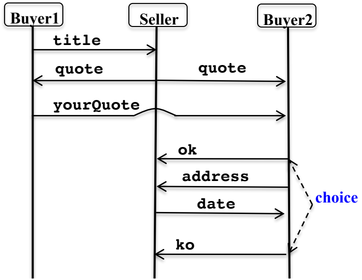
A multiparty session



- Buyer1 sends a message to Seller with the `title` of the book she wants to buy
- Seller after receiving a `title` sends to both buyers a `quote` of the price
- Buyer1 computes how much she wants to pay and sends to Buyer2 the amount she should contribute, `yourQuote`
- Buyer2 using this information may decide
  - either to send an `ok` message to the Seller followed by the `address` the book should be sent to, and then she waits for a `date` from the Seller,
  - or to give up and send a `ko` message.

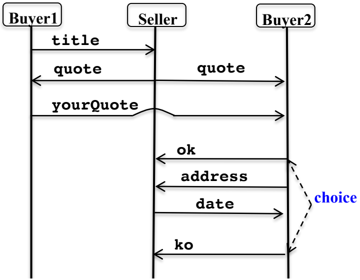
UPO
UNIVERSITÀ DEL PIEMONTE ORIENTALE

A multiparty session



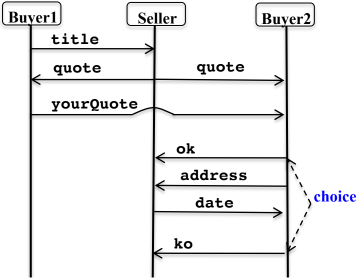
- Buyer1 sends a message to Seller with the `title` of the book she wants to buy
- Seller after receiving a `title` sends to both buyers a `quote` of the price
- Buyer1 computes how much she wants to pay and sends to Buyer2 the amount she should contribute, `yourQuote`
- Buyer2 using this information may decide
  - either to send an `ok` message to the Seller followed by the `address` the book should be sent to, and then she waits for a `date` from the Seller,
  - or to give up and send a `ko` message.

A multiparty session



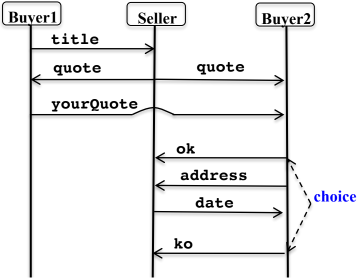
- Buyer1 sends a message to Seller with the `title` of the book she wants to buy
- Seller after receiving a `title` sends to both buyers a `quote` of the price
- Buyer1 computes how much she wants to pay and sends to Buyer2 the amount she should contribute, `yourQuote`
- Buyer2 using this information may decide
  - either to send an `ok` message to the Seller followed by the `address` the book should be sent to, and then she waits for a `date` from the Seller,
  - or to give up and send a `ko` message.

A multiparty session



- Buyer1 sends a message to Seller with the `title` of the book she wants to buy
- Seller after receiving a `title` sends to both buyers a `quote` of the price
- Buyer1 computes how much she wants to pay and sends to Buyer2 the amount she should contribute, `yourQuote`
- Buyer2 using this information may decide
  - either to send an `ok` message to the Seller followed by the `address` the book should be sent to, and then she waits for a `date` from the Seller,
  - or to give up and send a `ko` message.

A multiparty session



- Buyer1 sends a message to Seller with the `title` of the book she wants to buy
- Seller after receiving a `title` sends to both buyers a `quote` of the price
- Buyer1 computes how much she wants to pay and sends to Buyer2 the amount she should contribute, `yourQuote`
- Buyer2 using this information may decide
  - either to send an `ok` message to the Seller followed by the `address` the book should be sent to, and then she waits for a `date` from the Seller,
  - or to give up and send a `ko` message.

Global and Session Types

Global type of the session (where B1, B2 and S stand for Buyer1, Buyer2 and Seller) is

$$\begin{array}{l} \mathsf{B1} \to \mathsf{S} : \mathtt{title}; \\ \mathsf{S} \to \mathsf{B1} : \mathtt{quote}; \mathsf{S} \to \mathsf{B2} : \mathtt{quote}; \\ \mathsf{B1} \to \mathsf{B2} : \mathtt{yourQuote}; \\ \mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array}$$

Session types of participants: obtained by projection from the global type.

$$\begin{array}{lcl} \mathsf{T}_{\mathsf{B1}} & = & \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \mathbf{End} \\ \\ \mathsf{T}_{\mathsf{S}} & = & \begin{array}{l} \mathsf{B1}\,?\,\mathtt{title}; \\ \mathsf{B1}\,!\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{quote}; \\ \mathsf{B2}\,?\,\{\mathtt{ok}; \mathsf{B2}\,?\,\mathtt{address}; \mathsf{B2}\,!\,\mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array} \\ \\ \mathsf{T}_{\mathsf{B2}} & = & \begin{array}{l} \mathsf{S}\,?\,\mathtt{quote}; \\ \mathsf{B1}\,?\,\mathtt{yourQuote}; \\ \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array} \end{array}$$

- $\mathsf{B2}\,?\,\{\mathtt{ok}; \_\ ,\ \mathtt{ko}; \_\}$ receiving one out of a set of messages input/external choice
- $\mathsf{S}\,!\,\{\mathtt{ok}; \_\ ,\ \mathtt{ko}; \_\}$ sending one out of a set of messages output/internal choice

Global and Session Types

Global type of the session (where B1, B2 and S stand for Buyer1, Buyer2 and Seller) is

$$\begin{array}{l} \mathsf{B1} \to \mathsf{S} : \texttt{title}; \\ \mathsf{S} \to \mathsf{B1} : \texttt{quote}; \mathsf{S} \to \mathsf{B2} : \texttt{quote}; \\ \mathsf{B1} \to \mathsf{B2} : \texttt{yourQuote}; \\ \mathsf{B2} \to \mathsf{S} : \{\texttt{ok}; \mathsf{B2} \to \mathsf{S} : \texttt{address}; \mathsf{S} \to \mathsf{B2} : \texttt{date}; \mathbf{End}\ ,\ \texttt{ko}; \mathbf{End}\} \end{array}$$

Session types of participants: obtained by projection from the global type.

$$\mathsf{T}_{\mathsf{B1}} = \mathsf{S}\,!\,\texttt{title}; \mathsf{S}\,?\,\texttt{quote}; \mathsf{B2}\,!\,\texttt{yourQuote}; \mathbf{End}$$

$$\mathsf{T}_{\mathsf{S}} = \begin{array}{l} \mathsf{B1}\,?\,\texttt{title}; \\ \mathsf{B1}\,!\,\texttt{quote}; \mathsf{B2}\,!\,\texttt{quote}; \\ \mathsf{B2}\,?\,\{\texttt{ok}; \mathsf{B2}\,?\,\texttt{address}; \mathsf{B2}\,!\,\texttt{date}; \mathbf{End}\ ,\ \texttt{ko}; \mathbf{End}\} \end{array}$$

$$\mathsf{T}_{\mathsf{B2}} = \begin{array}{l} \mathsf{S}\,?\,\texttt{quote}; \\ \mathsf{B1}\,?\,\texttt{yourQuote}; \\ \mathsf{S}\,!\,\{\texttt{ok}; \mathsf{S}\,!\,\texttt{address}; \mathsf{S}\,?\,\texttt{date}; \mathbf{End}\ ,\ \texttt{ko}; \mathbf{End}\} \end{array}$$

- $\mathsf{B2}\,?\,\{\texttt{ok}; \_\ ,\ \texttt{ko}; \_\}$ receiving one out of a set of messages input/external choice
- $\mathsf{S}\,!\,\{\texttt{ok}; \_\ ,\ \texttt{ko}; \_\}$ sending one out of a set of messages output/internal choice

Global and Session Types

Global type of the session (where B1, B2 and S stand for Buyer1, Buyer2 and Seller) is

$$\begin{array}{l} \mathsf{B1} \to \mathsf{S} : \mathtt{title}; \\ \mathsf{S} \to \mathsf{B1} : \mathtt{quote}; \mathsf{S} \to \mathsf{B2} : \mathtt{quote}; \\ \mathsf{B1} \to \mathsf{B2} : \mathtt{yourQuote}; \\ \mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array}$$

Session types of participants: obtained by projection from the global type.

$$\mathsf{T}_{\mathsf{B1}} = \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \mathbf{End}$$

$$\mathsf{T}_{\mathsf{S}} = \begin{array}{l} \mathsf{B1}\,?\,\mathtt{title}; \\ \mathsf{B1}\,!\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{quote}; \\ \mathsf{B2}\,?\,\{\mathtt{ok}; \mathsf{B2}\,?\,\mathtt{address}; \mathsf{B2}\,!\,\mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array}$$

$$\mathsf{T}_{\mathsf{B2}} = \begin{array}{l} \mathsf{S}\,?\,\mathtt{quote}; \\ \mathsf{B1}\,?\,\mathtt{yourQuote}; \\ \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array}$$

- B2 ? {ok; _ , ko; _} receiving one out of a set of messages input/external choice
- S ! {ok; _ , ko; _} sending one out of a set of messages output/internal choice

Global and Session Types

Global type of the session (where B1, B2 and S stand for Buyer1, Buyer2 and Seller) is

$$\begin{array}{l} \mathsf{B1} \to \mathsf{S} : \mathtt{title}; \\ \mathsf{S} \to \mathsf{B1} : \mathtt{quote}; \mathsf{S} \to \mathsf{B2} : \mathtt{quote}; \\ \mathsf{B1} \to \mathsf{B2} : \mathtt{yourQuote}; \\ \mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array}$$

Session types of participants: obtained by projection from the global type.

$$T_{\mathsf{B1}} = \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \mathbf{End}$$

$$T_{\mathsf{S}} = \begin{array}{l} \mathsf{B1}\,?\,\mathtt{title}; \\ \mathsf{B1}\,!\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{quote}; \\ \mathsf{B2}\,?\,\{\mathtt{ok}; \mathsf{B2}\,?\,\mathtt{address}; \mathsf{B2}\,!\,\mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array}$$

$$T_{\mathsf{B2}} = \begin{array}{l} \mathsf{S}\,?\,\mathtt{quote}; \\ \mathsf{B1}\,?\,\mathtt{yourQuote}; \\ \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array}$$

- $\mathsf{B2}\,?\,\{\mathtt{ok}; \_\ ,\ \mathtt{ko}; \_\}$ receiving one out of a set of messages input/external choice
- $\mathsf{S}\,!\,\{\mathtt{ok}; \_\ ,\ \mathtt{ko}; \_\}$ sending one out of a set of messages output/internal choice

Global and Session Types

Global type of the session (where B1, B2 and S stand for Buyer1, Buyer2 and Seller) is

$$\begin{array}{l} \mathsf{B1} \to \mathsf{S} : \mathtt{title}; \\ \mathsf{S} \to \mathsf{B1} : \mathtt{quote}; \mathsf{S} \to \mathsf{B2} : \mathtt{quote}; \\ \mathsf{B1} \to \mathsf{B2} : \mathtt{yourQuote}; \\ \mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathbf{End} \ , \ \mathtt{ko}; \mathbf{End}\} \end{array}$$

Session types of participants: obtained by projection from the global type.

$$\begin{array}{lcl} \mathsf{T}_{\mathsf{B1}} & = & \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \mathbf{End} \\ \\ & & \mathsf{B1}\,?\,\mathtt{title}; \\ \mathsf{T}_{\mathsf{S}} & = & \mathsf{B1}\,!\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{quote}; \\ & & \mathsf{B2}\,?\,\{\mathtt{ok}; \mathsf{B2}\,?\,\mathtt{address}; \mathsf{B2}\,!\,\mathtt{date}; \mathbf{End} \ , \ \mathtt{ko}; \mathbf{End}\} \\ \\ & & \mathsf{S}\,?\,\mathtt{quote}; \\ \mathsf{T}_{\mathsf{B2}} & = & \mathsf{B1}\,?\,\mathtt{yourQuote}; \\ & & \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathbf{End} \ , \ \mathtt{ko}; \mathbf{End}\} \end{array}$$

- $\mathsf{B2}\,?\,\{\mathtt{ok};\ \_\ ,\ \mathtt{ko};\ \_\}$ receiving one out of a set of messages input/external choice
- $\mathsf{S}\,!\,\{\mathtt{ok};\ \_\ ,\ \mathtt{ko};\ \_\}$ sending one out of a set of messages output/internal choice

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Global and Session Types

Global type of the session (where B1, B2 and S stand for Buyer1, Buyer2 and Seller) is

$$\begin{array}{l} \mathsf{B1} \to \mathsf{S} : \mathtt{title}; \\ \mathsf{S} \to \mathsf{B1} : \mathtt{quote}; \mathsf{S} \to \mathsf{B2} : \mathtt{quote}; \\ \mathsf{B1} \to \mathsf{B2} : \mathtt{yourQuote}; \\ \mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array}$$

Session types of participants: obtained by projection from the global type.

$$\begin{array}{lcl} \mathsf{T}_{\mathsf{B1}} & = & \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \mathbf{End} \\ \\ \mathsf{T}_{\mathsf{S}} & = & \begin{array}{l} \mathsf{B1}\,?\,\mathtt{title}; \\ \mathsf{B1}\,!\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{quote}; \\ \mathsf{B2}\,?\,\{\mathtt{ok}; \mathsf{B2}\,?\,\mathtt{address}; \mathsf{B2}\,!\,\mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array} \\ \\ \mathsf{T}_{\mathsf{B2}} & = & \begin{array}{l} \mathsf{S}\,?\,\mathtt{quote}; \\ \mathsf{B1}\,?\,\mathtt{yourQuote}; \\ \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathbf{End}\ ,\ \mathtt{ko}; \mathbf{End}\} \end{array} \end{array}$$

- $\mathsf{B2}\,?\,\{\mathtt{ok}; \_\ ,\ \mathtt{ko}; \_\}$ receiving one out of a set of messages input/external choice
- $\mathsf{S}\,!\,\{\mathtt{ok}; \_\ ,\ \mathtt{ko}; \_\}$ sending one out of a set of messages output/internal choice

Projection

p, q, r participant names λ message label

- Global types

$$\mathsf{G} \quad ::=_{\rho} \quad \mathsf{p} \to \mathsf{q}{:}\{\lambda_i; \mathsf{G}_i\}_{i \in I} \mid \mathsf{End}$$

  where $I \neq \emptyset$, $\mathsf{p} \neq \mathsf{q}$ and $\lambda_j \neq \lambda_h$ for $j \neq h$.
  Coinductive definition. Only regular terms.

- Session types

$$\mathsf{T} \quad ::=_{\rho} \quad \mathsf{q}\,!\,\{\lambda_i; \mathsf{T}_i\}_{i \in N} \mid \mathsf{p}\,?\,\{\lambda_i; \mathsf{T}_i\}_{i \in N} \mid \mathsf{End}$$

- Projection

  - $(\mathsf{p} \to \mathsf{q}{:}\{\lambda_i; \mathsf{G}_i\}_{i \in I}) \upharpoonright \mathsf{r} = \begin{cases} \mathsf{q}\,!\,\{\lambda_i; \mathsf{G}_i \upharpoonright \mathsf{r}\}_{i \in I} & \text{if } \mathsf{r} = \mathsf{p} \neq \mathsf{q}, \\ \mathsf{p}\,?\,\{\lambda_i; \mathsf{G}_i \upharpoonright \mathsf{r}\}_{i \in I} & \text{if } \mathsf{r} = \mathsf{q} \neq \mathsf{p}, \\ \mathsf{G}_1 \upharpoonright \mathsf{r} & \text{if } \mathsf{r} \neq \mathsf{p} \text{ and } \mathsf{r} \neq \mathsf{q} \\ & \forall i, j \in I.\ \mathsf{G}_i \upharpoonright \mathsf{r} = \mathsf{G}_j \upharpoonright \mathsf{r} \end{cases}$
  - $\mathsf{End} \upharpoonright \mathsf{r} = \mathsf{End}$

Projection

p, q, r participant names λ message label

- Global types

$$G \quad ::=_\rho \quad p \to q{:}\{\lambda_i; G_i\}_{i \in I} \mid \text{End}$$

where $I \neq \emptyset$, $p \neq q$ and $\lambda_j \neq \lambda_h$ for $j \neq h$.
Coinductive definition. Only regular terms.

- Session types

$$T \quad ::=_\rho \quad q\,!\,\{\lambda_i; T_i\}_{i \in N} \mid p\,?\,\{\lambda_i; T_i\}_{i \in N} \mid \text{End}$$

- Projection

  - $$(p \to q{:}\{\lambda_i; G_i\}_{i \in I}) \upharpoonright r = \begin{cases} q\,!\,\{\lambda_i; G_i \upharpoonright r\}_{i \in I} & \text{if } r = p \neq q, \\ p\,?\,\{\lambda_i; G_i \upharpoonright r\}_{i \in I} & \text{if } r = q \neq p, \\ G_1 \upharpoonright r & \text{if } r \neq p \text{ and } r \neq q \\ & \forall i, j \in I \; G_i \upharpoonright r = G_j \upharpoonright r \end{cases}$$
  - $\text{End} \upharpoonright r = \text{End}$

Projection

p, q, r participant names $\qquad$ λ message label

- Global types

$$\mathsf{G} \quad ::=_\rho \quad \mathsf{p} \to \mathsf{q}{:}\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{End}$$

  where $I \neq \emptyset$, $\mathsf{p} \neq \mathsf{q}$ and $\lambda_j \neq \lambda_h$ for $j \neq h$.
  Coinductive definition. Only regular terms.

- Session types

$$\mathsf{T} \quad ::=_\rho \quad \mathsf{q}\,!\,\{\lambda_i; \mathsf{T}_i\}_{i\in N} \mid \mathsf{p}\,?\,\{\lambda_i; \mathsf{T}_i\}_{i\in N} \mid \mathsf{End}$$

- Projection
  - $(\mathsf{p} \to \mathsf{q}{:}\{\lambda_i; \mathsf{G}_i\}_{i\in I}) \restriction \mathsf{r} = \begin{cases} \mathsf{q}\,!\,\{\lambda_i; \mathsf{G}_i \restriction \mathsf{r}\}_{i\in I} & \text{if } \mathsf{r} = \mathsf{p} \neq \mathsf{q}, \\ \mathsf{p}\,?\,\{\lambda_i; \mathsf{G}_i \restriction \mathsf{r}\}_{i\in I} & \text{if } \mathsf{r} = \mathsf{q} \neq \mathsf{p}, \\ \mathsf{G}_1 \restriction \mathsf{r} & \text{if } \mathsf{r} \neq \mathsf{p} \text{ and } \mathsf{r} \neq \mathsf{q} \\ & \forall i,j \in I\ \mathsf{G}_i \restriction \mathsf{r} = \mathsf{G}_j \restriction \mathsf{r} \end{cases}$
  - $\mathsf{End} \restriction \mathsf{r} = \mathsf{End}$

Projection

p, q, r participant names λ message label

- Global types

$$\mathsf{G} \quad ::=_\rho \quad \mathsf{p} \to \mathsf{q}{:}\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{End}$$

  where $I \neq \emptyset$, $\mathsf{p} \neq \mathsf{q}$ and $\lambda_j \neq \lambda_h$ for $j \neq h$.
  Coinductive definition. Only regular terms.

- Session types

$$\mathsf{T} \quad ::=_\rho \quad \mathsf{q}\,!\,\{\lambda_i; \mathsf{T}_i\}_{i\in N} \mid \mathsf{p}\,?\,\{\lambda_i; \mathsf{T}_i\}_{i\in N} \mid \mathsf{End}$$

- Projection

  - $(\mathsf{p} \to \mathsf{q}{:}\{\lambda_i; \mathsf{G}_i\}_{i\in I})\upharpoonright \mathsf{r} = \begin{cases} \mathsf{q}\,!\,\{\lambda_i; \mathsf{G}_i\upharpoonright\mathsf{r}\}_{i\in I} & \text{if } \mathsf{r} = \mathsf{p} \neq \mathsf{q}, \\ \mathsf{p}\,?\,\{\lambda_i; \mathsf{G}_i\upharpoonright\mathsf{r}\}_{i\in I} & \text{if } \mathsf{r} = \mathsf{q} \neq \mathsf{p}, \\ \mathsf{G}_1\upharpoonright\mathsf{r} & \text{if } \mathsf{r} \neq \mathsf{p} \text{ and } \mathsf{r} \neq \mathsf{q} \\ & \forall i,j \in I\ \mathsf{G}_i\upharpoonright\mathsf{r} = \mathsf{G}_j\upharpoonright\mathsf{r} \end{cases}$
  - $\mathsf{End}\upharpoonright\mathsf{r} = \mathsf{End}$

Projection

p, q, r participant names λ message label

- Global types

$$G \quad ::=_\rho \quad p \to q{:}\{\lambda_i; G_i\}_{i\in I} \mid \text{End}$$

  where $I \neq \emptyset$, $p \neq q$ and $\lambda_j \neq \lambda_h$ for $j \neq h$.
  Coinductive definition. Only regular terms.

- Session types

$$T \quad ::=_\rho \quad q\,!\,\{\lambda_i; T_i\}_{i\in N} \mid p\,?\,\{\lambda_i; T_i\}_{i\in N} \mid \text{End}$$

- Projection

  - $$(p \to q{:}\{\lambda_i; G_i\}_{i\in I}) \upharpoonright r = \begin{cases} q\,!\,\{\lambda_i; G_i \upharpoonright r\}_{i\in I} & \text{if } r = p \neq q, \\ p\,?\,\{\lambda_i; G_i \upharpoonright r\}_{i\in I} & \text{if } r = q \neq p, \\ G_1 \upharpoonright r & \text{if } r \neq p \text{ and } r \neq q \\ & \forall i,j \in I\ G_i \upharpoonright r = G_j \upharpoonright r \end{cases}$$

  - $\text{End} \upharpoonright r = \text{End}$

Projection

p, q, r participant names λ message label

- Global types

$$G \quad ::=_\rho \quad \mathsf{p} \to \mathsf{q}{:}\{\lambda_i; G_i\}_{i \in I} \mid \mathsf{End}$$

where $I \neq \emptyset$, $\mathsf{p} \neq \mathsf{q}$ and $\lambda_j \neq \lambda_h$ for $j \neq h$.
Coinductive definition. Only regular terms.

- Session types

$$T \quad ::=_\rho \quad \mathsf{q}\,!\,\{\lambda_i; T_i\}_{i \in N} \mid \mathsf{p}\,?\,\{\lambda_i; T_i\}_{i \in N} \mid \mathsf{End}$$

- Projection
  - $$(\mathsf{p} \to \mathsf{q}{:}\{\lambda_i; G_i\}_{i \in I}) \restriction \mathsf{r} = \begin{cases} \mathsf{q}\,!\,\{\lambda_i; G_i \restriction \mathsf{r}\}_{i \in I} & \text{if } \mathsf{r} = \mathsf{p} \neq \mathsf{q}, \\ \mathsf{p}\,?\,\{\lambda_i; G_i \restriction \mathsf{r}\}_{i \in I} & \text{if } \mathsf{r} = \mathsf{q} \neq \mathsf{p}, \\ G_1 \restriction \mathsf{r} & \text{if } \mathsf{r} \neq \mathsf{p} \text{ and } \mathsf{r} \neq \mathsf{q} \\ & \forall i, j \in I \; G_i \restriction \mathsf{r} = G_j \restriction \mathsf{r} \end{cases}$$
  - $\mathsf{End} \restriction \mathsf{r} = \mathsf{End}$

Role of Projection

- Projectability of global types on all participants ensures realisability of the protocol.
- Crucial is projection of a choice on participants different from sender and receiver.

Example

Assume we add $\mathsf{B2} \to \mathsf{B1} : \mathtt{ko}$ in the branch $\mathtt{ko}$ of the choice

$$\ldots;$$
$$\mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathsf{End} \ ,\ \mathtt{ko}; \mathsf{B2} \to \mathsf{B1} : \mathtt{ko}; \mathsf{End}\}$$

This protocol is not realisable:

$$\mathsf{T}_{\mathsf{B2}} = \begin{array}{l} \mathsf{S}\,?\,\mathtt{quote};\\ \mathsf{B1}\,?\,\mathtt{yourQuote};\\ \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathsf{End} \ ,\ \mathtt{ko}; \mathsf{B1}\,!\,\mathtt{ko}; \mathsf{End}\} \end{array}$$

$$\mathsf{T}_{\mathsf{B1}} = \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \mathsf{B2}\,?\,\mathtt{ko}; \mathsf{End}$$

- More flexible projections have been proposed!
- We only consider G projectable on all participants.

Role of Projection

- Projectability of global types on all participants ensures realisability of the protocol.
- Crucial is projection of a choice on participants different from sender and receiver.

Example

Assume we add $\mathsf{B2} \to \mathsf{B1} : \mathtt{ko}$ in the branch $\mathtt{ko}$ of the choice

$$\ldots;$$
$$\mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathsf{End}\ ,\ \mathtt{ko}; \mathsf{B2} \to \mathsf{B1} : \mathtt{ko}; \mathsf{End}\}$$

This protocol is not realisable:

$$\mathsf{T}_{\mathsf{B2}} = \begin{array}{l} \mathsf{S}\,?\,\mathtt{quote}; \\ \mathsf{B1}\,?\,\mathtt{yourQuote}; \\ \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathsf{End}\ ,\ \mathtt{ko}; \mathsf{B1}\,!\,\mathtt{ko}; \mathsf{End}\} \end{array}$$

$$\mathsf{T}_{\mathsf{B1}} = \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \sout{\mathsf{B2}\,?\,\mathtt{ko}}; \mathsf{End}$$

- More flexible projections have been proposed!
- We only consider G projectable on all participants.

Role of Projection

- Projectability of global types on all participants ensures realisability of the protocol.
- Crucial is projection of a choice on participants different from sender and receiver.

Example

Assume we add $\mathsf{B2} \to \mathsf{B1} : \mathtt{ko}$ in the branch $\mathtt{ko}$ of the choice

$$\ldots;$$
$$\mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathsf{End}\ ,\ \mathtt{ko}; \mathsf{B2} \to \mathsf{B1} : \mathtt{ko}; \mathsf{End}\}$$

This protocol is not realisable:

$$\mathsf{T}_{\mathsf{B2}} = \begin{array}{l} \mathsf{S}\,?\,\mathtt{quote}; \\ \mathsf{B1}\,?\,\mathtt{yourQuote}; \\ \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathsf{End}\ ,\ \mathtt{ko}; \mathsf{B1}\,!\,\mathtt{ko}; \mathsf{End}\} \end{array}$$

$$\mathsf{T}_{\mathsf{B1}} = \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \sout{\mathsf{B2}\,?\,\mathtt{ko}}; \mathsf{End}$$

- More flexible projections have been proposed!
- We only consider G projectable on all participants.

Role of Projection

- Projectability of global types on all participants ensures realisability of the protocol.
- Crucial is projection of a choice on participants different from sender and receiver.

Example

Assume we add $\mathsf{B2} \to \mathsf{B1} : \texttt{ko}$ in the branch $\texttt{ko}$ of the choice

$$\ldots;$$
$$\mathsf{B2} \to \mathsf{S} : \{\texttt{ok}; \mathsf{B2} \to \mathsf{S} : \texttt{address}; \mathsf{S} \to \mathsf{B2} : \texttt{date}; \mathsf{End}\ ,\ \texttt{ko}; \mathsf{B2} \to \mathsf{B1} : \texttt{ko}; \mathsf{End}\}$$

This protocol is not realisable:

$$\mathsf{T}_{\mathsf{B2}} = \begin{array}{l} \mathsf{S}\,?\,\texttt{quote}; \\ \mathsf{B1}\,?\,\texttt{yourQuote}; \\ \mathsf{S}\,!\,\{\texttt{ok}; \mathsf{S}\,!\,\texttt{address}; \mathsf{S}\,?\,\texttt{date}; \mathsf{End}\ ,\ \texttt{ko}; \mathsf{B1}\,!\,\texttt{ko}; \mathsf{End}\} \end{array}$$

$$\mathsf{T}_{\mathsf{B1}} = \mathsf{S}\,!\,\texttt{title}; \mathsf{S}\,?\,\texttt{quote}; \mathsf{B2}\,!\,\texttt{yourQuote}; \cancel{\mathsf{B2}\,?\,\texttt{ko}}; \mathsf{End}$$

- More flexible projections have been proposed!
- We only consider G projectable on all participants.

Role of Projection

- Projectability of global types on all participants ensures realisability of the protocol.
- Crucial is projection of a choice on participants different from sender and receiver.

Example

Assume we add $\mathsf{B2} \to \mathsf{B1} : \mathtt{ko}$ in the branch $\mathtt{ko}$ of the choice

$$\ldots ;$$
$$\mathsf{B2} \to \mathsf{S} : \{\mathtt{ok}; \mathsf{B2} \to \mathsf{S} : \mathtt{address}; \mathsf{S} \to \mathsf{B2} : \mathtt{date}; \mathsf{End}\ ,\ \mathtt{ko}; \mathsf{B2} \to \mathsf{B1} : \mathtt{ko}; \mathsf{End}\}$$

This protocol is not realisable:

$$\mathsf{T}_{\mathsf{B2}} = \begin{array}{l} \mathsf{S}\,?\,\mathtt{quote}; \\ \mathsf{B1}\,?\,\mathtt{yourQuote}; \\ \mathsf{S}\,!\,\{\mathtt{ok}; \mathsf{S}\,!\,\mathtt{address}; \mathsf{S}\,?\,\mathtt{date}; \mathsf{End}\ ,\ \mathtt{ko}; \mathsf{B1}\,!\,\mathtt{ko}; \mathsf{End}\} \end{array}$$

$$\mathsf{T}_{\mathsf{B1}} = \mathsf{S}\,!\,\mathtt{title}; \mathsf{S}\,?\,\mathtt{quote}; \mathsf{B2}\,!\,\mathtt{yourQuote}; \sout{\mathsf{B2}\,?\,\mathtt{ko}}; \mathsf{End}$$

- More flexible projections have been proposed!
- We only consider G projectable on all participants.

Processes and Queues

- We focus on the core message-passing aspects of asynchronous multiparty sessions. We can define processes as session types.

$$P \quad ::=_\rho \quad \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \mid \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i\in I} \mid 0$$

- Projection of a global type onto a participant defined changing $\mathsf{End}\restriction \mathsf{r} = \mathsf{End}$ with $\mathsf{End}\restriction \mathsf{r} = 0$
- To hold messages in transit we use a queue defined by:

$$\mathcal{M} ::= \emptyset \mid \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \mathcal{M}$$

Order between messages matters only for messages with the same sender and receiver. We consider queues modulo the following structural equivalence:

$$\mathcal{M}\cdot\langle \mathsf{p}, \lambda, \mathsf{q}\rangle\cdot\langle \mathsf{r}, \lambda', \mathsf{s}\rangle\cdot\mathcal{M}' \equiv \mathcal{M}\cdot\langle \mathsf{r}, \lambda', \mathsf{s}\rangle\cdot\langle \mathsf{p}, \lambda, \mathsf{q}\rangle\cdot\mathcal{M}' \quad \text{if} \quad \mathsf{p}\neq\mathsf{r} \quad \text{or} \quad \mathsf{q}\neq\mathsf{s}$$

Processes and Queues

- We focus on the core message-passing aspects of asynchronous multiparty sessions. We can define processes as session types.

$$P \quad ::=_\rho \quad \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i \in I} \mid \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i \in I} \mid 0$$

- Projection of a global type onto a participant defined changing $\mathsf{End} \upharpoonright \mathsf{r} = \mathsf{End}$ with $\mathsf{End} \upharpoonright \mathsf{r} = 0$
- To hold messages in transit we use a queue defined by:

$$\mathcal{M} ::= \emptyset \mid \langle \mathsf{p}, \lambda, \mathsf{q} \rangle \cdot \mathcal{M}$$

Order between messages matters only for messages with the same sender and receiver. We consider queues modulo the following structural equivalence:

$$\mathcal{M} \cdot \langle \mathsf{p}, \lambda, \mathsf{q} \rangle \cdot \langle \mathsf{r}, \lambda', \mathsf{s} \rangle \cdot \mathcal{M}' \equiv \mathcal{M} \cdot \langle \mathsf{r}, \lambda', \mathsf{s} \rangle \cdot \langle \mathsf{p}, \lambda, \mathsf{q} \rangle \cdot \mathcal{M}' \quad \text{if} \quad \mathsf{p} \neq \mathsf{r} \quad \text{or} \quad \mathsf{q} \neq \mathsf{s}$$

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Processes and Queues

- We focus on the core message-passing aspects of asynchronous multiparty sessions. We can define processes as session types.

$$P \quad ::=_\rho \quad \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \mid \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i\in I} \mid 0$$

- Projection of a global type onto a participant defined changing $\mathsf{End}\upharpoonright \mathsf{r} = \mathsf{End}$ with $\mathsf{End}\upharpoonright \mathsf{r} = 0$
- To hold messages in transit we use a queue defined by:

$$\mathcal{M} ::= \emptyset \mid \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \mathcal{M}$$

Order between messages matters only for messages with the same sender and receiver. We consider queues modulo the following structural equivalence:

$$\mathcal{M} \cdot \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \langle \mathsf{r}, \lambda', \mathsf{s}\rangle \cdot \mathcal{M}' \equiv \mathcal{M} \cdot \langle \mathsf{r}, \lambda', \mathsf{s}\rangle \cdot \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \mathcal{M}' \quad \text{if} \quad \mathsf{p} \neq \mathsf{r} \quad \text{or} \quad \mathsf{q} \neq \mathsf{s}$$

Processes and Queues

- We focus on the core message-passing aspects of asynchronous multiparty sessions. We can define processes as session types.

$$P \quad ::=_\rho \quad \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \mid \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i\in I} \mid 0$$

- Projection of a global type onto a participant defined changing $\mathsf{End} \restriction \mathsf{r} = \mathsf{End}$ with $\mathsf{End} \restriction \mathsf{r} = 0$
- To hold messages in transit we use a queue defined by:

$$\mathcal{M} ::= \emptyset \mid \langle \mathsf{p}, \lambda, \mathsf{q} \rangle \cdot \mathcal{M}$$

Order between messages matters only for messages with the same sender and receiver. We consider queues modulo the following structural equivalence:

$$\mathcal{M} \cdot \langle \mathsf{p}, \lambda, \mathsf{q} \rangle \cdot \langle \mathsf{r}, \lambda', \mathsf{s} \rangle \cdot \mathcal{M}' \equiv \mathcal{M} \cdot \langle \mathsf{r}, \lambda', \mathsf{s} \rangle \cdot \langle \mathsf{p}, \lambda, \mathsf{q} \rangle \cdot \mathcal{M}' \quad \text{if} \quad \mathsf{p} \neq \mathsf{r} \quad \text{or} \quad \mathsf{q} \neq \mathsf{s}$$

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Multiparty Sessions Semantics

- A network $\mathbb{N}$ is a parallel composition of located processes

$$\mathbb{N} ::= \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!]$$

  where $n > 0$ and $\mathsf{p}_i \neq \mathsf{p}_j$ for $i \neq j$.

- A multiparty session is

$$\mathbb{N} \parallel \mathcal{M}$$

- Labelled Transition System

$$[\text{Send}]\quad \mathsf{p}[\![\, \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}\,!\,\lambda_h} \mathsf{p}[\![\, P_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \cdot \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \quad h \in I$$

$$[\text{Rcv}]\quad \mathsf{q}[\![\, \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}\,?\,\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad h \in I$$

Multiparty Sessions Semantics

- A network $\mathbb{N}$ is a parallel composition of located processes

$$\mathbb{N} ::= \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!]$$

  where $n > 0$ and $\mathsf{p}_i \neq \mathsf{p}_j$ for $i \neq j$.

- A multiparty session is

$$\mathbb{N} \parallel \mathcal{M}$$

- Labelled Transition System

$$[\text{Send}] \quad \mathsf{p}[\![\, \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}!\lambda_h} \mathsf{p}[\![\, P_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \cdot \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \quad h \in I$$

$$[\text{Rcv}] \quad \mathsf{q}[\![\, \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad h \in I$$

Multiparty Sessions Semantics

- A network $\mathbb{N}$ is a parallel composition of located processes

$$\mathbb{N} ::= \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!]$$

where $n > 0$ and $\mathsf{p}_i \neq \mathsf{p}_j$ for $i \neq j$.

- A multiparty session is

$$\mathbb{N} \parallel \mathcal{M}$$

- Labelled Transition System

$$[\text{Send}] \quad \mathsf{p}[\![\, \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}!\lambda_h} \mathsf{p}[\![\, P_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \cdot \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \quad h \in I$$

$$[\text{Rcv}] \; \mathsf{q}[\![\, \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad h \in I$$

Multiparty Sessions Semantics

- A network $\mathbb{N}$ is a parallel composition of located processes

$$\mathbb{N} ::= \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!]$$

  where $n > 0$ and $\mathsf{p}_i \neq \mathsf{p}_j$ for $i \neq j$.

- A multiparty session is

$$\mathbb{N} \parallel \mathcal{M}$$

- Labelled Transition System

$$[\text{Send}] \quad \mathsf{p}[\![\, \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}!\lambda_h} \mathsf{p}[\![\, P_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \cdot \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \quad h \in I$$

$$[\text{Rcv}] \quad \mathsf{q}[\![\, \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad h \in I$$

Multiparty Sessions Semantics

- A network $\mathbb{N}$ is a parallel composition of located processes

$$\mathbb{N} ::= \mathsf{p}_1[\![\, P_1 \,]\!] \parallel \cdots \parallel \mathsf{p}_n[\![\, P_n \,]\!]$$

where $n > 0$ and $\mathsf{p}_i \neq \mathsf{p}_j$ for $i \neq j$.

- A multiparty session is

$$\mathbb{N} \parallel \mathcal{M}$$

- Labelled Transition System

$$[\text{Send}] \quad \mathsf{p}[\![\, \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}!\lambda_h} \mathsf{p}[\![\, P_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \cdot \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \quad h \in I$$

$$[\text{Rcv}] \; \mathsf{q}[\![\, \mathsf{p}\,?\,\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad h \in I$$

Progress Property

A multiparty session $\mathbb{N} \parallel \mathcal{M}$ has the progress property iff it has

- no deadlocks
  all derivatives of $\mathbb{N} \parallel \mathcal{M}$ are
  - either terminated, i.e., $\mathbb{N} \equiv \mathsf{p}[\![\,0\,]\!]$ and $\mathcal{M} \equiv \emptyset$
  - or live, i.e. if $\mathbb{N} \parallel \mathcal{M} \xrightarrow{\beta}$ for some β;
- no locked inputs
  all inputs will eventually be satisfied
- no orphan messages
  all messages in the queue will eventually be read

Progress Property

A multiparty session $\mathbb{N} \parallel \mathcal{M}$ has the progress property iff it has

- no deadlocks
  all derivatives of $\mathbb{N} \parallel \mathcal{M}$ are
  - either terminated, i.e., $\mathbb{N} \equiv \mathsf{p}[\![\, 0 \,]\!]$ and $\mathcal{M} = \emptyset$
  - or live, i.e. if $\mathbb{N} \parallel \mathcal{M} \xrightarrow{\beta}$ for some β;
- no locked inputs
  all inputs will eventually be satisfied
- no orphan messages
  all messages in the queue will eventually be read

Progress Property

A multiparty session $\mathbb{N} \parallel \mathcal{M}$ has the progress property iff it has

- no deadlocks
  all derivatives of $\mathbb{N} \parallel \mathcal{M}$ are
  - either terminated, i.e., $\mathbb{N} \equiv \mathsf{p}[\![\, 0 \,]\!]$ and $\mathcal{M} = \emptyset$
  - or live, i.e. if $\mathbb{N} \parallel \mathcal{M} \xrightarrow{\beta}$ for some β;
- no locked inputs
  all inputs will eventually be satisfied
- no orphan messages
  all messages in the queue will eventually be read

Progress Property

A multiparty session $\mathbb{N} \parallel \mathcal{M}$ has the progress property iff it has

- no deadlocks
  all derivatives of $\mathbb{N} \parallel \mathcal{M}$ are
  - either terminated, i.e., $\mathbb{N} \equiv \mathsf{p}[\![\, 0 \,]\!]$ and $\mathcal{M} = \emptyset$
  - or live, i.e. if $\mathbb{N} \parallel \mathcal{M} \xrightarrow{\beta}$ for some β;
- no locked inputs
  all inputs will eventually be satisfied
- no orphan messages
  all messages in the queue will eventually be read

Progress Property

A multiparty session $\mathbb{N} \parallel \mathcal{M}$ has the progress property iff it has

- no deadlocks
  all derivatives of $\mathbb{N} \parallel \mathcal{M}$ are
  - either terminated, i.e., $\mathbb{N} \equiv \mathsf{p}[\![\, 0 \,]\!]$ and $\mathcal{M} = \emptyset$
  - or live, i.e. if $\mathbb{N} \parallel \mathcal{M} \xrightarrow{\beta}$ for some β;
- no locked inputs
  all inputs will eventually be satisfied
- no orphan messages
  all messages in the queue will eventually be read

Progress Property

A multiparty session $\mathbb{N} \parallel \mathcal{M}$ has the progress property iff it has

- no deadlocks
  all derivatives of $\mathbb{N} \parallel \mathcal{M}$ are
  - either terminated, i.e., $\mathbb{N} \equiv \mathsf{p}[\![\, 0 \,]\!]$ and $\mathcal{M} = \emptyset$
  - or live, i.e. if $\mathbb{N} \parallel \mathcal{M} \xrightarrow{\beta}$ for some β;
- no locked inputs
  all inputs will eventually be satisfied
- no orphan messages
  all messages in the queue will eventually be read

Well-typed Initial Multiparty Session

- Well-typed Networks

$$[\text{I-Net}]\ \frac{P_i \leq \mathsf{G}\upharpoonright \mathsf{p}_i \quad i \in I \qquad \mathsf{participants}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}}{\vdash \Pi_{i \in I}\mathsf{p}_i[\![\, P_i \,]\!] : \mathsf{G}}$$

- Subtyping

$$[\leq\text{-Out}]\ \frac{P_i \leq Q_i \qquad i \in I}{\mathsf{q}\,!\,\{\lambda_i; P_i\}_{i \in I} \leq \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i \in I \cup J}} \qquad [\leq\text{-In}]\ \frac{P_i \leq Q_i \qquad i \in I}{\mathsf{q}\,?\,\{\lambda_i; P_i\}_{i \in I \cup J} \leq \mathsf{q}\,?\,\{\lambda_i; P_i\}_{i \in I}}$$

  - Internal choices are better if they send less message labels.
  - External choices are better if they receive more input message labels.

Well-typed Initial Multiparty Session

- Well-typed Networks

$$[\text{I-Net}]\ \frac{P_i \leq \mathsf{G}\restriction \mathsf{p}_i \quad i \in I \qquad \text{participants}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}}{\vdash \Pi_{i\in I}\mathsf{p}_i[\![\, P_i \,]\!] : \mathsf{G}}$$

- Subtyping

$$[\leq\text{-Out}]\ \frac{P_i \leq Q_i \qquad i \in I}{\mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \leq \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I\cup J}} \qquad [\leq\text{-In}]\ \frac{P_i \leq Q_i \qquad i \in I}{\mathsf{q}\,?\,\{\lambda_i; P_i\}_{i\in I\cup J} \leq \mathsf{q}\,?\,\{\lambda_i; P_i\}_{i\in I}}$$

  - Internal choices are better if they send less message labels.
  - External choices are better if they receive more input message labels.

Well-typed Initial Multiparty Session

- Well-typed Networks

$$[\text{I-Net}]\ \frac{P_i \leq \mathsf{G}\restriction \mathsf{p}_i \quad i \in I \qquad \mathsf{participants}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}}{\vdash \Pi_{i\in I}\mathsf{p}_i[\![\, P_i \,]\!] : \mathsf{G}}$$

- Subtyping

$$[\leq\text{-Out}]\ \frac{P_i \leq Q_i \qquad i \in I}{\mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \leq \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I\cup J}} \qquad [\leq\text{-In}]\ \frac{P_i \leq Q_i \qquad i \in I}{\mathsf{q}\,?\,\{\lambda_i; P_i\}_{i\in I\cup J} \leq \mathsf{q}\,?\,\{\lambda_i; P_i\}_{i\in I}}$$

  - Internal choices are better if they send less message labels.
  - External choices are better if they receive more input message labels.

Well-typed Initial Multiparty Session

- Well-typed Networks

$$[\text{I-Net}]\ \frac{P_i \leq \mathsf{G}\restriction \mathsf{p}_i \quad i \in I \qquad \text{participants}(\mathsf{G}) \subseteq \{\mathsf{p}_i \mid i \in I\}}{\vdash \Pi_{i\in I}\mathsf{p}_i[\![\, P_i \,]\!] : \mathsf{G}}$$

- Subtyping

$$[\leq\text{-Out}]\ \frac{P_i \leq Q_i \qquad i \in I}{\mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I} \leq \mathsf{q}\,!\,\{\lambda_i; P_i\}_{i\in I\cup J}} \qquad [\leq\text{-In}]\ \frac{P_i \leq Q_i \qquad i \in I}{\mathsf{q}\,?\,\{\lambda_i; P_i\}_{i\in I\cup J} \leq \mathsf{q}\,?\,\{\lambda_i; P_i\}_{i\in I}}$$

  - Internal choices are better if they send less message labels.
  - External choices are better if they receive more input message labels.

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Progress of Multiparty Sessions

A global type G is bounded if all $\mathsf{p} \in \mathsf{G}$ occur at bounded depth in all paths of G (needed for no locked inputs)

Theorem

If $\vdash \mathbb{N} : \mathsf{G}$ for some bounded G and $\mathbb{N} \parallel \emptyset \longrightarrow^{*} \mathbb{N}' \parallel \mathcal{M}$ then $\mathbb{N}' \parallel \mathcal{M}$ has the progress property.

Progress of Multiparty Sessions

A global type G is bounded if all $p \in G$ occur at bounded depth in all paths of G (needed for no locked inputs)

Theorem

If $\vdash \mathbb{N} : G$ for some bounded G and $\mathbb{N} \parallel \emptyset \longrightarrow^* \mathbb{N}' \parallel \mathcal{M}$ then $\mathbb{N}' \parallel \mathcal{M}$ has the progress property.

Progress of Multiparty Sessions

A global type G is bounded if all $\mathsf{p} \in \mathsf{G}$ occur at bounded depth in all paths of G (needed for no locked inputs)

Theorem

If $\vdash \mathbb{N} : \mathsf{G}$ for some bounded G and $\mathbb{N} \parallel \emptyset \longrightarrow^* \mathbb{N}' \parallel \mathcal{M}$ then $\mathbb{N}' \parallel \mathcal{M}$ has the progress property.

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$\mathsf{p}[\![\,\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \emptyset$$

Let $\mathbb{N} = \mathsf{p}[\![\,\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!]$

$$\begin{array}{lll}\mathbb{N} \parallel \emptyset & \xrightarrow{\mathsf{p}\,\mathsf{q}!\mathtt{home}} & \mathsf{p}[\![\,\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\ & \xrightarrow{\mathsf{q}\,\mathsf{p}!\mathtt{home}} & \mathsf{p}[\![\,\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \\ & & \qquad \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \cdot \langle \mathsf{q}, \mathtt{home}, \mathsf{p}\rangle \equiv \langle \mathsf{q}, \mathtt{home}, \mathsf{p}\rangle \cdot \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\ & \xrightarrow{\mathsf{q}\,\mathsf{p}?\mathtt{home}} & \mathsf{p}[\![\,\mathbf{0}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\ & \xrightarrow{\mathsf{p}\,\mathsf{q}?\mathtt{home}} & \mathsf{p}[\![\,\mathbf{0}\,]\!] \parallel \mathsf{q}[\![\,\mathbf{0}\,]\!] \parallel \emptyset \end{array}$$

The two candidates

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home} \qquad \mathsf{G}_2 = \mathsf{q} \to \mathsf{p} : \mathtt{home}; \mathsf{p} \to \mathsf{q} : \mathtt{home}$$

fail to type $\mathbb{N}$!

$$\begin{array}{ll}\mathsf{G}_1 \restriction \mathsf{p} = \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} & \mathsf{G}_1 \restriction \mathsf{q} = \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home} \\ \mathsf{G}_2 \restriction \mathsf{p} = \mathsf{q}\,?\,\mathtt{home};\mathsf{q}\,!\,\mathtt{home} & \mathsf{G}_2 \restriction \mathsf{q} = \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\end{array}$$

Asynchronous Communications

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$p[\![\, q\,!\,\mathtt{home}; q\,?\,\mathtt{home} \,]\!] \parallel q[\![\, p\,!\,\mathtt{home}; p\,?\,\mathtt{home} \,]\!] \parallel \emptyset$$

Let $\mathbb{N} = p[\![\, q\,!\,\mathtt{home}; q\,?\,\mathtt{home} \,]\!] \parallel q[\![\, p\,!\,\mathtt{home}; p\,?\,\mathtt{home} \,]\!]$

$$\begin{array}{lll}
\mathbb{N} \parallel \emptyset & \xrightarrow{p\,q!\mathtt{home}} & p[\![\, q\,?\,\mathtt{home} \,]\!] \parallel q[\![\, p\,!\,\mathtt{home}; p\,?\,\mathtt{home} \,]\!] \parallel \langle p, \mathtt{home}, q\rangle \\
 & \xrightarrow{q\,p!\mathtt{home}} & p[\![\, q\,?\,\mathtt{home} \,]\!] \parallel q[\![\, p\,?\,\mathtt{home} \,]\!] \parallel \\
 & & \quad \langle p, \mathtt{home}, q\rangle \cdot \langle q, \mathtt{home}, p\rangle \equiv \langle q, \mathtt{home}, p\rangle \cdot \langle p, \mathtt{home}, q\rangle \\
 & \xrightarrow{q\,p?\mathtt{home}} & p[\![\, 0 \,]\!] \parallel q[\![\, p\,?\,\mathtt{home} \,]\!] \parallel \langle p, \mathtt{home}, q\rangle \\
 & \xrightarrow{p\,q?\mathtt{home}} & p[\![\, 0 \,]\!] \parallel q[\![\, 0 \,]\!] \parallel \emptyset
\end{array}$$

The two candidates

$$G_1 = p \to q : \mathtt{home}; q \to p : \mathtt{home} \qquad G_2 = q \to p : \mathtt{home}; p \to q : \mathtt{home}$$

fail to type $\mathbb{N}$!

$$\begin{array}{ll}
G_1 \restriction p = q\,!\,\mathtt{home}; q\,?\,\mathtt{home} & G_1 \restriction q = p\,?\,\mathtt{home}; p\,!\,\mathtt{home} \\
G_2 \restriction p = q\,?\,\mathtt{home}; q\,!\,\mathtt{home} & G_2 \restriction q = p\,!\,\mathtt{home}; p\,?\,\mathtt{home}
\end{array}$$

Asynchronous Communications

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$\mathsf{p}[\![\, \mathsf{q}\,!\,\mathrm{home}; \mathsf{q}\,?\,\mathrm{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathrm{home}; \mathsf{p}\,?\,\mathrm{home}\,]\!] \parallel \emptyset$$

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\mathrm{home}; \mathsf{q}\,?\,\mathrm{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathrm{home}; \mathsf{p}\,?\,\mathrm{home}\,]\!]$

$$\begin{aligned}
\mathbb{N} \parallel \emptyset \quad & \xrightarrow{\mathsf{p}\,\mathsf{q}!\mathrm{home}} \quad \mathsf{p}[\![\, \mathsf{q}\,?\,\mathrm{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathrm{home}; \mathsf{p}\,?\,\mathrm{home}\,]\!] \parallel \langle \mathsf{p}, \mathrm{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}!\mathrm{home}} \quad \mathsf{p}[\![\, \mathsf{q}\,?\,\mathrm{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathrm{home}\,]\!] \parallel \\
& \qquad\qquad \langle \mathsf{p}, \mathrm{home}, \mathsf{q} \rangle \cdot \langle \mathsf{q}, \mathrm{home}, \mathsf{p} \rangle \equiv \langle \mathsf{q}, \mathrm{home}, \mathsf{p} \rangle \cdot \langle \mathsf{p}, \mathrm{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}?\mathrm{home}} \quad \mathsf{p}[\![\, 0 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathrm{home}\,]\!] \parallel \langle \mathsf{p}, \mathrm{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{p}\,\mathsf{q}?\mathrm{home}} \quad \mathsf{p}[\![\, 0 \,]\!] \parallel \mathsf{q}[\![\, 0 \,]\!] \parallel \emptyset
\end{aligned}$$

The two candidates

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathrm{home}; \mathsf{q} \to \mathsf{p} : \mathrm{home} \qquad \mathsf{G}_2 = \mathsf{q} \to \mathsf{p} : \mathrm{home}; \mathsf{p} \to \mathsf{q} : \mathrm{home}$$

fail to type $\mathbb{N}$!

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathrm{home}; \mathsf{q}\,?\,\mathrm{home} \qquad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathrm{home}; \mathsf{p}\,!\,\mathrm{home}$$

$$\mathsf{G}_2 \upharpoonright \mathsf{p} = \mathsf{q}\,?\,\mathrm{home}; \mathsf{q}\,!\,\mathrm{home} \qquad \mathsf{G}_2 \upharpoonright \mathsf{q} = \mathsf{p}\,!\,\mathrm{home}; \mathsf{p}\,?\,\mathrm{home}$$

Asynchronous Communications

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$\mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \emptyset$$

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!]$

$$\begin{array}{lll}
\mathbb{N} \parallel \emptyset & \xrightarrow{\mathsf{p}\,\mathsf{q}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
 & \xrightarrow{\mathsf{q}\,\mathsf{p}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \\
 & & \qquad \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \cdot \langle \mathsf{q}, \mathtt{home}, \mathsf{p}\rangle \equiv \langle \mathsf{q}, \mathtt{home}, \mathsf{p}\rangle \cdot \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
 & \xrightarrow{\mathsf{q}\,\mathsf{p}?\mathtt{home}} & \mathsf{p}[\![\, 0\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
 & \xrightarrow{\mathsf{p}\,\mathsf{q}?\mathtt{home}} & \mathsf{p}[\![\, 0\,]\!] \parallel \mathsf{q}[\![\, 0\,]\!] \parallel \emptyset
\end{array}$$

The two candidates

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home} \qquad \mathsf{G}_2 = \mathsf{q} \to \mathsf{p} : \mathtt{home}; \mathsf{p} \to \mathsf{q} : \mathtt{home}$$

fail to type $\mathbb{N}$!

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \qquad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

$$\mathsf{G}_2 \upharpoonright \mathsf{p} = \mathsf{q}\,?\,\mathtt{home};\mathsf{q}\,!\,\mathtt{home} \qquad \mathsf{G}_2 \upharpoonright \mathsf{q} = \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}$$

Asynchronous Communications

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$\mathsf{p}[\![\, \mathsf{q}\,!\,\mathrm{home};\mathsf{q}\,?\,\mathrm{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathrm{home};\mathsf{p}\,?\,\mathrm{home}\,]\!] \parallel \emptyset$$

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\mathrm{home};\mathsf{q}\,?\,\mathrm{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathrm{home};\mathsf{p}\,?\,\mathrm{home}\,]\!]$

$$\begin{array}{lll}
\mathbb{N} \parallel \emptyset & \xrightarrow{\mathsf{p}\,\mathsf{q}!\mathrm{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathrm{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathrm{home};\mathsf{p}\,?\,\mathrm{home}\,]\!] \parallel \langle \mathsf{p}, \mathrm{home}, \mathsf{q}\rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}!\mathrm{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathrm{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathrm{home}\,]\!] \parallel \\
& & \qquad \langle \mathsf{p}, \mathrm{home}, \mathsf{q}\rangle \cdot \langle \mathsf{q}, \mathrm{home}, \mathsf{p}\rangle \equiv \langle \mathsf{q}, \mathrm{home}, \mathsf{p}\rangle \cdot \langle \mathsf{p}, \mathrm{home}, \mathsf{q}\rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}?\mathrm{home}} & \mathsf{p}[\![\, 0\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathrm{home}\,]\!] \parallel \langle \mathsf{p}, \mathrm{home}, \mathsf{q}\rangle \\
& \xrightarrow{\mathsf{p}\,\mathsf{q}?\mathrm{home}} & \mathsf{p}[\![\, 0\,]\!] \parallel \mathsf{q}[\![\, 0\,]\!] \parallel \emptyset
\end{array}$$

The two candidates

$$G_1 = \mathsf{p} \to \mathsf{q} : \mathrm{home}; \mathsf{q} \to \mathsf{p} : \mathrm{home} \qquad G_2 = \mathsf{q} \to \mathsf{p} : \mathrm{home}; \mathsf{p} \to \mathsf{q} : \mathrm{home}$$

fail to type $\mathbb{N}$!

$$\begin{array}{ll}
G_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathrm{home};\mathsf{q}\,?\,\mathrm{home} & G_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathrm{home};\mathsf{p}\,!\,\mathrm{home} \\
G_2 \upharpoonright \mathsf{p} = \mathsf{q}\,?\,\mathrm{home};\mathsf{q}\,!\,\mathrm{home} & G_2 \upharpoonright \mathsf{q} = \mathsf{p}\,!\,\mathrm{home};\mathsf{p}\,?\,\mathrm{home}
\end{array}$$

Asynchronous Communications

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$\mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \emptyset$$

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!]$

$$\begin{array}{lll}
\mathbb{N} \parallel \emptyset & \xrightarrow{\mathsf{p}\,\mathsf{q}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \\
& & \qquad \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \cdot \langle \mathsf{q}, \mathtt{home}, \mathsf{p}\rangle \equiv \langle \mathsf{q}, \mathtt{home}, \mathsf{p}\rangle \cdot \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}?\mathtt{home}} & \mathsf{p}[\![\, 0\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
& \xrightarrow{\mathsf{p}\,\mathsf{q}?\mathtt{home}} & \mathsf{p}[\![\, 0\,]\!] \parallel \mathsf{q}[\![\, 0\,]\!] \parallel \emptyset
\end{array}$$

The two candidates

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home} \qquad \mathsf{G}_2 = \mathsf{q} \to \mathsf{p} : \mathtt{home}; \mathsf{p} \to \mathsf{q} : \mathtt{home}$$

fail to type $\mathbb{N}$!

$$\mathsf{G}_1 \restriction \mathsf{p} = \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \qquad \mathsf{G}_1 \restriction \mathsf{q} = \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

$$\mathsf{G}_2 \restriction \mathsf{p} = \mathsf{q}\,?\,\mathtt{home};\mathsf{q}\,!\,\mathtt{home} \qquad \mathsf{G}_2 \restriction \mathsf{q} = \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}$$

Asynchronous Communications

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$\mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \emptyset$$

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home}\,]\!]$

$$\begin{array}{lll}
\mathbb{N} \parallel \emptyset & \xrightarrow{\mathsf{p}\,\mathsf{q}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
 & \xrightarrow{\mathsf{q}\,\mathsf{p}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \\
 & & \qquad \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \cdot \langle \mathsf{q}, \mathtt{home}, \mathsf{p}\rangle \equiv \langle \mathsf{q}, \mathtt{home}, \mathsf{p}\rangle \cdot \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
 & \xrightarrow{\mathsf{q}\,\mathsf{p}?\mathtt{home}} & \mathsf{p}[\![\, 0\,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathtt{home}\,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q}\rangle \\
 & \xrightarrow{\mathsf{p}\,\mathsf{q}?\mathtt{home}} & \mathsf{p}[\![\, 0\,]\!] \parallel \mathsf{q}[\![\, 0\,]\!] \parallel \emptyset
\end{array}$$

The two candidates

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home} \qquad \mathsf{G}_2 = \mathsf{q} \to \mathsf{p} : \mathtt{home}; \mathsf{p} \to \mathsf{q} : \mathtt{home}$$

fail to type $\mathbb{N}$!

$$\mathsf{G}_1 \restriction \mathsf{p} = \mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home} \qquad \mathsf{G}_1 \restriction \mathsf{q} = \mathsf{p}\,?\,\mathtt{home}; \mathsf{p}\,!\,\mathtt{home}$$

$$\mathsf{G}_2 \restriction \mathsf{p} = \mathsf{q}\,?\,\mathtt{home}; \mathsf{q}\,!\,\mathtt{home} \qquad \mathsf{G}_2 \restriction \mathsf{q} = \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home}$$

Asynchronous Communications

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$\mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home} \,]\!] \parallel \emptyset$$

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home} \,]\!]$

$$\begin{array}{lll}
\mathbb{N} \parallel \emptyset & \xrightarrow{\mathsf{p}\,\mathsf{q}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home} \,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q}\,?\,\mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathtt{home} \,]\!] \parallel \\
& & \qquad \langle \mathsf{p}, \mathtt{home}, \mathsf{q} \rangle \cdot \langle \mathsf{q}, \mathtt{home}, \mathsf{p} \rangle \equiv \langle \mathsf{q}, \mathtt{home}, \mathsf{p} \rangle \cdot \langle \mathsf{p}, \mathtt{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}?\mathtt{home}} & \mathsf{p}[\![\, \mathbf{0} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\mathtt{home} \,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{p}\,\mathsf{q}?\mathtt{home}} & \mathsf{p}[\![\, \mathbf{0} \,]\!] \parallel \mathsf{q}[\![\, \mathbf{0} \,]\!] \parallel \emptyset
\end{array}$$

The two candidates

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home} \qquad \mathsf{G}_2 = \mathsf{q} \to \mathsf{p} : \mathtt{home}; \mathsf{p} \to \mathsf{q} : \mathtt{home}$$

fail to type $\mathbb{N}$!

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home} \qquad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathtt{home}; \mathsf{p}\,!\,\mathtt{home}$$

$$\mathsf{G}_2 \upharpoonright \mathsf{p} = \mathsf{q}\,?\,\mathtt{home}; \mathsf{q}\,!\,\mathtt{home} \qquad \mathsf{G}_2 \upharpoonright \mathsf{q} = \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home}$$

Asynchronous Communications

Example

Participants p and q want to inform each other once they arrive home. Once they get home they send each other a message and wait to receive a similar one from the other.

$$\mathsf{p}[\![\, \mathsf{q} \,!\, \mathtt{home}; \mathsf{q} \,?\, \mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,!\, \mathtt{home}; \mathsf{p} \,?\, \mathtt{home} \,]\!] \parallel \emptyset$$

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q} \,!\, \mathtt{home}; \mathsf{q} \,?\, \mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,!\, \mathtt{home}; \mathsf{p} \,?\, \mathtt{home} \,]\!]$

$$\begin{array}{lll}
\mathbb{N} \parallel \emptyset & \xrightarrow{\mathsf{p}\,\mathsf{q}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q} \,?\, \mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,!\, \mathtt{home}; \mathsf{p} \,?\, \mathtt{home} \,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}!\mathtt{home}} & \mathsf{p}[\![\, \mathsf{q} \,?\, \mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,?\, \mathtt{home} \,]\!] \parallel \\
& & \qquad \langle \mathsf{p}, \mathtt{home}, \mathsf{q} \rangle \cdot \langle \mathsf{q}, \mathtt{home}, \mathsf{p} \rangle \equiv \langle \mathsf{q}, \mathtt{home}, \mathsf{p} \rangle \cdot \langle \mathsf{p}, \mathtt{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{q}\,\mathsf{p}?\mathtt{home}} & \mathsf{p}[\![\, \mathbf{0} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,?\, \mathtt{home} \,]\!] \parallel \langle \mathsf{p}, \mathtt{home}, \mathsf{q} \rangle \\
& \xrightarrow{\mathsf{p}\,\mathsf{q}?\mathtt{home}} & \mathsf{p}[\![\, \mathbf{0} \,]\!] \parallel \mathsf{q}[\![\, \mathbf{0} \,]\!] \parallel \emptyset
\end{array}$$

The two candidates

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home} \qquad \mathsf{G}_2 = \mathsf{q} \to \mathsf{p} : \mathtt{home}; \mathsf{p} \to \mathsf{q} : \mathtt{home}$$

fail to type $\mathbb{N}$!

$$\begin{array}{ll}
\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q} \,!\, \mathtt{home}; \mathsf{q} \,?\, \mathtt{home} & \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p} \,?\, \mathtt{home}; \mathsf{p} \,!\, \mathtt{home} \\
\mathsf{G}_2 \upharpoonright \mathsf{p} = \mathsf{q} \,?\, \mathtt{home}; \mathsf{q} \,!\, \mathtt{home} & \mathsf{G}_2 \upharpoonright \mathsf{q} = \mathsf{p} \,!\, \mathtt{home}; \mathsf{p} \,?\, \mathtt{home}
\end{array}$$

Index

Asynchronous Subtyping

- Asynchronous subtyping[2] enables controlled reordering of actions by anticipating outputs, e.g.,

$$\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home} \leq_A \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

- Let $\preceq$ be the transitive closure of $\leq$ and $\leq_A$

$$[\text{Net}]\ \frac{\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \preceq \mathsf{G}_1 \upharpoonright \mathsf{p} \qquad \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}; \preceq \mathsf{G}_1 \upharpoonright \mathsf{q}}{\vdash \mathsf{p}[\![\, \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home} \,]\!] : \mathsf{G}_1}$$

where

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home}$$

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \qquad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

[2] D. Mostrous, N. Yoshida, K. Honda: Global Principal Typing in Partially Commutative Asynchronous Sessions. ESOP 2009

Asynchronous Subtyping

- Asynchronous subtyping[2] enables controlled reordering of actions by anticipating outputs, e.g.,

$$\mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home} \leq_A \mathsf{p}\,?\,\mathtt{home}; \mathsf{p}\,!\,\mathtt{home}$$

- Let $\preceq$ be the transitive closure of $\leq$ and $\leq_A$

$$[\text{Net}]\ \frac{\mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home} \preceq \mathsf{G}_1 \upharpoonright \mathsf{p} \qquad \mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home}; \preceq \mathsf{G}_1 \upharpoonright \mathsf{q}}{\vdash \mathsf{p}[\![\,\mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,!\,\mathtt{home}; \mathsf{p}\,?\,\mathtt{home}\,]\!] : \mathsf{G}_1}$$

where

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home}$$

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathtt{home}; \mathsf{q}\,?\,\mathtt{home} \qquad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathtt{home}; \mathsf{p}\,!\,\mathtt{home}$$

Problem

[2] D. Mostrous, N. Yoshida, K. Honda: Global Principal Typing in Partially Commutative Asynchronous Sessions. ESOP 2009

Asynchronous Subtyping

- Asynchronous subtyping[2] enables controlled reordering of actions by anticipating outputs, e.g.,

$$\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home} \leq_A \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

- Let $\preceq$ be the transitive closure of $\leq$ and $\leq_A$

$$[\text{Net}]\ \frac{\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \preceq \mathsf{G}_1 \upharpoonright \mathsf{p} \qquad \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}; \preceq \mathsf{G}_1 \upharpoonright \mathsf{q}}{\vdash \mathsf{p}[\![\,\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] : \mathsf{G}_1}$$

where

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home}$$

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \qquad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

Problem

[2] D. Mostrous, N. Yoshida, K. Honda: Global Principal Typing in Partially Commutative Asynchronous Sessions. ESOP 2009

Asynchronous Subtyping

- Asynchronous subtyping[2] enables controlled reordering of actions by anticipating outputs, e.g.,

$$\mathsf{p}\,!\,\texttt{home};\mathsf{p}\,?\,\texttt{home} \leq_A \mathsf{p}\,?\,\texttt{home};\mathsf{p}\,!\,\texttt{home}$$

- Let $\preceq$ be the transitive closure of $\leq$ and $\leq_A$

$$[\mathrm{Net}]\ \frac{\mathsf{q}\,!\,\texttt{home};\mathsf{q}\,?\,\texttt{home} \preceq \mathsf{G}_1 \upharpoonright \mathsf{p} \qquad \mathsf{p}\,!\,\texttt{home};\mathsf{p}\,?\,\texttt{home}; \preceq \mathsf{G}_1 \upharpoonright \mathsf{q}}{\vdash \mathsf{p}[\![\,\mathsf{q}\,!\,\texttt{home};\mathsf{q}\,?\,\texttt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,!\,\texttt{home};\mathsf{p}\,?\,\texttt{home}\,]\!] : \mathsf{G}_1}$$

where

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \texttt{home}; \mathsf{q} \to \mathsf{p} : \texttt{home}$$

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\texttt{home};\mathsf{q}\,?\,\texttt{home} \qquad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\texttt{home};\mathsf{p}\,!\,\texttt{home}$$

Problem

[2] D. Mostrous, N. Yoshida, K. Honda: Global Principal Typing in Partially Commutative Asynchronous Sessions. ESOP 2009

Asynchronous Subtyping

- Asynchronous subtyping[2] enables controlled reordering of actions by anticipating outputs, e.g.,

$$\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home} \leq_A \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

- Let $\preceq$ be the transitive closure of $\leq$ and $\leq_A$

$$[\text{Net}]\ \frac{\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \preceq \mathsf{G}_1 \upharpoonright \mathsf{p} \qquad \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}; \preceq \mathsf{G}_1 \upharpoonright \mathsf{q}}{\vdash \mathsf{p}[\![\,\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] : \mathsf{G}_1}$$

where

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home}$$

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \quad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

Problem

asynchronous subtyping is undecidable[3], so $\vdash \mathbb{N} : \mathsf{G}$ is undecidable!

---

[2] D. Mostrous, N. Yoshida, K. Honda: Global Principal Typing in Partially Commutative Asynchronous Sessions. ESOP 2009

[3] M. Bravetti, M. Carbone, G. Zavattaro: Undecidability of asynchronous session subtyping. Information and Computation 2017

Asynchronous Subtyping

- Asynchronous subtyping[2] enables controlled reordering of actions by anticipating outputs, e.g.,

$$\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home} \leq_A \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

- Let $\preceq$ be the transitive closure of $\leq$ and $\leq_A$

$$[\text{Net}]\ \frac{\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \preceq \mathsf{G}_1 \upharpoonright \mathsf{p} \qquad \mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}; \preceq \mathsf{G}_1 \upharpoonright \mathsf{q}}{\vdash \mathsf{p}[\![\,\mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home}\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}\,!\,\mathtt{home};\mathsf{p}\,?\,\mathtt{home}\,]\!] : \mathsf{G}_1}$$

where

$$\mathsf{G}_1 = \mathsf{p} \to \mathsf{q} : \mathtt{home}; \mathsf{q} \to \mathsf{p} : \mathtt{home}$$

$$\mathsf{G}_1 \upharpoonright \mathsf{p} = \mathsf{q}\,!\,\mathtt{home};\mathsf{q}\,?\,\mathtt{home} \quad \mathsf{G}_1 \upharpoonright \mathsf{q} = \mathsf{p}\,?\,\mathtt{home};\mathsf{p}\,!\,\mathtt{home}$$

Problem

asynchronous subtyping is undecidable[3]*, so* $\vdash \mathbb{N} : \mathsf{G}$ *is undecidable!*

---

[2] D. Mostrous, N. Yoshida, K. Honda: Global Principal Typing in Partially Commutative Asynchronous Sessions. ESOP 2009

[3] M. Bravetti, M. Carbone, G. Zavattaro: Undecidability of asynchronous session subtyping. Information and Computation 2017

Our approach

reduce the gap between global types and asynchronous multiparty sessions

- split outputs and inputs in global types
- match global types with networks bypassing projection (decidable type-checking)
- give well-formedness conditions on global types to ensure good properties

Our approach

reduce the gap between global types and asynchronous multiparty sessions

- split outputs and inputs in global types
- match global types with networks bypassing projection (decidable type-checking)
- give well-formedness conditions on global types to ensure good properties

Our approach

reduce the gap between global types and asynchronous multiparty sessions

- split outputs and inputs in global types
- match global types with networks bypassing projection (decidable type-checking)
- give well-formedness conditions on global types to ensure good properties

Our approach

reduce the gap between global types and asynchronous multiparty sessions

- split outputs and inputs in global types
- match global types with networks bypassing projection (decidable type-checking)
- give well-formedness conditions on global types to ensure good properties

Asynchronous global types

$$G \quad ::=_{\rho} \quad \mathsf{p}\,\mathsf{q}!\{\lambda_i; G_i\}_{i\in I} \mid \mathsf{p}\,\mathsf{q}?\{\lambda_i; G_i\}_{i\in I} \mid \mathsf{End}$$

- $\mathsf{p}\,\mathsf{q}!\{\lambda_i; G_i\}_{i\in I}$ = output choice (p sends to q a label λ_i)
- $\mathsf{p}\,\mathsf{q}?\{\lambda_i; G_i\}_{i\in I}$ = input choice (q receives from p a label λ_i)
- End = termination

The active participants of a global type, players, are:

$$\mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i; G_i\}_{i\in I}) = \mathsf{players}(\mathsf{q}\,\mathsf{p}?\{\lambda_i; G_i\}_{i\in I}) = \{\mathsf{p}\} \cup \bigcup_{i\in I} \mathsf{players}(G_i)$$
$$\mathsf{players}(\mathsf{End}) = \emptyset$$

Example

$$G = \mathsf{p}\,\mathsf{q}!\mathrm{home}; \mathsf{q}\,\mathsf{p}!\mathrm{home}; \mathsf{p}\,\mathsf{q}?\mathrm{home}; \mathsf{q}\,\mathsf{p}?\mathrm{home}$$

Asynchronous global types

$$G \quad ::=_{\rho} \quad \mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{p}\,\mathsf{q}?\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{End}$$

- $\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}$ = output choice (p sends to q a label λ_i)
- $\mathsf{p}\,\mathsf{q}?\{\lambda_i; \mathsf{G}_i\}_{i\in I}$ = input choice (q receives from p a label λ_i)
- End = termination

The active participants of a global type, players, are:

$$\mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}) = \mathsf{players}(\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\} \cup \bigcup_{i\in I} \mathsf{players}(\mathsf{G}_i)$$
$$\mathsf{players}(\mathsf{End}) = \emptyset$$

Example

$$\mathsf{G} = \mathsf{p}\,\mathsf{q}!\mathrm{home}; \mathsf{q}\,\mathsf{p}!\mathrm{home}; \mathsf{p}\,\mathsf{q}?\mathrm{home}; \mathsf{q}\,\mathsf{p}?\mathrm{home}$$

Asynchronous global types

$$G \quad ::=_\rho \quad \mathsf{p}\,\mathsf{q}!\{\lambda_i; G_i\}_{i\in I} \mid \mathsf{p}\,\mathsf{q}?\{\lambda_i; G_i\}_{i\in I} \mid \mathsf{End}$$

- $\mathsf{p}\,\mathsf{q}!\{\lambda_i; G_i\}_{i\in I}$ = output choice (p sends to q a label λ_i)
- $\mathsf{p}\,\mathsf{q}?\{\lambda_i; G_i\}_{i\in I}$ = input choice (q receives from p a label λ_i)
- End = termination

The active participants of a global type, players, are:

$$\mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i; G_i\}_{i\in I}) = \mathsf{players}(\mathsf{q}\,\mathsf{p}?\{\lambda_i; G_i\}_{i\in I}) = \{\mathsf{p}\} \cup \bigcup_{i\in I} \mathsf{players}(G_i)$$
$$\mathsf{players}(\mathsf{End}) = \emptyset$$

Example

$$G = \mathsf{p}\,\mathsf{q}!\mathsf{home}; \mathsf{q}\,\mathsf{p}!\mathsf{home}; \mathsf{p}\,\mathsf{q}?\mathsf{home}; \mathsf{q}\,\mathsf{p}?\mathsf{home}$$

Asynchronous global types

$$\mathsf{G} \quad ::=_\rho \quad \mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{p}\,\mathsf{q}?\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{End}$$

- $\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}$ = output choice (p sends to q a label λ_i)
- $\mathsf{p}\,\mathsf{q}?\{\lambda_i; \mathsf{G}_i\}_{i\in I}$ = input choice (q receives from p a label λ_i)
- End = termination

The active participants of a global type, players, are:

$$\mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}) = \mathsf{players}(\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\} \cup \bigcup_{i\in I} \mathsf{players}(\mathsf{G}_i)$$
$$\mathsf{players}(\mathsf{End}) = \emptyset$$

Example

$$\mathsf{G} = \mathsf{p}\,\mathsf{q}!\mathrm{home}; \mathsf{q}\,\mathsf{p}!\mathrm{home}; \mathsf{p}\,\mathsf{q}?\mathrm{home}; \mathsf{q}\,\mathsf{p}?\mathrm{home}$$

Asynchronous global types

$$\mathsf{G} \quad ::=_{\rho} \quad \mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{p}\,\mathsf{q}?\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{End}$$

- $\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}$ = output choice (p sends to q a label λ_i)
- $\mathsf{p}\,\mathsf{q}?\{\lambda_i; \mathsf{G}_i\}_{i\in I}$ = input choice (q receives from p a label λ_i)
- End = termination

The active participants of a global type, players, are:

$$\mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}) = \mathsf{players}(\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\} \cup \bigcup_{i\in I} \mathsf{players}(\mathsf{G}_i)$$
$$\mathsf{players}(\mathsf{End}) = \emptyset$$

Example

$$\mathsf{G} = \mathsf{p}\,\mathsf{q}!\mathrm{home}; \mathsf{q}\,\mathsf{p}!\mathrm{home}; \mathsf{p}\,\mathsf{q}?\mathrm{home}; \mathsf{q}\,\mathsf{p}?\mathrm{home}$$

Asynchronous global types

$$\mathsf{G} \quad ::=_\rho \quad \mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{p}\,\mathsf{q}?\{\lambda_i; \mathsf{G}_i\}_{i\in I} \mid \mathsf{End}$$

- $\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}$ = output choice (p sends to q a label λ_i)
- $\mathsf{p}\,\mathsf{q}?\{\lambda_i; \mathsf{G}_i\}_{i\in I}$ = input choice (q receives from p a label λ_i)
- End = termination

The active participants of a global type, players, are:

$$\mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}) = \mathsf{players}(\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\} \cup \bigcup_{i\in I} \mathsf{players}(\mathsf{G}_i)$$
$$\mathsf{players}(\mathsf{End}) = \emptyset$$

Example

$$\mathsf{G} = \mathsf{p}\,\mathsf{q}!\mathrm{home}; \mathsf{q}\,\mathsf{p}!\mathrm{home}; \mathsf{p}\,\mathsf{q}?\mathrm{home}; \mathsf{q}\,\mathsf{p}?\mathrm{home}$$

Asynchronous global types

$$\mathsf{G} \quad ::=_\rho \quad \mathsf{p}\,\mathsf{q}!\{\lambda_i;\mathsf{G}_i\}_{i\in I} \mid \mathsf{p}\,\mathsf{q}?\{\lambda_i;\mathsf{G}_i\}_{i\in I} \mid \mathsf{End}$$

- $\mathsf{p}\,\mathsf{q}!\{\lambda_i;\mathsf{G}_i\}_{i\in I}$ = output choice (p sends to q a label λ_i)
- $\mathsf{p}\,\mathsf{q}?\{\lambda_i;\mathsf{G}_i\}_{i\in I}$ = input choice (q receives from p a label λ_i)
- End = termination

The active participants of a global type, players, are:

$$\mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i;\mathsf{G}_i\}_{i\in I}) = \mathsf{players}(\mathsf{q}\,\mathsf{p}?\{\lambda_i;\mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\} \cup \bigcup_{i\in I} \mathsf{players}(\mathsf{G}_i)$$
$$\mathsf{players}(\mathsf{End}) = \emptyset$$

Example

$$\mathsf{G} = \mathsf{p}\,\mathsf{q}!\mathsf{home}; \mathsf{q}\,\mathsf{p}!\mathsf{home}; \mathsf{p}\,\mathsf{q}?\mathsf{home}; \mathsf{q}\,\mathsf{p}?\mathsf{home}$$

Asynchronous global types

$$\mathsf{G} \quad ::=_\rho \quad \mathsf{p}\,\mathsf{q}!\{\lambda_i;\mathsf{G}_i\}_{i\in I} \mid \mathsf{p}\,\mathsf{q}?\{\lambda_i;\mathsf{G}_i\}_{i\in I} \mid \mathsf{End}$$

- $\mathsf{p}\,\mathsf{q}!\{\lambda_i;\mathsf{G}_i\}_{i\in I}$ = output choice (p sends to q a label λ_i)
- $\mathsf{p}\,\mathsf{q}?\{\lambda_i;\mathsf{G}_i\}_{i\in I}$ = input choice (q receives from p a label λ_i)
- End = termination

The active participants of a global type, players, are:

$$\mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i;\mathsf{G}_i\}_{i\in I}) = \mathsf{players}(\mathsf{q}\,\mathsf{p}?\{\lambda_i;\mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\} \cup \bigcup_{i\in I} \mathsf{players}(\mathsf{G}_i)$$
$$\mathsf{players}(\mathsf{End}) = \emptyset$$

Example

$$\mathsf{G} = \mathsf{p}\,\mathsf{q}!\mathtt{home};\mathsf{q}\,\mathsf{p}!\mathtt{home};\mathsf{p}\,\mathsf{q}?\mathtt{home};\mathsf{q}\,\mathsf{p}?\mathtt{home}$$

Typing Rules

$$[\text{End}]\ \frac{}{\mathsf{End} \vdash \mathsf{p}[\![\, 0 \,]\!]}$$

$$[\text{Out}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \mathsf{players}(\mathsf{G}_i) = \mathsf{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}!\{\lambda_i; P_i\}_{i \in I} \,]\!] \parallel \mathbb{N}}$$

$$[\text{In}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \mathsf{players}(\mathsf{G}_i) = \mathsf{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}?\{\lambda_j; P_j\}_{j \in J} \,]\!] \parallel \mathbb{N}}\ \ I \subseteq J$$

- Standard subtyping for input choices is incorporated in Rule [In]
- No need for subtyping in Rule [Out] , since no expressivity is lost.

Properties

Typing Rules

$$[\text{End}]\ \dfrac{}{\mathsf{End} \vdash \mathsf{p}[\![\, 0 \,]\!]}$$

$$[\text{Out}]\ \dfrac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \mathsf{players}(\mathsf{G}_i) = \mathsf{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}!\{\lambda_i; P_i\}_{i \in I} \,]\!] \parallel \mathbb{N}}$$

$$[\text{In}]\ \dfrac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \mathsf{players}(\mathsf{G}_i) = \mathsf{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}?\{\lambda_j; P_j\}_{j \in J} \,]\!] \parallel \mathbb{N}} \quad I \subseteq J$$

- Standard subtyping for input choices is incorporated in Rule [In]
- No need for subtyping in Rule [Out] , since no expressivity is lost.

Properties

Typing Rules

$$[\text{End}]\ \dfrac{}{\text{End} \vdash \mathsf{p}[\![\, 0 \,]\!]}$$

$$[\text{Out}]\ \dfrac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \text{players}(\mathsf{G}_i) = \text{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}!\{\lambda_i; P_i\}_{i \in I} \,]\!] \parallel \mathbb{N}}$$

$$[\text{In}]\ \dfrac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \text{players}(\mathsf{G}_i) = \text{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}?\{\lambda_j; P_j\}_{j \in J} \,]\!] \parallel \mathbb{N}} \quad I \subseteq J$$

- Standard subtyping for input choices is incorporated in Rule [In]
- No need for subtyping in Rule [Out] , since no expressivity is lost.

Properties

Typing Rules

$$[\text{End}]\ \frac{}{\text{End} \vdash \mathsf{p}[\![\, 0 \,]\!]}$$

$$[\text{Out}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \text{players}(\mathsf{G}_i) = \text{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}!\{\lambda_i; P_i\}_{i \in I} \,]\!] \parallel \mathbb{N}}$$

$$[\text{In}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \text{players}(\mathsf{G}_i) = \text{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}?\{\lambda_j; P_j\}_{j \in J} \,]\!] \parallel \mathbb{N}} \quad I \subseteq J$$

- Standard subtyping for input choices is incorporated in Rule [In]
- No need for subtyping in Rule [Out] , since no expressivity is lost.

Properties

- $\mathsf{G} \vdash \mathbb{N}$ is decidable
- for all $\mathbb{N}$ there is G such that $\mathsf{G} \vdash \mathbb{N}$

Typing Rules

$$[\text{End}]\ \frac{}{\text{End} \vdash \mathsf{p}[\![\, 0 \,]\!]}$$

$$[\text{Out}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \text{players}(\mathsf{G}_i) = \text{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\![\, \mathsf{q}!\{\lambda_i; P_i\}_{i\in I} \,]\!] \parallel \mathbb{N}}$$

$$[\text{In}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \text{players}(\mathsf{G}_i) = \text{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\![\, \mathsf{q}?\{\lambda_j; P_j\}_{j\in J} \,]\!] \parallel \mathbb{N}} \quad I \subseteq J$$

- Standard subtyping for input choices is incorporated in Rule [In]
- No need for subtyping in Rule [Out] , since no expressivity is lost.

Properties

- $\mathsf{G} \vdash \mathbb{N}$ is decidable
- for all $\mathbb{N}$ there is G such that $\mathsf{G} \vdash \mathbb{N}$

Typing Rules

$$[\text{End}]\ \frac{}{\mathsf{End} \vdash \mathsf{p}[\![\, 0 \,]\!]}$$

$$[\text{Out}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \mathsf{players}(\mathsf{G}_i) = \mathsf{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}!\{\lambda_i; P_i\}_{i \in I} \,]\!] \parallel \mathbb{N}}$$

$$[\text{In}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \mathsf{players}(\mathsf{G}_i) = \mathsf{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}?\{\lambda_j; P_j\}_{j \in J} \,]\!] \parallel \mathbb{N}} \quad I \subseteq J$$

- Standard subtyping for input choices is incorporated in Rule [In]
- No need for subtyping in Rule [Out] , since no expressivity is lost.

Properties

- $\mathsf{G} \vdash \mathbb{N}$ is decidable
- for all $\mathbb{N}$ there is G such that $\mathsf{G} \vdash \mathbb{N}$

Typing Rules

$$[\text{End}]\ \dfrac{}{\text{End} \vdash \mathsf{p}[\![\, 0 \,]\!]}$$

$$[\text{Out}]\ \dfrac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \text{players}(\mathsf{G}_i) = \text{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}!\{\lambda_i; P_i\}_{i \in I} \,]\!] \parallel \mathbb{N}}$$

$$[\text{In}]\ \dfrac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \qquad \text{players}(\mathsf{G}_i) = \text{players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I}{\mathsf{q}\,\mathsf{p}?\{\lambda_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\, \mathsf{q}?\{\lambda_j; P_j\}_{j \in J} \,]\!] \parallel \mathbb{N}} \quad I \subseteq J$$

- Standard subtyping for input choices is incorporated in Rule [In]
- No need for subtyping in Rule [Out] , since no expressivity is lost.

Properties

- $\mathsf{G} \vdash \mathbb{N}$ is decidable
- for all $\mathbb{N}$ there is G such that $\mathsf{G} \vdash \mathbb{N}$

Need for restrictions

Problem

assigning a global type to a network does not ensure progress

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\lambda_1 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\lambda_2 \,]\!]$ and $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2$

$\mathsf{G} \vdash \mathbb{N}$

$\mathbb{N} \parallel \emptyset$ is deadlocked and λ_1 is an orphan message

need for well-formedness conditions on global types

- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is well-formed
- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \emptyset$ is NOT well-formed

well-formedness depends on the queue $\mathcal{M}$

Need for restrictions

Problem

assigning a global type to a network does not ensure progress

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q} \,!\, \lambda_1 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,?\, \lambda_2 \,]\!]$ and $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2$

$\mathsf{G} \vdash \mathbb{N}$

$\mathbb{N} \parallel \emptyset$ is deadlocked and λ_1 is an orphan message

need for well-formedness conditions on global types

- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is well-formed
- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \emptyset$ is NOT well-formed

well-formedness depends on the queue $\mathcal{M}$

Need for restrictions

Problem

assigning a global type to a network does not ensure progress

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q} \,!\, \lambda_1 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,?\, \lambda_2 \,]\!]$ and $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2$

$$\mathsf{G} \vdash \mathbb{N}$$

$\mathbb{N} \parallel \emptyset$ is deadlocked and λ_1 is an orphan message

need for well-formedness conditions on global types

- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is well-formed
- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \emptyset$ is NOT well-formed

well-formedness depends on the queue $\mathcal{M}$

Need for restrictions

Problem

assigning a global type to a network does not ensure progress

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q} \,!\, \lambda_1 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,?\, \lambda_2 \,]\!]$ and $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1 ; \mathsf{p}\,\mathsf{q}?\lambda_2$

$$\mathsf{G} \vdash \mathbb{N}$$

$\mathbb{N} \parallel \emptyset$ is deadlocked and λ_1 is an orphan message

need for well-formedness conditions on global types

- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is well-formed
- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \emptyset$ is NOT well-formed

well-formedness depends on the queue $\mathcal{M}$

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Need for restrictions

Problem

assigning a global type to a network does not ensure progress

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\lambda_1 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\lambda_2 \,]\!]$ and $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2$

$$\mathsf{G} \vdash \mathbb{N}$$

$\mathbb{N} \parallel \emptyset$ is deadlocked and λ_1 is an orphan message

need for well-formedness conditions on global types

- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is well-formed
- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \emptyset$ is NOT well-formed

well-formedness depends on the queue $\mathcal{M}$

Need for restrictions

Problem

assigning a global type to a network does not ensure progress

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q}\,!\,\lambda_1 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\lambda_2 \,]\!]$ and $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2$

$$\mathsf{G} \vdash \mathbb{N}$$

$\mathbb{N} \parallel \emptyset$ is deadlocked and λ_1 is an orphan message

need for well-formedness conditions on global types

- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is well-formed
- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \emptyset$ is NOT well-formed

well-formedness depends on the queue $\mathcal{M}$

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Need for restrictions

Problem

assigning a global type to a network does not ensure progress

Let $\mathbb{N} = \mathsf{p}[\![\, \mathsf{q} \,!\, \lambda_1 \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p} \,?\, \lambda_2 \,]\!]$ and $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2$

$$\mathsf{G} \vdash \mathbb{N}$$

$\mathbb{N} \parallel \emptyset$ is deadlocked and λ_1 is an orphan message

need for well-formedness conditions on global types

- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is well-formed
- $\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End} \parallel \emptyset$ is NOT well-formed

well-formedness depends on the queue $\mathcal{M}$

Well-formedness and Progress

A type configuration G ‖ $\mathcal{M}$ is well-formed if

- G is bounded
- G ‖ $\mathcal{M}$ is balanced , i.e.,
  - at least one of the labels of every input choice of G is matched by either a message in $\mathcal{M}$ or a preceding output in G
  - every message in $\mathcal{M}$ will be eventually read by G.

Theorem

If G ⊢ $\mathbb{N}$ for some G and G ‖ $\mathcal{M}$ is well-formed, then $\mathbb{N}$ ‖ $\mathcal{M}$ has the progress property.

Well-formedness and Progress

A type configuration $\mathsf{G} \parallel \mathcal{M}$ is well-formed if

- G is bounded
- $\mathsf{G} \parallel \mathcal{M}$ is balanced , i.e.,
  - at least one of the labels of every input choice of G is matched by either a message in $\mathcal{M}$ or a preceding output in G
  - every message in $\mathcal{M}$ will be eventually read by G.

Theorem

If $\mathsf{G} \vdash \mathbb{N}$ for some G and $\mathsf{G} \parallel \mathcal{M}$ is well-formed, then $\mathbb{N} \parallel \mathcal{M}$ has the progress property.

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Well-formedness and Progress

A type configuration $\mathsf{G} \parallel \mathcal{M}$ is well-formed if

- G is bounded
- $\mathsf{G} \parallel \mathcal{M}$ is balanced , i.e.,
  - at least one of the labels of every input choice of G is matched by either a message in $\mathcal{M}$ or a preceding output in G
  - every message in $\mathcal{M}$ will be eventually read by G.

Theorem

If $\mathsf{G} \vdash \mathbb{N}$ for some G and $\mathsf{G} \parallel \mathcal{M}$ is well-formed, then $\mathbb{N} \parallel \mathcal{M}$ has the progress property.

Well-formedness and Progress

A type configuration $\mathsf{G} \parallel \mathcal{M}$ is well-formed if

- G is bounded
- $\mathsf{G} \parallel \mathcal{M}$ is balanced , i.e.,
  - at least one of the labels of every input choice of G is matched by either a message in $\mathcal{M}$ or a preceding output in G
  - every message in $\mathcal{M}$ will be eventually read by G.

Theorem

If $\mathsf{G} \vdash \mathbb{N}$ for some G and $\mathsf{G} \parallel \mathcal{M}$ is well-formed, then $\mathbb{N} \parallel \mathcal{M}$ has the progress property.

Well-formedness and Progress

A type configuration $\mathsf{G} \parallel \mathcal{M}$ is well-formed if

- G is bounded
- $\mathsf{G} \parallel \mathcal{M}$ is balanced , i.e.,
  - at least one of the labels of every input choice of G is matched by either a message in $\mathcal{M}$ or a preceding output in G
  - every message in $\mathcal{M}$ will be eventually read by G.

Theorem

If $\mathsf{G} \vdash \mathbb{N}$ for some G and $\mathsf{G} \parallel \mathcal{M}$ is well-formed, then $\mathbb{N} \parallel \mathcal{M}$ has the progress property.

Well-formedness and Progress

A type configuration $\mathsf{G} \parallel \mathcal{M}$ is well-formed if

- G is bounded
- $\mathsf{G} \parallel \mathcal{M}$ is balanced , i.e.,
  - at least one of the labels of every input choice of G is matched by either a message in $\mathcal{M}$ or a preceding output in G
  - every message in $\mathcal{M}$ will be eventually read by G.

Theorem

If $\mathsf{G} \vdash \mathbb{N}$ for some G and $\mathsf{G} \parallel \mathcal{M}$ is well-formed, then $\mathbb{N} \parallel \mathcal{M}$ has the progress property.

Well-formedness and Progress

A type configuration $\mathsf{G} \parallel \mathcal{M}$ is well-formed if

- G is bounded
- $\mathsf{G} \parallel \mathcal{M}$ is balanced , i.e.,
  - at least one of the labels of every input choice of G is matched by either a message in $\mathcal{M}$ or a preceding output in G
  - every message in $\mathcal{M}$ will be eventually read by G.

Theorem

If $\mathsf{G} \vdash \mathbb{N}$ for some G and $\mathsf{G} \parallel \mathcal{M}$ is well-formed, then $\mathbb{N} \parallel \mathcal{M}$ has the progress property.

Checking Well-formedness

- Balancing is undecidable.
- We defined a decidable restriction of weak balancing that allows to type multiparty sessions that are not typable by other decidable restrictions of asynchronous typing[4]
- We can type the running example of [4]
- However, we do not wether there is an example typable in [4] which is not typable in our system!

[4] M. Bravetti, M. Carbone, J. Lange, N. Yoshida, G. Zavattaro: A Sound Algorithm for Asynchronous Session Subtyping. CONCUR 2019

Checking Well-formedness

- Balancing is undecidable.
- We defined a decidable restriction of weak balancing that allows to type multiparty sessions that are not typable by other decidable restrictions of asynchronous typing[4]
- We can type the running example of [4]
- However, we do not wether there is an example typable in [4] which is not typable in our system!

[4] M. Bravetti, M. Carbone, J. Lange, N. Yoshida, G. Zavattaro: A Sound Algorithm for Asynchronous Session Subtyping. CONCUR 2019

Checking Well-formedness

- Balancing is undecidable.
- We defined a decidable restriction of weak balancing that allows to type multiparty sessions that are not typable by other decidable restrictions of asynchronous typing[4]
- We can type the running example of [4]
- However, we do not wether there is an example typable in [4] which is not typable in our system!

[4] M. Bravetti, M. Carbone, J. Lange, N. Yoshida, G. Zavattaro: A Sound Algorithm for Asynchronous Session Subtyping. CONCUR 2019

Checking Well-formedness

- Balancing is undecidable.
- We defined a decidable restriction of weak balancing that allows to type multiparty sessions that are not typable by other decidable restrictions of asynchronous typing[4]
- We can type the running example of [4]
- However, we do not wether there is an example typable in [4] which is not typable in our system!

[4] M. Bravetti, M. Carbone, J. Lange, N. Yoshida, G. Zavattaro: A Sound Algorithm for Asynchronous Session Subtyping. CONCUR 2019

Index

Contributions

- a new formalism of global types splitting outputs and inputs
- a decidable type-checking for asynchronous sessions
- an algorithm ensuring well-formedness of type configurations
- a prototype implementation in co-logic programming of a preliminary version of the type system

Contributions

- a new formalism of global types splitting outputs and inputs
- a decidable type-checking for asynchronous sessions
- an algorithm ensuring well-formedness of type configurations
- a prototype implementation in co-logic programming of a preliminary version of the type system

Contributions

- a new formalism of global types splitting outputs and inputs
- a decidable type-checking for asynchronous sessions
- an algorithm ensuring well-formedness of type configurations
- a prototype implementation in co-logic programming of a preliminary version of the type system

Contributions

- a new formalism of global types splitting outputs and inputs
- a decidable type-checking for asynchronous sessions
- an algorithm ensuring well-formedness of type configurations
- a prototype implementation in co-logic programming of a preliminary version of the type system

Other work on global types

- reversible multiparty sessions (with also optional participants [2])
  [1] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Concurrent Reversible Sessions. CONCUR 2017.
  [2] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Reversible sessions with flexible choices. Acta Informatica 2019.
- delegation
  [3] I. Castellani, M. Dezani-Ciancaglini, P. Giannini, R. Horne: Global types with internal delegation. Theoretical Computer Science 2020.
- asynchronous global types (preliminary version of those described here)
  [4] F. Dagnino, P. Giannini, M. Dezani-Ciancaglini: Deconfined Global Types for Asynchronous Sessions. COORDINATION 2021.
- processes with input races
  [5] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Asynchronous Sessions with Input Races. PLACES@ETAPS 2022
- a tool for multiparty-session-types coordination for core Erlang
  [6] L. Egidi , P. Giannini, L. Ventura: Multiparty-session-types coordination for core Erlang. ICSOFT 2022.

Other work on global types

- reversible multiparty sessions (with also optional participants [2])
  [1] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Concurrent Reversible Sessions. CONCUR 2017.
  [2] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Reversible sessions with flexible choices. Acta Informatica 2019.
- delegation
  [3] I. Castellani, M. Dezani-Ciancaglini, P. Giannini, R. Horne: Global types with internal delegation. Theoretical Computer Science 2020.
- asynchronous global types (preliminary version of those described here)
  [4] F. Dagnino, P. Giannini, M. Dezani-Ciancaglini: Deconfined Global Types for Asynchronous Sessions. COORDINATION 2021.
- processes with input races
  [5] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Asynchronous Sessions with Input Races. PLACES@ETAPS 2022
- a tool for multiparty-session-types coordination for core Erlang
  [6] L. Egidi , P. Giannini, L. Ventura: Multiparty-session-types coordination for core Erlang. ICSOFT 2022.

Other work on global types

- reversible multiparty sessions (with also optional participants [2])
  [1] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Concurrent Reversible Sessions. CONCUR 2017.
  [2] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Reversible sessions with flexible choices. Acta Informatica 2019.
- delegation
  [3] I. Castellani, M. Dezani-Ciancaglini, P. Giannini, R. Horne: Global types with internal delegation. Theoretical Computer Science 2020.
- asynchronous global types (preliminary version of those described here)
  [4] F. Dagnino, P. Giannini, M. Dezani-Ciancaglini: Deconfined Global Types for Asynchronous Sessions. COORDINATION 2021.
- processes with input races
  [5] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Asynchronous Sessions with Input Races. PLACES@ETAPS 2022
- a tool for multiparty-session-types coordination for core Erlang
  [6] L. Egidi , P. Giannini, L. Ventura: Multiparty-session-types coordination for core Erlang. ICSOFT 2022.

Other work on global types

- reversible multiparty sessions (with also optional participants [2])
  [1] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Concurrent Reversible Sessions. CONCUR 2017.
  [2] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Reversible sessions with flexible choices. Acta Informatica 2019.
- delegation
  [3] I. Castellani, M. Dezani-Ciancaglini, P. Giannini, R. Horne: Global types with internal delegation. Theoretical Computer Science 2020.
- asynchronous global types (preliminary version of those described here)
  [4] F. Dagnino, P. Giannini, M. Dezani-Ciancaglini: Deconfined Global Types for Asynchronous Sessions. COORDINATION 2021.
- processes with input races
  [5] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Asynchronous Sessions with Input Races. PLACES@ETAPS 2022
- a tool for multiparty-session-types coordination for core Erlang
  [6] L. Egidi , P. Giannini, L. Ventura: Multiparty-session-types coordination for core Erlang. ICSOFT 2022.

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Other work on global types

- reversible multiparty sessions (with also optional participants [2])
  [1] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Concurrent Reversible Sessions. CONCUR 2017.
  [2] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Reversible sessions with flexible choices. Acta Informatica 2019.
- delegation
  [3] I. Castellani, M. Dezani-Ciancaglini, P. Giannini, R. Horne: Global types with internal delegation. Theoretical Computer Science 2020.
- asynchronous global types (preliminary version of those described here)
  [4] F. Dagnino, P. Giannini, M. Dezani-Ciancaglini: Deconfined Global Types for Asynchronous Sessions. COORDINATION 2021.
- processes with input races
  [5] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Asynchronous Sessions with Input Races. PLACES@ETAPS 2022
- a tool for multiparty-session-types coordination for core Erlang
  [6] L. Egidi , P. Giannini, L. Ventura: Multiparty-session-types coordination for core Erlang. ICSOFT 2022.

Other work on global types

- reversible multiparty sessions (with also optional participants [2])
  [1] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Concurrent Reversible Sessions. CONCUR 2017.
  [2] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Reversible sessions with flexible choices. Acta Informatica 2019.
- delegation
  [3] I. Castellani, M. Dezani-Ciancaglini, P. Giannini, R. Horne: Global types with internal delegation. Theoretical Computer Science 2020.
- asynchronous global types (preliminary version of those described here)
  [4] F. Dagnino, P. Giannini, M. Dezani-Ciancaglini: Deconfined Global Types for Asynchronous Sessions. COORDINATION 2021.
- processes with input races
  [5] I. Castellani, M. Dezani-Ciancaglini, P. Giannini: Asynchronous Sessions with Input Races. PLACES@ETAPS 2022
- a tool for multiparty-session-types coordination for core Erlang
  [6] L. Egidi , P. Giannini, L. Ventura: Multiparty-session-types coordination for core Erlang. ICSOFT 2022.

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Use of global types in T-Ladies

- Coordination of IoT applications
- Are there properties of IoT applications we may want to enforce ?
- Multiparty-session-types coordination for JadeScript (we did it for Erlang!)

Use of global types in T-Ladies

- Coordination of IoT applications
- Are there properties of IoT applications we may want to enforce ?
- Multiparty-session-types coordination for JadeScript (we did it for Erlang!)

UPO UNIVERSITÀ DEL PIEMONTE ORIENTALE

Use of global types in T-Ladies

- Coordination of IoT applications
- Are there properties of IoT applications we may want to enforce ?
- Multiparty-session-types coordination for JadeScript (we did it for Erlang!)